TOSHIBA

USERS MANUAL

16-BIT MICROCONTROLLER

1992

TLCS-900 SERIES

TMP96C141

TMP96CM40

TMP96PM40

The information contained herein is subject to change without notice.

The information contained herein is presented only as a guide for the applications of our products. No responsibility is assumed by TOSHIBA for any infringements of patents or other rights of the third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of TOSHIBA or others.

The products described in this document contain strategic products subject to COCOM regulations. They should not be exported without authorization from the appropriate governmental authorities.

These TOSHIBA products are intended for usage in general electronic equipments (office equipment, communication equipment, measuring equipment, domestic electrification, etc.) . Please make sure that you consult with us before you use these TOSHIBA products in equipments which require high quality and/or reliability, and in equipments which could have major impact to the welfare of human life (atomic energy control, airplane, spaceship, traffic signal, combustion control, all types of safety devices, etc.) . TOSHIBA cannot accept liability to any damage which may occur in case these TOSHIBA products were used in the mentioned equipments without prior consultation with TOSHIBA.

PREFACE

Thank you very much for making use of Toshiba microcomputer LSIs and development systems.

Toshiba has a broad range of microcomputer LSIs which are applicable to various fields ranging from consumer to industrial. This document describes the 16-bit microcontroller TLCS-900 series with regard to system architecture, electrical characteristics and package dimensions.

The TLCS-900 series is the original Toshiba microcontroller which has features in multifunction I/Os and large-capacity program and data areas.

The current market tends to be compact, and demands the systems be simple and reasonably priced. Additionally, one-chip I/O peripherals and high-performance microcontrollers are demanded. The TLCS-900 series has been developed to satisfy all these needs clearly.

Because Toshiba constantly modifies systems to meet specific requests, updated 16-bit microcontroller families will always be available.

TABLE OF CONTENTS

CHAPTER 1 TLCS-900 CPU

CHAPTER 2 TLCS-900 LSI DEVICES

TLCS-900 SERIES

TOSHIBA

CHAPTER 1 TLCS-900 CPU

TOSHIBA CORPORATION

1. OUTLINE

The TLCS-900 series has an original Toshiba high-performance 16-bit CPU. Combining the CPU with various I/O function blocks (such as timers, serial I/Os, ADs) creates broad possibilities in application fields.

The TLCS-900 CPU, being 16-bit CPU, has a 32-bit/16-bit register bank configuration, therefore it is suitable as an embedded controller.

The TLCS-900 CPU features are as follows :

(1)TLCS-90 extended architecture

- Upward compatibility on mnemonic and register set levels

(2)General-purpose registers

- All 8 registers usable as accumulator

(3)Register bank system

- Minimum mode : eight 16-bit register banks
- Maximum mode : four 32-bit register banks

(4)16M-byte linear address space ; 9 types addressing modes

(5)Dynamic bus sizing system

- Can consist 8- / 16-bit external data bus together

(6)High reliability

- Supporting system mode and normal mode

(7)Orthogonal instruction sets

- 8-/16-/32-bit data transfer/arithmetic instructions
- 16-bit multiplication/division
  16×16 to 32-bits (signed/unsigned = 3.25μs @16 MHz)
  $32 \div 16$ to 16 bits (unsigned = 3.75μs, signed = 4.0μs@16 MHz)
- Bit processing including bit arithmetic
- Supporting instruction for C compiler
- Filter calculations : multiplication-addition arithmetic, modulo increment instruction

(8)High-speed processing

- Minimum instruction execution time: 250ns @16MHz (200ns@20MHz version : under development)
- Pipeline system with 4-byte instruction queue buffer
- 16-bit ALU

2. CPU OPERATING MODES (system and normal modes)

The TLCS-900 has two types of operating modes : system and normal. These modes are switched by instructions or interrupts. In system mode, there are no restrictions on using instructions or registers.

The CPU resources effective in system mode are as follows :

(1)General-purpose registers
- Four 16-bit general-purpose registers × 8 banks (minimum mode)
  or
  Four 32-bit general-purpose registers × 4 banks (maximum mode)
- Four 32-bit general-purpose registers (including system stack pointer : XSP)

(2)Status register (SR) : including system mode flag

(3)Program counter (PC): 32 bits for maximum mode, 16 bits for minimum mode

(4)Control register: parameter register for high-speed micro DMA, etc.

(5)Normal stack pointer: accessible as control register (XNSP)

(6)All CPU instructions

(7)All built-in I/O registers

(8)All built-in memories

In normal mode, the ineffective CPU resources are as follows :

(1)Privileged instructions (PUSH SR, POP SR, EI, DI, RETI, HALT, LDC, etc.)

(2)Controlling status register (SR) flags
- <SYSM>, <IFF0~2>, <MAX>

(3)Control register (CR) : parameter registers for high-speed micro DMA, etc.

(4)Built-in I/O registers (depending on products)

Product name	Built-in I/O registers which cannot be accessed in normal mode
96C141 96CM40 96PM40	Chip select/wait controller (B0CS, B1CS, B2CS registers)

(5)Built-in memories (depending on products)

Product name	Memories which cannot be accessed in normal mode
96C141 96CM40 96PM40	Memory blocks whose built-in chip select/wait controller's B0SYS, B1SYS, and B2SYS bits are set to 1 (memory space set to system mode)

The stack pointers (SP) are provided in both system mode and normal mode, named SYSTEM STACK POINTER and NORMAL STACK POINTER. These pointers are automatically switched when the CPU mode is changed by the NORMAL instruction or an interrupt. In system mode, the normal stack pointer (XNSP) is handled as a control register, and can be accessed by the LDC instruction.

The CPU enters system mode by system reset, as well as by interrupt. The CPU changes from system to normal mode by the NORMAL instruction. The NORMAL instruction resets the <SYSM> bit of the status register (SR) to "0", and sets the CPU to normal mode.

This makes it possible for an OS-less system to configure software by using system mode only. Because this mode changing to the NORMAL mode is only done by executing the "NORMAL" instruction.

3. REGISTERS

3.1 Register Structure

Figure 3.1 (1) and (2) illustrate the format of registers. The TLCS-900 series has two register modes.

① Minimum mode····64K-byte program area/16M-byte data area

Four 16-bit general-purpose registers × 8 banks
+
Four 32-bit general-purpose registers
+
16-bit program counter

② Maximum mode····16M-byte program area/16M-byte data area

Four 32-bit general-purpose registers × 4 banks
+
Four 32-bit general-purpose registers
+
32-bit program counter

Register mode is switched by changing the <MAX> bit of the status register with the MAX instruction. (After reset, the mode is set to minimum.)

The stack pointer (SP) is provided for each operating mode (System and Normal mode), and switched automatically when the CPU mode is changed. The flag register (F') is used for temporary register. To change the flag registers (F, F'), use the EX instruction (EX F, F').

After RESET, the system stack pointer (XSP) is set to 100H, the program counter (PC) to 8000H, and the upper bytes of the status register to F0H. But RESET does not initialize the normal stack pointer, the lower bytes of status register (flag register) and general-purpose registers.

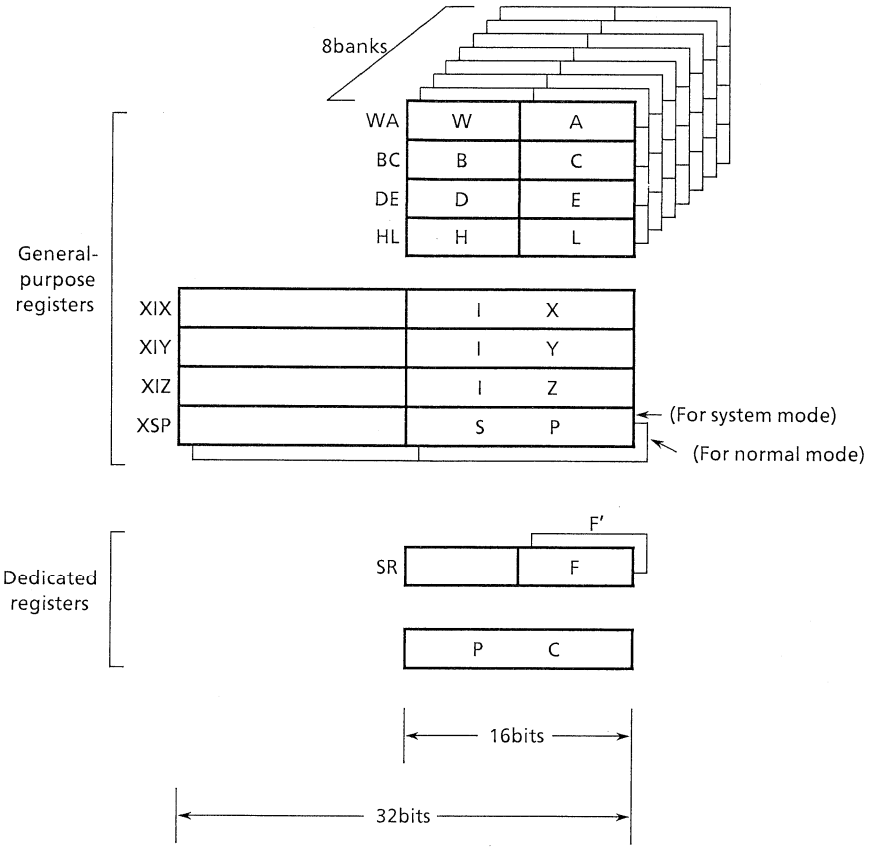


Figure 3.1 (1) Register Format (minimum mode: 64K-byte program area)

Note : The data memory area is 16M-byte.
The whole 16M-byte area can be accessed by using the registers (XIX, XIY, XIZ, XSP) or absolute addressing mode.

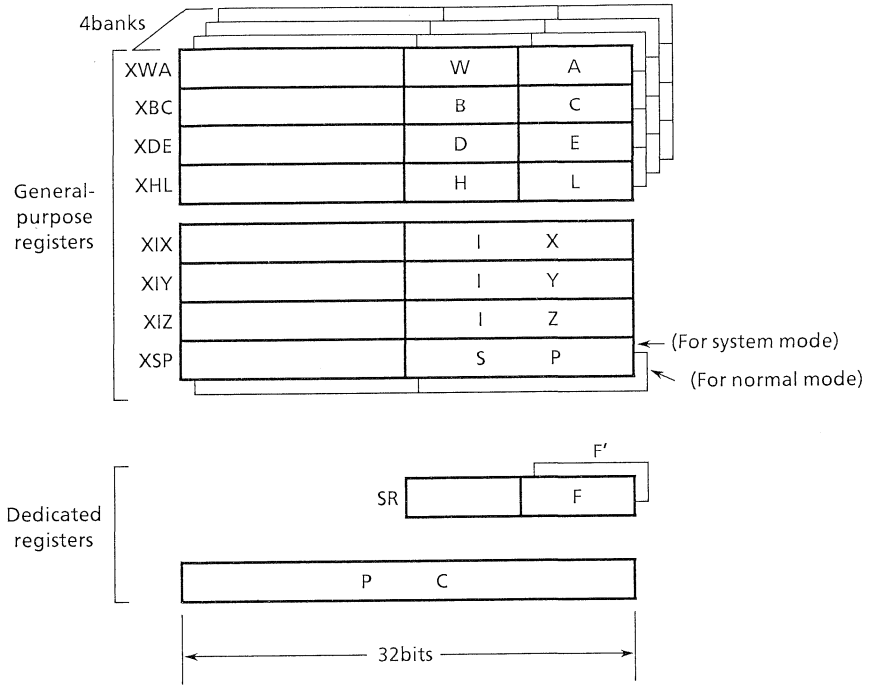


Figure 3.1 (2) Register Format (maximum mode: 16M-byte program area)

After reset, the CPU enters minimum mode. The register banks are 16 bits wide. To change to maximum mode, use the MAX instruction. To change from maximum to minimum mode there is no dedicated instruction; instead, the RETI or POP SR instruction changes the <MAX> bit of the status register.

When the mode changes from minimum to maximum, the 16-bit general-purpose registers (WA, BC, DE, and HL) are extended to 32-bit general-purpose registers (XWA, XBC, XDE, and XHL). The value of the upper 16 bits (that is, bit 16 to bit 31) are undefined. Those registers need to be initialized before use. Changing the mode from minimum to maximum also extends the program counter to 32 bits which automatically writes "0" to the upper 16 bits.

So doing ensures program continuity.

3.2 Register Details

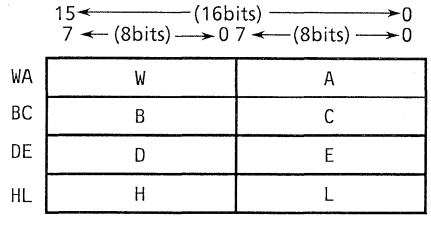
3.2.1 General-purpose bank registers

As explained in the previous section, the TLCS-900 series has two register formats. Which of the register formats is used depends on whether the mode is minimum or maximum. In either way, the register sets and registers in each bank are used exactly the same.

(1) General-purpose Bank Registers in Minimum Mode

In minimum mode, the following four 16-bit general-purpose registers consisting of 8 banks can be used. The register format in a bank is shown below.

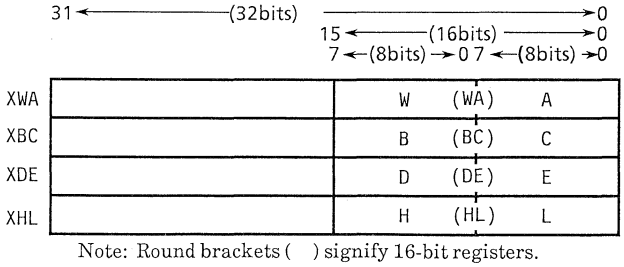
Four 16-bit registers (WA, BC, DE, and HL) are general-purpose registers and can be used as accumulators, index registers, and displacement registers. They can also be used as 8-bit registers (W, A, B, C, D, E, H, and L) to function for example as accumulators.



(2) General-purpose Bank Registers in Maximum Mode

In maximum mode, the following four 32-bit general-purpose registers consisting of 4 banks can be used. The register format in a bank is shown below.

Four 32-bit registers (XWA, XBC, XDE, and XHL) are general-purpose registers and can be used as an accumulators and index registers. They can also be used as 16-bit registers (WA, BC, DE, and HL), in which case, the lower 16 bits of the 32-bit registers are assigned.



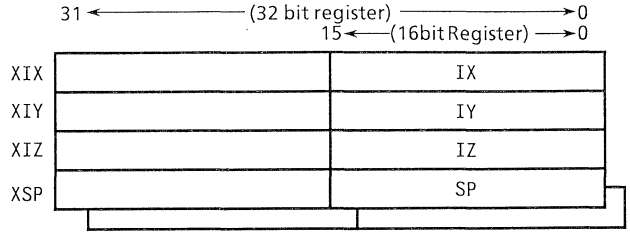
Note: Round brackets ( ) signify 16-bit registers.

16-bit registers can be used as accumulators, index registers in index addressing mode, and displacement registers. They can also be used as two 8-bit general-purpose registers (W, A, B, C, D, E, H, and L) to function for example as accumulators.

3.2.2 32-bit General-purpose Registers

The TLCS-900 series has four 32-bit general-purpose registers (XIX, XIY, XIZ, and XSP). They are fixed, independent of maximum or minimum mode. The register format is shown below.

These registers can also be used as accumulators, index registers, and displacement registers. They can be used either as 16-bit, or 8-bit registers. Names when registers are used as 8-bit registers are listed later.



The XSP register is utilized for stack pointers. This register is provided for both SYSTEM and NORMAL mode. Stack pointer for SYSTEM mode is called SYSTEM STACK POINTER, and for NORMAL mode is called NORMAL STACK POINTER. NORMAL and SYSTEM stack pointer are independent and switched automatically by change of the CPU operating mode. In both modes, they are referred to as XSP. The system stack pointer (XSP) is not able to be accessed from normal mode. The normal stack pointer (XSP) is able to be accessed from system mode as one of control registers (CR). In this case, it is referred to as XNSP. The XNSP can be accessed using the (privileged) LDC instruction.

When an interrupt occurs in the normal mode, the CPU enters system mode. At the same time, the normal stack pointer changes automatically to the system stack pointer (XSP). Then the previous normal stack pointer can be changed as a control register. After return using the RETI instruction, the mode and the stack pointers become normal.

After reset, the system stack pointer is initialized to 100H; the normal stack pointer remains undefined. Thus, when changing to normal mode it is necessary to initialize XSP.

3.2.3 Status Register (SR)

The status register contains flags indicating the status (operating mode, register format, etc.) of the CPU and operation results. This register consists of two parts. The lower byte (bits 0 to 7) can be accessed in either mode, normal or system. The upper byte (bits 8 to 15) can be accessed only in system mode. (However, in normal mode the LDF/INCF/DECF instruction can be used to overwrite bits 8 to 10.) The upper byte of the status register (bits 8 to 15) indicates the CPU status. The lower byte (bits 0 to 7) are referred to as the flag register (F). This indicates the status of the operation result. The TLCS-900 series has two flag registers (F and F'). They can be switched using the EX instruction.

(1) Upper Byte of Status Register

15	14	13	12	11	10	9	8	
SYSM	IFF2	IFF1	IFF0	MAX	RFP2	RFP1	RFP0	: R/W

① SYSM (SYStem Mode)

Indicates the CPU operating mode, system or normal. In system mode, all instructions can be executed. In normal mode, privileged instructions cannot be executed. (If forced, a privilege violation interrupt will occur.)

Initialized to 1 (system mode) by reset. To change to normal mode, use the NORMAL instruction. An interrupt automatically causes the mode to change from normal to system.

0	Normal mode
1	System mode

② IFF2 to IFF0 (Interrupt mask Flip-Flop2 to 0)

Mask registers with interrupt levels from 1 to 7. Level 7 has the highest priority.

Initialized to 111 by reset.

000	Enables interrupts with level 1 or higher.
001	Enables interrupts with level 1 or higher.
010	Enables interrupts with level 2 or higher.
011	Enables interrupts with level 3 or higher.
100	Enables interrupts with level 4 or higher.
101	Enables interrupts with level 5 or higher.
110	Enables interrupts with level 6 or higher.
111	Enables interrupts with level 7 only (non-maskable interrupt).

Same

Any value can be set using the EI instruction.

When an interrupt is received, the mask register sets a value higher by 1 than the interrupt level received. When an interrupt with level 7 is received, 111 is set. Unlike with the TLCS-90 series, the EI instruction becomes effective immediately after execution.

③ MAX (minimum/MAXimum)

Bit used to specify the register mode which determines the sizes of the register banks and the program counter.

0	Minimum mode
1	Maximum mode

If the program size exceeds 64K bytes, use the MAX instruction to set this register to 1 so that register mode becomes maximum mode.

Initialized to 0 by reset.

④ RFP2~RFP0 (Register File Pointer2~0)

Indicates the number of register file (register bank) currently being used. Initialized to 000 by reset.

The values in these registers can be operated on using the following three instructions. RFP2 is fixed to 0 in maximum mode. It remains 0 even if an attempt to change it to 1 using following instructions.

- LDF imm ; RFP←imm (0~7) (250ns @16MHz)
- INCF ; RFP←RFP+1 (250ns @16MHz)
- DECF ; RFP←RFP−1 (250ns @16MHz)

(2) Flag Register, F

7	6	5	4	3	2	1	0	
S	Z	"0"	H	"0"	V	N	C	: R/W

① S (Sign flag)

"1" is set when the operation result is negative, "0" when positive.
(The value of the most significant bit of the operation result is copied.)

② Z (Zero flag)

"1" is set when the operation result is zero, otherwise "0".

③ H (Half carry flag)

"1" is set when a carry or borrow from bit 3 to bit 4 occurs as a result of the operation, otherwise "0". With a 32-bit operation instruction, an undefined value is set.

④ V (Parity/over-flow flag)

Indicates either parity or overflow, depending on the operation type.
Parity (P): "0" is set when the number of bits set to 1 is odd, "1" when even.
An undefined value is set with a 32-bit operation instruction.
Overflow (V): "0" is set if no overflow, if overflow "1".

⑤ N (Negative)

ADD/SUB flag
"0" is set after an addition instruction such as ADD is executed, "1" after a subtraction instruction such as SUB.
Used when executing the DAA (decimal addition adjust accumulator) instruction.

⑥ C (Carry flag)

"1" is set when a carry or borrow occurs, otherwise "0".

3.2.4 Program Counter (PC)

The program counter is a pointer indicating the memory address to be executed next. The program counter bit length depends on whether the register format is in minimum or maximum mode.

In minimum mode, the program counter consists of 16 bits, and a maximum program area of 64K bytes (from addresses 000000H to 00FFFFH) can be accessed.

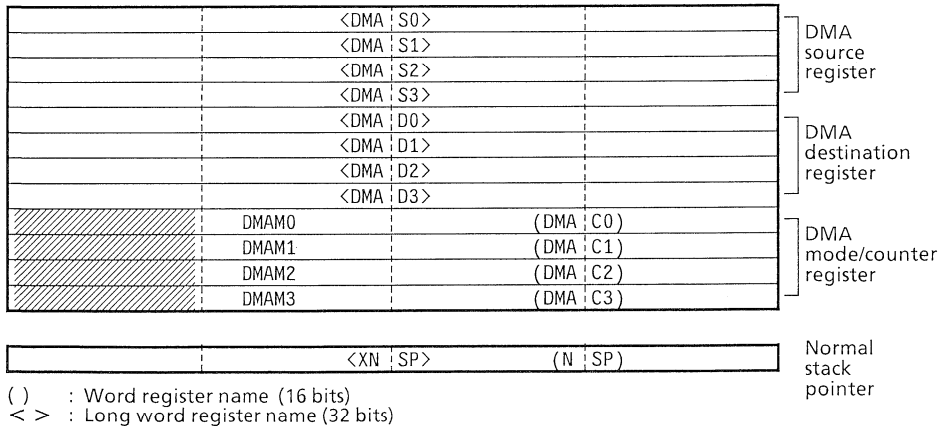
In maximum mode, the program counter consists of 32 bits. The size of the program area depends on the number of the address pins that the product has. With 24 address pins (A0 to A23), a maximum program area of 16M bytes can be accessed as a linear address space. In this case, the upper 8 bits of the program counter (bits 24 to 31) are ignored.

When the register format changes from minimum to maximum mode, the upper word of the program counter (bits 16 to 31) is extended so that the program counter becomes 32 bits long. This automaticaly writes "0" to the upper word of the program counter. So doing ensures program continuity. The program counter is initialized to 8000H by reset.

3.2.5 Control registers (CR)

The control registers consist of registers used to control high-speed micro DMA operation, a TLCS-900 series feature, and a normal stack pointer, which can be accessed in system mode. Control registers can be accessed using the (privileged) LDC instruction, which can only be used in system mode.

Control registers are illustrated below.



For high-speed micro DMA, refer to "Part 2 TLCS-900 LSI Devices".

3.3 Register Bank Switching

Register banks are classified into the following three types.

Current bank registers
Previous bank registers
Absolute bank registers

The current bank is indicated by the register file pointer, <RFP>, (status register bits 8 to 10). The registers in the current bank are used as general-purpose registers, as described in the previous section. By changing the contents of the <RFP>, another register bank becomes the current register bank.

The previous bank is indicated by the value obtained by subtracting 1 from the <RFP>. For example, if the current bank is bank 3, bank 2 is the previous bank. The names of registers in the previous bank are indicated with a dash (WA', BC', DE', HL'). The EX instruction (EX A,A') is used to switch between current and previous banks.

All bank registers, including the current and previous ones, have a numerical value (absolute bank number) to indicate the bank. With a register name which includes a numerical value such as RW0, RA0, etc., all bank registers can be used. These registers (that is, all registers) are called absolute bank registers.

The TLCS-900 series CPU is designed to perform optimally when the current bank registers are operated as the working registers. In other words, if the CPU uses other bank registers, its performance degrades somewhat. In order to obtain maximum CPU efficiency, the TLCS-900 series has a function which easily switches register banks.

The bank switching function provides the following advantages:

- Optimum CPU operating efficiency
- Reduced programming size (Object codes)
- Higher response speed and reduced programming size when used as a context switch for an interrupt service routine.

Bank switching is performed by the instructions listed below.

LDF imm : Sets the contents of the immediate value in <RFP>. imm: 0 to 7
INCF : Increments <RFP> by 1.
DECF : Decrements <RFP> by 1.

In minimum mode, the immediate values used by the LDF instruction are from 0 to 7, in maximum mode 0 to 3. If a carry or borrow occurs when the INCF or DECF instruction is executed, it is ignored. The value of the <RFP> rotates. For example, if the INCF instruction is executed with bank 7, the result is bank 0. If the DECF instruction is executed with bank 0, the result is bank 7. Note that careless execution of the INCF or DECF instruction may destroy the contents of the register bank.

- Example of Register Bank Usage

The TLCS-900 series registers are formatted in banks. Banks can be used for processing objectives or interrupt levels. Two examples are given below.

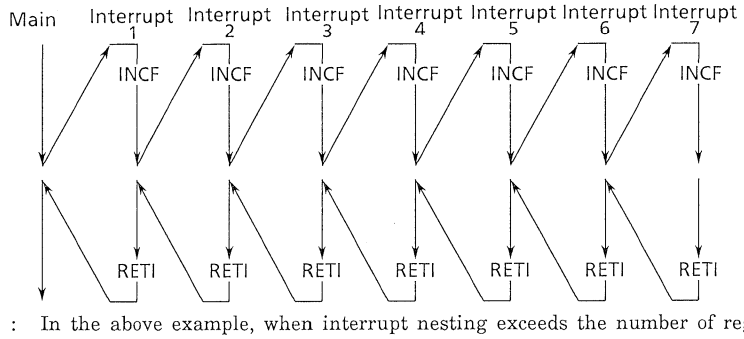
<Example 1> When assigning register banks to interrupt processing routines.

Register bank 0 = Used for the main program and interrupt processing other than that shown below.
Register bank 1 = Used for processing INT0 .
Register bank 2 = Used for processing timer 0.
Register bank 3 = Used for processing timer 1.
Register bank 4 = Used for processing A/D converter.
Register bank 5 = Used for processing serial I/O. (Data send)
Register bank 6 = Used for processing serial I/O. (Data receive)
Register bank 7 = Used for processing $\overline{\text{NMI}}$.

For example, if a timer 1 interrupt occurs during main program execution, processing jumps to a subroutine as follows. PUSH/POP processing for the register is unnecessary.

```
LDF 3     ; Sets register bank to 3. 0.25 μs (@16MHz)
 :
 :
RETI      ; Returns to previous status including <RFP>.
            1.5 μs (@ 16MHz)
```

<Example 2> When assigning register banks to their appropriate interrupt level nesting.



Note 1 : In the above example, when interrupt nesting exceeds the number of register banks (8), the <RFP> becomes 000 and the contents of register bank 0 are destroyed. However, interrupt levels are usually from 1 to 7, so in most cases nesting will not exceed 8 levels.Unless, that is, multiple NMIs occur. If there is any chance of multiple NMIs occurring, do not use the INCF instruction in the NMI processing routine.

Note 2 : The INCF instruction is used to execute <RFP>←<RFP> + 1.
0.25 μs (@16 MHz)

3.4 Accessing General-purpose Registers

The register access code is formatted in a varied code length on byte basis. The current bank registers can be accessed by the shortest code length. All general-purpose registers can be accessed by an instruction code which is 1 byte longer. General-purpose registers are as follows.

① General-purpose registers in current bank

(Minimum mode)

					W	(W	A)	A
					B	(B	C)	C
					D	(D	E)	E
					H	(H	L)	L

(Maximum mode)

QW	(Q	WA)	QA	<X	WA >	W	(W	A)	A
QB	(Q	BC)	QC	<X	BC >	B	(B	C)	C
QD	(Q	DE)	QE	<X	DE >	D	(D	E)	E
QH	(Q	HL)	QL	<X	HL >	H	(H	L)	L

( ) : Word register name (16 bits)
< > : Long word register name (32 bits)

② General-purpose registers in previous bank

(Minimum mode)

					W'	(W	A')	A'
					B'	(B	C')	C'
					D'	(D	E')	E'
					H'	(H	L')	L'

(Maximum mode)

QW'	(Q	WA')	QA'	<X	WA'>	W'	(W	A')	A'
QB'	(Q	BC')	QC'	<X	BC'>	B'	(B	C')	C'
QD'	(Q	DE')	QE'	<X	DE'>	D'	(D	E')	E'
QH'	(Q	HL')	QL'	<X	HL'>	H'	(H	L')	L'

③ 32-bit general-purpose registers

(Both minimum and maximum modes)

QIXH	(Q	IX)	QIXL	<X	IX>	IXH	(I	X)	IXL
QIYH	(Q	IY)	QIYL	<X	IY>	IYH	(I	Y)	IYL
QIZH	(Q	IZ)	QIZL	<X	IZ>	IZH	(I	Z)	IZL
QSPH	(Q	SP)	QSPL	<X	SP>	SPH	(S	P)	SPL

④ Absolute bank registers

(Minimum mode)

				RW0	(RWA 0)	RA0		Bank0
				RB0	(RBC 0)	RC0		
				RD0	(RDE 0)	RE0		
				RH0	(RHL 0)	RL0		
				RW1	(RWA 1)	RA1		Bank1
				RB1	(RBC 1)	RC1		
				RD1	(RDE 1)	RE1		
				RH1	(RHL 1)	RL1		
				RW2	(RWA 2)	RA2		Bank2
				RB2	(RBC 2)	RC2		
				RD2	(RDE 2)	RE2		
				RH2	(RHL 2)	RL2		
				RW3	(RWA 3)	RA3		Bank3
				RB3	(RBC 3)	RC3		
				RD3	(RDE 3)	RE3		
				RH3	(RHL 3)	RL3		
				RW4	(RWA 4)	RA4		Bank4
				RB4	(RBC 4)	RC4		
				RD4	(RDE 4)	RE4		
				RH4	(RHL 4)	RL4		
				RW5	(RWA 5)	RA5		Bank5
				RB5	(RBC 5)	RC5		
				RD5	(RDE 5)	RE5		
				RH5	(RHL 5)	RL5		
				RW6	(RWA 6)	RA6		Bank6
				RB6	(RBC 6)	RC6		
				RD6	(RDE 6)	RE6		
				RH6	(RHL 6)	RL6		
				RW7	(RWA 7)	RA7		Bank7
				RB7	(RBC 7)	RC7		
				RD7	(RDE 7)	RE7		
				RH7	(RHL 7)	RL7		

(Maximum mode)

QW0	(QWA 0)	QA0	<XWA 0>	RW0	(RWA 0)	RA0		Bank0
QB0	(QBC 0)	QC0	<XBC 0>	RB0	(RBC 0)	RC0		
QD0	(QDE 0)	QE0	<XDE 0>	RD0	(RDE 0)	RE0		
QH0	(QHL 0)	QL0	<XHL 0>	RH0	(RHL 0)	RL0		
QW1	(QWA 1)	QA1	<XWA 1>	RW1	(RWA 1)	RA1		Bank1
QB1	(QBC 1)	QC1	<XBC 1>	RB1	(RBC 1)	RC1		
QD1	(QDE 1)	QE1	<XDE 1>	RD1	(RDE 1)	RE1		
QH1	(QHL 1)	QL1	<XHL 1>	RH1	(RHL 1)	RL1		
QW2	(QWA 2)	QA2	<XWA 2>	RW2	(RWA 2)	RA2		Bank2
QB2	(QBC 2)	QC2	<XBC 2>	RB2	(RBC 2)	RC2		
QD2	(QDE 2)	QE2	<XDE 2>	RD2	(RDE 2)	RE2		
QH2	(QHL 2)	QL2	<XHL 2>	RH2	(RHL 2)	RL2		
QW3	(QWA 3)	QA3	<XWA 3>	RW3	(RWA 3)	RA3		Bank3
QB3	(QBC 3)	QC3	<XBC 3>	RB3	(RBC 3)	RC3		
QD3	(QDE 3)	QE3	<XDE 3>	RD3	(RDE 3)	RE3		
QH3	(QHL 3)	QL3	<XHL 3>	RH3	(RHL 3)	RL3		

( ) : Word register name (16 bits)
< > : Long word register name (32 bits)

4. ADDRESSING MODES

The TLCS-900 series has nine addressing modes. These are combined with most instructions to improve CPU processing capabilities.

TLCS-900 series addressing modes are listed below. They cover the entire TLCS-90 addressing modes.

No.	Addressing mode	Description
1.	Register	reg8 reg16 reg32
2.	Immediate	n8 n16 n32
3.	Register indirect	(reg)
4.	Register indirect pre-decrement	(– reg)
5.	Register indirect post-increment	(reg +)
6.	Index	(reg + d8) (reg + d16)
7.	Register index	(reg + reg8) (reg + reg16)
8.	Absolute	(n8) (n16) (n24)
9.	Relative	(PC + d8) (PC + d16)

reg 8 : All 8-bit registers such as W, A, B, C, D, E, H, L, etc.

reg 16 : All 16-bit registers such as WA, BC, DE, HL, IX, IY, IZ, SP, etc.

reg 32 : All 32-bit registers such as XWA, WBC, XDE, XHL, XIX, XIY, XIZ, XSP, etc.

reg : All 32-bit registers such as XWA, WBC, XDE, XHL, XIX, XIY, XIZ, XSP, etc. (Maximum mode)
All 16-bit bank registers such as WA, BC, DE, HL, etc. and XIX, XIY, XIZ, and XSP. (Minimum mode)

d8 : 8-bit displacement (–80H~+7FH)

d16 : 16-bit displacement (–8000H~+7FFFH)

n8 : 8-bit constant (00H~FFH)

n16 : 16-bit constant (0000H~FFFFH)

n32 : 32-bit constant (00000000H~FFFFFFFFH)

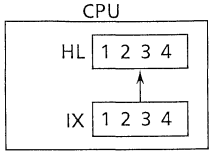
Note 1 : Relative addressing mode can only be used with the following instructions: LDAR, JR, JRL, DJNZ, and CALR

Note 2 : In minimum mode, register bank blocks (current bank registers and previous bank registers, and bank 0 to 7 registers) consist of 16 bits. When these 16-bit registers are used for addressing, the CPU extends bits 16 to 31 to 0000H for address calculations.

(1) Register Addressing Mode

In this mode, the operand is the specified register.

Example: LD HL,IX

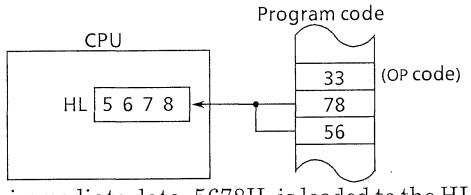


The IX register contents, 1234H, are loaded to the HL register.

(2) Immediate Addressing Mode

In this mode, the operand is in the instruction code.

Example: LD HL,5678H

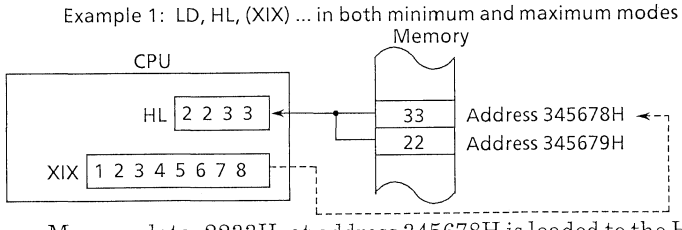


The immediate data, 5678H, is loaded to the HL register.

(3) Register Indirect Addressing Mode

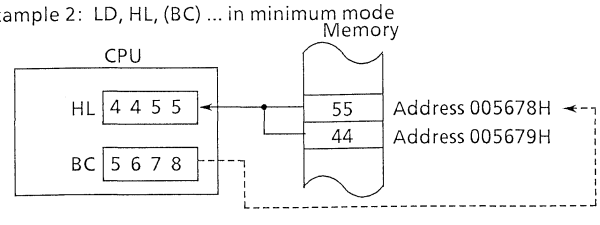
In this mode, the operand is the memory address specified by the contents of the register.

Example 1: LD, HL, (XIX) ... in both minimum and maximum modes



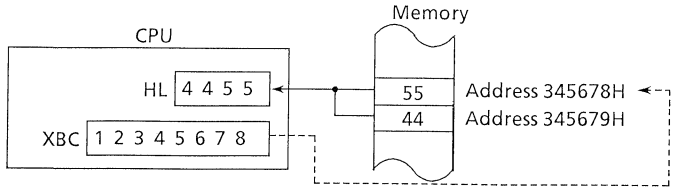
Memory data, 2233H, at address 345678H is loaded to the HL register.

Example 2: LD, HL, (BC) ... in minimum mode



In minimum mode, if a bank register (WA, BC, DE, or HL) is used for addressing, address bits 16 to 23 are set to 00H.

Example 3: LD HL,(XBC) ... in maximum mode

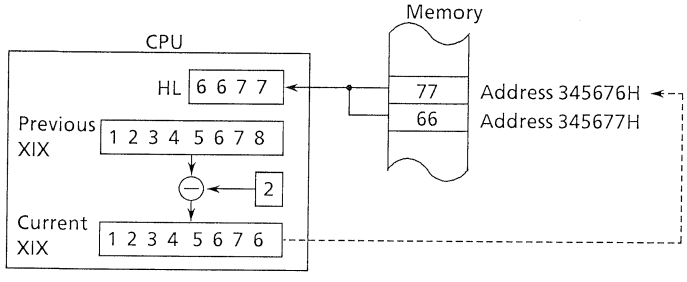


In maximum mode, if a bank register (XWA, XBC, XDE, or XHL) is used for addressing, the values of bits 0 to 23 are output to the address bus.

(4) Register Indirect Pre-decrement Addressing Mode

In this mode, the contents of the register is decremented by the pre-decrement values. In this case, the operand is the memory address specified by the decremented register.

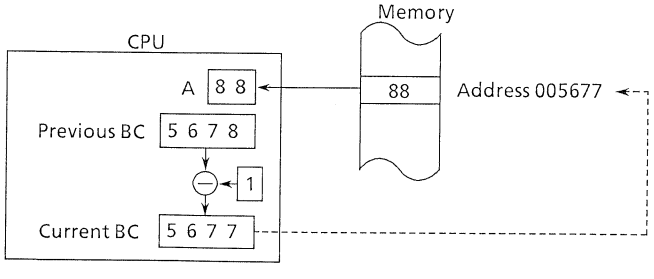
Example 1: LD HL, (– XIX) ... in both minimum and maximum modes



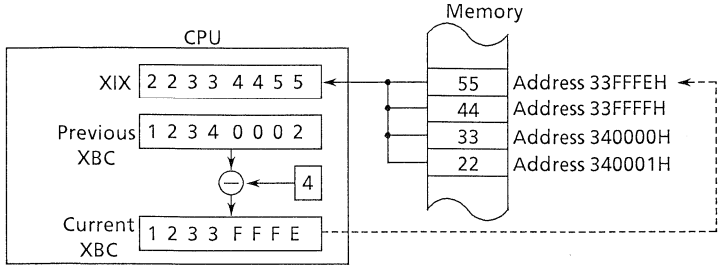
The pre-decrement values are as follows:

When the size of the operand is one byte (8 bits) : −1
When the size of the operand is one word (16 bits) : −2
When the size of the operand is one long word (32 bits) : −4

Example 2: LD A,(-BC) ... in minimum mode



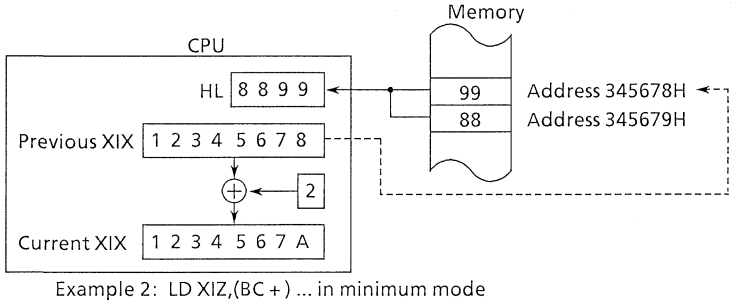
Example 3: LD XIX,(-XBC) ... in maximum mode



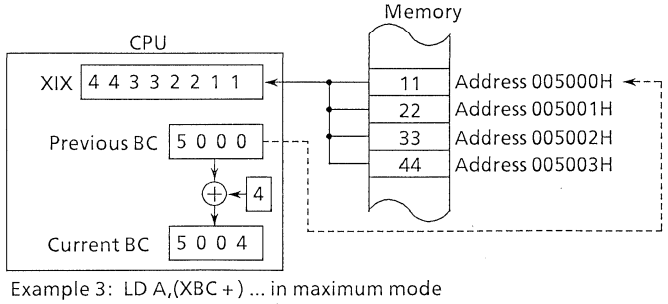
(5) Register Indirect Post-increment Addressing Mode

In this mode, the operand is the memory address specified by the contents of the register. After the operation, the contents of the register are incremented by the size of the operand.

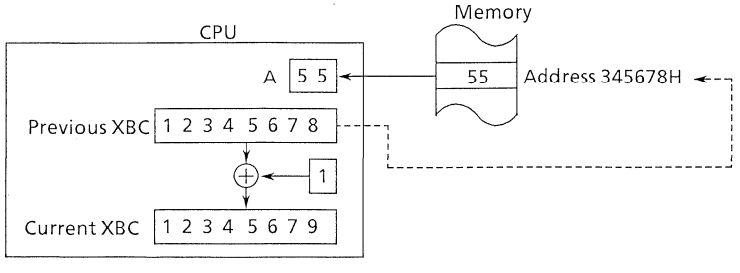
Example 1: LD HL,(XIX +) ... in both minimum and maximum modes



Example 2: LD XIZ,(BC +) ... in minimum mode



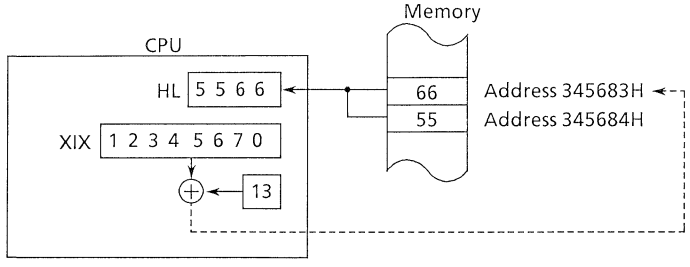
Example 3: LD A,(XBC +) ... in maximum mode



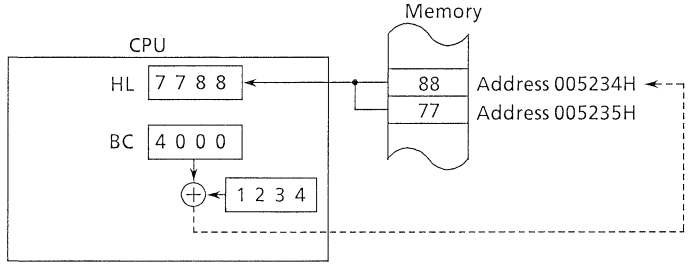
(6) Index Addressing Mode

In this mode, the operand is the memory address obtained by adding the contents of the specified register to the 8- or 16-bit displacement value in the instruction code.

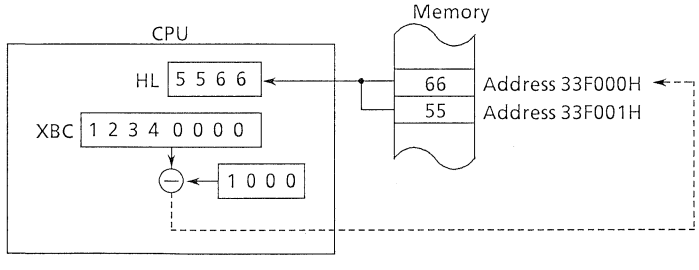
Example 1: LD HL,(XIX + 13H) ... in both minimum and maximum modes



Example 2: LD HL,(BC + 1234H) ... in minimum mode



Example 3: LD HL,(XBC-1000H) ... in maximum mode

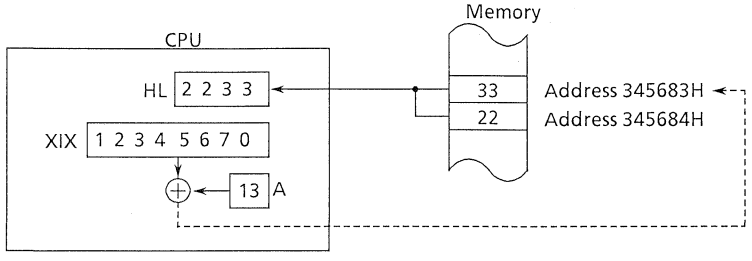


The displacement values range from −8000H to +7FFFH.

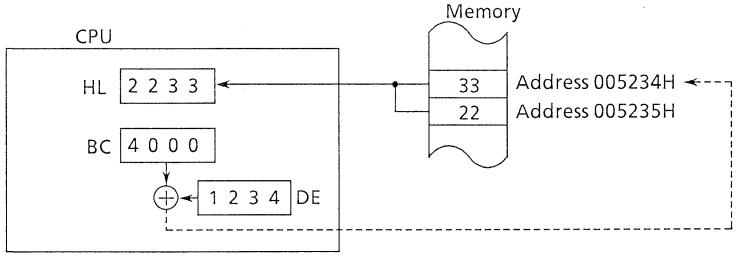
(7) Register Index Addressing Mode

In this mode, the operand is the memory address obtained by adding the contents of the register specified as the base to the register specified as the 8- or 16-bit displacement.

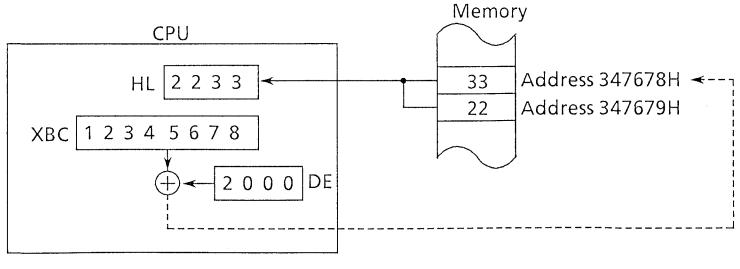
Example 1: LD HL,(XIX + A) ... in both minimum and maximum modes



Example 2: LD HL,(BC + DE) ... in minimum mode



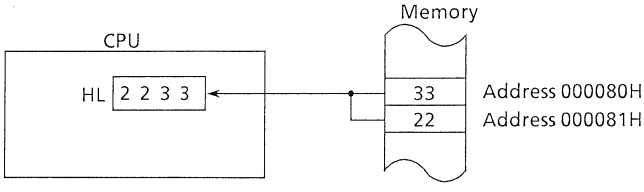
Example 3: LD HL,(XBC + DE) ... in maximum mode



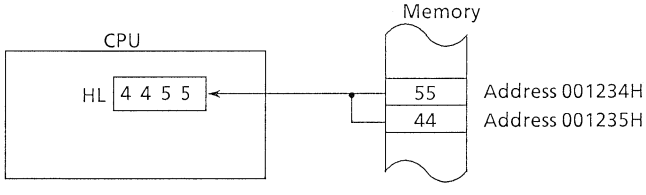
(8) Absolute Addressing Mode

In this mode, the operand is the memory address specified by 1 to 3 bytes in the instruction code. Addresses 000000H to 0000FFH can be specified by 1 byte. Addresses 000000H to 00FFFFH can be specified by 2 bytes. Addresses 000000H to FFFFFFH can be specified by 3 bytes.

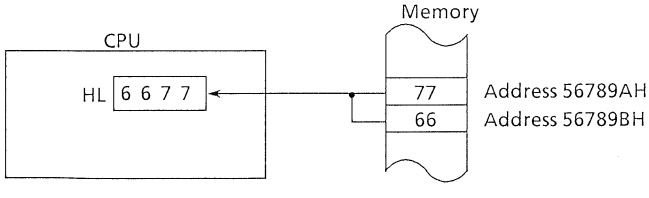
Example 1: LD HL,(80H)



Example 2: LD HL,(1234H)



Example 3: LD HL,(56789AH)



(9) Relative Addressing Mode

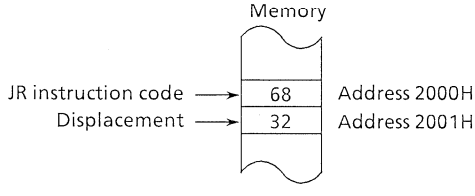
In this mode, the operand is the memory address obtained by adding the 8- or 16-bit displacement value to the address where the instruction code being executed is located.

In this mode, only the following five instructions can be used.

LDAR	R, $+4+d16	
JR	cc, $+2+d8	
JRL	cc, $+3+d16	
CALR	$+3+d16	
DJNZ	r, $+3+d8	($: start address of instruction code)

In calculating the displacement object code value, the adjustment value (+2 to +4) depends on the instruction type.

Example 1: JR 2034H



In the above example, the displacement object code value is:

2034H − (2000H + 2) = 32H.

5. INSTRUCTIONS

In addition to its various addressing modes, the TLCS-900 series also has a powerful instruction set. The basic instructions are classified into the following nine groups:

- Load instructions (8/16/32 bits)
- Exchange instructions (8/16 bits)
- Block transfer & Block search instructions (8/16 bits)
- Arithmetic operation instructions (8/16/32 bits)
- Logical operation instructions (8/16/32 bits)
- Bit operation instructions (1 bit)
- Special operations, CPU control instructions
- Rotate and Shift instructions (8/16/32 bits)
- Jump, Call, and Return instructions

Table 5 lists the basic instructions of the TLCS-900 series. For details of instructions, see Appendix A; for the instruction list, Appendix B; for the instruction code map, Appendix C; and for the differences between the TLCS-90 and TLCS-900 series, Appendix D.

Table 5 (1) TLCS-900 Series Basic Instructions

LD	dst, src	Load dst←src	
PUSH	src	Push src data to stack. SP←SP – size: (SP) ←src	
POP	dst	Pop data from stack to dst. dst←(SP) : SP←SP + size	
LDA	dst, src	Load address: set src effective address in dst.	
LDAR	dst, PC + dd	Load address relative: set program counter relative address value in dst. dst←PC + dd	
EX	dst1, dst2	Exchange dst1 and dst2 data.	
MIRR	dst	Mirror-invert dst bit pattern.	
LDI		Load increment	
LDIR		Load increment repeat	
LDD		Load decrement	
LDDR		Load decrement repeat	
CPI		Compare increment	
CPIR		Compare increment repeat	
CPD		Compare decrement	
CPDR		Compare decrement repeat	
ADD	dst, src	Add	dst←dst + src
ADC	dst, src	Add with carry	dst←dst + src + CY
SUB	dst, src	Subtract	dst←dst – src
SBC	dst, src	Subtract with carry	dst←dst – src – CY
CP	dst, src	Compare	dst – src
AND	dst, src	And	dst←dst AND src
OR	dst, src	Or	dst←dst OR src
XOR	dst, src	Exclusive-or	dst←dst XOR src
INC	imm, dst	Increment	dst←dst + imm
DEC	imm, dst	Decrement	dst←dst – imm
MUL	dst, src	Multiply unsigned	dst←dst (low) × src
MULS	dst, src	Multiply signed	dst←dst (low) × src
DIV	dst, src	Divide unsigned dst (low) ← dst ÷ src dst (high) ← remainder V flag set due to division by 0 or overflow.	
DIVS	dst, src	Divide signed dst (low) ← dst ÷ src dst (high) ← remainder: sign is same as that of dividend. V flag set due to division by 0 or overflow.	

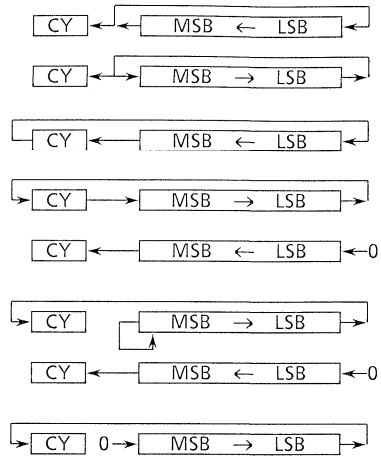
MULA	dst	Multiply and add $\underline{dst}_{32bit} \leftarrow \underline{dst}_{32bit} + \underline{(XDE)}_{16bit} \times \underline{(XHL-)}_{16bit}$
MINC1	num, dst	Modulo increment 1
MINC2	num, dst	Modulo increment 2
MINC4	num, dst	Modulo increment 4
MDEC1	num, dst	Modulo decrement 1
MDEC2	num, dst	Modulo decrement 2
MDEC4	num, dst	Modulo decrement 4
NEG	dst	Negate dst←0 – dst (Twos complement)
CPL	dst	Complement dst←not dst (Ones complement)
EXTZ	dst	Extend zero: set upper data of dst to 0.
EXTS	dst	Extend signed: copy the MSB of the lower data of dst to upper data.
DAA	dst	Decimal adjustment accumulator
PAA	dst	Pointer adjustment accumulator: when dst is odd, increment dst by 1 to make it even. if dst (0) = 1 then dst←dst + 1.
LDCF	bit, src	Load carry flag: copy src<bit> value to C flag.
STCF	bit, dst	Store carry flag: copy C flag value to dst<bit>.
ANDCF	bit, src	And carry flag: and src<bit> value and C flag, then load the result to C flag.
ORCF	bit, src	Or carry flag: or src<bit> and C flag, then load result to C flag.
XORCF	bit, src	Exclusive-or carry flag: exclusive-or src<bit> value and C flag, then load result to C flag.
RCF		Reset carry flag: reset C flag to 0.
SCF		Set carry flag: set C flag to 1.
CCF		Complement carry flag: invert C flag value.
ZCF		Zero flag to carry flag: copy inverted value of Z flag to C flag.
BIT	bit, src	Bit test: Z flag ← not src<bit>
RES	bit, dst	Bit reset
SET	bit, dst	Bit set
CHG	bit, dst	Bit change dst<bit>←not dst<bit>
TSET	bit, dst	Bit test and set: Z flag ← not dst<bit> dst<bit> ← 1

BS1F	A, dst	Bit search 1 forward: search dst for the first bit set to 1 starting from the LSB, then set the bit number in the A register.
BS1B	A,dst	Bit search 1 backward: search dst for the first bit set to 1 starting fom the MSB, then set the bit number in the A register.
NOP		No operation
NORMAL		Set CPU to normal mode.
MAX		Set CPU to maximum mode (32-bit bank register and PC).
EI	imm	Enable interrupt. IFF←imm
DI		Disable maskable interrupt. IFF←7
PUSH	SR	Push status registers.
POP	SR	Pop status registers.
SWI	imm	Software interrupt PUSH PC&SR : JP 8000H + 10H × imm
HALT		Halt CPU.
LDC	CTRL – REG, reg	Load control: copy the register contents to control register of CPU.
LDC	reg, CTRL – REG	Load control: copy the control register contents to register.
LDX	dst, src	Load extract. dst←src

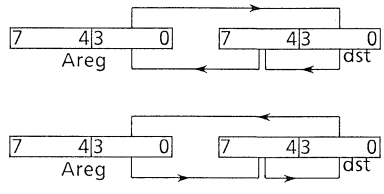
LINK	reg, dd	Link: generate stack frame.
		PUSH reg
		LD reg, XSP
		ADD XSP, dd

UNLK	reg	Unlink: delete stack frame. LD XSP, reg POP reg
LDF	imm	Load register file pointer: specify register bank. RFP←imm
INCF		Increment register file pointer: move to new register bank. RFP←RFP + 1
DECF		Decrement register file pointer: return to previous register bank. RFP←RFP – 1
SCC	cc, dst	Set dst with condition codes. if cc then dst ←1 else dst ←0.

RLC	num, dst	Rotate left without carry
RRC	num, dst	Rotate right without carry
RL	num, dst	Rotate left
RR	num, dst	Rotate right
SLA	num, dst	Shift left arithmetic
SRA	num, dst	Shift right arithmetic
SLL	num, dst	Shift left logical
SRL	num, dst	Shift right logical



RLD	dst	Rotate left digit
RRD	dst	Rotate right digit



JR	cc, PC + d	Jump relative (8-bit displacement) if cc then PC←PC + d.
JRL	cc, PC + dd	Jump relative long (16-bit displacement) if cc then PC←PC + dd.
JP	cc, dst	Jump if cc then PC←dst.
CALR	RC + dd	Relative call (16-bit displacement) PUSH PC: PC←PC + dd.
CALL	cc, dst	Call relative if cc then PUSH PC: PC←dst.
DJNZ	dst, PC + d	Decrement and jump if non-zero dst←dst − 1 if dst≠ 0 then PC←PC + d.
RET	cc	Return if cc then POP PC.
RETD	dd	Return and deallocate RET XSP←XSP + dd
RETI		Return from interrupt POP SR&PC

Table 5 (2) TLCS-900 Series Instruction List

BWL	LD	reg, reg	BWL	INC	imm3, reg	---	NOP	
BWL	LD	reg, imm		DEC	imm3, mem.B/W	---	*NORMAL	
BWL	LD	reg, mem				---	*MAX	
BWL	LD	mem, reg				---	EI	[imm3]
BW-	LD	mem, imm				---	DI	
BW-	LD	(nn), mem	BW-	MUL	reg, reg	-W-	*PUSH	SR
BW-	LD	mem, (nn)		*MULS	reg, imm	-W-	*POP	SR
				DIV	reg, mem	---	SWI	[imm3]
				*DIVS		---	HALT	
BWL	PUSH	reg/F				BWL	*LDC	CTRL – R, reg
BW-	PUSH	imm	-W-	*MULA	reg	BWL	*LDC	reg, CTRL – R
BW-	PUSH	mem				B--	*LDX	(n), n
			-W-	*MINC1	imm, reg			
BWL	POP	reg/F	-W-	*MINC2	imm, reg	--L	*LINK	reg, dd
BW-	POP	mem	-W-	*MINC4	imm, reg	--L	*UNLK	reg
			-W-	*MDEC1	imm, reg	---	*LDF	imm3
			-W-	*MDEC2	imm, reg	---	*INCF	
-WL	LDA	reg, mem	-W-	*MDEC4	imm, reg	---	*DECF	
-WL	LDAR	reg, PC + dd				BW-	*SCC	cc, reg
			BW-	NEG	reg			
			BW-	CPL	reg	BWL	RLC	imm, reg
			-WL	*EXTZ	reg		RRC	A, reg
B--	EX	F, F'	-WL	*EXTS	reg		RL	mem. B/W
BW-	EX	reg, reg	B--	DAA	reg		RR	
BW-	EX	mem, reg	-WL	*PAA	reg		SLA	
							SRA	
							SLL	
-W-	*MIRR	reg	BW-	*LDCF	imm, reg		SRL	
				*STCF	A, reg			
				*ANDCF	imm, mem.B	B--	RLD	[A,] mem
				*ORCF	A, mem.B	B--	RRD	[A,] mem
BW-	LDI			*XORCF				
BW-	LDIR							
BW-	LDD		---	RCF		---	JR	[cc,] PC + d
BW-	LDDR		---	SCF		---	JRL	[cc,] PC + dd
			---	CCF		---	JP	[cc,] mem
			---	*ZCF		---	CALR	PC + dd
BW-	CPI					---	CALL	[cc,] mem
BW-	CPIR		BW-	BIT	imm, reg			
BW-	CPD			RES	imm, mem.B	BW-	DJNZ	[reg], PC + d
BW-	CPDR			SET				
				*CHG		---	RET	[cc]
				TSET		---	*RETD	dd
						---	RETI	
BWL	ADD	reg, reg						
	ADC	reg, imm	-W-	*BS1F	A, reg			
	SUB	reg, mem		*BS1B				
	SBC	mem, reg						
	CP	mem, imm.B/W						
	AND							
	OR							
	XOR							

B = Byte (8bit), W = Word (16bit), L = Long-Word (32bit).

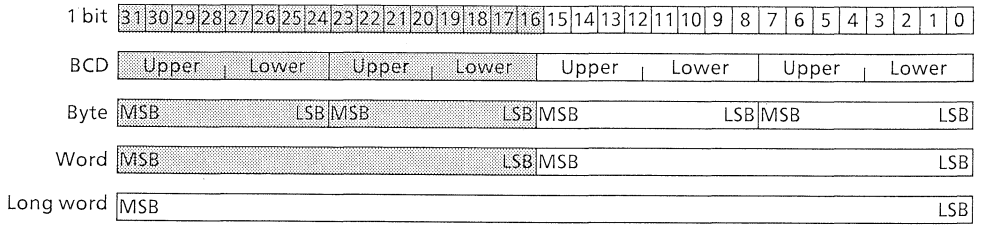
* : Indicates instruction added to the TLCS-90 series.
: Indicates privileged instruction.
[ ] : Indicates can be omitted.

6. DATA FORMATS

The TLCS-900 series can handle 1/4/8/16/32-bit data.

(1) Register Data Format

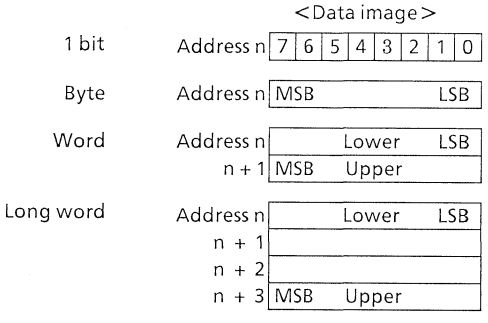
<Data image>



Note 1 : To access the shaded parts, the instruction code is one byte longer than when accessing the non-shaded parts.

(2) Memory Data Format

<Data image>



Note 2 : There are no restrictions on the location of word or long word data in memory. They can be located from even or odd numbered address.

Note 3 : When the PUSH instruction is used to save data to the stack area, the stack pointer is decremented, then the data is saved.

Example: PUSH HL; XSP←XSP−2
(XSP) ←L
(XSP+1)←H

This is the same in register indirect pre-decrement mode. The order is reversed in the TLCS-90 series: data is saved first, then the stack pointer is decremented.

Example: PUSH HL; (XSP−1)←H
(XSP−2)←L
XSP←XSP−2

(3) Dynamic Bus Sizing

The TLCS-900 series can switch between 8- and 16-bit data buses dynamically during each bus cycle. This is called dynamic bus sizing. The function enables external memory extension using both 8- and 16-bit data bus memories. Products with a built-in chip select/wait controller can control external data bus size for each address area.

Table 6 (1) Dynamic Bus Sizing

Operand data size	Operand start address	Data size at memory side	CPU address	CPU data	
				D15 – D8	D7 – D0
8 bits	2n + 0 (even)	8 bits	2n + 0	xxxxx	b7 – b0
		16 bits	2n + 0	xxxxx	b7 – b0
	2n + 1 (odd)	8 bits	2n + 1	xxxxx	b7 – b0
		16 bits	2n + 1	b7 – b0	xxxxx
16 bits	2n + 0 (even)	8 bits	2n + 0	xxxxx	b7 – b0
			2n + 1	xxxxx	b15 – b8
		16 bits	2n + 0	b15 – b8	b7 – b0
	2n + 1 (odd)	8 bits	2n + 1	xxxxx	b7 – b0
			2n + 2	xxxxx	b15 – b8
		16 bits	2n + 1	b7 – b0	xxxxx
			2n + 2	xxxxx	b15 – b8
32 bits	2n + 0 (even)	8 bits	2n + 0	xxxxx	b7 – b0
			2n + 1	xxxxx	b15 – b8
			2n + 2	xxxxx	b23 – b16
			2n + 3	xxxxx	b31 – b24
		16 bits	2n + 0	b15 – b8	b7 – b0
			2n + 2	b31 – b24	b23 – b16
	2n + 1 (odd)	8 bits	2n + 1	xxxxx	b7 – b0
			2n + 2	xxxxx	b15 – b8
			2n + 3	xxxxx	b23 – b16
			2n + 4	xxxxx	b31 – b24
		16 bits	2n + 1	b7 – b0	xxxxx
			2n + 2	b23 – b16	b15 – b8
			2n + 4	xxxxx	b31 – b24

xxxxx : During read, indicates the data input to the bus are ignored. During write, indicates the bus is at high impedance and the write strobe signal is non-active.

(4) Internal Data Bus Format

With the TLCS-900 series, the CPU and the internal memory (built-in ROM or RAM) are connected via a 16-bit internal data bus. The internal memory operates with 0 wait. The CPU and the built-in I/Os are connected using an 8-bit internal data bus. This is

because the built-in I/O access speed has little influence on the overall system operation speed.

Overall system operation speed depends largely on the speed of program memory access. The built-in I/O operates in sync with the signal phase of the CLK pin. It is synchronized so that the CLK rises (⎍) in the middle of the bus cycle. (Figure 7 (1) shows signal phases.) If the CLK is 1 when the ALE signal rises, 1 wait is inserted automatically for synchronization.

Figure 6 (1) shows an example of external memory connection with the TMP96C141 using the dynamic bus sizing function. In this example, ROM is connected with a 16-bit data bus, RAM, with an 8-bit bus. For details of connections, see a chip select/wait controller section of each product.

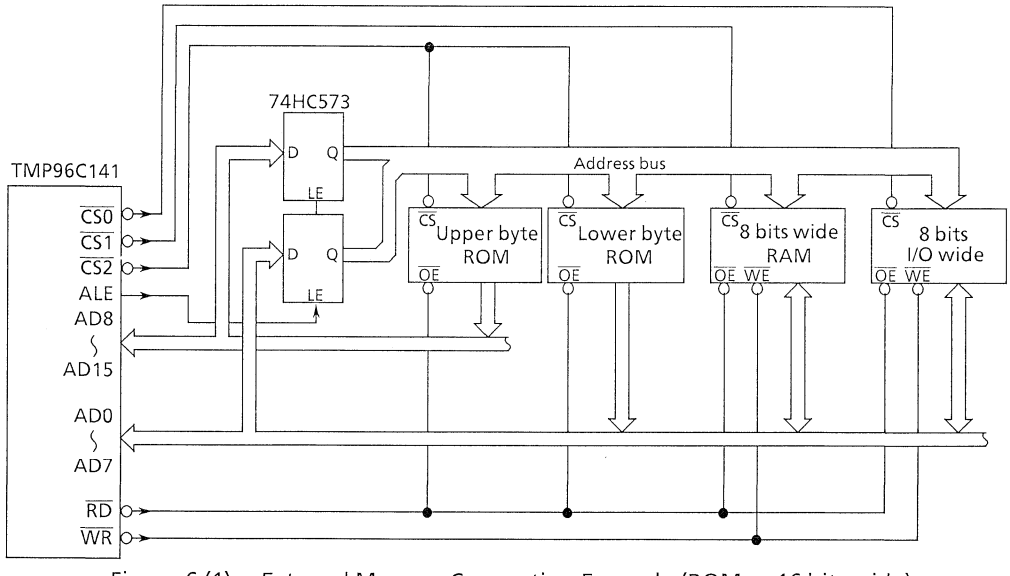


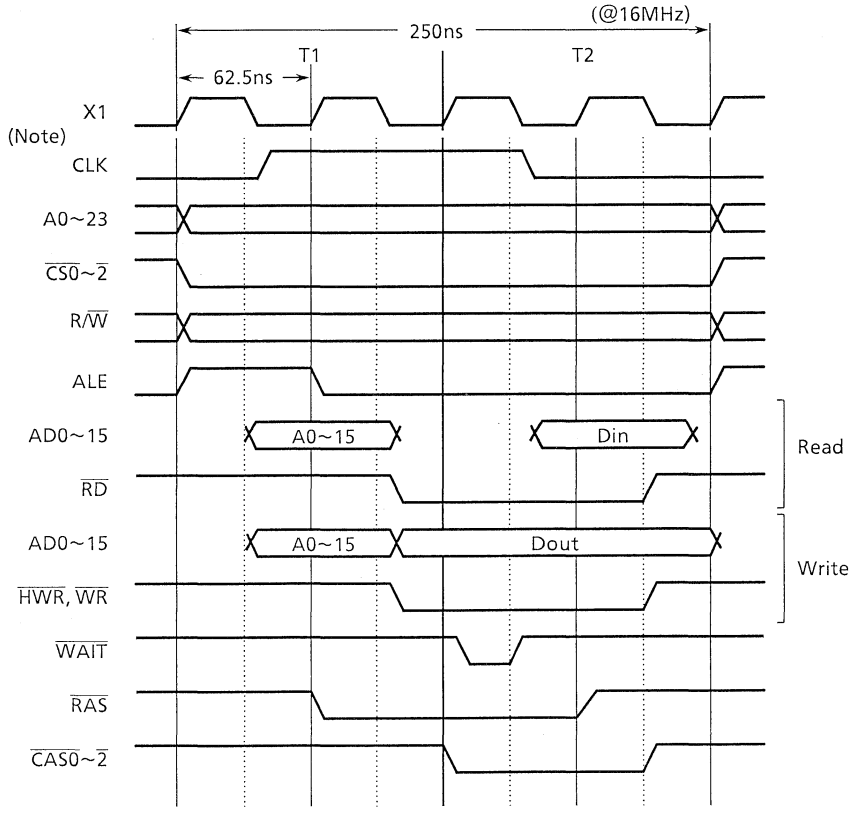
Figure 6 (1) External Memory Connection Example (ROM = 16 bits wide), RAM & I/O = 8 bits wide)

7. BASIC TIMINGS

The TLCS-900 series runs the following basic timings.

- Read cycle
- Write cycle
- Dummy cycle
- Interrupt receive timing
- Reset

Figures 7 (1) to (8) show the basic timings.



Note : CLK outputs are not always the same as the above phases.

Figure 7 (1) Read/Write Cycle

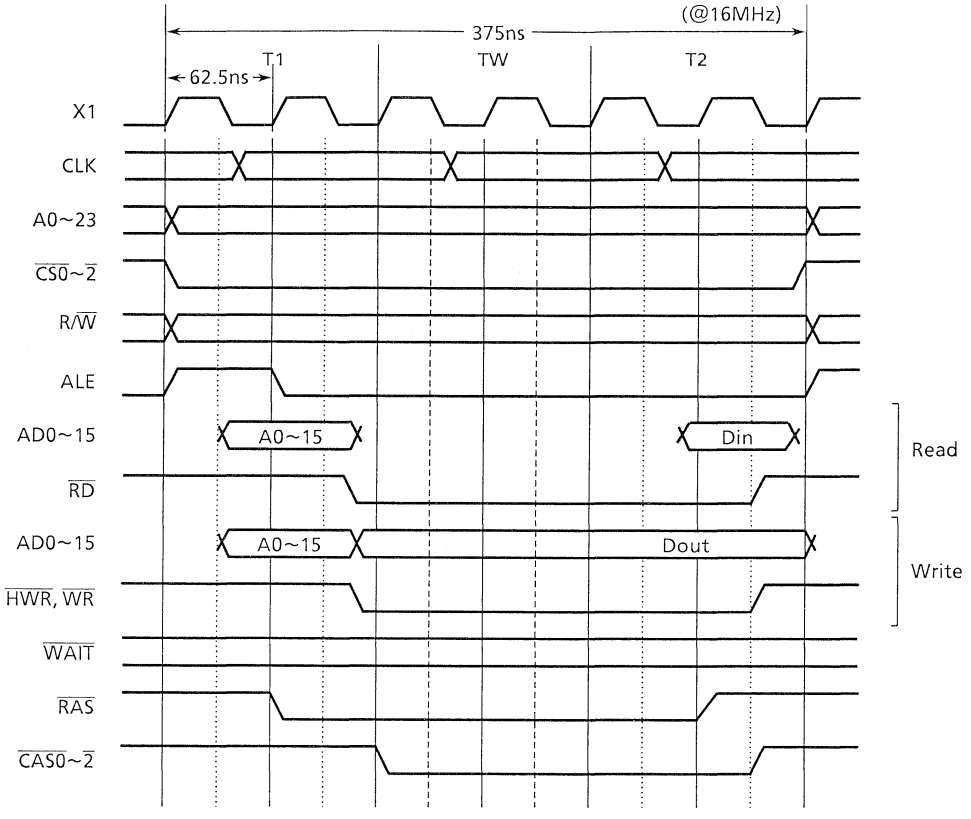


Figure 7 (2) 1WAIT Read/Write Cycle

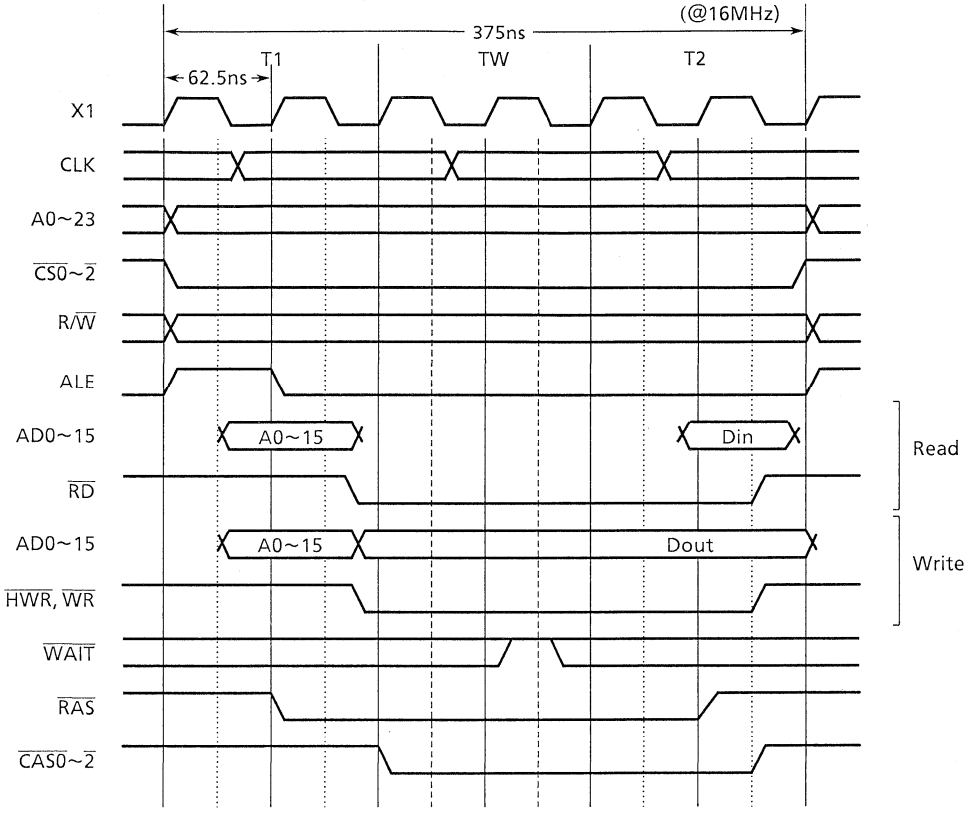


Figure 7 (3) 1WAIT + n Read/Write Cycle (n = 0)

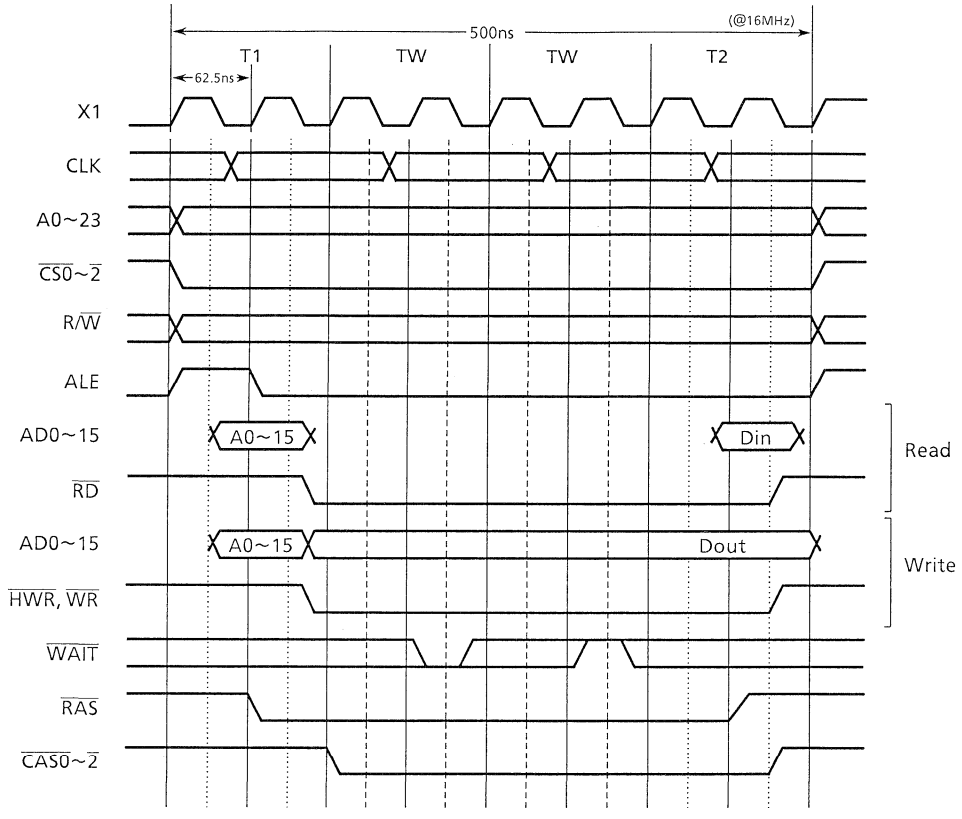


Figure 7 (4) 1WAIT + n Read/Write Cycle (n = 1)

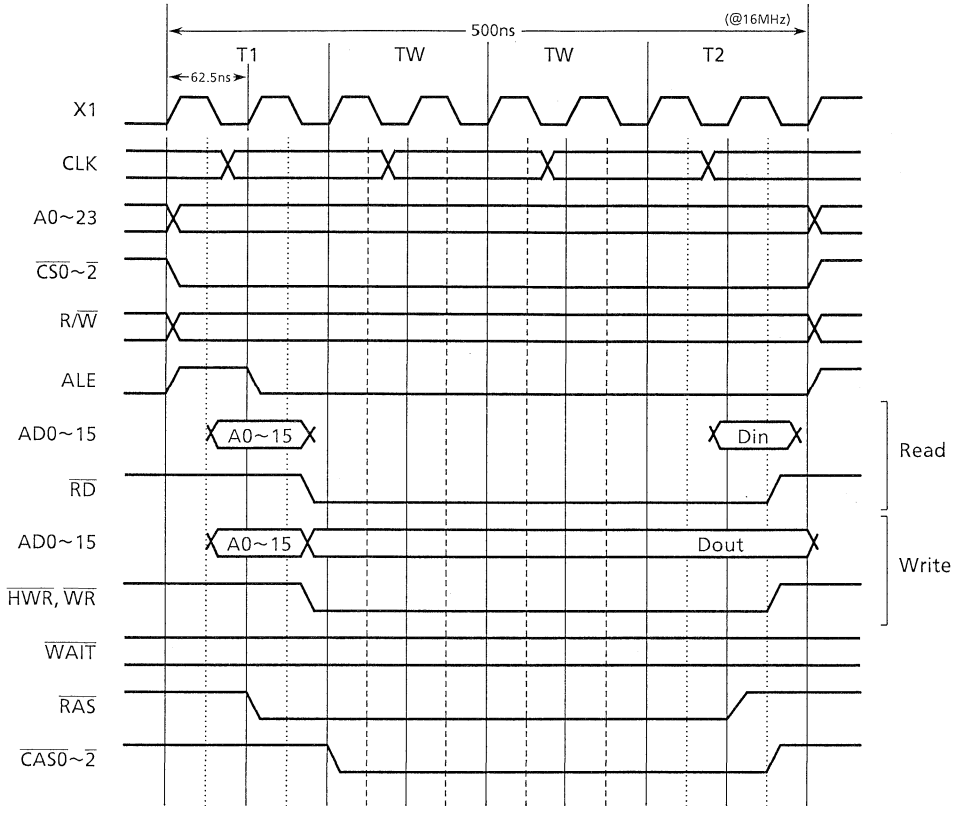


Figure 7 (5) 2 WAIT Read/Write Cycle

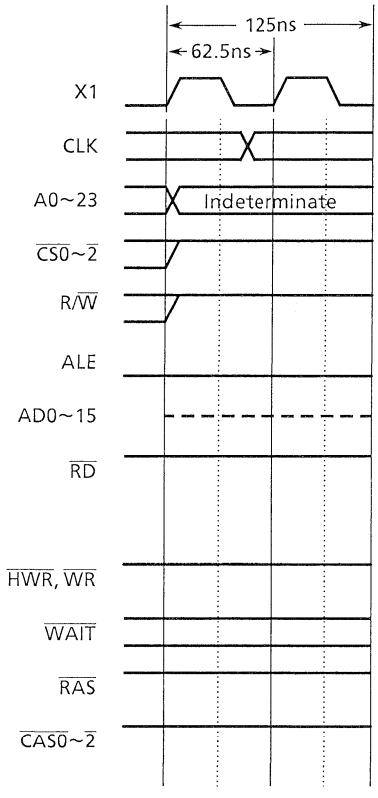
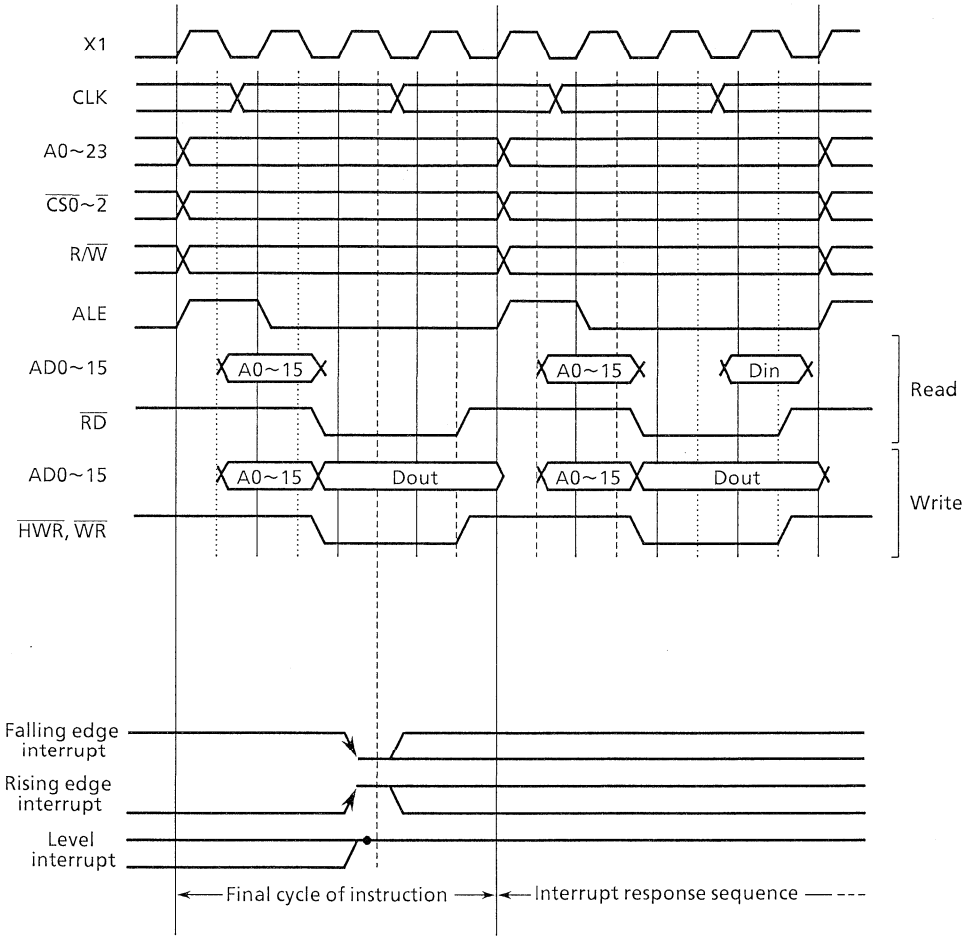


Figure 7 (6) 1 State Dummy Cycle



Note : This timing chart is a theoretical example. In practice, due to the operation of the bus interface unit in the CPU, external bus and internal interrupt receive timings do not correspond one to one.

Figure 7 (7) Interrupt Receive Timing

Figure 7 (8) Reset Timings (external ROM operation: TMP96C141)

Appendix A: Details of Instructions

■ Instruction List

① Load

LD	PUSH	POP	LDA	LDAR

② Exchange

EX	MIRR

③ Load Increment/Decrement & Compare Increment/Decrement

LDI	LDIR	LDD	LDDR	CPI	CPIR	CPD	CPDR

④ Arithmetic operations

ADD	ADC	SUB	SBC	CP	INC	DEC	NEG
EXTZ	EXTS	DAA	PAA	MUL	MULS	DIV	DIVS
MULA	MINC	MDEC					

⑤ Logical operations

AND	OR	XOR	CPL

⑥ Bit operations

LDCF	STCF	ANDCF	ORCF	XORCF	RCF	SCF	CCF
ZCF	BIT	RES	SET	CHG	TSET	BS1	

⑦ Special operations and CPU control

NOP	NORMAL	MAX	EI	DI	PUSH_SR	POP_SR	SWI
HALT	LDC	LDX	LINK	UNLK	LDF	INCF	DECF
SCC							

⑧ Rotate and shift

RLC	RRC	RL	RR	SLA	SRA	SLL	SRL
RLD	RRD						

⑨ Jump, call, and return

JP	JR	JRL	CALL	CALR	DJNZ	RET	RETD
RETI							

■ Explanations of symbols used in this document

dst	Destination: destination of data transfer or operation result load.
src	Source: source of data transfer or operation data read.
num	Number: numerical value.
condition	Condition: based on flag status.
R	Eight general-purpose registers including 8/16/32-bit current bank registers. 8-bit registers : W, A, B, C, D, E, H, L 16-bit registers : WA, BC, DE, HL, IX, IY, IZ, SP 32-bit registers : XWA, XBC, XDE, XHL, XIX, XIY, XIZ, XSP
r	8/16/32-bit general-purpose registers
r16	16-bit general-purpose registers
r32	32-bit general-purpose registers
cr	All 8/16/32-bit CPU control registers
A	A register (8 bits)
F	Flag registers (8 bits)
F'	Inverse flag registers (8 bits)
SR	Status registers (16 bits)
PC	Program counter (in minimum mode, 16 bits; in maximum mode, 32 bits)
(mem)	8/16/32-bit memory data
mem	Effective address value
<W>	When the operand size is a word, W must be specified.
[ ]	Operands enclosed in square brackets can be omitted.
#	8/16/32-bit immediate data.
#3	3-bit immediate data : 0 to 7 or 1 to 8 ... for abbreviated codes.
#4	4-bit immediate data : 0 to 15 or 1 to 16
d8	8-bit displacement : −80H~ +7FH
d16	16-bit displacement : −8000H~ +7FFFH
cc	Condition code
CY	Carry flag
Z	Zero flag
(#8)	Direct addressing: (00H) to (0FFH) ... 256-byte area
(#16)	64K-byte area addressing: (0000H) to (0FFFFH)
(−r32)	Pre-decrement addressing
(r32+)	Post-increment addressing
$	Start address of instruction

■ Explanations of symbols in object codes

z
zz
zzz
s

Operand size specify code

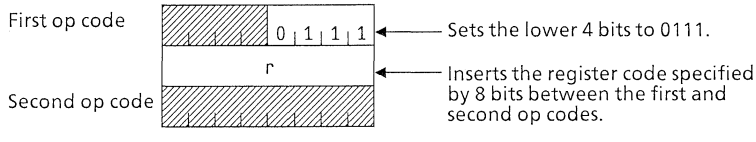
	Byte	Word	Long word
z	0	1	—
zz	00	01	10
zzz	010	011	100
s	—	0	1

R
r

Register specify code

Code	Byte	Word	Long word
000	W	WA	XWA
001	A	BC	XBC
010	B	DE	XDE
011	C	HL	XHL
100	D	IX	XIX
101	E	IY	XIY
110	H	IZ	XIZ
111	L	SP	XSP

Note: In addition to the above, all registers can be specified by "r" using extension codes. (In this case, the number of execution states increases by 1.) The format is shown below.



The code value in "r" must be:

Multiple of 2, if accessed as a word register.

Multiple of 4 ,if accessed as a long word.

For registers specified by 8 bits, see Register Maps shown on pages 48 and 49.

mem

Memory addressing mode specify code

(XWA)	=	-0--0000				
(XBC)	=	-0--0001				
(XDE)	=	-0--0010				
(XHL)	=	-0--0011				
(XIX)	=	-0--0100				
(XIY)	=	-0--0101				
(XIZ)	=	-0--0110				
(XSP)	=	-0--0111				
(XWA+d8)	=	-0--1000	d<7:0>			;+2 states
(XBC+d8)	=	-0--1001	d<7:0>			;+2 states
(XDE+d8)	=	-0--1010	d<7:0>			;+2 states
(XHL+d8)	=	-0--1011	d<7:0>			;+2 states
(XIX+d8)	=	-0--1100	d<7:0>			;+2 states
(XIY+d8)	=	-0--1101	d<7:0>			;+2 states
(XIZ+d8)	=	-0--1110	d<7:0>			;+2 states
(XSP+d8)	=	-0--1111	d<7:0>			;+2 states
(#8)	=	-1--0000	#<7:0>			;+2 states
(#16)	=	-1--0001	#<7:0>	#<15:8>		;+2 states
(#24)	=	-1--0010	#<7:0>	#<15:8>	#<23:16>	;+3 states
(r32)	=	-1--0011	r32'	00		;+5 states
(r32+d16)	=	-1--0011	r32'	01	d<7:0> d<15:8>	;+5 states
(r32+r8)	=	-1--0011	000000	11	r32 r8	;+8 states
(r32+r16)	=	-1--0011	000001	11	r32 r16	;+8 states
(-r32)	=	-1--0100	r32'	zz		;+3 states
(r32+)	=	-1--0101	r32'	zz		;+3 states

<7:0>= Indicates the data bit range. This example means 8-bit data from bit 0 to bit 7.

r32: 32-bit register
r16: Signed 16-bit register
r8: Signed 8-bit register

zz= Code used to specify the value of increments or decrements.

00: ±1
01: ±2
10: ±4
11: (Not defined)

r32' = Upper 6 bits of register code

cc

Condition codes

Code	Symbol	Description	Conditional expression
0000	F	always False	-
1000	(none)	always True	-
0110	Z	Zero	Z=1
1110	NZ	Not Zero	Z=0
0111	C	Carry	C=1
1111	NC	Not Carry	C=0
1101	PL or P	PLus	S=0
0101	MI or M	MInus	S=1
1110	NE	Not Equal	Z=0
0110	EQ	EQual	Z=1
0100	OV	OVerflow	P/V=1
1100	NOV	No OVerflow	P/V=0
0100	PE	Parity is Even	P/V=1
1100	PO	Parity is Odd	P/V=0
1001	GE	Greater than or Equal (signed)	(S xor P/V) =0
0001	LT	Less Than (signed)	(S xor P/V) =1
1010	GT	Greater Than (signed)	[Z or (S xor P/V)]=0
0010	LE	Less than or Equal (signed)	[Z or (S xor P/V)]=1
1111	UGE	Unsigned Greater than or Equal	C=0
0111	ULT	Unsigned Less Than	C=1
1011	UGT	Unsigned Greater Than	(C or Z) =0
0011	ULE	Unsigned Less than or Equal	(C or Z) =1

■ Flag changes

0	Reset to "0".
1	Set to "1".
-	No change.
*	"0" or "1" depending on the result of the calculation.
X	Indeterminate value.
P	Parity result is set.
V	Overflow result is set.

■ Register map "r" (minimum mode)

	+3	+2	+1	+0	
00H			RW0 (RWA 0)	RA0	Bank 0
04H			RB0 (RBC 0)	RC0	
08H			RD0 (RDE 0)	RE0	
0CH			RH0 (RHL 0)	RL0	
10H			RW1 (RWA 1)	RA1	Bank 1
14H			RB1 (RBC 1)	RC1	
18H			RD1 (RDE 1)	RE1	
1CH			RH1 (RHL 1)	RL1	
20H			RW2 (RWA 2)	RA2	Bank 2
24H			RB2 (RBC 2)	RC2	
28H			RD2 (RDE 2)	RE2	
2CH			RH2 (RHL 2)	RL2	
30H			RW3 (RWA 3)	RA3	Bank 3
34H			RB3 (RBC 3)	RC3	
38H			RD3 (RDE 3)	RE3	
3CH			RH3 (RHL 3)	RL3	
40H			RW4 (RWA 4)	RA4	Bank 4
44H			RB4 (RBC 4)	RC4	
48H			RD4 (RDE 4)	RE4	
4CH			RH4 (RHL 4)	RL4	
50H			RW5 (RWA 5)	RA5	Bank 5
54H			RB5 (RBC 5)	RC5	
58H			RD5 (RDE 5)	RE5	
5CH			RH5 (RHL 5)	RL5	
60H			RW6 (RWA 6)	RA6	Bank 6
64H			RB6 (RBC 6)	RC6	
68H			RD6 (RDE 6)	RE6	
6CH			RH6 (RHL 6)	RL6	
70H			RW7 (RWA 7)	RA7	Bank 7
74H			RB7 (RBC 7)	RC7	
78H			RD7 (RDE 7)	RE7	
7CH			RH7 (RHL 7)	RL7	
D0H			W' (W A')	A'	Previous bank
D4H			B' (B C')	C'	
D8H			D' (D E')	E'	
DCH			H' (H L')	L'	
E0H			W (W A)	A	Current bank
E4H			B (B C)	C	
E8H			D (D E)	E	
ECH			H (H L)	L	
F0H	QIXH (Q IX)	QIXL	<X IX>	IXH (I X)	IXL
F4H	QIYH (Q IY)	QIYL	<X IY>	IYH (I Y)	IYL
F8H	QIZH (Q IZ)	QIZL	<X IZ>	IZH (I Z)	IZL
FCH	QSPH (Q SP)	QSPL	<X SP>	SPH (S P)	SPL

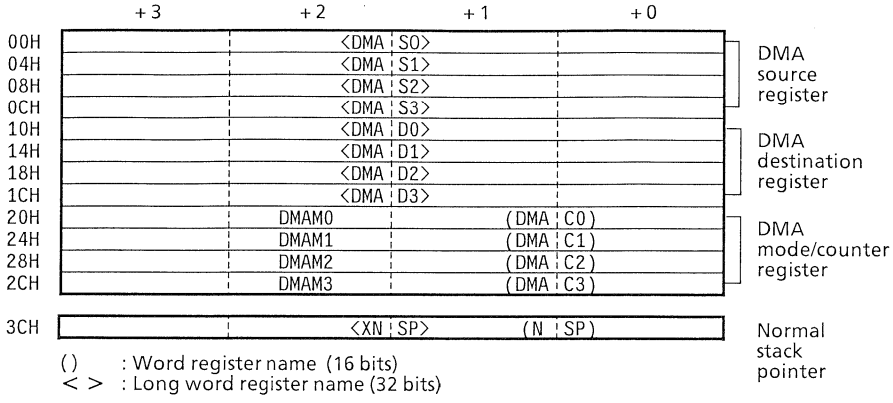
( ) : Word register name (16 bits)
< > : Long word register name (32 bits)

■ Register map "r" (maximum mode)

	+3	+2	+1	+0	
00H	QW0 (QWA 0)	QA0 <XWA 0>	RW0 (RWA 0)	RA0	Bank 0
04H	QB0 (QBC 0)	QC0 <XBC 0>	RB0 (RBC 0)	RC0	
08H	QD0 (QDE 0)	QE0 <XDE 0>	RD0 (RDE 0)	RE0	
0CH	QH0 (QHL 0)	QL0 <XHL 0>	RH0 (RHL 0)	RL0	
10H	QW1 (QWA 1)	QA1 <XWA 1>	RW1 (RWA 1)	RA1	Bank 1
14H	QB1 (QBC 1)	QC1 <XBC 1>	RB1 (RBC 1)	RC1	
18H	QD1 (QDE 1)	QE1 <XDE 1>	RD1 (RDE 1)	RE1	
1CH	QH1 (QHL 1)	QL1 <XHL 1>	RH1 (RHL 1)	RL1	
20H	QW2 (QWA 2)	QA2 <XWA 2>	RW2 (RWA 2)	RA2	Bank 2
24H	QB2 (QBC 2)	QC2 <XBC 2>	RB2 (RBC 2)	RC2	
28H	QD2 (QDE 2)	QE2 <XDE 2>	RD2 (RDE 2)	RE2	
2CH	QH2 (QHL 2)	QL2 <XHL 2>	RH2 (RHL 2)	RL2	
30H	QW3 (QWA 3)	QA3 <XWA 3>	RW3 (RWA 3)	RA3	Bank 3
34H	QB3 (QBC 3)	QC3 <XBC 3>	RB3 (RBC 3)	RC3	
38H	QD3 (QDE 3)	QE3 <XDE 3>	RD3 (RDE 3)	RE3	
3CH	QH3 (QHL 3)	QL3 <XHL 3>	RH3 (RHL 3)	RL3	
40H					
44H					
48H					
4CH					
50H					
54H					
58H					
5CH					
60H					
64H					
68H					
6CH					
70H					
74H					
78H					
7CH					
D0H	QW' (Q WA')	QA' <X WA'>	W' (W A')	A'	Previous bank
D4H	QB' (Q BC')	QC' <X BC'>	B' (B C')	C'	
D8H	QD' (Q DE')	QE' <X DE'>	D' (D E')	E'	
DCH	QH' (Q HL')	QL' <X HL'>	H' (H L')	L'	
E0H	QW (Q WA)	QA <X WA >	W (W A)	A	Current bank
E4H	QB (Q BC)	QC <X BC >	B (B C)	C	
E8H	QD (Q DE)	QE <X DE >	D (D E)	E	
ECH	QH (Q HL)	QL <X HL >	H (H L)	L	
F0H	QIXH (Q IX)	QIXL <X IX>	IXH (I X)	IXL	
F4H	QIYH (Q IY)	QIYL <X IY>	IYH (I Y)	IYL	
F8H	QIZH (Q IZ)	QIZL <X IZ>	IZH (I Z)	IZL	
FCH	QSPH (Q SP)	QSPL <X SP>	SPH (S P)	SPL	

( ) : Word register name (16 bits)
< > : Long word register name (32 bits)

■ Control register map "cr"



■ Execution time

The number of states in the table of appendix B are shown in bytes, words, or long words in order of operand size. Execution time is calculated as follows.

At 16 MHz oscillation : number of states x 125 ns

At 20 MHz oscillation : number of states x 100 ns

The number of states is the average value when the program and data memory data buses are 16-bit; when buses are 8-bit, the execution time is slower by between 20 to 40%.

If the operand of a read or write instruction is a word or long word, add the number of adjustment states (the number of additional read/write cycles of the operand) below. For the read modify write instruction, add double the number of adjustment states.

Operand size	Memory size	Start address	Number of adjustment states
Word	Byte	— —	+2
Word	Word	Even number	0
Word	Word	Odd number	+2
Long word	Byte	— —	+4
Long word	Word	Even number	0
Long word	Word	Odd number	+2

The number of states of the SWI/CALL/CALR/RET/RETD/RETI instruction is the number in minimum mode. In maximum mode, PUSH /POP for the program counter takes 4 bytes; calculate by adding 2 states.

The TLCS-900 series CPU comes with a built-in 4-byte instruction queue buffer. Execution time is approximately 10 to 20% faster than the time calculated according to the above conditions.

ADC dst, src

< Add with Carry >

Operation : dst ← dst + src + CY

Description : Adds the contents of dst, src, and carry flag, and transfers the result to dst.

Details :

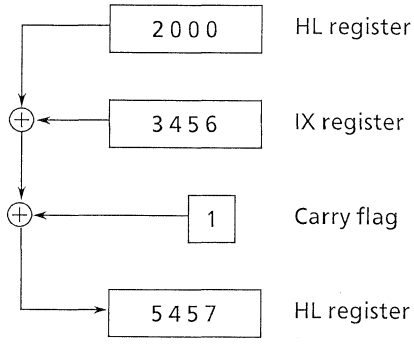
State			Mnemonic		Code
Byte	Word	Long word			
4	4	7	ADC	R, r	1 1 z z 1 r 1 0 0 1 0 R
4	4	7	ADC	r, #	1 1 z z 1 r 1 1 0 0 1 0 0 1 #<7:0> #<15:8> #<23:16> #<31:24>
4	4	6	ADC	R, (mem)	1 m z z m m m m 1 0 0 1 0 R
6	6	10	ADC	(mem), R	1 m z z m m m m 1 0 0 1 1 R
7	8	–	ADC<W>	(mem), #	1 m 0 z m m m m 0 0 1 1 1 0 0 1 #<7:0> #<15:8>

Flags:

S	Z	H	V	N	C
*	*	*	*	0	*

S = MSB value of the result is set.
Z = 1 is set if the result is 0, otherwise 0.
H = 1 is set if a carry from bit 3 to bit 4 occurs as a result of the operation; otherwise, 0. If the operand is 32-bit, an undefined value is set.
V = 1 is set if an overflow occurs as a result of the operation; otherwise, 0.
N = Cleared to zero.
C = 1 is set if a carry occurs from the MSB, otherwise 0.

Execution example: ADC HL,IX
When the HL register = 2000H, the IX register = 3456H, and the carry flag = 1, execution sets the HL register to 5457H.



ADD dst, src

< Add >

Operation : dst ← dst + src

Description : Adds the contents of dst to those of src and transfers the result to dst.

Details :

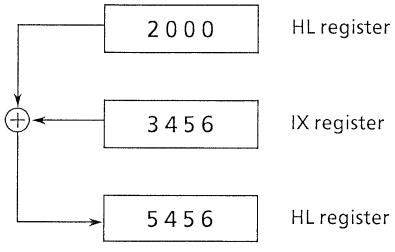
State			Mnemonic		Code
Byte	Word	Long word			
4	4	7	ADD	R, r	1 1 z z 1 r 1 0 0 0 0 R
4	4	7	ADD	r, #	1 1 z z 1 r 1 1 0 0 1 0 0 0 #<7:0> #<15:8> #<23:16> #<31:24>
4	4	6	ADD	R, (mem)	1 m z z m m m m 1 0 0 0 0 R
6	6	10	ADD	(mem), R	1 m z z m m m m 1 0 0 0 1 R
7	8	–	ADD<W>	(mem), #	1 m 0 z m m m m 0 0 1 1 1 0 0 0 #<7:0> #<15:8>

Flags : S Z H V N C

S	Z	H	V	N	C
*	*	*	*	0	*

S = MSB value of the result is set.
Z = 1 is set if the result is 0, otherwise 0.
H = 1 is set if a carry from bit 3 to bit 4 occurs as a result of the operation, otherwise 0. If the operand is 32-bit, an undefined value is set.
V = 1 is set if an overflow occurs as a result of the operation, otherwise 0.
N = Cleared to zero.
C = 1 is set if a carry occurs from the MSB, otherwise 0.

Execution example: ADD HL,IX
When the HL register = 2000H and the IX register = 3456H, execution sets the HL register to 5456H.



AND dst, src

< And >

Operation : dst ← dst AND src

Description : Ands the contents of dst and src, then transfers the result to dst.

(Truth table)

A	B	A and B
0	0	0
0	1	0
1	0	0
1	1	1

Details :

State			Mnemonic		Code
Byte	Word	Long word			
4	4	7	AND	R, r	1 1 z z 1 r 1 1 0 0 0 R
4	4	7	AND	r, #	1 1 z z 1 r 1 1 0 0 1 1 0 0 #<7:0> #<15:8> #<23:16> #<31:24>
4	4	6	AND	R, (mem)	1 m z z m m m m 1 1 0 0 0 R
6	6	10	AND	(mem), R	1 m z z m m m m 1 1 0 0 1 R
7	8	–	AND<W>	(mem), #	1 m 0 z m m m m 0 0 1 1 1 1 0 0 #<7:0> #<15:8>

Flags : S Z H V N C

S	Z	H	V	N	C
*	*	1	*	0	0

S = MSB value of the result is set.
Z = 1 is set if the result is 0, otherwise 0.
H = 1 is set.
V = 1 is set if a parity of the result is even, 0 if odd. If the operand is 32 bits, an undefined value is set.
N = Cleared to zero.
C = Cleared to zero.

Execution example: AND HL,IX
When the HL register = 7350H and the IX register = 3456H, execution sets the HL register to 3050H.

```
        0111  0011  0101  0000  ←  HL register (before execution)
AND)    0011  0100  0101  0110  ←  IX register (before execution)
        0011  0000  0101  0000  ←  HL register (after execution)
```

ANDCF num, src

< And Carry Flag >

Operation : CY ← CY and src<num>

Description : Ands the contents of the carry flag and bit num of src, and transfers the result to the carry flag.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
4	4	–	ANDCF	#4, r	1 1 0 z 1 r 0 0 1 0 0 0 0 0 0 0 0 0 #4
4	4	–	ANDCF	A, r	1 1 0 z 1 r 0 0 1 0 1 0 0 0
8	–	–	ANDCF	#3, (mem)	1 m 1 1 m m m m 1 0 0 0 0 #3
8	–	–	ANDCF	A, (mem)	1 m 1 1 m m m m 0 0 1 0 1 0 0 0

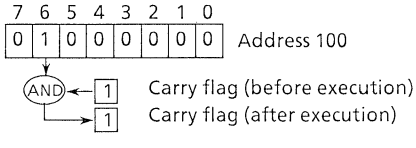
Notes : When bit num is specified by the A register, the value of the lower 4 bits of the A register is used as bit num. When the operand is a byte and the value of the lower 4 bits of bit num is from 8 to 15, the result is undefined.

Flags :

S	Z	H	V	N	C
–	–	–	–	–	*

S = No change
Z = No change
H = No change
V = No change
N = No change
C = The value obtained by anding the contents of the carry flag and the bit num of src is set.

Execution example: ANDCF 6,(100H)
When the contents of memory address 100 = 01000000B (binary) and the carry flag = 1, execution sets the carry flag to 1.



BIT num, src

< Bit test >

Operation : Z flag ← inverted value of src<num>

Description : Transfers the inverted value of the bit num of src to the Z flag.

Details :

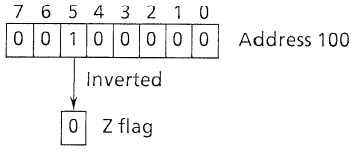
State			Mnemonic		Code
Byte	Word	Long word			
4	4	–	BIT	#4, r	1 1 0 z 1 r 0 0 1 1 0 0 1 1 0 0 0 0 # 4
8	–	–	BIT	#3, (mem)	1 m 1 1 m m m m 1 1 0 0 1 #3

Flags :

S	Z	H	V	N	C
×	*	1	×	0	–

S = An undefined value is set.
Z = The inverted value of src<num> is set.
H = 1 is set.
V = An undefined value is set.
N = Reset to 0.
C = No change

Execution example: BIT 5,(100H)
When the contents of memory address 100 = 00100000B (binary), execution sets the Z flag to 0.



BS1B dst, src

< Bit Search 1 Backward >

Operation : dst ← src backward searched value

Description : Searches the src bit pattern backward (from MSB to LSB) for the first bit set to 1 and transfers the bit number to dst.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
–	4	–	BS1B	A, r	1 1 0 1 1 r 0 0 0 0 1 1 1 1

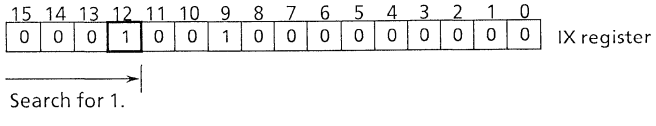
Note : dst in the operand must be the A register; src must be the register in words. If no bit set to 1 is found in the searched bit pattern, sets the A register to an undefined value and the V flag to 1.

Flags :

S	Z	H	V	N	C
–	–	–	*	–	–

S = No change
Z = No change
H = No change
V = 1 is set if the contents of src are all 0s (no bit is set to 1), otherwise 0.
N = No change
C = No change

Execution example: BS1B A,IX
When the IX register = 1200H, execution sets the A register to 0CH.



BS1F dst, src

< Bit Search 1 Forward >

Operation : dst ← src forward searched result

Description : Searches the src bit pattern forward (from LSB to MSB) for the first bit set to 1 and transfers the bit number to dst.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
–	4	–	BS1F	A, r	1 1 0 1 1 r 0 0 0 0 1 1 1 0

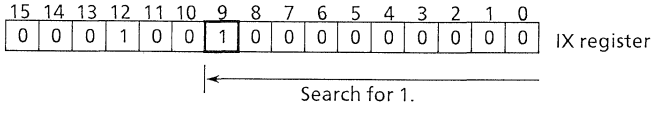
Note : dst in the operand must be the A register; src must be a register in words. If no bit set to 1 is found in the searched bit pattern, sets the A register to an undefined value and the V flag to 1.

Flags :

S	Z	H	V	N	C
–	–	–	*	–	–

S = No change
Z = No change
H = No change
V = 1 is set if the contents of src are all 0s (no bit is set to 1), otherwise 0.
N = No change
C = No change

Execution example: BS1F A,IX
When the IX register = 1200H, execution sets the A register to 09H.



CALL condition, dst

< Call subroutine >

Operation : In minimum mode, if cc is true, then XSP ← XSP−2,(XSP) ← 16-bit PC, PC ← dst.
In maximum mode, if cc is true, then XSP ← XSP−4,(XSP) ← 32-bit PC, PC ← dst.

Description : If the operand condition is true, saves the contents of the program counter to the stack area and jumps to the program address specified by dst.

Details :

State	Mnemonic	Code
12 (minimum mode) 14 (maximum mode)	CALL #16	0 0 0 1 1 1 0 0 #<7:0> #<15:8>
14 (maximum mode)	CALL #24	0 0 0 1 1 1 0 1 #<7:0> #<15:8> #<23:16>
12 (If cc is true, in minimum mode) 14 (If cc is true, in maximum mode.) 6 (cc is false.)	CALL [cc,] mem	1 m 1 1 m m m m 1 1 1 0 c c

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example: CALL 9000H
When the stack pointer XSP is 100H in minimum mode, executing this instruction at memory address 8000H writes the return address 8003H (word data) to memory address 0FEH, sets the stack pointer XSP to 0FEH, and jumps to address 9000H.

CALR dst

< Call Relative >

Operation : In minimum mode, XSP ← XSP − 2,(XSP) ← 16-bit PC,PC ← dst.
In maximum mode, XSP ← XSP − 4,(XSP) ← 32-bit PC,PC ← dst.

Description : Saves the contents of the program counter to the stack area and makes a relative jump to the program address specified by dst.

Details :

State	Mnemonic	Code
12 (minimum mode) 14 (maximum mode)	CALR $+3+d16	0 0 0 1 1 1 1 0 d<7:0> d<15:8>

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

CCF

< Complement Carry Flag >

Operation : CY ← inverted value of CY

Description : Inverts the contents of the carry flag.

Details :

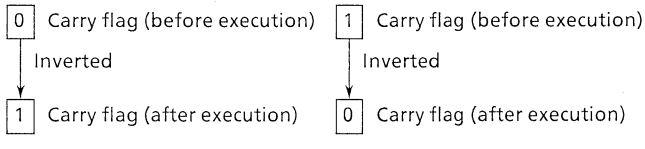
State	Mnemonic	Code
2	CCF	0 0 0 1 0 0 1 0

Flags :

S	Z	H	V	N	C
–	–	×	–	0	*

S = No change
Z = No change
H = An undefined value is set.
V = No change
N = Reset to 0.
C = Inverted value of itself is set.

Execution example: When the carry flag = 0, executing CCF sets the carry flag to 1; executing CCF again sets the carry flag to 0.



CHG num, dst

< Change >

Operation : dst<num> ← Inverted value of dst<num>

Description : Inverts the value of bit num of dst.

Details :

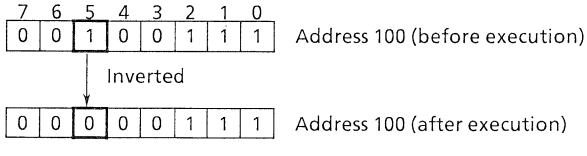
State			Mnemonic		Code
Byte	Word	Long word			
4	4	–	CHG	#4, r	1 1 0 z 1 r 0 0 1 1 0 0 1 0 0 0 0 0 #4
8	–	–	CHG	#3, (mem)	1 m 1 1 m m m m 1 1 0 0 0 #3

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example: CHG 5,(100H)
When the contents of memory address 100 = 00100111B (binary), execution sets the contents to 00000111B (binary).



CP src1, src2

< Compare >

Operation : src1 − src2

Description : Compares the contents of src1 with those of src2 and indicates the results in flag register F.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
4	4	7	CP	R, r	1 1 z z 1 r 1 1 1 1 0 R
4	4	–	CP	r, #3	1 1 0 z 1 r 1 1 0 1 1 #3
4	4	7	CP	r, #	1 1 z z 1 r 1 1 0 0 1 1 1 1 #<7:0> #<15:8> #<23:16> #<31:24>
4	4	6	CP	R, (mem)	1 m z z m m m m 1 1 1 1 0 R
6	6	6	CP	(mem), R	1 m z z m m m m 1 1 1 1 1 R
6	6	–	CP<W>	(mem), #	1 m 0 z m m m m 0 0 1 1 1 1 1 1 #<7:0> #<15:8>

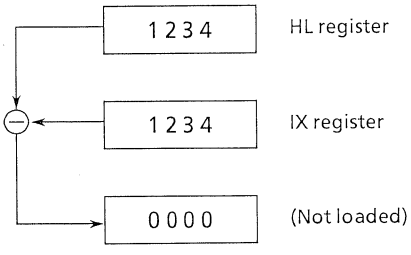
Note : #3 in operands indicates from 0 to 7.

Flags : S Z H V N C

S	Z	H	V	N	C
*	*	*	*	1	*

S = MSB value of the result is set.
Z = 1 is set if the result is 0, otherwise 0.
H = 1 is set if a borrow from bit 3 to bit 4 occurs as a result of the operation, otherwise 0. If the operand is 32 bits, an undefined value is set.
V = 1 is set if an overflow occurs as a result of the operation, otherwise 0.
N = 1 is set.
C = 1 is set if a borrow occurs from the MSB bit as a result of the operation, otherwise 0.

Execution example: CP HL,IX
When the HL register = 1234H and the IX register = 1234H, execution sets the Z and N flags to 1 and clears the S, H, V, and C flags to zero.



CPD src1, src2

< Compare Decrement >

Operation : src1 − src2, BC ← BC − 1

Description : Compares the contents of src1 with those of src2, then decrements the contents of the BC register by 1. src1 must be the A or WA register. src2 must be in post-decrement register indirect addressing mode.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
8	8	–	CPD	[A/WA, (R−)]	1 0 0 z 0 R 0 0 0 1 0 1 1 0

Note : Omitting operands in square brackets [] specifies A,(XHL−).

Flags :

S	Z	H	V	N	C
*	*	*	*	1	–

S = MSB value of the result of src1-src2 is set.
Z = 1 is set if the result of src1-src2 is 0, otherwise 0.
H = 1 is set if a borrow from bit 3 to bit 4 occurs as a result of src1-src2, otherwise 0.
V = 0 is set if the BC register value is 0 after execution, otherwise 1.
N = 1 is set.
C = No change

Execution example: CPI A,(XIX−)
When the XIX register = 00123456H and the BC register = 0200H, execution compares the contents of the A register with those of memory address 123456H, then sets the XIX register to 00123455H, the BC register to 01FFH.

CPDR src1, src2

< Compare Decrement Repeat >

Operation : src1 − src2, BC ← BC − 1, Repeat until src1 = src2 or BC = 0

Description : Compares the contents of src1 with those of src2. Then decrements the contents of the BC register by 1. Repeats until src1 = src2 or BC = 0. src1 must be the A or WA register. src2 must be in post-decrement register indirect addressing mode.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
10 (At non-repeat)	10	−	CPDR	[A/WA, (R−)]	1 0 0 z 0 R 0 0 0 1 0 1 1 1
14 (At repeat)	14	−			

Note : Omitting operands in square brackets [] specifies A,(XHL−).

Flags :

S	Z	H	V	N	C
*	*	*	*	1	−

S = MSB value of the result of src1 - src2 is set.
Z = 1 is set if the result of src1 - src2 is 0, otherwise 0.
H = 1 is set if a borrow from bit 3 to bit 4 occurs as a result of src1 - src2, otherwise 0.
V = 0 is set if the BC register value is 0 after execution, otherwise 1.
N = 1 is set.
C = No change

Execution example: CPIR A,(XIX−)

Under the following conditions, execution reads the contents of memory addresses 123456H, 123455H, and 123454H. The instruction ends with condition BC = 0 and sets the XIX register to 00123453H and the BC register to 0000H.

Conditions : A register = 55H
XIX register = 00123456H
BC register = 0003H
Memory address 123456H = 11H
Memory address 123455H = 22H
Memory address 123454H = 33H

CPI src1, src2

< Compare Increment >

Operation : src1-src2, BC ← BC − 1

Description : Compares the contents of src1 with those of src2, then decrements the contents of the BC register by 1. src1 must be the A or WA register. src2 must be in post-increment register indirect addressing mode.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
8	8	–	CPI	[A/WA, (R+)]	1 0 0 z 0 R 0 0 0 1 0 1 0 0

Note : Omitting operands enclosed in square brackets [] specifies A,(XHL+).

Flags :

S	Z	H	V	N	C
*	*	*	*	1	–

S = MSB value of the result of src1-src2 is set.
Z = 1 is set if the result of src1-src2 is 0, otherwise 0.
H = 1 is set if a borrow from bit 3 to bit 4 occurs as a result of src1-src2, otherwise 0.
V = 0 is set if the BC register value is 0 after execution, otherwise 1.
N = 1 is set.
C = No change

Execution example: CPI A, (XIX+)
When the XIX register = 00123456H and the BC register = 0200H, execution compares the contents of the A register with those of memory address 123456H, and sets the XIX register to 00123457H and the BC register to 01FFH.

CPIR src1, src2

< Compare Increment Repeat >

Operation : src1-src2, BC ← BC − 1, repeat until src1 = src2 or BC = 0

Description : Compares the contents of src1 with those of src2. Then decrements the contents of the BC register by 1. Repeats until src1 = src2 or BC = 0. src1 must be the A or WA register. src2 must be in post-increment register indirect addressing mode.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
10 (At non-repeat)	10	–	CPIR	[A/WA, (R+)]	1 0 0 z 0 R 0 0 0 1 0 1 0 1
14 (At repeat)	14	–			

Note : Omitting operands in square brackets [ ] specifies A,(XHL+).

Flags :

S	Z	H	V	N	C
*	*	*	*	1	–

S = MSB value of the result of src1-src2 is set.
Z = 1 is set if the result of src1-src2 is 0, otherwise 0.
H = 1 is set if a borrow from bit 3 to bit 4 occurs as a result of src1-src2, otherwise 0.
V = 0 is set if the BC register value is 0 after execution, otherwise 1.
N = 1 is set.
C = No change

Execution example: CPIR A,(XIX+)

Under the following conditions, execution reads memory addresses 123456H, 123457H, and 123458H. The instruction ends with condition src1 = src2, sets the XIX register to 00123459H and the BC register to 01FDH.

Conditions : A register = 33H
XIX register = 00123456
HBC register = 0200H
Memory address 123456H = 11H
Memory address 123457H = 22H
Memory address 123458H = 33H

CPL dst

< Complement >

Operation : dst ← Ones complement of dst

Description : Transfers the value of ones complement (inverted bit of 0/1) of dst to dst.

Details :

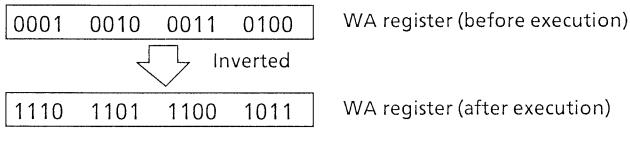
State			Mnemonic		Code
Byte	Word	Long word			
4	4	–	CPL	r	1 1 0 z 1 r 0 0 0 0 0 1 1 0

Flags :

S	Z	H	V	N	C
–	–	1	–	1	–

S = No change
Z = No change
H = 1 is set.
V = No change
N = 1 is set.
C = No change

Execution example: CPL WA
When the WA register = 1234H, execution sets the WA register to EDCBH.



DAA dst

< Decimal Adjust Accumulator >

Operation : dst ← decimal adjustment of dst

Description : Decimal adjusts the contents of dst depending on the states of the C, H, and N flags. Used to adjust the execution result of the add or subtract instruction as binary-coded decimal (BCD).

Details :

State			Mnemonic		Code
Byte	Word	Long word			
6	—	—	DAA	r	1 1 0 0 1 r 0 0 0 1 0 0 0 0

Opera - tion	N flag before DAA instruction execution	C flag before DAA instruction execution	Upper 4 bits of dst	H flag before DAA instruction execution	Lower 4 bits of dst	Added value	C flag after DAA instruction execution
	0	0	0~9	0	0~9	00	0
	0	0	0~8	0	A~F	06	0
ADD	0	0	0~9	1	0~3	06	0
	0	0	A~F	0	0~9	60	1
ADC	0	0	9~F	0	A~F	66	1
	0	0	A~F	1	0~3	66	1
	0	1	0~2	0	0~9	60	1
	0	1	0~2	0	A~F	66	1
	0	1	0~3	1	0~3	66	1
SUB	1	0	0~9	0	0~9	00	0
SBC	1	0	0~8	1	6~F	FA	0
NEG	1	1	7~F	0	0~9	A0	1
	1	1	6~F	1	6~F	9A	1

Note : Decimal adjustment cannot be performed for the INC or DEC instruction. This is because the C flag does not change.

Flags : S Z H V N C

S	Z	H	V	N	C
*	*	*	*	–	*

S = MSB value of the result is set.
Z = 1 is set if the result is 0, otherwise 0.
H = 1 is set if a carry from bit 3 to bit 4 occurs as a result of the operation, otherwise 0.
V = 1 is set if the parity (number of 1s) of the result is even, otherwise 0.
N = No change
C = 1 is set if a carry occurs from the MSB as a result of the operation or a carry was 1 before operation, otherwise 0.

Execution example:
```
ADD  A,B
DAA  A
```
When the A register = 59H and the B register = 13 H, execution sets the A register to 72H.

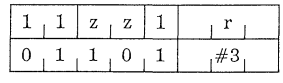
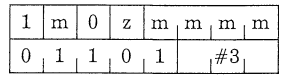
DEC num, dst

< Decrement >

Operation : dst ← dst − num

Description : Decrements dst by the contents of num and transfers the result to dst.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
4	4	5	DEC	#3, r	
6	6	–	DEC<W>	#3, (mem)	

Note : #3 in operands indicates from 1 to 8; object codes correspond from 1 to 7,0.

Flags :

S	Z	H	V	N	C
*	*	*	*	1	–

S = MSB value of the result is set.
Z = 1 is set if the result is 0, otherwise 0.
H = 1 is set if a borrow from bit 3 to bit 4 occurs as a result of the operation, otherwise 0.
V = 1 is set if an overflow occurs as a result of the operation, otherwise 0.
N = 1 is set.
C = No change

Note : With the DEC #3, r instruction, if the operand is a word or a long word, no flags change.

Execution example: DEC 4, HL
When the HL register = 5678H, execution sets the HL register to 5674H.

DECF

< Decrement Register File Pointer >

Operation : RFP<2:0> ← RFP<2:0> − 1

Description : Decrements the contents of register file pointer RFP <2:0> in the status register by 1. In maximum mode, RFP2 is fixed to 0.

Details :

State	Mnemonic	Code
2	DECF	0 0 0 0 1 1 0 1

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example: DECF
When the contents of RFP<2:0> = 2, execution sets the contents of RFP<2:0> to 1.

DI
<Disable Interrupt>
(Privileged instruction)

Operation : IFF<2:0> ← 7

Description : Sets the contents of the interrupt enable flag (IFF) <2:0> in status register to 7. After execution, only non-maskable interrupts (interrupt level 7) can be received.

Details :

State	Mnemonic	Code
5	DI	0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 1

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

DIV dst, src

< Divide >

Operation : dst<lower half> ← dst ÷ src, dst<upper half> ← remainder (unsigned)

Description : Divides unsigned the contents of dst by those of src and transfers the quotient to the lower half of dst, the remainder to the upper half of dst.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
22	30	–	DIV	RR, r	1 1 0 z 1 r 0 1 0 1 0 R
22	30	–	DIV	rr, #	1 1 0 z 1 r 0 0 0 0 1 0 1 0 #<7:0> #<15:8>
22	30	–	DIV	RR, (mem)	1 m 0 z m m m m 0 1 0 1 0 R

*For RR, see the following page.

Notes : When the operation is in bytes, dst (lower byte) ← dst (word) ÷ src (byte), dst (upper byte) ← remainder.
When the operation is in words, dst (lower word) ← dst (long word) ÷ src (word), dst (upper word) ← remainder. Match coding of the operand dst with the size of the dividend.

Flags :

S	Z	H	V	N	C
–	–	–	V	–	–

S = No change
Z = No change
H = No change
V = 1 is set when divided by 0 or the quotient exceeds the numerals which can be expressed in bits of dst for load; otherwise, 0 is set.
N = No change
C = No change

Execution example: DIV XIX,IY
When the XIX register = 12345678H and the IY register = 89ABH, execution results in a quotient of 21DAH and a remainder of 0FDAH, and sets the XIX register to 0FDA21DAH.

Note : : "RR" of the DIV RR,r and DIV RR,(mem) instruction is as listed below.

Operation size in bytes
(8 bits ← 16 bits ÷ 8 bits)

RR	Code "R"
WA	001
BC	011
DE	101
HL	111
IX	Specification not possible!
IY	
IZ	
SP	

Operation size in words
(16 bits ← 32 bits ÷ 16 bits)

RR	Code "R"
XWA	000
XBC	001
XDE	010
XHL	011
XIX	100
XIY	101
XIZ	110
XSP	111

*1 When the CPU is in minimum mode, XWA, XBC, XDE, and XHL cannot be used.

"rr" of the DIV rr,# instruction is as listed below.

Operation size in bytes
(8 bits ← 16 bits ÷ 8 bits)

rr	Code "r"
WA	001
BC	011
DE	101
HL	111
IX	C7H : F0H
IY	C7H : F4H
IZ	C7H : F8H
SP	C7H : FCH
	1st byte 2nd byte

*2 Any other word registers can be specified in the same extension coding as IX to SP.

Operation size in words
(16 bits ← 32 bits ÷ 16 bits)

rr	Code "r"
XWA	000
XBC	001
XDE	010
XHL	011
XIX	100
XIY	101
XIZ	110
XSP	111

*3 When the CPU is in minimum mode, XWA, XBC, XDE, and XHL cannot be used.
*4 Any other long word registers can be specified in the extension coding.

DIVS dst, src

< Divide Signed >

Operation : dst<lower half> ← dst ÷ src, dst<upper half> ← remainder (signed)

Description : Divides signed the contents of dst by those of src and transfers the quotient to the lower half of dst, the remainder to the upper half of dst.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
24	32	–	DIVS	RR, r	1 1 0 z 1 r 0 1 0 1 1 R
24	32	–	DIVS	rr, #	1 1 0 z 1 r 0 0 0 0 1 0 1 1 #<7:0> #<15:8>
24	32	–	DIVS	RR, (mem)	1 m 0 z m m m m 0 1 0 1 1 R

* For RR, see the following page.

Notes : When the operation is in bytes, dst (lower byte) ← dst (word) ÷ src (byte), dst (upper byte) ← remainder.
When the operation is in words, dst (lower word) ← dst (long word) ÷ src (word), dst (upper word) ← remainder.
Match coding of the operand dst with the size of the dividend. The sign of the remainder is the same as that of the dividend.

Flags :

S	Z	H	V	N	C
–	–	–	*	–	–

S = No change
Z = No change
H = No change
V = 1 is set when divided by 0, or the quotient exceeds the value which can be expressed in bits of the dst used for loading, otherwise 0.
N = No change
C = No change

Execution example: DIVS XIX,IY
When the XIX register = 12345678H and the IY register = 89ABH, execution results in the quotient as 16EEH and the remainder as D89EH, and sets the XIX register to 16EED89EH.

Note : "RR" of the DIVS RR,r and DIVS RR,(mem) instruction is as listed below.

Operation size in bytes
(8 bits ← 16 bits ÷ 8 bits)

RR	Code "R"
WA	001
BC	011
DE	101
HL	111
IX	Specification not possible!
IY	
IZ	
SP	

Operation size in words
(16 bits ← 32 bits ÷ 16 bits)

RR	Code "R"
XWA	000
XBC	001
XDE	010
XHL	011
XIX	100
XIY	101
XIZ	110
XSP	111

*1 When the CPU is in minimum mode, XWA, XBC, XDE, or XHL cannot be used.

"rr" of the DIVS rr,# instruction is as listed below.

Operation size in bytes
(8 bits ← 16 bits ÷ 8 bits)

rr	Code "r"
WA	001
BC	011
DE	101
HL	111
IX	C7H : F0H
IY	C7H : F4H
IZ	C7H : F8H
SP	C7H : FCH
	1st byte 2nd byte

*2 Any other word registers can be specified in the same extension coding as those for IX to SP.

Operation size in words
(16 bits ← 32 bits ÷ 16 bits)

rr	Code "r"
XWA	000
XBC	001
XDE	010
XHL	011
XIX	100
XIY	101
XIZ	110
XSP	111

*3 When the CPU is in minimum mode, XWA, XBC, XDE, or XHL cannot be used.
*4 Any other long word registers can be specified in the extension coding.

DJNZ dst1, dst2

< Decrement and Jump if Non Zero >

Operation : dst1 ← dst1 − 1. if dst1 ≠ 0, then PC ← dst2.

Description : Decrements the contents of dst1 by 1. Makes a relative jump to the program address specified by dst2 if the result is other than 0.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
11 (dst1 ≠ 0)	11	–	DJNZ	[r,] $+3+d8	1 1 0 z 1 r 0 0 0 1 1 1 0 0 d<7:0>
7 (dst1 = 0)	7	–			

Note : Omitting "r" of the operand in square brackets [ ] is regarded as specifying the B register.

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example:
```
LOOP: ADD   A, A
      DJNZ  W, LOOP
```
When the A register = 12H and the W register = 03H, execution loops three times and sets the A register to 24H→48→90H and the W register to 02H→01H→00H.

EI num

<Enable Interrupt>

(Privileged instruction)

Operation : IFF <2:0> ← num

Description : Sets the contents of the IFF<2:0> in the status register to num. After execution, the CPU interrupt receive level becomes num.

Details :

State	Mnemonic	Code
5	EI [#3]	0 0 0 0 0 1 1 0 0 0 0 0 0 #3

Note : A value from 0 to 7 can be specified as the operand value. If the operand is omitted, the default value is "0" (EI 0).

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

EX dst, src

< Exchange >

Operation : dst ↔ src

Description : Exchanges the contents of dst and src.

Details :

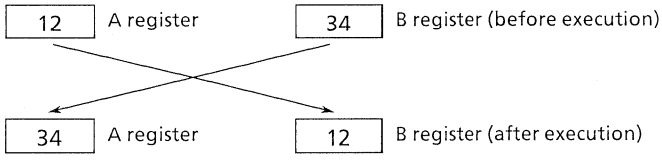
State			Mnemonic		Code
Byte	Word	Long word			
2	–	–	EX	F, F'	0 0 0 1 0 1 1 0
5	5	–	EX	R, r	1 1 z z 1 r 1 0 1 1 1 R
6	6	–	EX	(mem), r	1 m z z m m m m 0 0 1 1 0 R

Flags : S Z H V N C

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change
* Executing EX F,F' changes all flags.

Execution example: EX A,B
When the A register = 12H and the B register = 34H, execution sets the A register to 34H and the B register to 12H.



EXTS dst

< Extend Sign >

Operation : dst <upper half> ← signed bit of dst <lower half>

Description : Transfers (copies) the signed bit (bit 7 when the operand size is a word, bit 15 when a long word) of the lower half of dst to all bits of the upper half of dst.

Details :

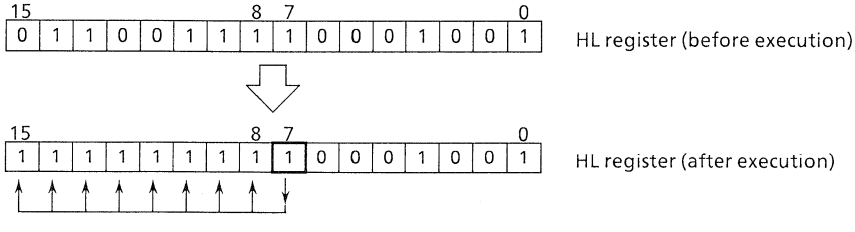
State			Mnemonic		Code
Byte	Word	Long word			
–	5	5	EXTS	r	1 1 z z 1 r / 0 0 0 1 0 0 1 1

フラグ :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example: EXTS HL
When the HL register = 6789H, execution sets the HL register to FF89H.



EXTZ dst

< Extend Zero >

Operation : dst<upper half> ← 0

Description : Clears the upper half of dst to zero. Used for making the operand sizes the same when they are different.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
–	4	4	EXTZ	r	1 1 z z 1 r 0 0 0 1 0 0 1 0

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example: EXTZ HL
When the HL register = 6789H, execution sets the HL register to 0089H.

EXTZ XIX
When the XIX register = 12345678H, execution sets the XIX register to 00005678H.

HALT

< Halt CPU >
(Privileged instruction)

Operation : CPU halt

Description : Halts the instruction execution. To resume, an interrupt must de received.

Details :

State	Mnemonic	Code
8	HALT	0 0 0 0 0 1 0 1

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

INC num, dst

< Increment >

Operation : dst←dst+num

Description : Adds the contents of dst and num and transfers the result to dst.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
4	4	4	INC	#3, r	1 1 z z 1 r 0 1 1 0 0 #3
6	6	–	INC<W>	#3, (mem)	1 m 0 z m m m m 0 1 1 0 0 #3

Note : #3 in operands indicates from 1 to 8 and object codes correspond from 1 to 7,0.

Flags :

S	Z	H	V	N	C
*	*	*	*	0	–

S = MSB value of the result is set.
Z = 1 is set if the result is 0, otherwise 0.
H = 1 is set if a carry occurs from bit 3 to bit 4 as a result of the operation, otherwise 0.
V = 1 is set if an overflow occurs as a result of the operation, otherwise 0.
N = Cleared to zero.
C = No change

Note : With the INC #3,r instruction, if the operand is a word or a long word, no flags change.

Execution exampl : INC 5,WA
When the WA register = 1234H, execution sets the WA register to 1239H.

INCF

< Increment Register File Pointer >

Operation : RFP<2:0> ← RFP<2:0> + 1
Description : Increments the contents of RFP<2:0> in the status register by 1. In maximum mode, RFP2 is fixed to 0.

Details :

State	Mnemonic	Code
2	INCF	0 0 0 0 1 1 0 0

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example: INCF
When the contents of RFP<2:0> = 2, execution sets the contents of RFP<2:0> to 3.

JP condition, dst

< Jump >

Operation : If cc is true, then PC ← dst.

Description : If the operand condition is true, jumps to the program address specified by dst.

Details :

State	Mnemonic		Code
7	JP	#16	0 0 0 1 1 0 1 0 #<7:0> #<15:8>
7	JP	#24	0 0 0 1 1 0 1 1 #<7:0> #<15:8> #<23:16>
9 (cc is true) 6 (cc is false)	JP	[cc,] mem	1 m 1 1 m m m m 1 1 0 1 c c

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example: JP 2000H
Execution jumps unconditionally to address 2000H.

JR condition, dst

< Jump Relative >

Operation : If cc is true, then PC ← dst.

Description : If the operand condition is true, makes a relative jump to the program address specified by dst.

Details :

State	Mnemonic		Code
8 (cc is true) 4 (cc is false)	JR	[cc,] $+2+d8	0 1 1 0 c c d<7:0>
8 (cc is true) 4 (cc is false)	JRL	[cc,] $+3+d16	0 1 1 1 c c #<7:0> #<15:8>

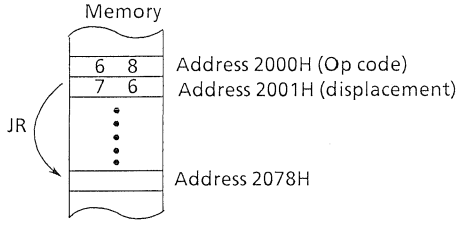
Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example: JR 2078H

When this instruction is executed at memory address 2000H, execution relative jumps unconditionally to address 2078H. The object code of the instruction is 68H : 76H.



LD dst, src

< Load >

Operation : dst ← src

Description : Loads the contents of src to dst.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
4	4	4	LD	R, r	1 1 z z 1 r 1 0 0 0 1 R
4	4	4	LD	r, R	1 1 z z 1 r 1 0 0 1 1 R
4	4	4	LD	r, #3	1 1 z z 1 r 1 0 1 0 1 #3
2	3	5	LD	R, #	0 z z z 0 R #<7:0> #<15:8> #<23:16> #<31:24>
4	4	6	LD	r, #	1 1 z z 1 r 0 0 0 0 0 0 1 1 #<7:0> #<15:8> #<23:16> #<31:24>
4	4	6	LD	R, (mem)	1 m z z m m m m 0 0 1 0 0 R
4	4	6	LD	(mem), R	1 m 1 1 m m m m 0 1 z z 0 R
5	6	–	LD<W>	(#8), #	0 0 0 0 1 0 z 0 #8 #<7:0> #<15:8>

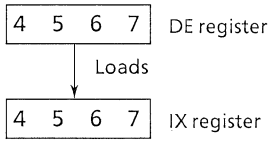
State			Mnemonic		Code
Byte	Word	Long word			
5	6	–	LD<W>	(mem), #	1 m 1 1 m m m m 0 0 0 0 0 0 z 0 #<7:0> #<15:8>
8	8	–	LD<W>	(#16), (mem)	1 m 0 z m m m m 0 0 0 1 1 0 0 1 #16<7:0> #16<15:8>
8	8	–	LD<W>	(mem), (#16)	1 m 1 1 m m m m 0 0 0 1 0 1 z 0 #16<7:0> #16<15:8>

Flags : S Z H V N C

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example: LD IX, DE
When the DE register = 4567H, execution sets the IX register to 4567H.



LDA dst, src

< Load Address >

Operation : dst ← src effective address value

Description : Loads the src effective address value to dst.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
–	4	4	LDA	R, mem	1 m 1 1 m m m m / 0 0 1 s 0 R

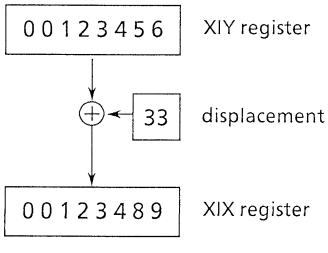
Note : This instruction operates much like the ADD instruction; the difference is that dst is specified independently from src. Mainly used for handling the pointer with the C compiler.

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example: LDA XIX, XIY+33H
When the XIY register = 00123456H, execution sets the XIX register to 00123489H.



LDAR dst, src

< Load Address Relative >

Operation : dst ← src relative address value

Description : Loads the relative address value specified in src to dst.

Details :

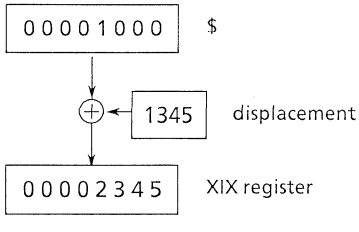
State		Mnemonic		Code
Byte	Word	Long word		
–	11	11	LDAR R, $+4+d16	1 1 1 1 0 0 1 1 0 0 0 1 0 0 1 1 d<7:0> d<15:8> 0 0 1 s 0 R

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example: LDAR XIX, $+1345H
When this instruction is executed at memory address 1000H, execution sets the XIX register to 00002345H. $ indicates the start address of the instruction. The instruction's object codes are: F3H:13H:41H:13H:34H.



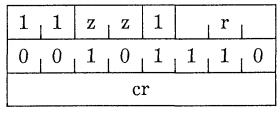
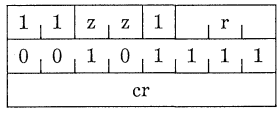
LDC dst, src

< Load Control >
(Privileged instruction)

Operation : dst ← src

Description : Loads the contents of src to dst.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
8	8	8	LDC	cr, r	
8	8	8	LDC	r, cr	

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example: LDC DMAC0, WA
When the WA register = 1234H, execution sets control register DMAC0 to 1234H.

LDCF num, src

< Load Carry Flag >

Operation : CY ← src<num>

Description : Loads the contents of bit num of src to the carry flag.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
4	4	–	LDCF	#4, r	1 1 0 z 1 r 0 0 1 0 0 0 1 1 0 0 0 0 #4
4	4	–	LDCF	A, r	1 1 0 z 1 r 0 0 1 0 1 0 1 1
8	–	–	LDCF	#3, (mem)	1 m 1 1 m m m m 1 0 0 1 1 #3
8	–	–	LDCF	A, (mem)	1 m 1 1 m m m m 0 0 1 0 1 0 1 1

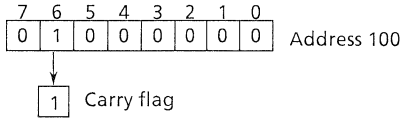
Notes : When bit num is specified by the A register, the value of the lower 4 bits of the A register is used as bit num. When the operand is a byte and the value of the lower 4 bits of bit num is from 8 to 15, the value of the carry flag is undefined.

Flags :

S	Z	H	V	N	C
–	–	–	–	–	*

S = No change
Z = No change
H = No change
V = No change
N = No change
C = Contents of bit num of src is set.

Execution example: LDCF 6, (100H)
When the contents of memoryad address 100 = 01000000B (binary), execution sets the carry flag to 1.



LDD dst, src

< Load Decrement >

Operation : dst ← src, BC ← BC − 1

Description : Loads the contents of src to dst, then decrements the contents of the BC register by 1. src and dst must be in post-decrement register indirect addressing mode.

Details :

State			Mnemonic	Code
Byte	Word	Long word		
10	10	−	LDD<W> [(XDE−), (XHL−)]	1 0 0 z 0 0 1 1 0 0 0 1 0 0 1 0
			LDD<W> (XIX−), (XIY−)	1 0 0 z 0 1 0 1 0 0 0 1 0 0 1 0

* Coding in square brackets [ ] can be omitted.

Flags :

S	Z	H	V	N	C
−	−	0	*	0	−

S = No change
Z = No change
H = Cleared to 0.
V = 0 is set if the BC register value is 0 after execution, otherwise 1.
N = Cleared to zero.
C = No change

Execution example: LDD (XIX−), (XIY−)
When the XIX register = 00123456H, the XIY register = 00335577H, and the BC register = 0700H, execution loads the contents at address 335577 to address 123456H and sets the XIX register to 123455H, the XIY register to 00335576H, and the BC register to 06FFH.

LDDR dst, src

< Load Decrement Repeat >

Operation : dst ← src, BC ← BC − 1, Repeat until BC = 0

Description : Loads the contents of src to dst, then decrements the contents of the BC register by 1. If the result is other than 0, the operation is repeated. src and dst must be in post-decrement register indirect addressing mode.

Details :

State			Mnemonic	Code
Byte	Word	Long word		
10 (Non-repeat)	10	–	LDDR<W>[(XDE−), (XHL−)]	1 0 0 z 0 0 1 1 0 0 0 1 0 0 1 1
14 (Repeat)	14	–	LDDR<W> (XIX−), (XIY−)	1 0 0 z 0 1 0 1 0 0 0 1 0 0 1 1

* Coding in square brackets [ ] can be omitted.

Note : Interrupt requests are sampled every time 1 item of data is loaded.

Flags :

S	Z	H	V	N	C
–	–	0	0	0	–

S = No change
Z = No change
H = Cleared to zero.
V = Cleared to zero.
N = Cleared to zero.
C = No change

Execution example: LDDR (XIX−), (XIY−)
When the XIX register = 00123456H, the XIY register = 00335577H, and the BC register = 0003H, the results of the execution are as follows:
Loads the contents of address 335577H to 123456H.
Loads the contents of address 335576H to 123455H.
Loads the contents of address 335575H to 123454H.
Sets the XIX register to 00123453H.
Sets the XIY register to 00335574H.
Sets the BC register to 0000H.

LDF num

< Load Register File Pointer >

Operation : RFP<2: 0> ← num

Description : Loads the num value to the register file pointer RFP<2:0> in status register. Since RFP2 is fixed to 0 in maximum mode, when the num value is from 4 to 7, RFP is set to from 0 to 3.

Details :

State	Mnemonic		Code
2	LDF	#3	0 0 0 1 0 1 1 1 0 0 0 0 0 #3

Note : In minimum mode, the operand value can be specified from 0 to 7; in maximum mode, from 0 to 3.

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

LDI dst, src

< Load Increment >

Operation : dst ← src, BC ← BC − 1

Description : Loads the contents of src to dst, then decrements the contents of the BC register by 1. src and dst must be in post-increment register indirect addressing mode.

Details :

State			Mnemonic	Code
Byte	Word	Long word		
10	10	–	LDI<W>[(XDE+),(XHL+)]	1 0 0 z 0 0 1 1 0 0 0 1 0 0 0 0
			LDI<W>(XIX+),(XIY+)	1 0 0 z 0 1 0 1 0 0 0 1 0 0 0 0

* Coding in square brackets [ ] can be omitted.

Flags :

S	Z	H	V	N	C
–	–	0	*	0	–

S = No change
Z = No change
H = Cleared to zero.
V = 0 is set when the BC register value is 0 after execution, otherwise 1.
N = Cleared to zero.
C = No change

Execution example: LDI (XIX+),(XIY+)
When the XIX register = 00123456H, the XIY register = 00335577H, and the BC register = 0700H, execution loads the contents of address 335577H to 123456H and sets the XIX register to 00123457H, the XIY register to 00335578H, and the BC register to 06FFH.

LDIR dst, src

< Load Increment Repeat >

Operation : dst ← src, BC ← BC − 1, Repeat until BC = 0

Description : Loads the contents of src to dst, then decrements the contents of the BC register by 1. If the result is other than 0, the operation is repeated. src and dst must be in post-increment register indirect addressing mode.

Details :

State			Mnemonic	Code
Byte	Word	Long word		
10 (Non-repeat)	10	–	LDIR<W>[(XDE+),(XHL+)]	1 0 0 z 0 0 1 1 0 0 0 1 0 0 0 1
14 (Repeat)	14	–	LDIR<W>(XIX+),(XIY+)	1 0 0 z 0 1 0 1 0 0 0 1 0 0 0 1

* Coding in square brackets [ ] can be omitted.

Note : Interrupt requests are sampled every time 1 item of data is loaded.

Flags :

S	Z	H	V	N	C
–	–	0	0	0	–

S = No change
Z = No change
H = Cleared to zero.
V = Cleared to zero.
N = Cleared to zero.
C = No change

Execution example: LDIR (XIX+),(XIY+)
When the XIX register = 00123456H, the XIY register = 00335577H, and the BC register = 0003H, execution results as follows:
Loads the contents of address 335577H to 123456H.
Loads the contents of address 335578H to 123457H.
Loads the contents of address 335579H to 123458H.
Sets the XIX register to 00123459H.
Sets the XIY register to 0033557AH.
Sets the BC register to 0000H.

LDX dst, src

< Load eXtract >

Operation : dst ← src

Description : Loads the contents of src to dst. The effective code is assigned to this instruction every other byte. Used to fetch the code from 8-bit data bus memory in 16-bit data bus mode.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
9	–	–	LDX	(#8), #	1 1 1 1 1 0 1 1 1 0 0 0 0 0 0 0 0 0 #8 0 0 0 0 0 0 0 0 # 0 0 0 0 0 0 0 0

Note : Even if the second, fourth, or sixth instruction code value is not 00H, the instruction operates correctly.

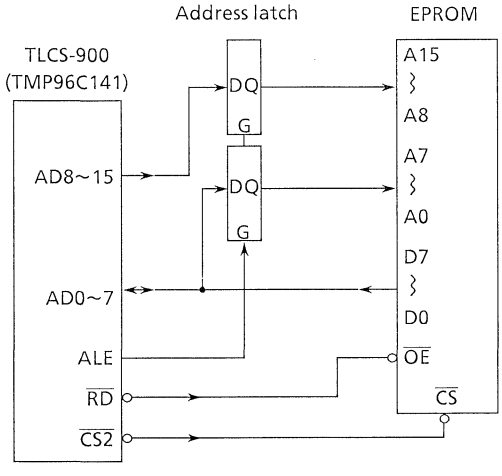
Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example: LDX (6AH), 9CH

Using the TMP96C141, the example executes the program using an EPROM which has an 8-bit data bus. After reset, starts fetching the program code in 16-bit data bus mode. When the program starts with an external memory with an 8-bit data bus, loads the above instruction to the start address, 8000H. Execution writes the 9CH data to the control register at address 6AH of the built-in programmable chip select/wait controller. As a result, memory addresses 8000H to 3FFFFFH enters 8-bit data bus 0WAIT mode.



After reset movement

TLCS-900		EPROM	
F 7		F 7	Address 8000H
	← ?	0 0	Address 8001H
6 A		6 A	Address 8002H
	← ?	0 0	Address 8003H
9 C		9 C	Address 8004H
	← ?	0 0	Address 8005H

? indicates code input from pins AD8 to 15.

LINK dst, num

< Link >

Operation : (−XSP) ← dst, dst ← XSP, XSP ← XSP + num

Description : Saves the contents of dst to the stack area. Loads the contents of stack pointer XSP to dst. Adds the contents of XSP to those of num (signed) and loads the result to XSP. Used for obtaining a local variable area in the stack area for -num bytes.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
−	−	10	LINK	r, d16	1 1 1 0 1 r 0 0 0 0 1 1 0 0 d<7:0> d<15:8>

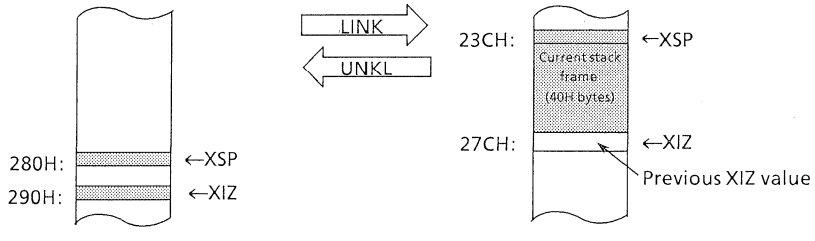
Flags :

S	Z	H	V	N	C
−	−	−	−	−	−

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example: LINK XIZ, −40H

When stack pointer XSP = 280H and the XIZ register = 290H, execution writes 00000290H (long data) at memory address 27CH and sets the XIZ register to 27CH and the stack pointer to XSP 23CH.



MAX

<Maximum>

(Privileged instruction)

Operation : Max bit ← 1

Description : Sets the MAX bit in status register to 1. Changes the CPU operation mode to maximum.

Details :

State	Mnemonic	Code
4	MAX	0 0 0 0 0 1 0 0

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Note : Basically, there is no instruction for changing from maximum to minimum mode. However, if it is absolutely necessary, execute either of the following two ways :

```
(1) PUSH   SR
    RES    3, (XSP+1)
    POP    SR
(2) SWI    n
```

RES 3, (XSP+1)
RETI
} At address 8000H+n× 10H (n = 0 to 7)

(1) in system mode only
(2) in both system and normal modes.

MDEC1 num, dst

< Modulo Decrement 1 >

Operation : if (dst mod num) = 0 then dst ← dst + (num − 1) else dst ← dst − 1.

Description : When the modulo num of dst is 0, increments dst by num -1 .
Otherwise, decrements dst by 1. Used to operate pointers for cyclic memory table.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
–	7	–	MDEC1	#, r	1 1 0 1 1 r 0 0 1 1 1 1 0 0 #<7:0> − 1 #<15:8>

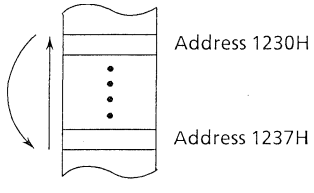
Note : The operand # must be 2 to the nth power. (n = 1 to 15)

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example: Decrements the IX register by cycling from 1230H to 1237H.
MDEC1 8, IX
When the IX register = 1231H, execution sets the IX register to 1230H. Further execution increments the IX register by 8-1 and sets the IX register to 1237H, since the IX register modulo 8 = 0.



MDEC2 num, dst

< Modulo Decrement 2 >

Operation : if (dst mod num) = 0 then dst ← dst + (num − 2) else dst ← dst − 2.

Description : When the modulo num of dst is 0, increments dst by num − 2. Otherwise, decrements dst by 2. Used to operate pointers for cyclic memory table.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
–	7	–	MDEC2	#, r	1 1 0 1 1 r 0 0 1 1 1 1 0 1 #<7:0> − 2 #<15:8>

Note : The operand # must be 2 to the nth power. (n = 2 to 15)

Flags :

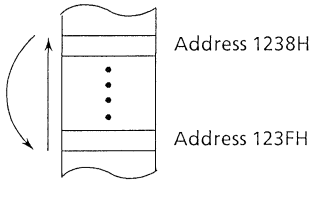
S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example: Decrements the IX register by cycling from 1238H to 123FH.

```
MDEC2  8,IX
```

When the IX register = 123AH, execution sets the IX register to 1238H. Further execution increments the IX register by 8-2 and sets the IX register to 123EH, since the IX register modulo 8 = 0.



MDEC4 num, dst

< Modulo Decrement 4 >

Operation : if (dst mod num)=0 then dst←dst+(num−4) else dst←dst−4.

Description : When the modulo num of dst is 0, increments dst by num−4. Otherwise, decrements dst by 4. Used to operate pointers for cyclic memory table.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
–	7	–	MDEC4	#, r	

1	1	0	1	1	r			
0	0	1	1	1	1	1	0	
#<7:0>−4								
#<15:8>								

Note : The operand # must be 2 to the nth power. (n = 3 to 15)

Flags :

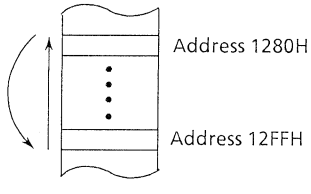
S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example: Decrements the IX register by cycling from 1280H to 12FFH.

```
MDEC4  80H,IX
```

When the IX register = 1284H, execution sets the IX register to 1280H. Further execution increments the IX register by 80H−4 and sets the IX register to 12FCH, since the IX register modulo 80H = 0.



MINC1 num, dst

< Modulo Increment 1 >

Operation : if (dst mod num) = (num − 1) then dst ← dst − (num − 1) else dst ← dst + 1.

Description : When the modulo num of dst is num − 1, decrements dst by num − 1. Otherwise, increments dst by 1. Used to operate pointers for cyclic memory table.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
–	8	–	MINC1	#, r	1 1 0 1 1 r 0 0 1 1 1 0 0 0 #<7:0> − 1 #<15:8>

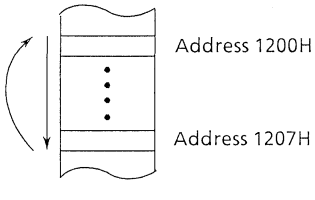
Note : The operand # must be 2 to the nth power. (n = 1 to 15)

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example: Increments the IX register by cycling from 1200H to 1207H.
MINC1 8, IX
When the IX register = 1206H, execution sets the IX register to 1207H. Further execution decrements the IX register by 8 − 1 and sets the IX register to 1200H, since the IX register modulo 8 = 8 − 1.



MINC2 num, dst

< Modulo Increment 2 >

Operation : if (dst mod num) = (num−2) then dst ← dst− (num−2) else dst ← dst+2.

Description : When the modulo num of dst is num−2, decrements dst by num−2. Otherwise, increments dst by 2. Used to operate pointers for cyclic memory table.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
–	8	–	MINC2	#, r	1 1 0 1 1 r 0 0 1 1 1 0 0 1 #<7:0> −2 #<15:8>

Note : The operand # must be 2 to the nth power. (n = 2 to 15)

Flags :

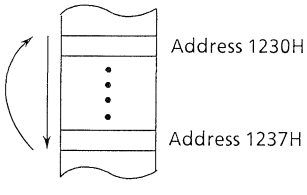
S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example: Increments the IX register by cycling from 1230H to 1237H.

```
MINC2  8,IX
```

When the IX register = 1234H, execution sets the IX register to 1236H. Further execution decrements the IX register by 8−2 and sets the IX Register to 1230H, since the IX register modulo 8 = 8−2.



MINC4 num, dst

< Modulo Increment 4 >

Operation : if (dst mod num) = (num−4) then dst ← dst− (num−4) else dst ← dst+4.

Description : When the modulo num of dst is num−4, decrements dst by num−4. Otherwise, increments dst by 4. Used to operate pointers for cyclic memory table.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
–	8	–	MINC4	#, r	1 1 0 1 1 r 0 0 1 1 1 0 1 0 #<7:0>−4 #<15:8>

Note : The operand # must be 2 to the nth power. (n = 3 to 15)

Flags :

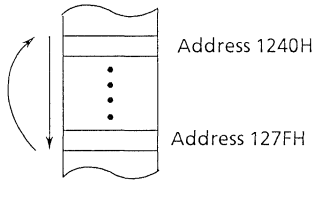
S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example: Increments the IX register by cycling from 1240H to 127FH.

MINC4 40H,IX

When the IX register = 1278H, execution sets the IX register to 127CH. Further execution decrements the IX register by 40H−4 and sets the IX register to 1240H, since the IX register modulo 40H = 40H−4.



MIRR dst

< Mirror >

Operation : dst<MSB:LSB> ← dst<LSB : MSB>

Description : Mirror-exchanges the contents of dst using the bit pattern image.

Details :

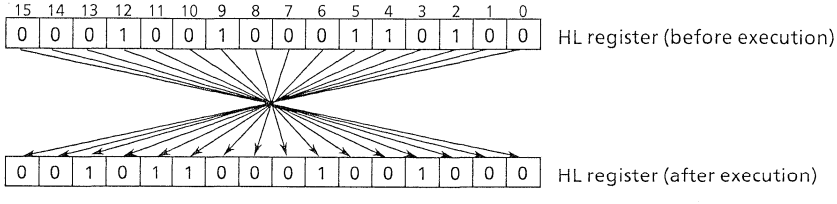
State			Mnemonic		Code
Byte	Word	Long word			
–	4	–	MIRR	r	1 1 0 1 1 r 0 0 0 1 0 1 1 0

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example: MIRR HL
When the HL register = 0001 0010 0011 0100B (binary), execution sets the HL register to 0010 1100 0100 1000B (binary).



MUL dst, src

< Multiply >

Operation : dst ← dst<lower half> × src (unsigned)

Description : Multiplies unsigned the contents of lower half of dst by those of src and loads the result to dst.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
18	26	–	MUL	RR, r	1 1 0 z 1 r 0 1 0 0 0 R
18	26	–	MUL	rr, #	1 1 0 z 1 r 0 0 0 0 1 0 0 0 # <7:0> # <15:8>
18	26	–	MUL	RR, (mem)	1 m 0 z m m m m 0 1 0 0 0 R

Note : When the operation is in bytes, dst (word) ← dst (byte) × src (byte).
When the operation is in words, dst (long word) ← dst (word) × src (word).
Match coding of the operand dst with the size of the result.

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example: MUL XIX, IY
When the IX register = 1234H and the IY register = 89ABH, execution multiplies unsigned the contents of the IX register by those of the IY register and sets the XIX register to 09C9FCBCH.

Note : "RR" for the MUL RR,r and MUL RR,(mem) instructions is as listed below:

Operation size in bytes
(16bits ← 8bits × 8bits)

RR	Code R
WA	001
BC	011
DE	101
HL	111
IX	Specification not possible!
IY	
IZ	
SP	

Operation size in words
(32bits ← 16bits × 16bits)

RR	Code R
XWA	000
XBC	001
XDE	010
XHL	011
XIX	100
XIY	101
XIZ	110
XSP	111

*1 When the CPU is in minimum mode, XWA, XBC, XDE, or XHL cannot be used.

"rr" for the MUL rr,# instruction is as listed below.

Operation size in bytes
(16bits ← 8bits × 8bits)

rr	Code r
WA	001
BC	011
DE	101
HL	111
IX	C7H : F0H
IY	C7H : F4H
IZ	C7H : F8H
SP	C7H : FCH
	1st byte 2nd byte

*2 Any other word registers can be specified in the same extension coding as those for IX to SP.

Operation size in words
(32bits ← 16bits × 16bits)

rr	Code r
XWA	000
XBC	001
XDE	010
XHL	011
XIX	100
XIY	101
XIZ	110
XSP	111

*3 When the CPU is in minimum mode, XWA, XBC, XDE, or XHL cannot be used.

*4 Any other long word registers can be specified in the extension coding.

MULA dst

< Multiply and Add >

Operation : dst ← dst + (XDE) × (XHL), XHL ← XHL − 2

Description : Multiplies signed the memory data (16 bits) specified by the XDE register by the memory data (16 bits) specified by the XHL register. Adds the result (32 bits) to the contents of dst (32 bits) and loads the sum to dst (32 bits). Then, decrements the contents of the XHL register by 2.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
–	31	–	MULA	rr	1 1 0 1 1 r 0 0 0 1 1 0 0 1

Note : Match coding of the operand dst with the operation size (long word).

Flags :

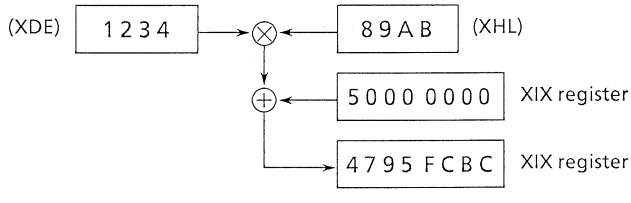
S	Z	H	V	N	C
*	*	–	*	–	–

S = MSB value of the result is set.
Z = 1 is set when the result is 0, otherwise 0.
H = No change.
V = 1 is set when an overflow occurs as a result, otherwise 0.
N = No change.
C = No change.

Execution example: MULA XIX

Under the following conditions, execution sets the XIX register to 4795FCBCH and the XHL register to 1FEH.

Conditions: XIX register = 50000000H
XDE register = 100H
XHL register = 200H
Memory data (word) at address 100H = 1234H
Memory data (word) at address 200H = 89ABH



MULS dst, src

< Multiply Signed >

Operation : dst ← dst<lower half> × src (signed)

Description : Multiplies signed the contents of the lower half of dst by those of src and loads the result to dst.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
18	26	–	MULS	RR, r	1 1 0 z 1 r 0 1 0 0 1 R
18	26	–	MULS	rr, #	1 1 0 z 1 r 0 0 0 0 1 0 0 1 # <7:0> # <15:8>
18	26	–	MULS	RR, (mem)	1 m 0 z m m m m 0 1 0 0 1 R

Note : When the operation is in bytes, dst(word) ← dst (byte) × src (byte).
When the operation is in words, dst (long word) ← dst (word) × src (word).
Match coding of the operand dst with the size of the result.

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example: MULS XIX, IY
When the IX register = 1234H and the IY register = 89ABH, execution multiplies signed the contents of the IX register by those of the IY register and sets the XIX register to F795FCBCH.

Note : "RR" for the MULS RR,r and MULS RR,(mem) instructions is as listed below:

Operation size in bytes
(16bits←8bits × 8bits)

RR	Code R
WA	001
BC	011
DE	101
HL	111
IX	Specification not possible!
IY	
IZ	
SP	

Operation size in words
(32bits←16bits × 16bits)

RR	Code R
XWA	000
XBC	001
XDE	010
XHL	011
XIX	100
XIY	101
XIZ	110
XSP	111

*1 When the CPU is in minimum mode, XWA, XBC, XDE, or XHL cannot be used.

"rr" for the MULS rr,# instruction is as listed below.

Operation size in bytes
(16bits←8bits × 8bits)

rr	Code r
WA	001
BC	011
DE	101
HL	111
IX	C7H : F0H
IY	C7H : F4H
IZ	C7H : F8H
SP	C7H : FCH
	1st byte 2nd byte

*2 Any other word registers can be specified in the same extension coding as those for IX to SP.

Operation size in words
(32bits←16bits × 16bits)

rr	Code r
XWA	000
XBC	001
XDE	010
XHL	011
XIX	100
XIY	101
XIZ	110
XSP	111

*3 When the CPU is in minimum mode, XWA, XBC, XDE, or XHL cannot be used.

*4 Any other long word registers can be specified in the extension coding.

NEG dst

< Negate >

Operation : dst←0−dst

Description : Decrements 0 by the contents of dst and loads the result to dst. (Twos complement)

Details :

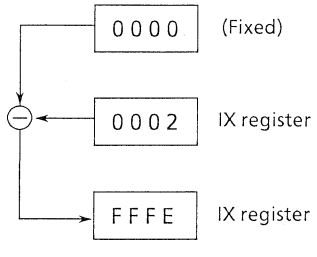
State			Mnemonic		Code
Byte	Word	Long word			
5	5	–	NEG	r	1 1 0 z 1 r 0 0 0 0 0 1 1 1

Flags :

S	Z	H	V	N	C
*	*	*	*	1	*

S = MSB value of the result is set.
Z = 1 is set when the result is 0, otherwise 0.
H = 1 is set when a borrow from bit 3 to bit 4 occurs as a result, otherwise 0.
V = 1 is set when an overflow occurs as a result, otherwise 0.
N = 1 is set.
C = 1 is set when a borrow from the MSB occurs as a result, otherwise 0.

Execution example: NEG IX
When the IX register = 0002H, execution sets the IX register to FFFEH.



NOP

<No Operation>

Operation : None.

Description : Does nothing but moves execution to the next instruction. The object code of this instruction is 00H.

Details :

State	Mnemonic	Code
2	NOP	0 0 0 0 0 0 0 0

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

NORMAL

< Normal >
(Privileged instruction)

Operation : SYSM bit ←0

Description : Resets the SYSM bit in status register to 0 and changes the CPU to normal mode.

Details :

State	Mnemonic	Code
4	NORMAL	0 0 0 0 0 0 0 1

Flags :

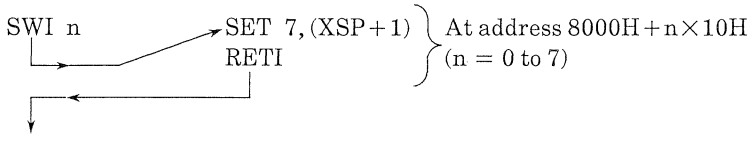
S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Note : Basically, only the software interrupt (SWI) instruction changes the mode from normal to system by software. The following change the mode :

(1) SWI
(2) Privilege violation interrupt
(3) Illegal instruction interrupt
(4) Hardware interrupt
(5) Reset

The SWI instruction is used to change the mode to system in the middle of program execution, as shown below.



OR dst, src

< Logical OR >

Operation : dst← dst OR src

Description : Ors the contents of dst with those of src and loads the result to dst.

(Truth table)

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

Details :

State			Mnemonic		Code
Byte	Word	Long word			
4	4	7	OR	R, r	1 1 z z 1 r 1 1 1 0 0 R
4	4	7	OR	r, #	1 1 z z 1 r 1 1 0 0 1 1 1 0 #<7:0> #<15:8> #<23:16> #<31:24>
4	4	6	OR	R, (mem)	1 m z z m m m m 1 1 1 0 0 R
6	6	10	OR	(mem), R	1 m z z m m m m 1 1 1 0 1 R
7	8	–	OR<W>	(mem), #	1 m 0 z m m m m 0 0 1 1 1 1 1 0 #<7:0> #<15:8>

Flags : S Z H V N C

S	Z	H	V	N	C
*	*	0	*	0	0

S = MSB value of the result is set.
Z = 1 is set when the result is 0, otherwise 0.
H = 0 is set.
V = 1 is set when the parity (number of 1s) of the result is even, 0 when odd. When the operand is 32-bit, an undefined value is set.
N = Cleared to 0.
C = Cleared to 0.

Execution example: OR HL, IX
When the HL register = 7350H and the IX register is 3456H, execution sets the HL register to 7756H.

```
        0111  0011  0101  0000  ←  HL register (before execution)
OR  )   0011  0100  0101  0110  ←  IX register (before execution)
        0111  0111  0101  0110  ←  HL register (after execution)
```

ORCF num, src

< OR Carry Flag >

Operation : CY ← CY OR src<num>

Description : Ors the contents of the carry flag with those of bit num of src and loads the result to the carry flag.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
4	4	–	ORCF	#4, r	1 1 0 z 1 r 0 0 1 0 0 0 0 1 0 0 0 0 #4
4	4	–	ORCF	A, r	1 1 0 z 1 r 0 0 1 0 1 0 0 1
8	–	–	ORCF	#3, (mem)	1 m 1 1 m m m m 1 0 0 0 1 #3
8	–	–	ORCF	A, (mem)	1 m 1 1 m m m m 0 0 1 0 1 0 0 1

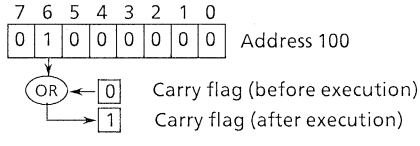
Note : When bit num is specified by the A register, the value of the lower 4 bits of the A register is used as bit num. When the operand is a byte and the value of the lower bits of bit num is from 8 to 15, the result is undefined.

flag :

S	Z	H	V	N	C
–	–	–	–	–	*

S = No change
Z = No change
H = No change
V = No change
N = No change
C = The result of or-ing the contents of the carry flag with those of bit num of src is set.

Execution example: ORCF 6, (100H)
When the contents of memory at address 100H = 01000000B (binary) and the carry flag = 0, execution sets the carry flag to 1.



PAA dst

< Pointer Adjust Accumulator >

Operation : if dst <LSB> = 1 then dst ← dst + 1

Description : Increments dst by 1 when the LSB of dst is 1. Does nothing when the LSB of dst is 0.
Used to make the contents of dst even. With the TLCS-900 series, when accessing 16- or 32-bit data in memory, if the data are loaded from an address starting with an even number, the number of bus cycles is 1 less than that of the data loaded from an address starting with an odd number.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
–	4	4	PAA	r	1 1 z z 1 r 0 0 0 1 0 1 0 0

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example: PAA XIZ
When the XIZ register = 00234567H, execution increments the XIZ register by 1 so that it becomes 00234568H.

POP dst

< Pop >

Operation : dst ← (XSP+)

In bytes	: dst←(XSP), XSP←XSP+1
In words	: dst←(XSP), XSP←XSP+2
In long words	: dst←(XSP), XSP←XSP+4

Description : First loads the contents of memory address specified by the stack pointer XSP to dst. Then increments the stack pointer XSP by the number of bytes in the operand.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
4	–	–	POP	F	0 0 0 1 1 0 0 1
4	–	–	POP	A	0 0 0 1 0 1 0 1
–	4	6	POP	R	0 1 0 s 1 R
6	6	8	POP	r	1 1 z z 1 r 0 0 0 0 0 1 0 1
6	6	–	POP<W>	(mem)	1 m 1 1 m m m m 0 0 0 0 0 1 z 0

Flags :

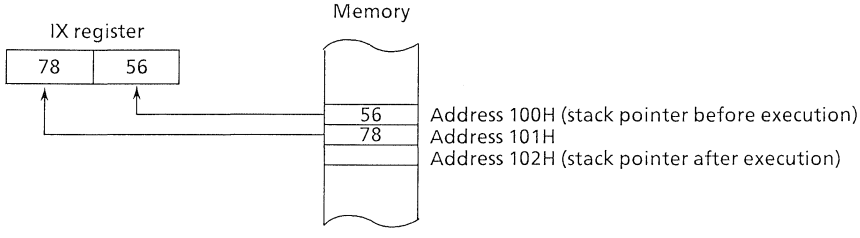
S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

(Note) Executing POP F changes all flags.

Execution example: POP IX

When the stack pointer XSP = 0100H, the contents of address 100H = 56H, and the contents of address 101H = 78H, execution sets the IX register to 7856H and the stack pointer XSP to 0102H.



POP SR

< Pop SR >
(Privileged instruction)

Operation : SR←(XSP+)

Description : Loads the contents of the address specified by the stack pointer XSP to status register. Then increments the contents of the stack pointer XSP by 2.

Details :

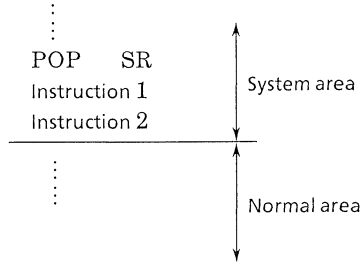
State			Mnemonic		Code
Byte	Word	Long word			
–	6	–	POP	SR	0 0 0 0 0 0 1 1

Flags :

S	Z	H	V	N	C
*	*	*	*	*	*

S =
Z =
H =
V =
N =
C =
} Contents of the memory address specified by the stack pointer XSP are set.

Note : The timing for executing this instruction is delayed by several states than that for fetching the instruction. This is because an instruction queue (4 bytes) and pipeline processing method is used.
Note, when changing from the system area to the normal area using this instruction, that the access area of the instruction code immediately after this instruction is used as the access area before execution of this instruction. The figure below is an example.



PUSH SR

<Push SR>

(Privileged instruction)

Operation : (−XSP)←SR

Description : Decrements the contents of the stack pointer XSP by 2. Then loads the contents of status register to the memory address specified by the stack pointer XSP.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
−	4	−	PUSH	SR	0 0 0 0 0 0 1 0

Flags :

S	Z	H	V	N	C
−	−	−	−	−	−

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

PUSH src

< Push >

Operation : (−XSP) ← src

In bytes	: XSP←XSP−1, (XSP)←src
In words	: XSP←XSP−2, (XSP)←src
In long words	: XSP←XSP−4, (XSP)←src

Description : Decrements the stack pointer XSP by the byte length of the operand. Then loads the contents of src to the memory address specified by the stack pointer XSP.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
3	–	–	PUSH	F	0 0 0 1 1 0 0 0
3	–	–	PUSH	A	0 0 0 1 0 1 0 0
–	3	5	PUSH	R	0 0 1 s 1 R
5	5	7	PUSH	r	1 1 z z 1 r 0 0 0 0 0 1 0 0
4	5	–	PUSH<W>	#	0 0 0 0 1 0 z 1 #<7:0> #<15:8>
7	7	–	PUSH<W>	(mem)	1 m 0 z m m m m 0 0 0 0 0 1 0 0

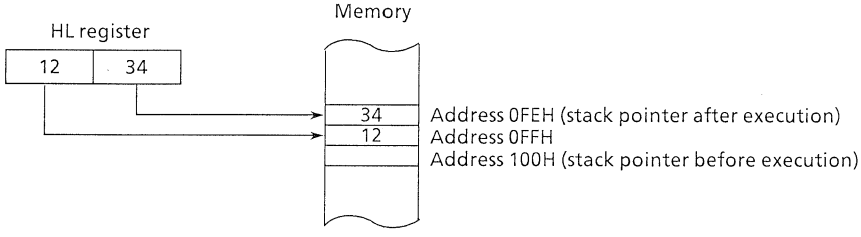
Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example: PUSH HL

When the stack pointer XSP = 0100H and the HL register = 1234H, execution changes address 00FEH to 34H, address 00FFH to 12H, and sets the stack pointer XSP to 00FEH.



RCF

< Reset Carry Flag >

Operation : CY ← 0

Description : Resets the carry flag to 0.

Details :

State	Mnemonic	Code
2	RCF	0 0 0 1 0 0 0 0

Flags :

S	Z	H	V	N	C
–	–	0	–	0	0

S = No change
Z = No change
H = Reset to 0.
V = Reset to 0.
N = No change
C = Reset to 0.

RES num, dst

< Reset >

Operation : dst <num> ← 0

Description : Resets bit num of dst to 0.

Details :

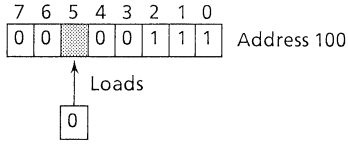
State			Mnemonic		Code
Byte	Word	Long word			
4	4	–	RES	#4, r	1 1 0 z 1 r 0 0 1 1 0 0 0 0 0 0 0 0 # 4
8	–	–	RES	#3, (mem)	1 m 1 1 m m m m 1 0 1 1 0 #3

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example: RES 5, (100H)
When the contents of memory at address 100H = 00100111B (binary), execution sets the contents to 00000111B (binary).



RET condition

< Return >

Operation : In minimum mode : If cc is true, then the 16-bit PC ← (XSP), XSP ← XSP + 2.
In maximum mode : If cc is true, then the 32-bit PC ← (XSP), XSP ← XSP + 4.

Description : Pops the return address from the stack area to the program counter when the operand condition is true.

Details :

State	Mnemonic	Code
9 (minimum mode) 11 (maximum mode)	RET	0 0 0 0 1 1 1 0
12 (cc is true, in minimum mode) 14 (cc is true, in maximum mode) 6 (cc is false)	RET cc	1 0 1 1 0 0 0 0 1 1 1 1 c c

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example: RET
When the stack pointer XSP = 0FEH and the contents of memory at address 0FEH = 9000H (word data) in minimum mode, execution sets the stack pointer XSP to 100H and jumps (returns) to address 9000H.

RETD num

< Return and Deallocate >

Operation : In minimum mode : 16-bit PC ← (XSP), XSP ← XSP+2, XSP ← XSP+num
In maximum mode : 32-bit PC ← (XSP), XSP ← XSP+4, XSP ← XSP+num

Description : Pops the return address from the stack area to the program counter. Then increments the stack pointer XSP by signed num.

Details :

State	Mnemonic		Code
9 (minimum mode) 11 (maximum mode)	RETD	d16	0 0 0 0 1 1 1 1 d<7:0> d<15:8>

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example: RETD 8

When the stack pointer XSP = 0FEH and the contents of memory at address 0FEH = 9000H (word data) in minimum mode, execution sets the stack pointer XSP to 0FEH + 2 + 8 → 108H and jumps (returns) to address 9000H.

Usage of the RETD instruction is shown below. In this example, the 8-bit parameter is pushed to the stack before the subroutine call. After the subroutine processing complete, the used parameter area is deleted by the RETD instruction.

```
PUSH  WA
PUSH  BC
PUSH  XIX
CALL  SAMPLE
  :
  :
```

```
SAMPLE :
  :
  :
  :
RETD  8
```

RETI

< Return from Interrupt >
(Privileged instruction)

Operation : In minimum mode : 2-byte Temp ← (XSP), 16-bit PC ← (XSP + 2),
SR ← Temp, XSP ← XSP + 4
In maximum mode : 2-byte Temp ← (XSP), 32-bit PC ← (XSP + 2),
SR ← Temp, XSP ← XSP + 6

Description : Pops data from the stack area to the 2-byte Temp register and program counter. Next, loads the contents of the Temp register to status register.

Note : The reason that data is not popped to status register directly from the stack area is to avoid changing the mode from normal to system while reading the stack area where the RETI instruction is being executed.

Details :

State	Mnemonic	Code
12 (minimum mode) 14 (maximum mode)	RETI	0 0 0 0 0 1 1 1

Flags :

S	Z	H	V	N	C
*	*	*	*	*	*

S = The value popped from the stack area is set.
Z = The value popped from the stack area is set.
H = The value popped from the stack area is set.
V = The value popped from the stack area is set.
N = The value popped from the stack area is set.
C = The value popped from the stack area is set.

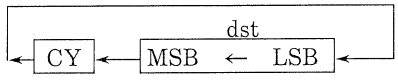
RL num, dst

< Rotate Left >

Operation : {CY & dst ← left rotates the value of CY & dst} Repeat num

Description : Rotates left the contents of the linked carry flag and dst.
Repeats the number of times specified in num.

Description figure:



Details :

State			Mnemonic		Code
Byte	Word	Long word			
6	6	8	RL	#4, r	1 1 z z 1 r 1 1 1 0 1 0 1 0 0 0 0 0 # 4
6	6	8	RL	A, r	1 1 z z 1 r 1 1 1 1 1 0 1 0
8	8	–	RL <W>	(mem)	1 m 0 z m m m m 0 1 1 1 1 0 1 0
+2n	+2n	+2n			

Note : When the number of rotates is specified by the A register, the value of the lower 4 bits of the A register is used. Specifying 0 rotates 16 times.
When dst is memory, rotating is performed only once.

Flags :

S	Z	H	V	N	C
*	*	0	*	0	*

S = MSB value of dst after rotate is set.
Z = 1 is set when the contents of dst after rotate is 0, otherwise 0.
H = Reset to 0.
V = 1 is set when the parity (number of 1s) of dst is even after rotate, otherwise 0. If the operand is 32 bits, an undefined value is set.
N = Reset to 0.
C = The value after rotate is set.

Execution example: RL 4, HL
When the HL register = 6230H and the carry flag = 1, execution sets the HL register to 230BH and the carry flag to 0.

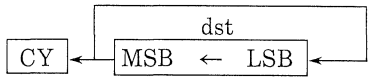
RLC num, dst

< Rotate Left without Carry >

Operation : {CY← dst <MSB>, dst← left rotate value of dst} Repeat num

Description : Loads the contents of the MSB of dst to the carry flag and rotates left the contents of dst. Repeats the number of times specified in num.

Description figure :



Details :

State Byte	State Word	State Long word	Mnemonic		Code
6 +2n	6 +2n	8 +2n	RLC	#4, r	1 1 z z 1 r 1 1 1 0 1 0 0 0 0 0 0 0 # 4
6 +2n	6 +2n	8 +2n	RLC	A, r	1 1 z z 1 r 1 1 1 1 1 0 0 0
8	8	–	RLC <W>	(mem)	1 m 0 z m m m m 0 1 1 1 1 0 0 0

Note : When the number of rotates is specified by the A register, the value of the lower 4 bits of the A register is used. Specifying 0 rotates 16 times.
When dst is memory, rotating is performed only once.

Flags :

S	Z	H	V	N	C
*	*	0	*	0	*

S = MSB value of dst after rotate is set.
Z = 1 is set when the contents of dst after rotate is 0, otherwise, 0.
H = Reset to 0.
V = 1 is set when the parity (number of 1s) of dst is even after rotate. If the operand is 32 bits, an undefined value is set.
N = Reset to 0.
C = MSB value of dst before the last rotate is set.

Execution example: RLC 4, HL
When the HL register = 1230H, execution sets the HL register to 2301H and the carry flag to 1.

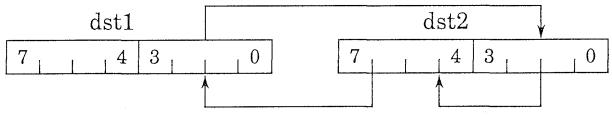
RLD dst1, dst2

< Rotate Left Digit >

Operation : dst1<3:0> ← dst2<7:4>, dst2<7:4> ← dst2<3:0>,
dst2<3:0> ← dst1 <3:0>

Description : Rotates left the lower 4 bits of dst1 and the contents of dst2 in units of 4 bits.

Description figure :



Details :

State			Mnemonic		Code
Byte	Word	Long word			
12	–	–	RLD	[A,] (mem)	1 m 0 0 m m m m 0 0 0 0 0 1 1 0

Flags :

S	Z	H	V	N	C
*	*	0	*	0	–

S = MSB value of the A register after rotate is set.
Z = 1 is set when the contents of the A register after the rotate are 0, otherwise 0.
H = Reset to 0.
V = 1 is set when the parity (number of 1s) of the A register is even after the rotate, otherwise 0.
N = Reset to 0.
C = No change

Execution example: RLD A, (100H)
When the A register = 12H and the contents of memory at address 100H = 34H, execution sets the A register to 13H and the contents of memory at address 100H to 42H.

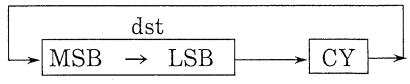
RR num, dst

< Rotate Right >

Operation : {CY & dst ← right rotates the value of CY & dst} Repeat num

Description : Rotates right the linked contents of the carry flag and dst.
Repeats the number of times specified in num.

Description figure:



Details :

State			Mnemonic		Code
Byte	Word	Long word			
6 +2n	6 +2n	8 +2n	RR	#4, r	1 1 z z 1 r 1 1 1 0 1 0 1 1 0 0 0 0 # 4
6 +2n	6 +2n	8 +2n	RR	A, r	1 1 z z 1 r 1 1 1 1 1 0 1 1
8	8	–	RR <W>	(mem)	1 m 0 z m m m m 0 1 1 1 1 0 1 1

Note : When the number of rotates is specified by the A register, the value of the lower 4 bits of the A register is used. Specifying 0 rotates 16 times.
When dst is memory, rotating is performed only once.

Flags :

S	Z	H	V	N	C
*	*	0	*	0	*

S = MSB value of dst after rotate is set.
Z = 1 is set when the contents of dst after rotate is 0, otherwise 0.
H = Reset to 0.
V = 1 is set when the parity (number of 1s) of dst is even after the rotate, otherwise 0. If the operand is 32 bits, an undefined value is set.
N = Reset to 0.
C = The value after rotate is set.

Execution example: RR 4, HL
When the HL register = 6230H and the carry flag = 1, execution sets the HL register to 1623H and the carry flag to 0.

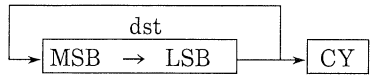
RRC num, dst

< Rotate Right without Carry >

Operation : {CY ← dst <LSB>, dst ← right rotate value of dst} Repeat num

Description : Loads the contents of the LSB of dst to the carry flag and rotates the contents of dst to the right. Repeats the number of times specified in num.

Description figure:



Details :

State Byte	State Word	State Long word	Mnemonic		Code
6 +2n	6 +2n	8 +2n	RRC	#4, r	1 1 z z 1 r 1 1 1 0 1 0 0 1 0 0 0 0 #4
6 +2n	6 +2n	8 +2n	RRC	A, r	1 1 z z 1 r 1 1 1 1 1 0 0 1
8	8	–	RRC <W>	(mem)	1 m 0 z m m m m 0 1 1 1 1 0 0 1

Note : When the number of rotates num is specified by the A register, the value of the lower 4 bits of the A register is used as the number of rotates.
Specifying 0 rotates 16 times. When dst is memory, rotating is only once.

Flags :

S	Z	H	V	N	C
*	*	0	*	0	*

S = MSB value of dst after rotate is set.
Z = 1 is set when the contents of dst after rotate is 0, otherwise 0.
H = Reset to 0.
V = 1 is set when the parity (number of 1s) of dst is even after rotate, otherwise 0. If the operand is 32 bits, an undefined value is set.
N = Reset to 0.
C = MSB value of dst before the last rotate is set.

Execution example: RLC 4, HL
When the HL register = 1230H, execution sets the HL register to 0123H and the carry flag to 0.

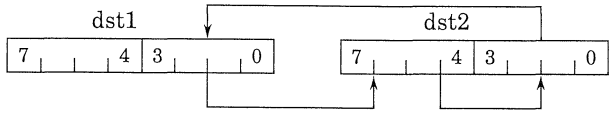
RRD dst1, dst2

< Rotate Right Digit >

Operation : dst1<3:0>←dst2<3:0>, dst2<7:4>←dst1<3:0>,
dst2<3:0>←dst2<7:4>

Description : Rotates right the lower 4 bits of dst1 and the contents of dst2 in units of 4 bits.

Description figure :



Details :

State			Mnemonic		Code
Byte	Word	Long word			
12	–	–	RRD	[A,] (mem)	1 m 0 0 m m m m 0 0 0 0 0 1 1 1

Flags :

S	Z	H	V	N	C
*	*	0	*	0	–

S = MSB value of the A register after rotate is set.
Z = 1 is set when the contents of the A register after rotate is 0, otherwise 0.
H = Reset to 0.
V = 1 is set when the parity (number of 1s) of the A register is even after rotate, otherwise 0.
N = Reset to 0.
C = No change

Execution example: RRD A, (100H)
When the A register = 12H and the contents of memory at address 100H = 34H, execution sets the A register to 14H and the contents of memory at address 100H to 23H.

SBC dst, src

< Subtract with Carry >

Operation : dst ← dst − src − CY

Description : Subtracts the contents of src and the carry flag from those of dst, and loads the result to dst.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
4	4	7	SBC	R, r	1 1 z z 1 r 1 0 1 1 0 R
4	4	7	SBC	r, #	1 1 z z 1 r 1 1 0 0 1 0 1 1 #<7:0> #<15:8> #<23:16> #<31:24>
4	4	6	SBC	R, (mem)	1 m z z m m m m 1 0 1 1 0 R
6	6	10	SBC	(mem), R	1 m z z m m m m 1 0 1 1 1 R
7	8	–	SBC<W>	(mem), #	1 m 0 z m m m m 0 0 1 1 1 0 1 1 #<7:0> #<15:8>

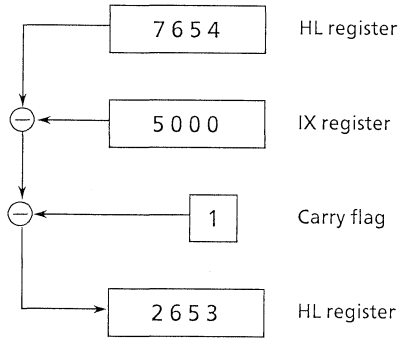
Flags : S Z H V N C

S	Z	H	V	N	C
*	*	*	*	1	*

S = MSB value of the result is set.
Z = 1 is set when the result is 0, otherwise 0.
H = 1 is set when a borrow from bit 3 to bit 4 occurs as a result, otherwise 0. When the operand is 32 bits, an undefined value is set.
V = 1 is set when an overflow occurs as a result, otherwise 0.
N = 1 is set.
C = 1 is set when a borrow from the MSB occurs as a result, otherwise 0.

Execution example: SBC HL, IX

When the HL register is 7654H, the IX register = 5000H, and the carry flag = 1, execution sets the HL register to 2653H.



SCC condition, dst

< Set Condition Code >

Operation : If cc is true, then dst ← 1 else dst ← 0.

Description : Loads 1 to dst when the operand condition is true; when false, 0 is loaded to dst.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
6	6	–	SCC	cc, r	1 1 0 z 1 r 0 1 1 1 c c

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example: SCC OV, HL
When the contents of the V flag = 1, execution sets the HL register to 0001H.

SCF

< Set Carry Flag >

Operation : CY ← 1

Description : Sets the carry flag to 1.

Details :

State	Mnemonic	Code
2	SCF	0 0 0 1 0 0 0 1

Flags :

S	Z	H	V	N	C
–	–	0	–	0	1

S = No change
Z = No change
H = Reset to 0.
V = No change
N = Reset to 0.
C = Set to 1.

SET num, dst

< Set >

Operation : dst <num> ← 1

Description : Sets bit num of dst to 1.

Details :

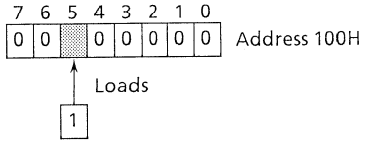
State			Mnemonic		Code
Byte	Word	Long word			
4	4	–	SET	#4, r	1 1 0 z 1 r 0 0 1 1 0 0 0 1 0 0 0 0 #4
8	–	–	SET	#3, (mem)	1 m 1 1 m m m m 1 0 1 1 1 #3

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example: SET 5, (100H)
When the contents of memory at address 100H = 00000000B (binary), execution sets the contents of memory at address 100H to 00100000B (binary).



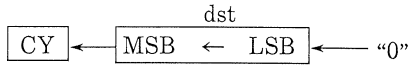
SLA num, dst

< Shift Left Arithmetic >

Operation : {CY ← dst<MSB>, dst ← left shift value of dst, dst<LSB> ← 0} Repeat num

Description : Loads the contents of the MSB of dst to the carry flag, shifts left the contents of dst, and loads 0 to the LSB of dst. Repeats the number of times specified in num.

Description chart:



Details :

State			Mnemonic		Code
Byte	Word	Long word			
6 +2n	6 +2n	8 +2n	SLA	#4, r	1 1 z z 1 r 1 1 1 0 1 1 0 0 0 0 0 0 # 4
6 +2n	6 +2n	8 +2n	SLA	A, r	1 1 z z 1 r 1 1 1 1 1 1 0 0
8	8	–	SLA <W>	(mem)	1 m 0 z m m m m 0 1 1 1 1 1 0 0

Note : When the number of shifts, num, is specified by the A register, the value of the lower 4 bits of the A register is used. Specifying 0 shifts 16 times. When dst is memory, shifting is performed only once.

Flags :

S	Z	H	V	N	C
*	*	0	*	0	*

S = MSB value of dst after shift is set.
Z = 1 is set when the contents of dst after shift is 0, otherwise 0.
H = Reset to 0.
V = 1 is set when the parity (number of 1s) of dst is even after shifting, otherwise 0. If the operand is 32 bits, an undefined value is set.
N = Reset to 0.
C = MSB value of dst before the last shift is set.

Execution example: SLA 4, HL
When the HL register = 1234H, execution sets the HL register to 2340H and the carry flag to 1.

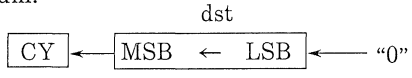
SLL num, dst

< Shift Left Logical >

Operation : {CY ← dst<MSB>, dst ← left shift value of dst, dst<LSB> ← 0} Repeat num

Description : Loads the contents of the MSB of dst to the carry flag, shifts left the contents of dst, and loads 0 to the MSB of dst. Repeats the number of times specified in num.

Description chart:



Details :

State			Mnemonic		Code
Byte	Word	Long word			
6 +2n	6 +2n	8 +2n	SLL	#4, r	1 1 z z 1 r 1 1 1 0 1 1 1 0 0 0 0 0 # 4
6 +2n	6 +2n	8 +2n	SLL	A, r	1 1 z z 1 r 1 1 1 1 1 1 1 0
8	8	–	SLL <W>	(mem)	1 m 0 z m m m m 0 1 1 1 1 1 1 0

Note : When the number of shifts, num, is specified by the A register, the value of the lower 4 bits of the A register is used. Specifying 0 shifts 16 times. When dst is memory, shifting is performed only once.

Flags :

S	Z	H	V	N	C
*	*	0	*	0	*

S = MSB value of dst after shift is set.
Z = 1 is set when the contents of dst after shift is 0, otherwise 0.
H = Reset to 0.
V = 1 is set when the parity (number of 1s) of dst is even after shifting, otherwise 0. If the operand is 32 bits, an undefined value is set.
N = Reset to 0.
C = MSB value of dst before the last shift is set.

Execution example: SLL 4, HL
When the HL register = 1234H, execution sets the HL register to 2340H and the carry flag to 1.

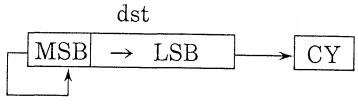
SRA num, dst

< Shift Right Arithmetic >

Operation : {CY ← dst<MSB>, dst ← right shift value of dst, dst <MSB> is fixed} Repeat num

Description : Loads the contents of the LSB of dst to the carry flag and shifts right the contents of dst (MSB is fixed). Repeats the number of times specified in num.

Description chart:



Details :

State Byte	State Word	State Long word	Mnemonic		Code
6 +2n	6 +2n	8 +2n	SRA	#4, r	1 1 z z 1 r 1 1 1 0 1 1 0 1 0 0 0 0 # 4
6 +2n	6 +2n	8 +2n	SRA	A, r	1 1 z z 1 r 1 1 1 1 1 1 0 1
8	8	–	SRA <W>	(mem)	1 m 0 z m m m m 0 1 1 1 1 1 0 1

Note : When the number of shifts, num, is specified by the A register, the value of the lower 4 bits of the A register is used. Specifying 0 shifts 16 times. When dst is memory, shifting is performed only once.

Flags :

S	Z	H	V	N	C
*	*	0	*	0	*

S = MSB value of dst after shift is set.
Z = 1 is set when the contents of dst after shift is 0, otherwise 0.
H = Reset to 0.
V = 1 is set when the parity (number of 1s) of dst is even after shift, otherwise 0. If the operand is 32 bits, an undefined value is set.
N = Reset to 0.
C = LSB value of dst before the last shift is set.

Execution example: SRA 4, HL
When the HL register = 8230H, execution sets the HL register to F823H and the carry flag to 0.

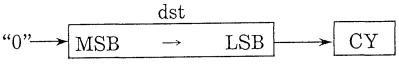
SRL num, dst

< Shift Right Logical >

Operation : {CY ← dst<LSB>, dst ← right shift value of dst, dst <MSB> ← 0} Repeat num

Description : Loads the contents of the LSB of dst to the carry flag, shifts right the contents of dst, and loads 0 to the MSB of dst. Repeats the number of times specified in num.

Description chart :



Details :

State Byte	State Word	State Long word	Mnemonic		Code
6 +2n	6 +2n	8 +2n	SRL	#4, r	1 1 z z 1 r 1 1 1 0 1 1 1 1 0 0 0 0 #4
6 +2n	6 +2n	8 +2n	SRL	A, r	1 1 z z 1 r 1 1 1 1 1 1 1 1
8	8	–	SRL<W>	(mem)	1 m 0 z m m m m 0 1 1 1 1 1 1 1

Note : When the number of shifts, num, is specified by the A register, the value of the lower 4 bits of the A register is used. Specifying 0 shifts 16 times. When dst is memory, shifting is performed only once.

Flags :

S	Z	H	V	N	C
*	*	0	*	0	*

S = MSB value of dst after shift is set.
Z = 1 is set when the contents of dst after shift is 0, otherwise 0.
H = Reset to 0.
V = 1 is set when the parity (number of 1s) of dst is even after shift, otherwise 0. If the operand is 32 bits, an undefined value is set.
N = Reset to 0.
C = LSB value of dst before the last shift is set.

Execution example: SRL 4, HL
When the HL register = 1238H, execution sets the HL register to 0123H and the carry flag to 1.

STCF num, dst

< Store Carry Flag >

Operation : dst<num>←CY

Description : Loads the contents of the carry flag to bit num of dst.

Details :

State Byte	State Word	State Long word	Mnemonic		Code
4	4	–	STCF	#4, r	1 1 0 z 1 r 0 0 1 0 0 1 0 0 0 0 0 0 #4
4	4	–	STCF	A, r	1 1 0 z 1 r 0 0 1 0 1 1 0 0
8	–	–	STCF	#3, (mem)	1 m 1 1 m m m m 1 0 1 0 0 #3
8	–	–	STCF	A, (mem)	1 m 1 1 m m m m 0 0 1 0 1 1 0 0

Note : When bit num is specified by the A register, the value of the lower 4 bits of the A register is used. When the operand is a byte and the value of the lower 4 bits of bit num is from 8 to 15, the operand value does not change.

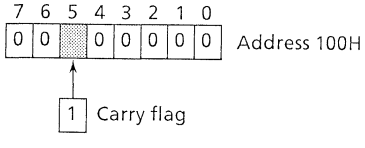
Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example: STCF 5, (100H)

When the contents of memory at address 100H = 00H and the carry flag = 1, execution sets the contents of memory at address 100H to 00100000B (binary).



SUB dst, src

< Subtract >

Operation : dst←dst−src

Description : Subtracts the contents of src from those of dst and loads the result to dst.

Details :

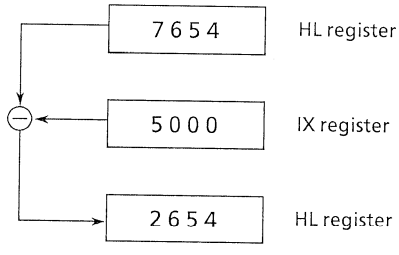
State			Mnemonic		Code
Byte	Word	Long word			
4	4	7	SUB	R, r	1 1 z z 1 r 1 0 1 0 0 R
4	4	7	SUB	r, #	1 1 z z 1 r 1 1 0 0 1 0 1 0 #<7:0> #<15:8> #<23:16> #<31:24>
4	4	6	SUB	R, (mem)	1 m z z m m m m 1 0 1 0 0 R
6	6	10	SUB	(mem), R	1 m z z m m m m 1 0 1 0 1 R
7	8	–	SUB<W>	(mem), #	1 m 0 z m m m m 0 0 1 1 1 0 1 0 #<7:0> #<15:8>

Flags : S Z H V N C

S	Z	H	V	N	C
*	*	*	*	1	*

S = MSB value of the result is set.
Z = 1 is set when the result is 0, otherwise 0.
H = 1 is set when a borrow from bit 3 to bit 4 occurs as a result, otherwise 0. When the operand is 32 bits, an undefined value is set.
V = 1 is set when an overflow occurs as a result, otherwise 0.
N = 1 is set.
C = 1 is set when a borrow from MSB occurs as a result, otherwise 0.

Execution example: SUB HL, IX
When the HL register = 7654H and the IX register = 5000H, execution sets the HL register to 2654H.



SWI num

< Software Interrupt >

Operation : In minimum mode: Temp ← SR, SYSMbit ← 1, XSP←XSP−4,
(XSP) ←Temp, (XSP+2) ←16-bit PC, PC←8000H+num×10H
In maximum mode: Temp ← SR, SYSMbit ← 1 , XSP←XSP−6,
(XSP) ←Temp, (XSP+2) ← 32-bit PC, PC←8000H+num×10H

Description : Sets SYSM bit to 1 to enter system mode. Saves to the stack area the contents of the status register before execution of the SWI instruction and contents of the program counter which indicate the address next to the SWI instruction. Finally, jumps to address 8000H+num×10H.

Details :

State	Mnemonic		Code
16 (Minimum mode)	SWI	[#3]	1 1 1 1 1 #3
18 (Maximum mode)			

Note 1 : A value from 0 to 7 can be specified as the operand value. When the operand coding is omitted, SWI 7 is assumed.

Note 2 : The status register structure is as shown below.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SYSM	IFF2	IFF1	IFF0	MAX	RFP2	RFP1	RFP0	S	Z	"0"	H	"0"	V	N	C

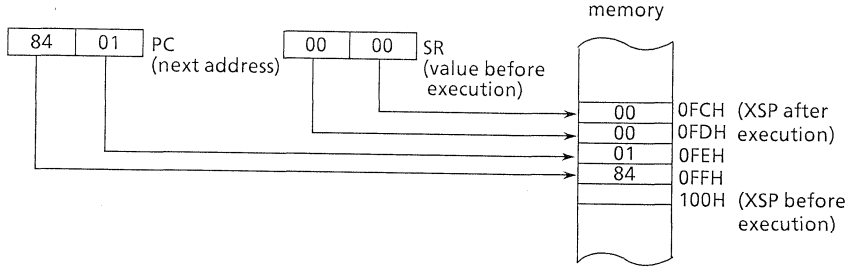
Flags :

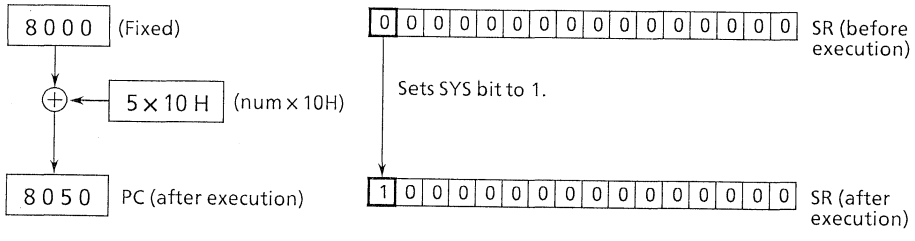
S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example: SWI 5

In minimum mode, when the stack pointer XSP = 100H, the status register = 0000H, executing the above instruction at memory address 8400H sets the status register to 8000H, writes the contents of the previous status register 0000H in memory address 00FCH, and the contents of the program counter 8401H in memory address 00FEH, then jumps to address 8050H.





UNLK dst

< Unlink >

Operation : XSP ← dst, dst ← (XSP+)

Description : Loads the contents of dst to the stack pointer XSP, then pops long word data from the stack area to dst. Used paired with the Link instruction.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
–	–	8	UNLK	r	1 1 1 0 1 r 0 0 0 0 1 1 0 1

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example: UNLK XIZ
As a result of executing this instruction after executing the Link instruction, the stack pointer XSP and the XIZ register revert to the same values they had before the Link instruction was executed. (For details of the Link instruction, see page 104)

XOR dst, src

< Exclusive OR >

Operation : dst←dst XOR src

Description : Exclusive ors the contents of dst with those of src and loads the result to dst.

(Truth table)

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Details :

State Byte	State Word	State Long word	Mnemonic		Code
4	4	7	XOR	R, r	1 1 z z 1 r 1 1 0 1 0 R
4	4	7	XOR	r, #	1 1 z z 1 r 1 1 0 0 1 1 0 1 #<7:0> #<15:8> #<23:16> #<31:24>
4	4	6	XOR	R, (mem)	1 m z z m m m m 1 1 0 1 0 R
6	6	10	XOR	(mem), R	1 m z z m m m m 1 1 0 1 1 R
7	8	–	XOR<W>	(mem), #	1 m 0 z m m m m 0 0 1 1 1 1 0 1 #<7:0> #<15:8>

Flags : S Z H V N C

S	Z	H	V	N	C
*	*	0	*	0	0

S = MSB value of the result is set.
Z = 1 is set when the result is 0, otherwise 0.
H = Reset to 0.
V = 1 is set when the parity (number of 1s) of dst is even as a result, otherwise 0. If the operand is 32 bits, an undefined value is set.
N = Cleared to 0.
C = Cleared to 0.

Execution example: XOR HL, IX
When the HL register = 7350H and the IX register = 3456H, execution sets the HL register to 4706H.

```
        0111  0011  0101  0000  ←  HL register (before execution)
XOR)    0011  0100  0101  0110  ←  IX register (before execution)
        0100  0111  0000  0110  ←  HL register (after execution)
```

XORCF num, src

< Exclusive OR Carry Flag >

Operation : CY ← CY XOR src<num>

Description : Exclusive ors the contents of the carry flag and bit num of src, and loads the result to the carry flag.

Details :

State			Mnemonic		Code
Byte	Word	Long word			
4	4	–	XORCF	#4, r	1 1 0 z 1 r 0 0 1 0 0 0 1 0 0 0 0 0 #4
4	4	–	XORCF	A, r	1 1 0 z 1 r 0 0 1 0 1 0 1 0
8	–	–	XORCF	#3, (mem)	1 m 1 1 m m m m 1 0 0 1 0 #3
8	–	–	XORCF	A, (mem)	1 m 1 1 m m m m 0 0 1 0 1 0 1 0

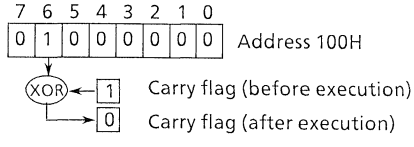
Note : When bit num is specified by the A register, the value of the lower 4 bits of the A register is used. When the operand is a byte and the value of the lower 4 bits of bit num is from 8 to 15, the result is undefined.

Flags :

S	Z	H	V	N	C
–	–	–	–	–	*

S = No change
Z = No change
H = No change
V = No change
N = No change
C = The value obtained by exclusive or-ing the contents of the carry flag with those of bit num of src is set.

Execution example: XORCF 6, (100H)
When the contents of memory at address 100H = 01000000B (binary) and the carry flag = 1, execution sets the carry flag to 0.



ZCF

< Zero flag to Carry Flag >

Operation : CY ← inverted value of Z flag

Description : Loads the inverted value of the Z flag to the carry flag.

Details :

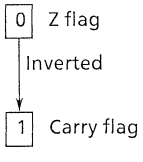
State	Mnemonic	Code
2	ZCF	0 0 0 1 0 0 1 1

Flags :

S	Z	H	V	N	C
–	–	×	–	0	*

S = No change
Z = No change
H = An undefined value is set.
V = No change
N = Reset to 0.
C = The inverted value of the Z flag is set.

Execution example: ZCF
When the Z flag = 0, execution sets the carry flag to 1.



Appendix B Instruction Lists (1/9)

(1) Load

Group	Size	Mnemonic	Codes (hex.)	Function	SZHVNC	Length (byte)	State
LD	BWL	LD R,r	C8+zz+r :88+R	R ← r	------	2	4. 4. 4
	BWL	LD r,R	C8+zz+r :98+R	r ← R	------	2	4. 4. 4
	BWL	LD r,#3	C8+zz+r :A8+#3	r ← #3	------	2	4. 4. 4
	BWL	LD R,#	20+zz+R :#	R ← #	------	1+#	2. 3. 5
	BWL	LD r,#	C8+zz+r :03:#	r ← #	------	2+#	4. 4. 6
	BWL	LD R,(mem)	80+zz+mem:20+R	R ← (mem)	------	2+M	4. 4. 6
	BWL	LD (mem),R	B0+mem :40+zz+R	(mem) ← R	------	2+M	4. 4. 6
	BW-	LD<W> (#8) ,#	08+z :#8:#	(#8) ← #	------	2+#	5. 6. -
	BW-	LD<W> (mem),#	B0+mem :00+z:#	(mem) ← #	------	2+M#	5. 6. -
	BW-	LD<W> (#16),(mem)	80+zz+mem:19:#16	(#16) ← (mem)	------	4+M	8. 8. -
	BW-	LD<W> (mem),(#16)	B0+mem :14+z:#16	(mem) ← (#16)	------	4+M	8. 8. -
PUSH	B--	PUSH F	18	(-XSP) ← F	------	1	3. -. -
	B--	PUSH A	14	(-XSP) ← A	------	1	3. -. -
	-WL	PUSH R	18+zz+R	(-XSP) ← R	------	1	-. 3. 5
	BWL	PUSH r	C8+zz+r :04	(-XSP) ← r	------	2	5. 5. 7
	BW-	PUSH<W> #	09+z :#	(-XSP) ← #	------	1+#	4. 5. -
	BW-	PUSH<W> (mem)	80+zz+mem:04	(-XSP) ← (mem)	------	2+M	7. 7. -
POP	B--	POP F	19	F ← (XSP+)	******	1	4. -. -
	B--	POP A	15	A ← (XSP+)	------	1	4. -. -
	-WL	POP R	38+zz+R	R ← (XSP+)	------	1	-. 4. 6
	BWL	POP r	C8+zz+r :05	r ← (XSP+)	------	2	6. 6. 8
	BW-	POP<W> (mem)	B0+mem :04+z	(mem) ← (XSP+)	------	2+M	6. 6. -
LDA	-WL	LDA R,mem	B0+mem :10+zz+R	R ← mem	------	2+M	-. 4. 4
LDAR	-WL	LDAR R,$+4+d16	F3:13:d16:10+zz+R	R ← PC+d16	------	5	-.11.11

(2) Exchange

Group	Size	Mnemonic	Codes (hex.)	Function	SZHVNC	Length (byte)	State
EX	B--	EX F,F'	16	F ↔ F'	******	1	2. -. -
	BW-	EX R,r	C8+zz+r :B8+R	R ↔ r	------	2	5. 5. -
	BW-	EX (mem),R	80+zz+mem:30+R	(mem) ↔ R	------	2+M	6. 6. -
MIRR	-W-	MIRR r	D8+r :16	r<0:MSB>←r<MSB:0>	------	2	-. 4. -

Note 1: B, W, and L used in the above tables in the size column represent byte (8 bits), word (16 bits), and long word (32 bits).

Note 2: z and zz used in the code column represent the operand sizes: B/W/L = 0/2/4 and B/W/L = 00H/10H/20H.

Note 3: Symbols for calculating object code length in the instruction length column mean as follows:
+# = adds immediate data length
+M = adds addressing code length
M# = adds immediate data length and addressing code length
$ symbol represents the start address.

Appendix B Instruction Lists (2/9)

(3) Load Increment/Decrement & Compare Increment/Decrement Size

Group	Size	Mnemonic	Codes (hex.)	Function	SZHVNC	Length (byte)	State
LDxx	BW-	LDI<W> [(XDE+),(XHL+)]	83+zz :10	(XDE+) ← (XHL+) BC ← BC-1	--0M0-	2	10.10. -
	BW-	LDI<W> (XIX+),(XIY+)	85+zz :10	(XIX+) ← (XIY+) BC ← BC-1	--0M0-	2	10.10. -
	BW-	LDIR<W> [(XDE+),(XHL+)]	83+zz :11	repeat (XDE+) ← (XHL+) BC ← BC-1 until BC=0	--000-	2	10.10. - (end) 14.14. - (repeat)
	BW-	LDIR<W> (XIX+),(XIY+)	85+zz :11	repeat (XIX+) ← (XIY+) BC ← BC-1 until BC=0	--000-	2	10.10. - (end) 14.14. - (repeat)
	BW-	LDD<W> [(XDE-),(XHL-)]	83+zz :12	(XDE-) ← (XHL-) BC ← BC-1	--0M0-	2	10.10. -
	BW-	LDD<W> (XIX-),(XIY-)	85+zz :12	(XIX-) ← (XIY-) BC ← BC-1	--0M0-	2	10.10. -
	BW-	LDDR<W> [(XDE-),(XHL-)]	83+zz :13	repeat (XDE-) ← (XHL-) BC ← BC-1 until BC=0	--000-	2	10.10. - (end) 14.14. - (repeat)
	BW-	LDDR<W> (XIX-),(XIY-)	85+zz :13	repeat (XIX-) ← (XIY-) BC ← BC-1 until BC=0	--000-	2	10.10. - (end) 14.14. - (repeat)
CPxx	BW-	CPI [A/WA,(R+)]	80+zz+R :14	A/WA - (R+) BC ← BC-1	*N*M1-	2	8. 8. -
	BW-	CPIR [A/WA,(R+)]	80+zz+R :15	repeat A/WA - (R+) BC ← BC-1 until A/WA=(R) or BC=0	*N*M1-	2	10.10. - (end) 14.14. - (repeat)
	BW-	CPD [A/WA,(R-)]	80+zz+R :16	A/WA - (R-) BC ← BC-1	*N*M1-	2	8. 8. -
	BW-	CPDR [A/WA,(R-)]	80+zz+R :17	repeat A/WA - (R-) BC ← BC-1 until A/WA=(R) or BC=0	*N*M1-	2	10.10. - (end) 14.14. - (repeat)

Note 1: Flag M; If BC = 0 after execution, the P/V flag is set to 0, otherwise 1.
Flag N; If A/WA = (R), the Z flag is set to 1, otherwise, 0 is set.

Note 2: When the operand is omitted in the CPI, CPIR, CPD, or CPDR instruction, A,(XHL +/−) is used as the default value.

Appendix B Instruction Lists (3/9)

(4) Arithmetic Operations

Group	Size	Mnemonic	Codes (hex.)	Function	SZHVNC	Length (byte)	State
ADD	BWL	ADD R,r	C8+zz+r :80+R	R ← R + r	***V0*	2	4. 4. 7
	BWL	ADD r,#	C8+zz+r :C8:#	r ← r + #	***V0*	2+#	4. 4. 7
	BWL	ADD R,(mem)	80+zz+mem:80+R	R ← R + (mem)	***V0*	2+M	4. 4. 6
	BWL	ADD (mem),R	80+zz+mem:88+R	(mem) ← (mem) + R	***V0*	2+M	6. 6.10
	BW-	ADD<W> (mem),#	80+zz+mem:38:#	(mem) ← (mem) + #	***V0*	2+M#	7. 8. -
ADC	BWL	ADC R,r	C8+zz+r :90+R	R ← R + r + CY	***V0*	?	4. 4. 7
	BWL	ADC r,#	C8+zz+r :C9:#	r ← r + # + CY	***V0*	2+#	4. 4. 7
	BWL	ADC R,(mem)	80+zz+mem:90+R	R ← R+(mem)+CY	***V0*	2+M	4. 4. 6
	BWL	ADC (mem),R	80+zz+mem:98+R	(mem) ←(mem)+R+CY	***V0*	2+M	6. 6.10
	BW-	ADC<W> (mem),#	80+zz+mem:39:#	(mem) ←(mem)+#+CY	***V0*	2+M#	7. 8. -
SUB	BWL	SUB R,r	C8+zz+r :A0+R	R ← R - r	***V1*	2	4. 4. 7
	BWL	SUB r,#	C8+zz+r :CA:#	r ← r - #	***V1*	2+#	4. 4. 7
	BWL	SUB R,(mem)	80+zz+mem:A0+R	R ← R - (mem)	***V1*	2+M	4. 4. 6
	BWL	SUB (mem),R	80+zz+mem:A8+R	(mem) ← (mem) - R	***V1*	2+M	6. 6.10
	BW-	SUB<W> (mem),#	80+zz+mem:3A:#	(mem) ← (mem) - #	***V1*	2+M#	7. 8. -
SBC	BWL	SBC R,r	C8+zz+r :B0+R	R ← R - r - CY	***V1*	2	4. 4. 7
	BWL	SBC r,#	C8+zz+r :CB:#	r ← r - # - CY	***V1*	2+#	4. 4. 7
	BWL	SBC R,(mem)	80+zz+mem:B0+R	R ←R-(mem)-CY	***V1*	2+M	4. 4. 6
	BWL	SBC (mem),R	80+zz+mem:B8+R	(mem) ←(mem)-R-CY	***V1*	2+M	6. 6.10
	BW-	SBC<W> (mem),#	80+zz+mem:3B:#	(mem) ←(mem)-#-CY	***V1*	2+M#	7. 8. -
CP	BWL	CP R,r	C8+zz+r :F0+R	R - r	***V1*	2	4. 4. 7
	BW-	CP r,#3	C8+zz+r :D8+#3	r - #3	***V1*	2	4. 4. -
	BWL	CP r,#	C8+zz+r :CF:#	r - #	***V1*	2+#	4. 4. 7
	BWL	CP R,(mem)	80+zz+mem:F0+R	R - (mem)	***V1*	2+M	4. 4. 6
	BWL	CP (mem),R	80+zz+mem:F8+R	(mem) - R	***V1*	2+M	6. 6. 6
	BW-	CP<W> (mem),#	80+zz+mem:3F:#	(mem) - #	***V1*	2+M#	6. 6. -
INC	B--	INC #3,r	C8+r :60+#3	r ← r + #3	***V0-	2	4. -. -
	-WL	INC #3,r	C8+zz+r :60+#3	r ← r + #3	------	2	-. 4. 4
	BW-	INC<W> #3,(mem)	80+zz+mem:60+#3	(mem) ← (mem) + #3	***V0-	2+M	6. 6. -
DEC	B--	DEC #3,r	C8+r :68+#3	r ← r - #3	***V1-	2	4. -. -
	-WL	DEC #3,r	C8+zz+r :68+#3	r ← r - #3	------	2	-. 4. 5
	BW-	DEC<W> #3,(mem)	80+zz+mem:68+#3	(mem) ← (mem) - #3	***V1-	2+M	6. 6. -
NEG	BW-	NEG r	C8+zz+r :07	r ← 0 - r	***V1*	2	5. 5. -
EXTZ	-WL	EXTZ r	C8+zz+r :12	r<high> ← 0	------	2	-. 4. 4
EXTS	-WL	EXTS r	C8+zz+r :13	r<high> ← r<low. MSB>	------	2	-. 5. 5
DAA	B--	DAA r	C8+r :10	Decimal adjustment after addition or subtraction.	***P-*	2	6. -. -
PAA	-WL	PAA r	C8+zz+r :14	if r<0>=1 then INC r	------	2	-. 4. 4

Note 1: With the INC/DEC instruction, when the code value of #3 = 0, functions as +8/−8.

Note 2: When the ADD R, r (word type) instruction is used in the TLCS-90, the S, Z, and V flags do not change. In the TLCS-900, these flags change.

Appendix B Instruction Lists (4/9)

Group	Size	Mnemonic	Codes (hex.)	Function	SZHVNC	Length (byte)	State

MUL	BW-
	BW-
	BW-
		MUL RR,r
		MUL rr,#
		MUL RR,(mem)
			C8+zz+r :40+R
			C8+zz+r :08:#
			80+zz+mem:40+R
				RR ← R×r
				rr ← r×#
				RR ← R×(mem)
					------ 	
					------ 	
					------ 	
						2
						2+#
						2+M
							18.26. -
							18.26. -
							18.26. -

MULS	BW-
	BW-
	BW-
		MULS RR,r
		MULS rr,#
		MULS RR,(mem)
			C8+zz+r :48+R
			C8+zz+r :09:#
			80+zz+mem:48+R
				RR ← R×r ;signed
				rr ← r×# ;signed
				RR ←R×(mem);signed
					------ 	
					------ 	
					------ 	
						2
						2+#
						2+M
							18.26. -
							18.26. -
							18.26. -

DIV	BW- BW- BW-	DIV RR,r DIV rr,# DIV RR,(mem)	C8+zz+r :50+R C8+zz+r :0A:# 80+zz+mem:50+R	R ← RR÷r r ← rr÷# R ← RR÷(mem)	---V-- ---V-- ---V--	2 2+# 2+M	22.30. - 22.30. - 22.30. -
DIVS	BW- BW- BW-	DIVS RR,r DIVS rr,# DIVS RR,(mem)	C8+zz+r :58+R C8+zz+r :0B:# 80+zz+mem:58+R	R ← RR÷r ;signed r ← rr÷# ;signed R ← RR÷(mem);signed	---V-- ---V-- ---V--	2 2+# 2+M	24.32. - 24.32. - 24.32. -
MULA	-W-	MULA rr	D8+r :19	Multiply and add signed rr ← rr+(XDE)×(XHL) 32bit 32bit 16bit 16bit XHL ← XHL-2	**-V--	2	-.31. -
MINC	-W-	MINC1 #,r (#=2**n) (1<=n<=15)	D8+r :38:#-1	Modulo increment;+1 if (r mod #)=(#-1) then r←r-(#-1) else r←r+1	------	4	-. 8. -
	-W-	MINC2 #,r (#=2**n) (2<=n<=15)	D8+r :39:#-2	Modulo increment;+2 if (r mod #)=(#-2) then r←r-(#-2) else r←r+2	------	4	-. 8. -
	-W-	MINC4 #,r (#=2**n) (3<=n<=15)	D8+r :3A:#-4	Modulo increment;+4 if (r mod #)=(#-4) then r←r-(#-4) else r←r+4	------	4	-. 8. -
MDEC	-W-	MDEC1 #,r (#=2**n) (1<=n<=15)	D8+r :3C:#-1	Modulo decrement;-1 if (r mod #)=0 then r←r+(#-1) else r←r-1	------	4	-. 7. -
	-W-	MDEC2 #,r (#=2**n) (2<=n<=15)	D8+r :3D:#-2	Modulo decrement;-2 if (r mod #)=0 then r←r+(#-2) else r←r-2	------	4	-. 7. -
	-W-	MDEC4 #,r (#=2**n) (3<=n<=15)	D8+r :3E:#-4	Modulo decrement;-4 if (r mod #)=0 then r←r+(#-4) else r←r-4	------	4	-. 7. -

Note: Operand RR of the MUL, MULS, DIV, and DIVS instructions indicates that a register twice the size of the operation is specified. When the operation is in bytes (8 bits x 8 bits, 16/8 bits), word register (16 bits) is specified; when the operation is in words (16 bits x 16 bits, 32/16 bits), long word register (32 bits) is specified.

Appendix B Instruction Lists (5/9)

(5) Logical operations

Group	Size	Mnemonic	Codes (hex.)	Function	SZHVNC	Length (byte)	State
AND	BWL	AND R,r	C8+zz+r :C0+R	R ← R and r	**1P00	2	4. 4. 7
	BWL	AND r,#	C8+zz+r :CC:#	r ← r and #	**1P00	2+#	4. 4. 7
	BWL	AND R,(mem)	80+zz+mem:C0+R	R ← R and (mem)	**1P00	2+M	4. 4. 6
	BWL	AND (mem),R	80+zz+mem:C8+R	(mem) ← (mem) and R	**1P00	2+M	6. 6. 10
	BW-	AND<w> (mem),#	80+zz+mem:3C:#	(mem) ← (mem) and #	**1P00	2+M#	7. 8. -
OR	BWL	OR R,r	C8+zz+r :E0+R	R ← R or r	**0P00	2	4. 4. 7
	BWL	OR r,#	C8+zz+r :CE:#	r ← r or #	**0P00	2+#	4. 4. 7
	BWL	OR R,(mem)	80+zz+mem:E0+R	R ← R or (mem)	**0P00	2+M	4. 4. 6
	BWL	OR (mem),R	80+zz+mem:E8+R	(mem) ← (mem) or R	**0P00	2+M	6. 6. 10
	BW-	OR<W> (mem),#	80+zz+mem:3E:#	(mem) ← (mem) or #	**0P00	2+M#	7. 8. -
XOR	BWL	XOR R,r	C8+zz+r :D0+R	R ← R xor r	**0P00	2	4. 4. 7
	BWL	XOR r,#	C8+zz+r :CD:#	r ← r xor #	**0P00	2+#	4. 4. 7
	BWL	XOR R,(mem)	80+zz+mem:D0+R	R ← R xor (mem)	**0P00	2+M	4. 4. 6
	BWL	XOR (mem),R	80+zz+mem:D8+R	(mem) ← (mem) xor R	**0P00	2+M	6. 6. 10
	BW-	XOR<W> (mem),#	80+zz+mem:3D:#	(mem) ← (mem) xor #	**0P00	2+M#	7. 8. -
CPL	BW-	CPL r	C8+zz+r :06	r ← not r	--1-1-	2	4. 4. -

Appendix B Instruction Lists (6/9)

(6) Bit operations

Group	Size	Mnemonic	Codes (hex.)	Function	SZHVNC	Length (byte)	State
LDCF	BW-	LDCF #4,r	C8+zz+r :23:#4	CY ← r<#4>	-----*	3	4.4.-
	BW-	LDCF A ,r	C8+zz+r :2B	CY ← r<A>	-----*	2	4.4.-
	B--	LDCF #3,(mem)	B0+men :98+#3	CY ← (mem)<#3>	-----*	2+M	8.-.-
	B--	LDCF A ,(mem)	B0+men :2B	CY ← (mem)<A>	-----*	2+M	8.-.-
STCF	BW-	STCF #4,r	C8+zz+r :24:#4	r<#4> ← CY	------	3	4.4.-
	BW-	STCF A ,r	C8+zz+r :2C	r<A> ← CY	------	2	4.4.-
	B--	STCF #3,(mem)	B0+mem :A0+#3	(mem)<#3> ← CY	------	2+M	8.-.-
	B--	STCF A ,(mem)	B0+mem :2C	(mem)<A> ← CY	------	2+M	8.-.-
ANDCF	BW-	ANDCF #4,r	C8+zz+r :20:#4	CY ← CY and r<#4>	-----*	3	4.4.-
	BW-	ANDCF A ,r	C8+zz+r :28	CY ← CY and r<A>	-----*	2	4.4.-
	B--	ANDCF #3,(mem)	B0+mem :80+#3	CY ← CY and (mem)<#3>	-----*	2+M	8.-.-
	B--	ANDCF A ,(mem)	B0+mem :28	CY ← CY and (mem)<A>	-----*	2+M	8.-.-
ORCF	BW-	ORCF #4,r	C8+zz+r :21:#4	CY ← CY or r<#4>	-----*	3	4.4.-
	BW-	ORCF A ,r	C8+zz+r :29	CY ← CY or r<A>	-----*	2	4.4.-
	B--	ORCF #3,(mem)	B0+mem :88+#3	CY ← CY or (mem)<#3>	-----*	2+M	8.-.-
	B--	ORCF A ,(mem)	B0+mem :29	CY ← CY or (mem)<A>	-----*	2+M	8.-.-
XORCF	BW-	XORCF #4,r	C8+zz+r :22:#4	CY ← CY xor r<#4>	-----*	3	4.4.-
	BW-	XORCF A ,r	C8+zz+r :2A	CY ← CY xor r<A>	-----*	2	4.4.-
	B--	XORCF #3,(mem)	B0+mem :90+#3	CY ← CY xor (mem)<#3>	-----*	2+M	8.-.-
	B--	XORCF A ,(mem)	B0+mem :2A	CY ← CY xor (mem)<A>	-----*	2+M	8.-.-
RCF	---	RCF	10	CY ← 0	--0-00	1	2
SCF	---	SCF	11	CY ← 1	--0-01	1	2
CCF	---	CCF	12	CY ← not CY	--X-0*	1	2
ZCF	---	ZCF	13	CY ← not Z flag	--X-0*	1	2
BIT	BW-	BIT #4,r	C8+zz+r :33:#4	Z ← not r<#4>	X*1X0-	3	4.4.-
	B--	BIT #3,(mem)	B0+mem :C8+#3	Z ← not (mem)<#3>	X*1X0-	2+M	8.-.-
RES	BW-	RES #4,r	C8+zz+r :30:#4	r<#4> ← 0	------	3	4.4.-
	B--	RES #3,(mem)	B0+mem :B0+#3	(mem)<#3> ← 0	------	2+M	8.-.-
SET	BW-	SET #4,r	C8+zz+r :31:#4	r<#4> ← 1	------	3	4.4.-
	B--	SET #3,(mem)	B0+mem :B8+#3	(mem)<#3> ← 1	------	2+M	8.-.-
CHG	BW-	CHG #4,r	C8+zz+r :32:#4	r<#4> ← not r<#4>	------	3	4.4.-
	B--	CHG #3,(mem)	B0+mem :C0+#3	(mem)<#3>←not (mem)<#3>	------	2+M	8.-.-
TSET	BW-	TSET #4,r	C8+zz+r :34:#4	Z←not r<#4> : r<#4>←1	X*1X0-	3	6.6.-
	B--	TSET #3,(mem)	B0+mem :A8+#3	Z ← not (mem)<#3> (mem)<#3> ← 1	X*1X0-	2+M	10.-.-
BS1	-W-	BS1F A,r	D8+r :0E	A ← 1 search r;Forward	---N--	2	-.4.-
	-W-	BS1B A,r	D8+r :0F	A ← 1 search r;Backward	---N--	2	-.4.-

Note: Flag N ;0 is set when the bit searched for is found, otherwise 1 is set and an undefined value is set in the A register.

Appendix B Instruction Lists (7/9)

(7) Special operations and CPU control

Group	Size	Mnemonic	Codes (hex.)	Function	SZHVNC	Length (byte)	State
NOP	---	NOP	00	no operation	------	1	2
NORMAL	---	NORMAL	01	Changes to normal mode. SYSM←0 [privileged]	------	1	4
MAX	---	MAX	04	Changes to maximum mode. MAX←1 [privileged]	------	1	4
EI	---	EI [#3]	06 :#3	Sets interrupt enable flag. IFF←#3 [privileged]	------	2	5
DI	---	DI	06 :07	Disables interrupt. IFF←7 [privileged]	------	2	5
PUSH	-W-	PUSH SR	02	(-XSP)←SR [privileged]	------	1	-.4.-
POP	-W-	POP SR	03	SR←(XSP+) [privileged]	******	1	-.6.-
SWI	---	SWI [#3]	F8+#3	Software interrupt PUSH PC&SR JP 8000H+10H×#3	------	1	16
HALT	---	HALT	05	CPU halt [privileged]	------	1	8
LDC	BWL BWL	LDC cr,r LDC r,cr	C8+zz+r :2E:cr C8+zz+r :2F:cr	cr ← r [privileged] r ← cr [privileged]	------ ------	3 3	8.8.8 8.8.8
LDX	B--	LDX (#8),#	F7:00:#8:00:#:00	(#8) ← #	------	6	9.-.-

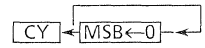
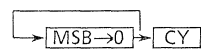
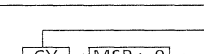
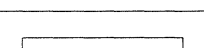
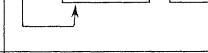
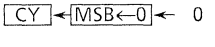
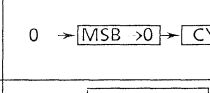
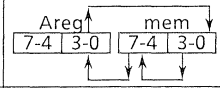
LINK	--L	LINK r,d16	E8+r :0C:d16	PUSH r
				LD r,XSP
				ADD XSP,d16
					------	4	-.-.10

UNLK	--L	UNLK r	E8+r :0D	LD XSP,r POP r	------	2	-.-.8
LDF	---	LDF #3	17 :#3	Sets register bank. RFP ← #3 (0 at reset)	------	2	2
INCF	---	INCF	0C	Switches register banks. RFP ← RFP + 1	------	1	2
DECF	---	DECF	0D	Switches register banks. RFP ← RFP - 1	------	1	2
SCC	BW-	SCC cc,r	C8+zz+r :70+cc	if cc then r ← 1 else r ← 0	------	2	6.-6.-

Note 1: When operand #3 coding in the EI instruction is omitted, 0 is used as the default value.
Note 2: When operand #3 coding in the SWI instruction is omitted, 7 is used as the default value.
Note 3: The value in the state column for the SWI instruction represents the number of states when the CPU is in minimum mode. In maximum mode, add + 2.

Appendix B Instruction Lists (8/9)

(8) Rotate and shift

Group	Size	Mnemonic	Codes (hex.)	Function	SZHVNC	Length (byte)	State
RLC	BWL	RLC #4,r	C8+zz+r :E8:#4		**0P0*	3	6.6.8+2n
	BWL	RLC A,r	C8+zz+r :F8		**0P0*	2	6.6.8+2n
	BW-	RLC<W> (mem)	80+zz+mem:78		**0P0*	2+M	8.8.-
RRC	BWL	RRC #4,r	C8+zz+r :E9:#4		**0P0*	3	6.6.8+2n
	BWL	RRC A,r	C8+zz+r :F9		**0P0*	2	6.6.8+2n
	BW-	RRC<W> (mem)	80+zz+mem:79		**0P0*	2+M	8.8.-
RL	BWL	RL #4,r	C8+zz+r :EA:#4		**0P0*	3	6.6.8+2n
	BWL	RL A,r	C8+zz+r :FA		**0P0*	2	6.6.8+2n
	BW-	RL<W> (mem)	80+zz+mem:7A		**0P0*	2+M	8.8.-
RR	BWL	RR #4,r	C8+zz+r :EB:#4		**0P0*	3	6.6.8+2n
	BWL	RR A,r	C8+zz+r :FB		**0P0*	2	6.6.8+2n
	BW-	RR<W> (mem)	80+zz+mem:7B		**0P0*	2+M	8.8.-
SLA	BWL	SLA #4,r	C8+zz+r :EC:#4		**0P0*	3	6.6.8+2n
	BWL	SLA A,r	C8+zz+r :FC		**0P0*	2	6.6.8+2n
	BW-	SLA<W> (mem)	80+zz+mem:7C		**0P0*	2+M	8.8.-
SRA	BWL	SRA #4,r	C8+zz+r :ED:#4		**0P0*	3	6.6.8+2n
	BWL	SRA A,r	C8+zz+r :FD		**0P0*	2	6.6.8+2n
	BW-	SRA<W> (mem)	80+zz+mem:7D		**0P0*	2+M	8.8.-
SLL	BWL	SLL #4,r	C8+zz+r :EE:#4		**0P0*	3	6.6.8+2n
	BWL	SLL A,r	C8+zz+r :FE		**0P0*	2	6.6.8+2n
	BW-	SLL<W> (mem)	80+zz+mem:7E		**0P0*	2+M	8.8.-
SRL	BWL	SRL #4,r	C8+zz+r :EF:#4		**0P0*	3	6.6.8+2n
	BWL	SRL A,r	C8+zz+r :FF		**0P0*	2	6.6.8+2n
	BW-	SRL<W> (mem)	80+zz+mem:7F		**0P0*	2+M	8.8.-
RLD	B--	RLD [A,](mem)	80+mem :06		**0P0-	2+M	12.-.-
RRD	B--	RRD [A,](mem)	80+mem :07		**0P0-	2+M	12.-.-

Note 1: When #4/A is used to specify the number of shifts, modulo 16 (0 to 15) is used. Code 0 means 16 shifts.

Note 2: When the following instructions are used in the TLCS-90, the S, Z and V flags do not change.
RLCA, RRCA, RLA, RRA, SLAA, SRAA, SLLA, and SRLA
In the TLCS-900, these flags change.

Appendix B Instruction Lists (9/9)

(9) Jump, call and return

Group	Size	Mnemonic	Codes (hex.)	Function	SZHVNC	Length (byte)	State
JP	---	JP #16	1A :#16	PC ← #16	------	3	7
	---	JP #24	1B :#24	PC ← #24	------	4	7
	---	JR [cc,]$+2+d8	60+cc :d8	if cc then PC ← PC+d8	------	2	8/4 (T/F)
	---	JRL [cc,]$+3+d16	70+cc :d16	if cc then PC ← PC+d16	------	3	8/4 (T/F)
	---	JP [cc,]mem	B0+mem :D0+cc	if cc then PC ← mem	------	2+M	9/6 (T/F)
CALL	---	CALL #16	1C :#16	PUSH PC : JP #16	------	3	12
	---	CALL #24	1D :#24	PUSH PC : JP #24	------	4	12
	---	CALR $+3+d16	1E :d16	PUSH PC : JR $+3+d16	------	3	12
	---	CALL [cc,]mem	B0+mem :E0+cc	if cc then PUSH PC : JP mem	------	2+M	12/6 (T/F)
DJNZ	BW-	DJNZ [r,]$+3+d8	C8+zz+r :1C:d8	r←r-1 if r≠0 then JR $+3+d8	------	3	11 (r≠0) 7 (r=0)
RET	---	RET	0E	POP PC	------	1	9
	---	RET cc	B0 :F0+cc	if cc then POP PC	------	2	12/6 (T/F)
	---	RETD d16	0F :d16	RET : ADD XSP,d16	------	3	9
	---	RETI	07	POP SR&PC [privileged]	******	1	12

Note 1: The value in the state column for the CALL, CALR, RET, RETD, and RETI instructions represents the number of states when the CPU is in minimum mode. In maximum mode, add + 2.

Note 2: (T/F) represents the number of states at true/false.

Appendix C Instruction Code Maps (1/4)

1-byte op code instructions

H/L	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	NORMAL	PUSH SR	POP SR	MAX	HALT	EI n	RETI	LD (n), n	PUSH n	LDW (n), nn	PUSHW nn	INCF	DECF	RET	RETD dd
1	RCF	SCF	CCF	ZCF	PUSH A	POP A	EX F, F'	LDF n	PUSH F	POP F	JP nn	JP nnn	CALL nn	CALL nnn	CALR PC + dd	
2	LD R, n								PUSH RR							
3	LD RR, nn								PUSH XRR							
4	LD XRR, nnnn								POP RR							
5									POP XRR							
6	F	LT	LE	ULE	PE/OV	M/MI	Z	JR C	cc,PC + d (T)	GE	GT	UGT	PO/NOV	P/PL	NZ	NC
7	F	LT	LE	ULE	PE/OV	M/MI	Z	JRL C	cc,PC + dd (T)	GE	GT	UGT	PO/NOV	P/PL	NZ	NC
8	scr. B (XWA)	(XBC)	(XDE)	(XHL)	(XIX)	(XIY)	(XIZ)	(XSP)	scr. B (XWA + d)	(XBC + d)	(XDE + d)	(XHL + d)	(XIX + d)	(XIY + d)	(XIZ + d)	(XSP + d)
9	scr. W (XWA)	(XBC)	(XDE)	(XHL)	(XIX)	(XIY)	(XIZ)	(XSP)	scr. W (XWA + d)	(XBC + d)	(XDE + d)	(XHL + d)	(XIX + d)	(XIY + d)	(XIZ + d)	(XSP + d)
A	scr. L (XWA)	(XBC)	(XDE)	(XHL)	(XIX)	(XIY)	(XIZ)	(XSP)	scr. L (XWA + d)	(XBC + d)	(XDE + d)	(XHL + d)	(XIX + d)	(XIY + d)	(XIZ + d)	(XSP + d)
B	dst (XWA)	(XBC)	(XDE)	(XHL)	(XIX)	(XIY)	(XIZ)	(XSP)	dst (XWA + d)	(XBC + d)	(XDE + d)	(XHL + d)	(XIX + d)	(XIY + d)	(XIZ + d)	(XSP + d)
C	scr. B (n)	(nn)	(nnn)	(mem)	(– xrr)	(xrr +)		reg. B r	reg. B W	A	B	C	D	E	H	L
D	scr. W (n)	(nn)	(nnn)	(mem)	(– xrr)	(xrr +)		reg. W rr	reg. W WA	BC	DE	HL	IX	IY	IZ	SP
E	scr. L (n)	(nn)	(nnn)	(mem)	(– xrr)	(xrr +)		reg. L xrr	reg. L XWA	XBC	XDE	XHL	XIX	XIY	XIZ	XSP
F	dst (n)	(nn)	(nnn)	(mem)	(– xrr)	(xrr +)		LDX (n), n	SWI n 0	1	2	3	4	5	6	7

Note 1: Codes in shaded parts are privileged instructions.
Note 2: Codes in blank parts are undefined instructions (i.e., illegal instructions).

Appendix C Instruction Code Maps (2/4)

1st byte: reg

H/L	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0				LD r,#	PUSH r	POP r	CPL BW r	NEG BW r	MUL rr,#	MULS rr,#	DIV rr,#	DIVS BW rr,#	LINK L r, dd	LINK L r	BS1F A, r	BS1B W A, r
1	DAA B r		EXTZ WL r	EXTS WL r	PAA WL r		MIRR W r			MULA W r			DJNZ BW r, d			
2	ANDCF #, r	ORCF #, r	XORCF #, r	LDCF #, r	STCF BW #, r				ANDCF A, r	ORCF A, r	XORCF A, r	LDCF A, r	STCF BW A, r		LDC cr, r	LDC r, cr
3	RES #, r	SET #, r	CHG #, r	BIT #, r	TSET BW #, r				MINC1 #, r	MINC2 #, r	MINC4 W		MDEC1 #, r	MDEC2 #, r	MDEC4 W	
4				MUL R, r				BW				MULS R, r				BW
5				DIV R, r				BW				DIVS R, r				BW
6				INC #3, r								DEC #3, r				
	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7
7								SCC C	cc, r (T)							BW NC
	F	LT	LE	ULE	PE/OV	M/MI	Z			GE	GT	UGT	PO/NOV	P/PL	NZ	
8				ADD R, r								LD R, r				
9				ADC R, r								LD r, R				
A				SUB R, r								LD r, #3				
									0	1	2	3	4	5	6	7
B				SBC R, r								EX R, r				BW
C				AND R, r					ADD r, #	ADC r, #	SUB r, #	SBC r, #	AND r, #	XOR r, #	OR r, #	CP r, #
D				XOR R, r								CP r, #3				BW
									0	1	2	3	4	5	6	7
E				OR R, r					RLC #, r	RRC #, r	RL #, r	RR #, r	SLA #, r	SRA #, r	SLL #, r	SRL #, r
F				CP R, r					RLC A, r	RRC A, r	RL A, r	RR A, r	SLA A, r	SRA A, r	SLL A, r	SRL A, r

r : Register specified by the 1st byte code. (Any CPU registers can be specified.)
R : Register specified by the 2nd byte code. (Only eight current registers can be specified.)
B : Operand size is a byte.
W : Operand size is a word.
L : Operand size is a long word.

Note : Dummy instructions are assigned to codes 1AH, 1BH, 3BH, and 3FH. Do not use them.

Appendix C Instruction Code Maps (3/4)

1st byte: src (mem)

H/L	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0					PUSH BW (mem)		RLD A, (mem)	RLD B								
1	LDI	LDIR	LDD	LDDR BW	CPI	CPIR	CPD	CPDR BW		LD BW (nn), (m)						
2	LD R, (mem)															
3	EX (mem), R							BW	ADD	ADC	SUB	SBC (mem), #	AND	XOR	OR	CP BW
4	MUL R, (mem)							BW	MULS R, (mem)							BW
5	DIV R, (mem)							BW	DIVS R, (mem)							BW
6	INC #3, (mem)							BW	DEC #3, (mem)							BW
	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7
7									RLC	RRC	RL	RR (mem)	SLA	SRA	SLL	SRL BW
8	ADD R, (mem)								ADD (mem), R							
9	ADC R, (mem)								ADC (mem), R							
A	SUB R, (mem)								SUB (mem), R							
B	SBC R, (mem)								SBC (mem), R							
C	AND R, (mem)								AND (mem), R							
D	XOR R, (mem)								XOR (mem), R							
E	OR R, (mem)								OR (mem), R							
F	CP R, (mem)								CP (mem), R							

B : Operand size is a byte.
W : Operand size is a word.

Appendix C Instruction Code Maps (4/4)

1st byte: dst (mem)

H/L	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	LD B (m), #		LD W (m), #		POP B (mem)		POP W (mem)									
1										LD B (m), (nn)		LD W (m), (nn)				
2	LDA R, mem W								ANDCF A, (mem)	ORCF A, (mem)	XORCF A, (mem)	LDCF A, (mem)	STCF B A, (mem)			
3	LDA R, mem L															
4	LD (mem), R B															
5	LD (mem), R W															
6	LD (mem), R L															
7																
8	ANDCF #3, (mem) B								ORCF #3, (mem) B							
	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
9	XORCF #3, (mem) B								LDCF #3, (mem) B							
	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
A	STCF #3, (mem) B								TSET #3, (mem) B							
	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
B	RES #3, (mem) B								SET #3, (mem) B							
	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
C	CHG #3, (mem) B								BIT #3, (mem) B							
	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
D	JP cc, mem															
	F	LT	LE	ULE	PE/OV	M/MI	Z	C	(T)	GE	GT	UGT	PO/NOV	P/PL	NZ	NC
E	CALL cc, mem															
	F	LT	LE	ULE	PE/OV	M/MI	Z	C	(T)	GE	GT	UGT	PO/NOV	P/PL	NZ	NC
F	RET cc (1st byte code is B0H.)															
	F	LT	LE	ULE	PE/OV	M/MI	Z	C	(T)	GE	GT	UGT	PO/NOV	P/PL	NZ	NC

B : Operand size is a byte.
W : Operand size is a word.
L : Operand size is a long word.

Appendix D Differences between TLCS-90 and TLCS-900 Series

Item \ Series	TLCS-90	TLCS-900
CPU architecture	8-bit CPU	16-bit CPU
Built-in ROM/built-in RAM	8-bit data bus	16-bit data bus
Built-in I/O	8-bit data bus	8-bit data bus
External data bus	8-bit data bus	8-bit/16-bit data bus (can be mixed)
Program space (except devices with MMU)	64KB	16MB (linear)
Instruction set/instruction mnemonic	TLCS-90	TLCS-90 + α α = enhancement of 16-bit multiply / divide instructions and bit operation instruction. 32-bit load/operation instructions, C compiler instructions, register bank operation instructions, etc.
Instruction code (object code)	Unique to TLCS-90	Unique to TLCS-900 (Different from TLCS-90.)
Addressing mode	TLCS-90	TLCS-90 + α α = (– Reg), (Reg +), (Reg + disp16), (Reg + Reg16), (nnn)
General-purpose register	TLCS-90	TLCS-90 + α α = Uses as 32 bits and register bank, and adds a system stack pointer.
Flag (F)	S Z I H X V N C	S Z "0" H "0" V N C I flag is extended to IFF2 to 0 of status register.X flag is deleted.
Reset	PC ← 0000H (SP does not change.)	PC ← 8000H XSP ← 100H
Built-in ROM address Built-in RAM address Built-in I/O address Direct addressing area (n)	0000H~ ~FFxxH FFxxH~FFFFH FF00H~FFFFH	8000H~ 0080H~ 0000H~007FH 0000H~00FFH
Interrupt Interrupt start address Register to be saved Mask register Mask level	 0000H + (8 × V) PC & AF IFF 0~1	 8000H + (10H × V) PC & SR IFF2~0 0~7

Note : The TLCS-900 series is essentially the same as the TLCS-90 series but with a 16-bit CPU. Built-in I/Os are completely compatible with those of the TLCS-90.
However, six types of instructions used in the TLCS-90 series do not directly correspond with those used in the TLCS-900 series. Thus, when transfering programs designed for the TLCS-90 to the TLCS-900, replace them with equivalents as follows:

Instructions in TLCS-90 but not in TLCS-900	Equivalent instructions in TLCS-900
EXX	EX BC, BC' EX DE, DE' EX HL, HL'
EX AF, AF'	EX A, A' EX F, F'
PUSH AF	PUSH A PUSH F
POP AF	POP F POP A
INCX	(32-bit INC instruction)
DECX	(32-bit DEC instruction)

Some TLCS-900 series instructions, though basically the same as TLCS-90 instructions, have more functions and more specification items in their operands. They are listed below.

TLCS-90	TLCS-900
INC reg INC mem	INC imm3, reg INC imm3, mem
DEC reg DEC mem	DEC imm3, reg DEC imm3, mem
RLC reg RRC reg RL reg RR reg SLA reg SRA reg SLL reg SRL reg	RLC imm, reg RRC imm, reg RL imm, reg RR imm, reg SLA imm, reg SRA imm, reg SLL imm, reg SRL imm, reg

TOSHIBA

CHAPTER 2 TLCS-900 LSI DEVICES

TOSHIBA CORPORATION

CMOS 16-bit MICROCONTROLLERS

TMP96C141F/TMP96CM40F/TMP96PM40F

1. OUTLINE AND DEVICE CHARACTERISTICS

TMP96C141F/TMP96CM40F/TMP96PM40F are high-speed advanced 16-bit microcontrollers developed for controlling medium to large-scale equipment. The TMP96C141 does not have a ROM, the TMP96CM40F has a built-in ROM, and the TMP96PM40 has a built-in OTP. Otherwise, the devices function in the same way.

TMP96C141F/TMP96CM40F/TMP96PM40F are housed in an 80-pin flat package.

Device characteristics are as follows:

(1) Original 16-bit CPU
- TLCS-90 instruction mnemonic upward compatible.
- 16M-byte linear address space
- General-purpose registers and register bank system
- 16-bit multiplication / division and bit transfer/arithmetic instructions
- High-speed micro DMA : 4 channels (2 μs/2 bytes @16MHz)

(2) Minimum instruction execution time : 250ns @16 MHz (20 MHz version under development)

(3) Internal RAM : 1K byte

Internal ROM :

TMP96C141	None
TMP96CM40	32K-byte ROM
TMP96PM40	32K-byte OTP

(4) External memory expansion
- Can be expanded up to16M bytes (for both programs and data).
- Can mix 8- and 16-bit external data buses.
  ··· Dynamic data bus sizing

(5) 8-bit timers : 2 channels
(6) 8-bit PWM timers : 2 channels
(7) 16-bit timers : 2 channels
(8) Pattern generators : 4 bits, 2 channels
(9) Serial interface : 2 channels
(10) 10-bit A/D converter : 4 channels
(11) Watchdog timer
(12) Chip select/wait controller : 3 blocks
(13) Interrupt functions
- 3 CPU interrupts····· SWI instruction, priviledged violation, and Illegal instruction
- 14 internal interrupts
- 6 external interrupts

7-level priority can be set.

(14) I/O ports
65 pins for TMP96CM40/TMP96PM40 and 47 pins for TMP96C141

(15) Standby function : 3 halt modes (RUN, IDLE, STOP)

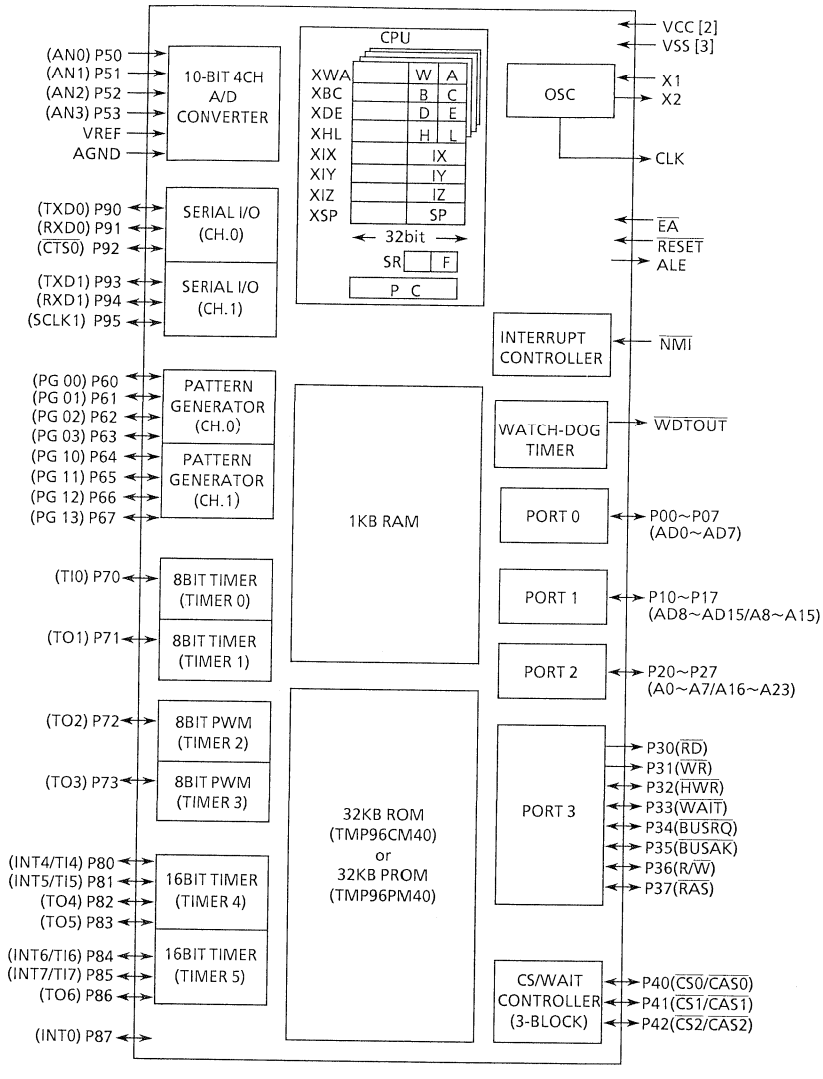


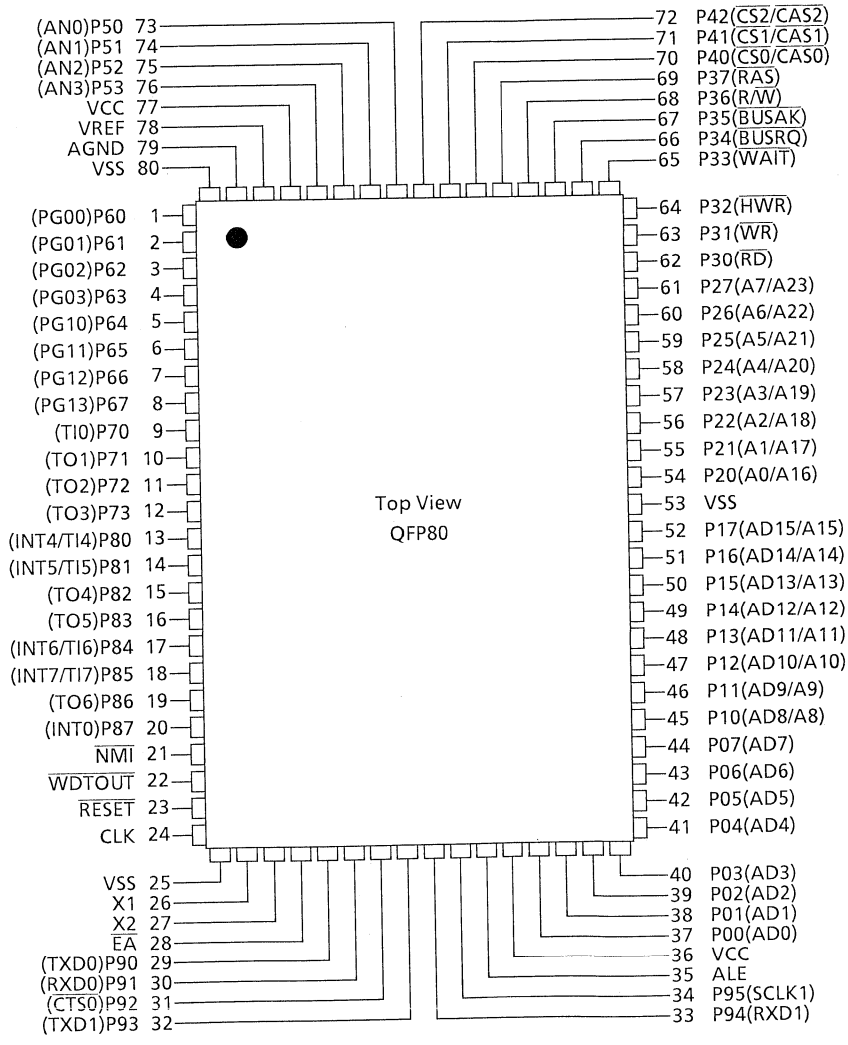
Figure1 TMP96C141/TMP96CM40/TMP96PM40 Block Diagram

2. PIN ASSIGNMENT AND FUNCTIONS

The assignment of input / output pins for TMP96C141/TMP96CM40/TMP96PM40, their name and outline functions are described below.

2.1 Pin Assignment

Figure 2.1 shows pin assignment of TMP96C141F/TMP96CM40F/TMP96PM40F.



Note : Because the TMP96C141 has an external ROM, P00 to P17 pins are fixed to AD0 to AD15; P30 to $\overline{RD}$; and P31 to $\overline{WR}$.

Figure 2.1 Pin Assignment (80-pin QFP)

2.2 Pin Names and Functions

The names of input/output pins and their functions are described below.

Table 2.2 Pin Names and Functions.

Pin name	Number of pins	I/O	Functions
P00~P07 AD0~AD7	8	I/O Tri-state	Port 0: I/O port that allows I/O to be selected on a bit basis Address/data (lower): 0 - 7 for address/data bus
P10~P17 AD8~AD15 A8~A15	8	I/O Tri-state Output	Port 1: I/O port that allows I/O to be selected on a bit basis Address data (upper): 8 - 15 for address/data bus Address: 8 to 15 for address bus
P20~P27 A0~A7 A16~A23	8	I/O Output Output	Port 2: I/O port that allows selection of I/O on a bit basis (with pull-down resistor) Address: 0 - 7 for address bus Address: 16 - 23 for address bus
P30 $\overline{RD}$	1	Output Output	Port 30: Output port Read: Strobe signal for reading external memory
P31 $\overline{WR}$	1	Output Output	Port 31: Output port Write: Strobe signal for writing data on pins AD0 - 7
P32 $\overline{HWR}$	1	I/O Output	Port 32: I/O port (with pull-up resistor) High write: Strobe signal for writing data on pins AD8 - 15
P33 $\overline{WAIT}$	1	I/O Input	Port 33: I/O port (with pull-up resistor) Wait: Pin used to request CPU bus wait
P34 $\overline{BUSRQ}$	1	I/O Input	Port34: I/O port (with pull-up resistor) Bus request: Signal used to request high impedance for AD0 - 15, A0 - 23, $\overline{RD}$, $\overline{WR}$,$\overline{HWR}$, R/$\overline{W}$, $\overline{RAS}$, $\overline{CS0}$, $\overline{CS1}$, and $\overline{CS2}$ pins. (For external DMAC)
P35 $\overline{BUSAK}$	1	I/O Output	Port 35: I/O port (with pull-up resistor) Bus acknowledge: Signal indicating that AD0-15, A0-23, $\overline{RD}$, $\overline{WR}$,$\overline{HWR}$, R/$\overline{W}$, $\overline{RAS}$, $\overline{CS0}$, $\overline{CS1}$, and $\overline{CS2}$ pins are at high impedance after receiving BUSRQ. (For external DMAC)
P36 R/$\overline{W}$	1	I/O Output	Port 36: I/O port (with pull-up resistor) Read/write: 1 represents read or dummy cycle; 0, write cycle.
P37 $\overline{RAS}$	1	I/O Output	Port 37: I/O port (with pull-up resistor) Row address strobe: Outputs RAS strobe for DRAM.
P40 $\overline{CS0}$ $\overline{CAS0}$	1	I/O Output Output	Port 40: I/O port (with pull-up resistor) Chip select 0: Outputs 0 when address is within specified address area. Column address strobe 0: Outputs CAS strobe for DRAM when address is within specified address area.

Note : With the external DMA controller, this device's built-in memory or built-in I/O cannot be accessed using the $\overline{BUSRQ}$ and $\overline{BUSAK}$ pins.

Pin name	Number of pins	I/O	Functions
P41 $\overline{CS1}$ $\overline{CAS1}$	1	I/O Output Output	Port 41: I/O port (with pull-up resistor) Chip select 1: Outputs 0 if address is within specified address area. Column address strobe 1: Outputs $\overline{CAS}$ strobe for DRAM if address is within specified address area.
P42 $\overline{CS2}$ $\overline{CAS2}$	1	I/O Output Output	Port 42: I/O port (with pull-down resistor) Chip select 2: Outputs 0 if address is within specified address area. Column address strobe 2: Outputs $\overline{CAS}$ strobe for DRAM if address is within specified address area.
P50~P53 AN0~AN3	4	Input Input	Port 5: Input port Analog input: Input to A/D converter
VREF	1	Input	Pin for reference voltage input to A/D converter
AGND	1	Input	Ground pin for A/D converter
P60~P63 PG00~PG03	4	I/O Output	Ports 60 - 63: I/O ports that allow selection of I/O on a bit basis (with pull-up resistor) Pattern generator ports: 00 - 03
P64~P67 PG10~PG13	4	I/O Output	Ports 64 - 67: I/O ports that allow selection of I/O on a bit basis (with pull-up resistor) Pattern generator ports: 10 - 13
P70 TI0	1	I/O Input	Port 70: I/O port (with pull-up resistor) Timer input 0: Timer 0 input
P71 TO1	1	I/O Output	Port 71: I/O port (with pull-up resistor) Timer output 1: Timer 0 or 1 output
P72 TO2	1	I/O Output	Port 72: I/O port (with pull-up resistor) PWM output 2: 8-bit PWM timer 2 output
P73 TO3	1	I/O Output	Port 73: I/O port (with pull-up resistor) PWM output 3: 8-bit PWM timer 3 output
P80 TI4 INT4	1	I/O Input Input	Port 80: I/O port (with pull-up resistor) Timer input 4: Timer 4 count/capture trigger signal input Interrupt request pin 4: Interrupt request pin with programmable rising/falling edge
P81 TI5 INT5	1	I/O Input Input	Port 81: I/O port (with pull-up resistor) Timer input 5: Timer 4 count/capture trigger signal input Interrupt request pin 5: Interrupt request pin with rising edge
P82 TO4	1	I/O Output	Port 82: I/O port (with pull-up resistor) Timer output 4: Timer 4 output pin
P83 TO5	1	I/O Output	Port 83: I/O port (with pull-up resistor) Timer output 5: Timer 4 output pin

Pin name	Number of pins	I/O	Functions
P84 TI6 INT6	1	I/O Input Input	Port 84: I/O port (with pull-up resistor) Timer input 6: Timer 5 count/capture trigger signal input Interrupt request pin 6: Interrupt request pin with programmable rising/falling edge
P85 TI7 INT7	1	I/O Input Input	Port 85: I/O port (with pull-up resistor) Timer input 7: Timer 5 count/capture trigger signal input Interrupt request pin 7: Interrupt request pin with rising edge
P86 TO6	1	I/O Output	Port 86: I/O port (with pull-up resistor) Timer output 6: Timer 5 output pin
P87 INT0	1	I/O Input	Port 87: I/O port (with pull-up resistor) Interrupt request pin 0: Interrupt request pin with programmable level/rising edge
P90 TXD0	1	I/O Output	Port 90: I/O port (with pull-up resistor) Serial send data 0
P91 RXD0	1	I/O Input	Port 91: I/O port (with pull-up resistor) Serial receive data 0
P92 $\overline{CTS0}$	1	I/O Input	Port 92: I/O port (with pull-up resistor) Serial data send enable 0 (Clear to Send)
P93 TXD1	1	I/O Output	Port 93: I/O port (with pull-up resistor) Serial send data 1
P94 RXD1	1	I/O Input	Port 94: I/O port (with pull-up resistor) Serial receive data 1
P95 SCLK1	1	I/O I/O	Port 95: I/O port (with pull-up resistor) Serial clock I/O 1
$\overline{WDTOUT}$	1	Output	Watchdog timer output pin
$\overline{NMI}$	1	Input	Non-maskable interrupt request pin: Interrupt request pin with falling edge. Can also be operated at rising edge by program.
CLK	1	Output	Clock output: Outputs ⌈ X1 ÷ 4 ⌋ clock. Pulled-up during reset.
$\overline{EA}$	1	Input	External access: 0 should be inputted with TMP96C141 1, with TMP96CM40/TMP96PM40.
ALE	1	Output	Address latch enable
$\overline{RESET}$	1	Input	Reset: Initializes LSI. (With pull-up resistor)
X1/X2	2	I/O	Oscillator connecting pin
VCC	2		Power supply pin (+ 5V)
VSS	3		GND pin (0V)

Note : Pull-up/pull-down resistor can be released from the pin by software.

3. OPERATION

This section describes in blocks the functions and basic operations of TMP96C141/TMP96CM40/TMP96PM40 devices.

3.1 CPU

TMP96C141/TMP96CM40/TMP96PM40 devices have a built-in high-performance 16-bit CPU. (For CPU operation, see TLCS-900 CPU in the previous section).

This section describes CPU functions unique to TMP96C141/TMP96CM40/TMP96PM40 that are not described in the previous section.

3.1.1 Reset

To reset the TMP96C141/TMP96CM40/TMP96PM40, the $\overline{\text{RESET}}$ input must be kept at 0 for at least 10 system clocks (10 states: 1μs with a 20 MHz system clock) within an operating voltage range and with a stable oscillation.

When reset is accepted, the CPU sets as follows:

- Program counter (PC) to 8000H.
- Stack pointer (XSP) for system mode to 100H.
- SYSM bit of status register (SR) to 1. (Sets to system mode.)
- IFF2 to 0 bits of status register to 111. (Sets mask register to interrupt level 7.)
- MAX bit of status register to 0. (Sets to minimum mode)
- Bits RFP2 to 0 of status register to 000. (Sets register banks to 0.)

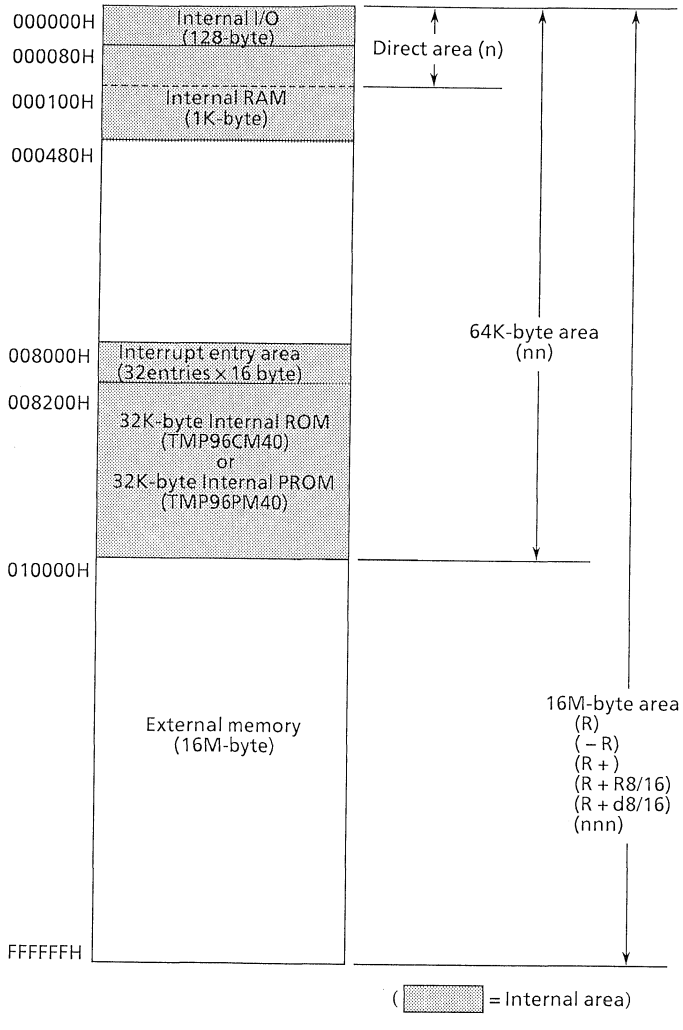
When reset is released, instruction execution starts from address 8000H. CPU internal registers other than the above are not changed.

When reset is accepted, processing for built-in I/Os, ports, and other pins is as follows

- Initializes built-in I/O registers as per specifications.
- Sets port pins (including pins also used as built-in I/Os) to general-purpose input/output port mode (sets I/O ports to input ports).
- Sets the $\overline{\text{WDTOUT}}$ pin to 0. (Watchdog timer is set to enable after reset.)
- Pulls up the CLK pin to 1.
- Sets the ALE pin to 0.

3.2 Memory Map

Figure 3.2 is a memory map of the TMP96C141/TMP96CM40/TMP96PM40.



Note : The start address after reset is 8000H. Resetting sets the stack pointer (XSP) on the system mode side to 100H.

Figure3.2 Memory map

3.3 Interrupts

TLCS-900 interrupts are controlled by the CPU interrupt mask flip-flop (IFF2 to 0) and the built-in interrupt controller.

TMP96C141F/TMP96CM40F/TMP96PM40F have altogether the following 23 interrupt sources:

- Interrupts from the CPU··· 3
  (Software interrupts, privileged violations, and Illegal (undefined) instruction execution)
- Interrupts from external pins (NMI, INT0, and INT4 to 7)··· 6
- Interrupts from built-in I/Os··· 14

A fixed individual interrupt vector number is assigned to each interrupt source; six levels of priority (variable) can also be assigned to each maskable interrupt. Non-maskable interrupts have a fixed priority of 7.

When an interrupt is generated, the interrupt controller sends the value of the priority of the interupt source to the CPU. When more than one interupt is generated simultaneously, the interrupt controller sends the value of the highest priority (7 for non-maskable interrupts is the highest) to the CPU.

The CPU compares the value of the priority sent with the value in the CPU interrupt mask register (IFF2 to 0). If the value is greater than that the CPU interrupt mask register, the interrupt is accepted. The value in the CPU interrupt mask register (IFF2 to 0) can be changed using the EI instruction (contents of the EI num/IFF<2:0> = num). For example, programming EI 3 enables acceptance of maskable interrupts with a priority of 3 or greater, and non-maskable interrupts which are set in the interrupt controller. The DI instruction (IFF<2:0> = 7) operates in the same way as the EI 7 instruction. Since the priority values for maskable interrupts are 0 to 6, the DI instruction is used to disable maskable interrupts to be accepted. The EI instruction becomes effective immediately after execution. (With the TLCS-90, the EI instruction becomes effective after execution of the subsequent instruction.)

In addition to the general-purpose interrupt processing mode described above, there is also a high-speed micro DMA processing mode . High-speed micro DMA is a mode used by the CPU to automatically transfer byte or word data. It enables the CPU to process interrupts such as data saves to built-in I/Os at high speed.

Figure 3.3 (1) is a flowchart showing overall interrupt processing.

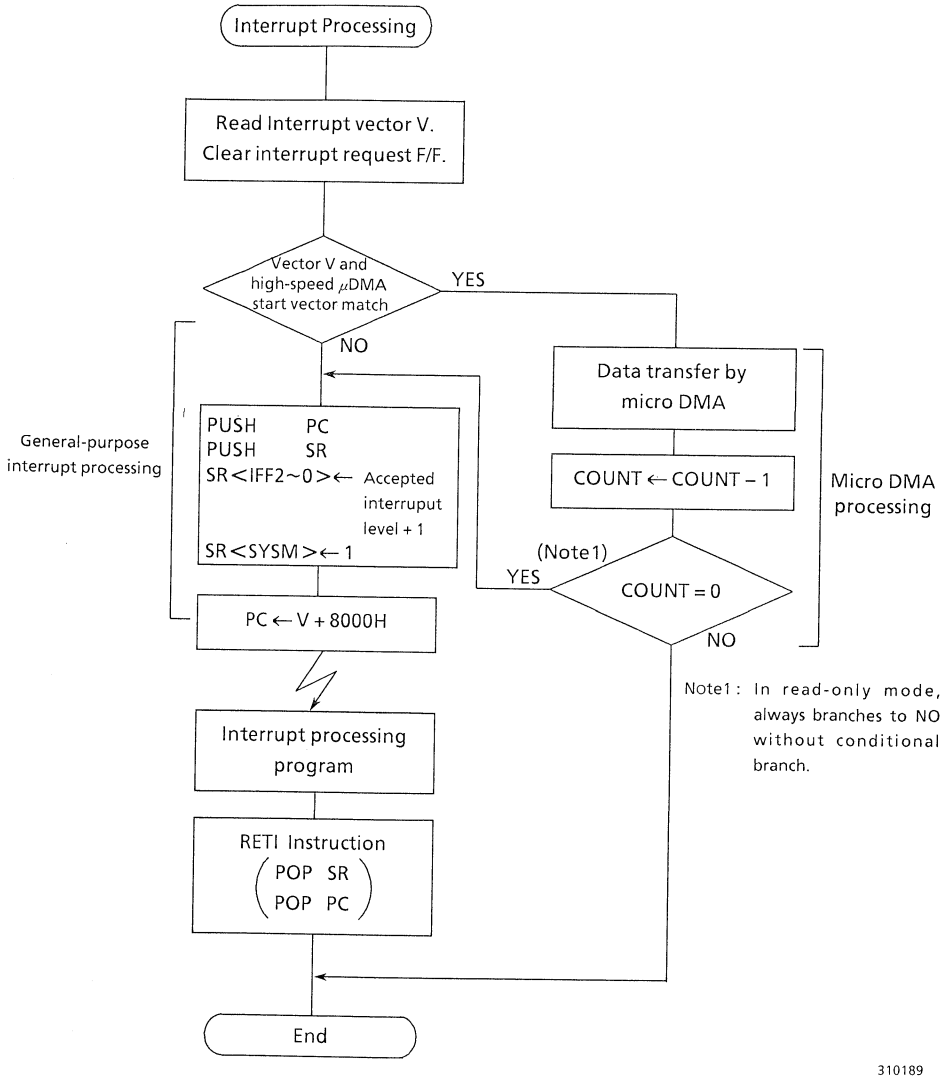


Figure3.3 (1) Interrupt Processing Flowchart

3.3.1 General-Purpose Interrupt Processing

When accepting an interrupt, the CPU operates as follows:

(1) The CPU reads the interrupt vector from the interrupt controller. When more than one interrupt with the same level is generated simultaneously, the interrupt controller generates interrupt vectors in accordance with the default priority (which is fixed as follows: the smaller the vector value, the higher the priority), then clears the interrupt request.

(2) The CPU pushes the program counter and the status register to the system stack area (area indicated by the system mode stack pointer).

(3) The CPU sets a value in the CPU interrupt mask register <IFF2 to 0> that is higher by 1 than the value of the accepted interrupt level. However, if the value is 7, 7 is set without an increment.

(4) The CPU sets the <SYSM> flag of the status register to 1 and enter the system mode.

(5) The CPU jumps to address 8000H + interrupt vector, then starts the interrupt processing routine.

In minimum mode, all the above processing is completed in 15 states (1.875 us @16 MHz). In maximum mode, it is completed in 17 states.

To return to the main routine after completion of the interrupt processing, the RETI instruction is usually used. Executing this instruction restores the contents of the program counter and the status registers.

Though acceptance of non-maskable interrupts cannot be disabled by program, acceptance of maskable interrupts can. A priority can be set for each source of maskable interrupts. The CPU accepts an interrupt request with a priority higher than the value in the CPU mask register <IFF2 to 0>. The CPU mask register <IFF2 to 0> is set to a value higher by 1 than the priority of the accepted interrupt. Thus, if an interrupt with a level higher than the interrupt being processed is generated, the CPU accepts the interrupt with the higher level, causing interrupt processing to nest. The CPU does not accept an interrupt request of the same level as that of the interrupt being processed.

Resetting initializes the CPU mask registers <IFF2 to 0> to 7; therefore, maskable interrupts are disabled.

The addresses 008000H to 0081FFH (512 bytes) of the TLCS-900 are assigned for interrupt processing entry area.

Table3.3 (1) TMP96C141/TMP96CM40/TMP96PM40 Interrupt Table

Default priority	Type	Interrupt source	Vector value "V"	Start address	High-speed micro DMA start vector
1	Non-maskable	Reset , or SWI0 instruction	0 0 0 0 H	8 0 0 0 H	–
2		INTPREV : Privileged violation, or SWI1	0 0 1 0 H	8 0 1 0 H	–
3		INTUNDEF : Illegal instruction, or SWI2	0 0 2 0 H	8 0 2 0 H	–
4		SWI 3 instruction	0 0 3 0 H	8 0 3 0 H	–
5		SWI 4 instruction	0 0 4 0 H	8 0 4 0 H	–
6		SWI 5 instruction	0 0 5 0 H	8 0 5 0 H	–
7		SWI 6 instruction	0 0 6 0 H	8 0 6 0 H	–
8		SWI 7 instruction	0 0 7 0 H	8 0 7 0 H	–
9		$\overline{\text{NMI}}$ Pin	0 0 8 0 H	8 0 8 0 H	08H
10		INTWD : Watchdog timer	0 0 9 0 H	8 0 9 0 H	09H
11	Maskable	INT0 pin	0 0 A 0 H	8 0 A 0 H	0AH
12		INT4 pin	0 0 B 0 H	8 0 B 0 H	0BH
13		INT5 pin	0 0 C 0 H	8 0 C 0 H	0CH
14		INT6 pin	0 0 D 0 H	8 0 D 0 H	0DH
15		INT7 pin	0 0 E 0 H	8 0 E 0 H	0EH
–		(Reserved)	0 0 F 0 H	8 0 F 0 H	0FH
16		INTT0 : 8-bit timer0	0 1 0 0 H	8 1 0 0 H	10H
17		INTT1 : 8-bit timer1	0 1 1 0 H	8 1 1 0 H	11H
18		INTT2 : 8-bit timer2 / PWM0	0 1 2 0 H	8 1 2 0 H	12H
19		INTT3 : 8-bit timer3 / PWM1	0 1 3 0 H	8 1 3 0 H	13H
20		INTTR4 : 16-bit timer4 (TREG4)	0 1 4 0 H	8 1 4 0 H	14H
21		INTTR5 : 16-bit timer4 (TREG5)	0 1 5 0 H	8 1 5 0 H	15H
22		INTTR6 : 16-bit timer5 (TREG6)	0 1 6 0 H	8 1 6 0 H	16H
23		INTTR7 : 16-bit timer5 (TREG7)	0 1 7 0 H	8 1 7 0 H	17H
24		INTRX0 : Serial receive (Channel.0)	0 1 8 0 H	8 1 8 0 H	18H
25		INTTX0 : Serial send (Channel.0)	0 1 9 0 H	8 1 9 0 H	19H
26		INTRX1 : Serial receive (Channel.1)	0 1 A 0 H	8 1 A 0 H	1AH
27		INTTX1 : Serial send (Channel.1)	0 1 B 0 H	8 1 B 0 H	1BH
28		INTAD : A/D conversion completion	0 1 C 0 H	8 1 C 0 H	1CH
–		(Reserved)	0 1 D 0 H	8 1 D 0 H	1DH
–		(Reserved)	0 1 E 0 H	8 1 E 0 H	1EH
–		(Reserved)	0 1 F 0 H	8 1 F 0 H	1FH

3.3.2 High-speed Micro DMA

In addition to the conventional interrupt processing, the TLCS-900 also has a high-speed micro DMA function. When an interrupt is accepted, in addition to an interrupt vector, the CPU receives data indicating whether processing is high-speed micro DMA mode or general-purpose interrupt. If high-speed micro DMA mode is requested, the CPU performs high-speed micro DMA processing.

The TLCS-900 can process at very high speed compared with the TLCS-90 micro DMA because it has transfer parameters in dedicated registers in the CPU. Since those dedicated registers are assigned as CPU control registers, they can only be accessed by the LDC (privileged) instruction.

(1) High-speed micro DMA operation

High-speed micro DMA operation starts when the accepted interrupt vector value matches the micro DMA start vector value set in the interrupt controller. The high-speed micro DMA has four channels so that it can be set for up to four types of interrupt source.

When a high-speed micro DMA interrupt is accepted, data is automatically transferred from the transfer source address to the transfer destination address set in the control register, and the transfer counter is decremented. If the value in the counter after decrementing is other than 0, high-speed micro DMA processing is completed; if the value in the counter after decrementing is 0, general-purpose interrupt processing is performed. In read-only mode, which is provided for DRAM refresh, the value in the counter is ignored and dummy read is repeated.

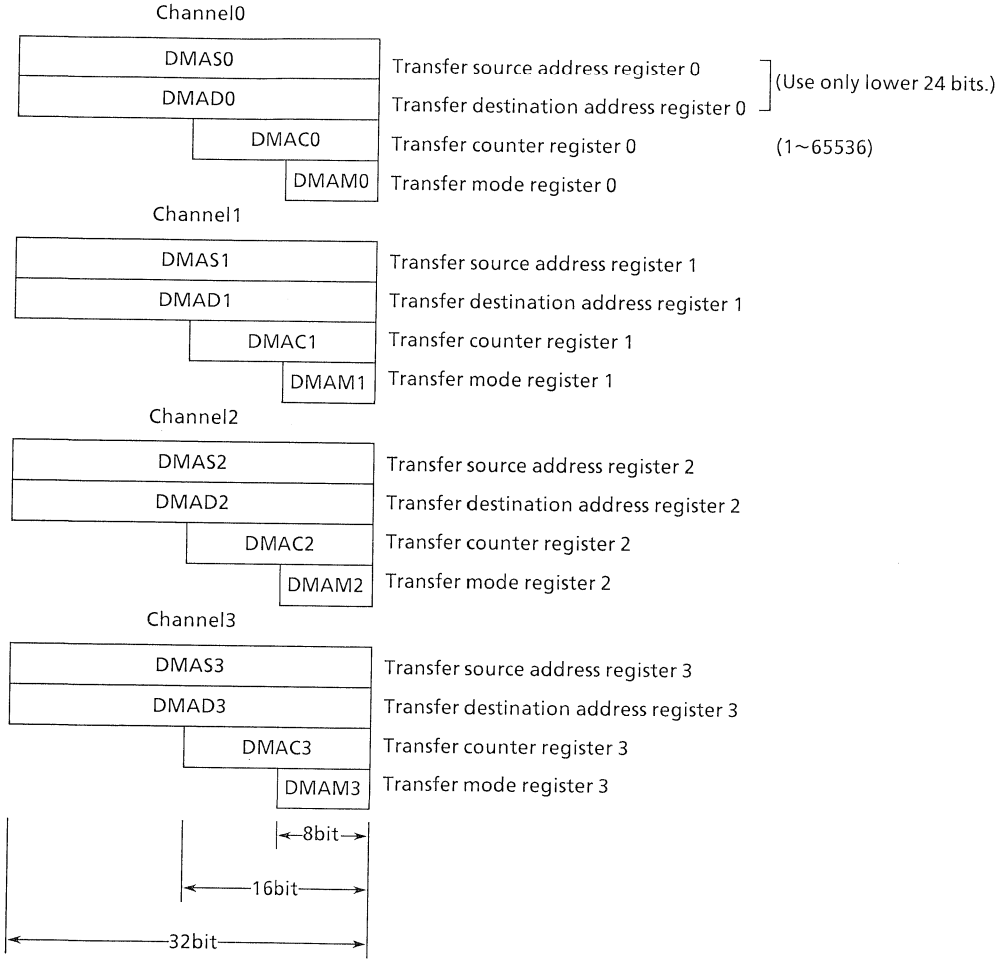
32-bit control registers are used for setting transfer source/destination addresses. However, the TLCS-900 has only 24 address pins for output. A 16M-byte space is available for the high-speed micro DMA. Also in normal mode operation, the all address space (In other words, the space for system mode which is set by the CS/WAIT controller) can be accessed by high-speed micro DMA processing.

There are two data transfer modes: one-byte mode and one-word mode. Incrementing, decrementing, and fixing the transfer source/destination address after transfer can be done in both modes. Therefore data can easily be transferred between I/O and memory and between I/Os. For details of transfer modes, see the description of transfer mode registers.

The transfer counter has 16 bits, so up to 65536 transfers (the maximum when the initial value of the transfer counter is 0000H) can be performed for one interrupt source by high-speed micro DMA processing.

Interrupt sources processed by high-speed micro DMA processing are those with the high-speed micro DMA start vectors listed in Table 3.3 (1).

(2) Register configuration (CPU control register)



(3) Transfer mode register details

(DMAM0~3)

0	0	0	0	Mode

Note : When specifying values for this register, set the upper 4 bits to 0.

Z: 0 = byte transfer, 1 = word transfer

					execution time (Min.)
0	0	0	Z	Transfer destination address INC mode ..... for I/O to memory (DMADn +) ← (DMASn) DMACn←DMACn – 1 if DMACn = 0 then INT.	16 states (2μs @16MHz)
0	0	1	Z	Transfer destination address DEC mode ..... for I/O to memory (DMADn –) ← (DMASn) DMACn←DMACn – 1 if DMACn = 0 then INT.	16 states (2μs @16MHz)
0	1	0	Z	Transfer source address INC mode ..... for I/O to memory (DMADn) ← (DMASn +) DMACn←DMACn – 1 if DMACn = 0 then INT.	16 states (2μs @16MHz)
0	1	1	Z	Transfer source address DEC mode ..... for I/O to memory (DMADn) ← (DMASn –) DMACn←DMACn – 1 if DMACn = 0 then INT.	16 states (2μs @16MHz)
1	0	0	Z	Fixed address mode ..... I/O to I/O (DMADn) ← (DMASn) DMACn←DMACn – 1 if DMACn = 0 then INT.	16 states (2μs @16MHz)
1	0	1	0	Read-only mode ..... for DRAM refresh Dummy← (DMASn) ; Reads 4 bytes. DMASn←DMASn + 4 ; Increments lower word only. DMACn←DMACn – 1	14 states (1.75μs @16MHz)
1	0	1	1	Counter mode ..... for interrupt counter DMASn←DMASn + 1 DMACn←DMACn – 1 if DMACn = 0 then INT.	11 states (1.375μs @16MHz)

(1 states = 125ns @ 16MHz)

Note : n : corresponds to high-speed μDMA channels 0-3.
DMADn + / DMASn + : Post-increment (Increments register value after transfer.)
DMADn – / DMASn – : Post-decrement (Decrement register value after transfer.)

All address space (the space for system mode) can be accessed by high-speed μDMA.
Do not use undefined codes for transfer mode control.

< Usage of read only mode (DRAM refresh) >

When the hardware configuration is as follows:
DRAM mapping size: = 1MB
DRAM data bus size: = 8 bits
DRAM mapping address range: = 100000H to 1FFFFFH

Set the following registers first; refresh is performed automatically.

① Register initial value setting

```
LD    XIX, 100000H
LDC   DMAS0, XIX     ··· mapping start address
LD    A, 00001010B
LDC   DMAM0, A       ··· read only mode (for DRAM refresh)
```

② Timer setting

Set the timers so that interrupts are generated at intervals of 62.5μs or less.

③ Interrupt controller setting

Set the timer interrupt mask at the desired interrupt request level (1 to 6). Write the above timer interrupt vector value in the micro DMA start vector register, DMA0V.

(Operation description)

The DRAM data bus is an 8-bit bus and the micro DMA is in read-only mode (4 bytes), so refresh is performed for four times per interrupt.

When a 512 refresh/8ms DRAM is connected, DRAM refresh is performed sufficiently if the micro DMA is started every 15.625μs × 4 = 62.4μs or less, since the timing is15.625μs/refresh.

(Overhead)

Each processing time by the micro DMA is 2.25μs (18 states) @16 MHz with an 8-bit data bus.
In the above example, the micro DMA is started every 62.5μs, 2.25μs/62.5μs = 0.036; thus, the overhead is 3.6%.

3.3.3 Interrupt Controller

Figure 3.3.3 (1) is a block diagram of the interrupt circuits. The left half of the diagram shows the interrupt controller; the right half includes the CPU interrupt request signal circuit and the HALT release signal circuit.

Each interrupt channel (total of 20 channels) in the interrupt controller has an interrupt request flip-flop, interrupt priority setting register, and a register for storing the high-speed micro DMA start vector. The interrupt request fip-flop is used to latch interrupt requests from peripheral devices. The flip-flop is cleared to 0 at reset, when the CPU reads the interrupt channel vector after the acceptance of interrupt, or when the CPU executes an instruction that clears the interrupt of that channel (writes 0 in the clear bit of the interrupt priority setting register).

For example, to clear the INT0 interrupt request, set the register after the DI instruction as follows.

INTE0AD ← ---- 0 --- Zero-clears the INT0 Flip Flop.

The status of the interrupt request flip-flop is detected by reading the clear bit. Detects whether there is an interrupt request for an interrupt channel.

The interrupt priority can be set by writing the priority in the interrupt priority setting register (eg, INTE0AD, INTE45, etc.) provided for each interrupt source. Interrupt levels to be set are from 1 to 6. Writing 0 or 7 as the interrupt priority disables the corresponding interrupt request. The priority of the non-maskable interrupt ($\overline{\text{NMI}}$ pin, watchdog timer, etc.) is fixed to 7. If interrupt requests with the same interrupt level are generated simultaneously, interrupts are accepted in accordance with the default priority (the smaller the vector value, the higher the priority).

The interrupt controller sends the interrupt request with the highest priority among the simultaneous interrupts and its vector address to the CPU. The CPU compares the priority value <IFF2 to 0> set in the Status Register by the interrupt request signal with the priority value sent; if the latter is higher, the interrupt is accepted. Then the CPU sets a value higher than the priority value by 1 in the CPU SR<IFF2 to 0>. Interrupt requests where the priority value equals or is higher than the set value are accepted simultaneously during the previous interrupt routine. When interrupt processing is completed (after execution of the RETI instruction) , the CPU restores the priority value saved in the stack before the interrupt was generated to the CPU SR<IFF2 to 0>.

The interrupt controller also has four registers used to store the high-speed micro DMA start vector. These are I/O registers; unlike other micro DMA registers (DMAS, DMAD, DMAM, and DMAC), they can be accessed in either normal or system mode. Writing the start vector of the interrupt source for the micro DMA processing (see Table 3.3.(1)), enables the corresponding interrupt to be processed by micro DMA processing. The values must be set in the micro DMA parameter registers (eg, DMAS and DMAD) prior to the micro DMA processing.

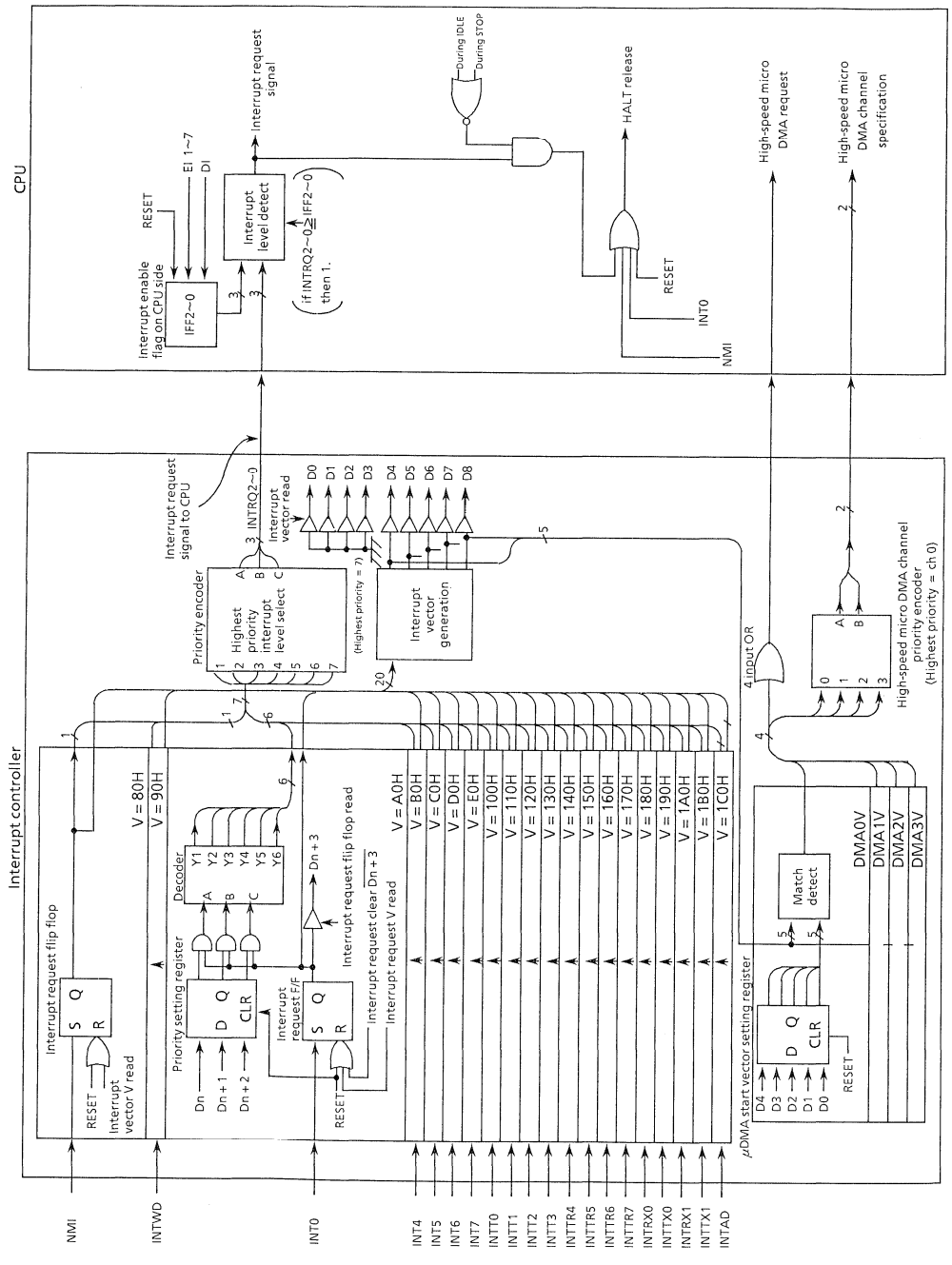


Figure3.3.3 (1) Block Diagram of Interrupt Controller

(1) Interrupt priority setting register

(Read-modify-write prohibited.)

Symbol	Address	7	6	5	4	3	2	1	0	
INTE0AD	0070H	INTAD				INT0				←Interrupt source
		IADC	IADM2	IADM1	IADM0	I0C	I0M2	I0M1	I0M0	←bit Symbol
		R/W	W			R/W	W			←Read/Write
		0	0	0	0	0	0	0	0	←After reset
INTE45	0071H	INT5				INT4				
		I5C	I5M2	I5M1	I5M0	I4C	I4M2	I4M1	I4M0	
		R/W	W			R/W	W			
		0	0	0	0	0	0	0	0	
INTE67	0072H	INT7				INT6				
		I7C	I7M2	I7M1	I7M0	I6C	I6M2	I6M1	I6M0	
		R/W	W			R/W	W			
		0	0	0	0	0	0	0	0	
INTET10	0073H	INTT1 (Timer1)				INTT0 (Timer0)				
		IT1C	IT1M2	IT1M1	IT1M0	IT0C	IT0M2	IT0M1	IT0M0	
		R/W	W			R/W	W			
		0	0	0	0	0	0	0	0	
INTEPW10	0074H	INTT3 (Timer3/PWM1)				INTT2 (Timer2/PWM0)				
		IPW1C	IPW1M2	IPW1M1	IPW1M0	IPW0C	IPW0M2	IPW0M1	IPW0M0	
		R/W	W			R/W	W			
		0	0	0	0	0	0	0	0	
INTET54	0075H	INTTR5 (TREG5)				INTTR4 (TREG4)				
		IT5C	IT5M2	IT5M1	IT5M0	IT4C	IT4M2	IT4M1	IT4M0	
		R/W	W			R/W	W			
		0	0	0	0	0	0	0	0	
INTET76	0076H	INTTR7 (TREG7)				INTTR6 (TREG6)				
		IT7C	IT7M2	IT7M1	IT7M0	IT6C	IT6M2	IT6M1	IT6M0	
		R/W	W			R/W	W			
		0	0	0	0	0	0	0	0	
INTES0	0077H	INTTX0				INTRX0				
		ITX0C	ITX0M2	ITX0M1	ITX0M0	IRX0C	IRX0M2	IRX0M1	IRX0M0	
		R/W	W			R/W	W			
		0	0	0	0	0	0	0	0	
INTES1	0078H	INTTX1				INTRX1				
		ITX1C	ITX1M2	ITX1M1	ITX1M0	IRX1C	IRX1M2	IRX1M1	IRX1M0	
		R/W	W			R/W	W			
		0	0	0	0	0	0	0	0	

IxxM2	IxxM1	IxxM0	Function (Write)
0	0	0	Prohibits interrupt request.
0	0	1	Sets interrupt request level to "1".
0	1	0	Sets interrupt request level to "2".
0	1	1	Sets interrupt request level to "3".
1	0	0	Sets interrupt request level to "4".
1	0	1	Sets interrupt request level to "5".
1	1	0	Sets interrupt request level to "6".
1	1	1	Prohibits interrupt request.

IxxC	Function (Read)	Function (Write)
0	Indicates no interrupt request.	Clears interrupt request flag.
1	Indicates interrupt request.	----- Don't care -----

(2) External interrupt control

Interrupt Input Mode Control Register

IIMC (007BH)

	7	6	5	4	3	2	1	0
bit Symbol						I0IE	I0LE	NMIREE
Read/Write						W	W	W
After reset						0	0	0
Function						1: INT0 input enable	0: INT0 edge mode 1: INT0 level mode	1: Can be operated in $\overline{NMI}$ rising edge.

Read-modify-write is prohibited.

INT0 input enable (Note)

0	INT0 disable (P87 function only)
1	Input enable

$\overline{NMI}$ rising edge enable

0	Interrupt request generation at falling edge
1	Interrupt request generation at rising/falling edge

INT0 level enable

0	Rising edge detect interrupt
1	High level interrupt

Note : The INT0 pin can also be used for standby release as described later. Even if the pin is not used for standby release, setting this register to "0" maintains the port function during standby mode.

Setting of External Interrupt Pin Functions

Interrupt	Pin name	Mode	Setting method
$\overline{NMI}$	—	Falling edge	IIMC<NMIREE> = 0
		Rising and falling edges	IIMC<NMIREE> = 1
INT0	P87	Rising edge	IIMC<I0LE> = 0, <I0IE> = 1
		Level	IIMC<I0LE> = 1, <I0IE> = 1
INT4	P80	Rising edge	T4MOC<CAP12M1,0> = 0,0 or 0,1 or 1,1
		Falling edge	T4MOD<CAP12M1, 0> = 1, 0
INT5	P81	Rising edge	———
INT6	P84	Rising edge	T5MOC<CAP34M1,0> = 0,0 or 0,1 or 1,1
		Falling edge	T5MOD<CAP34M1, 0> = 1, 0
INT7	P85	Rising edge	———

(3) High-speed micro DMA start vector

When the CPU reads the interrupt vector after accepting an interrupt, it simultaneously compares the interrupt vector with each channel's micro DMA start vector (bits 4 to 8 of the interrupt vector). When both match, the interrupt is processed in micro DMA mode for the channel whose value matched.

If the interrupt vector matches more than one channel, the channel with the lower channel number has a higher priority.

Micro DMA0 Start Vector (read-modify-write is not possible.)

DMA0V (007CH)

	7	6	5	4	3	2	1	0
bit Symbol				DMA0V8	DMA0V7	DMA0V6	DMA0V5	DMA0V4
Read/Write				W				
After reset				0	0	0	0	0

Micro DMA1 Start Vector (read-modify-write is not possible.)

DMA1V (007DH)

	7	6	5	4	3	2	1	0
bit Symbol				DMA1V8	DMA1V7	DMA1V6	DMA1V5	DMA1V4
Read/Write				W				
After reset				0	0	0	0	0

Micro DMA2 Start Vector (read-modify-write is not possible.)

DMA2V (007EH)

	7	6	5	4	3	2	1	0
bit Symbol				DMA2V8	DMA2V7	DMA2V6	DMA2V5	DMA2V4
Read/Write				W				
After reset				0	0	0	0	0

Micro DMA3 Start Vector (read-modify-write is not possible.)

DMA3V (007FH)

	7	6	5	4	3	2	1	0
bit Symbol				DMA3V8	DMA3V7	DMA3V6	DMA3V5	DMA3V4
Read/Write				W				
After reset				0	0	0	0	0

(4) Notes

The instruction execution unit and the bus interface unit of this CPU operate independently of each other. Therefore, if the instruction used to clear an interrupt request flag of an interrupt is fetched before the interrupt is generated, it is possible that the CPU might execute the fetched instruction to clear the interrupt request flag while reading the interrupt vector after accepting the interrupt. If so, the CPU would read the default vector 00A0 and start the interrupt processing from the address 80A0.

To avoid this, make sure that the instruction used to clear the interrupt request flag comes after the DI instruction.

3.4 Standby Function

When the HALT instruction is executed, the TMP96C141F / TMP96CM40F / TMP96PM40F enters RUN, IDLE, or STOP mode depending on the contents of the HALT mode setting register.

(1) RUN : Only the CPU halts; power consumption remains unchanged.

(2) IDLE : Only the built-in oscillator operates, while all other built-in circuits halt. Power consumption is reduced to 1/10 or less than that during normal operation.

(3) STOP : All internal circuits including the built-in oscillator halt. This greatly reduces power consumption.

The states of the port pins in STOP mode can be set as listed in Table 3.4 (1) using the I/O register WDMOD<DRVE> bit.

WDMOD (005CH)

	7	6	5	4	3	2	1	0
bit Symbol	WDTE	WDTP1	WDTP0	WARM	HALTM1	HALTM0	RESCR	DRVE
Read/Write	R/W							
After reset	1	0	0	0	0	0	0	0
Function	1 : WDT Enable	00 : 2^{16}/fc 01 : 2^{18}/fc 10 : 2^{20}/fc 11 : 2^{22}/fc Detection time		Warming up time 0 : 2^{16}/fc 1 : 2^{18}/fc	Standby mode 00 : RUN mode 01 : STOP mode 10 : IDLE mode 11 : Don't care		1 : Connects watchdog timer output to $\overline{\text{RESET}}$ pin internally.	1 : Drive pin even in STOP mode.

When STOP mode is released by other than a reset, the system clock output starts after allowing some time for warming up set by the warming-up counter for stabilizing the built-in oscillator. To release STOP mode by a reset, it is necessary to allow a reset time long enough to allow the oscillator to stablilize.

To release standby mode, a reset or an interrupt is used. To release IDLE or STOP mode, only an interrupt by the $\overline{\text{NMI}}$ or INT0 pin, or a reset can be used. The details are described below.

Standby Release by Interrupt

Standby mode \ Interrupt level	Interrupt mask (IFF2 to 0) ≦ interrupt request level	Interrupt mask (IFF2 to 0) > interrupt request level
RUN	Can be released by any interrupt. After standby mode is released, interrupt processing starts.	Can only be released by INT0 pin. Processing resumes from address next to HALT instruction.
IDLE	Can only be released by $\overline{\text{NMI}}$ or INT0 pin. After standby mode is released, interrupt processing starts.	↑
STOP	↑	↑

Table 3.4 (1) Pin states in STOP mode

Pin name	I/O	96CM40 / 96PM40		96C141	
		DRVE = 0	DRVE = 1	DRVE = 0	DRVE = 1
P0	Input mode / AD0~7 Output mode	– –	– Output	– ×	– ×
P1	Input mode / AD8~15 Output mode / A8~15	– –	– Output	– ×	– ×
P2	Input mode Output mode / A0~7, A16~23	PD* PD*	PD* Output	PD* PD*	PD* Output
P30 ($\overline{RD}$), P31 ($\overline{WR}$)	Output	–	Output	–	"1" Output
P32~P37	Input mode Output mode	PU PU	PU Output	←	
P40, P41	Input mode Output mode	PU* PU*	PU Output		
P42 ($\overline{CS2}$ / $\overline{CAS2}$)	Input mode Output mode	PD* PD*	PD Output		
P5	Input	–	–		
P6	Input mode Output mode	PU* PU*	PU Output		
P7	Input mode Output mode	PU* PU*	PU Output		
P80~P86	Input mode Output mode	PU* PU*	PU Output		
P87 (INT0)	Input mode Output mode	PU PU	PU Output		
P9	Input mode Output mode	PU* PU*	PU Output		
$\overline{NMI}$	Input	Input	Input		
$\overline{WDTOUT}$	Output	Output	Output		
ALE	Output	"0"	"0"		
CLK	Output	–	"1"		
$\overline{RESET}$	Input	Input	Input		
$\overline{EA}$	Input	Input	Input		
X1	Input	–	–		
X2	Output	"1"	"1"		

– : Input for input mode/input pin is invalid; output mode/output pin is at high impedance.
Input : Input enable state
Input : Input gate in operation. Fix input voltage to 0 or 1 so that input pin stays constant.
Output : Output state
PU : Programmable pull-up pin. Fix the pin to avoid through current since the input gate operates when a pull-up resistor is not set.
PD : Programmable pull-down pin. Fix the pin like a pull-up pin when a pull-down resistor is not set.
* : Input gate disable state. No through current even if the pin is set to high impedance.
× : Cannot set.

Note : Port registers are used for controlling programmable pull-up/pull-down. If a pin is also used for an output function (eg, TO1) and the output function is specified, whether pull-up or pull-down is selected depends on the output function data. If a pin is also used for an input function, whether pull-up or pull-down is selected depends on the port register setting value only.

3.5 Functions of Ports

The TMP96CM40/TMP96PM40 has 65 bits for I/O ports. The TMP96C141 has 47 bits for I/O ports because Port0, Port1, P30, and P31 are dedicated pins for AD0 to 7, AD8 to 15, $\overline{RD}$, and $\overline{WR}$.

These port pins have I/O functions for the built-in CPU and internal I/Os as well as general-purpose I/O port functions. Table 3.5 lists the function of each port pin.

Table 3.5 Functions of Ports (R: ↑ = With programmable pull-up resistor, ↓ = With programmable pull-down)

Port name	Pin name	Number of pins	Direction	R	Direction setting unit	Pin name for built-in function
Port0	P00~P07	8	I/O	–	Bit	AD0~AD7
Port1	P10~P17	8	I/O	–	Bit	AD8~AD15/A8~A15
Port2	P20~P27	8	I/O	↓	Bit	A0~A7/A16~A23
Port3	P30	1	Output	–	(Fixed)	$\overline{RD}$
	P31	1	Output	–	(Fixed)	$\overline{WR}$
	P32	1	I/O	↑	Bit	$\overline{HWR}$
	P33	1	I/O	↑	Bit	$\overline{WAIT}$
	P34	1	I/O	↑	Bit	$\overline{BUSRQ}$
	P35	1	I/O	↑	Bit	$\overline{BUSAK}$
	P36	1	I/O	↑	Bit	R/$\overline{W}$
	P37	1	I/O	↑	Bit	$\overline{RAS}$
Port4	P40	1	I/O	↑	Bit	$\overline{CS0}$ / $\overline{CAS0}$
	P41	1	I/O	↑	Bit	$\overline{CS1}$ / $\overline{CAS1}$
	P42	1	I/O	↓	Bit	$\overline{CS2}$ / $\overline{CAS2}$
Port5	P50~P53	4	Input	–	(Fixed)	AN0~AN3
Port6	P60~P67	8	I/O	↑	Bit	PG00~PG03, PG10~PG13
Port7	P70	1	I/O	↑	Bit	TI0
	P71	1	I/O	↑	Bit	TO1
	P72	1	I/O	↑	Bit	TO2
	P73	1	I/O	↑	Bit	TO3
Port8	P80	1	I/O	↑	Bit	TI4/INT4
	P81	1	I/O	↑	Bit	TI5/INT5
	P82	1	I/O	↑	Bit	TO4
	P83	1	I/O	↑	Bit	TO5
	P84	1	I/O	↑	Bit	TI6/INT6
	P85	1	I/O	↑	Bit	TI7/INT7
	P86	1	I/O	↑	Bit	TO6
	P87	1	I/O	↑	Bit	INT0
Port9	P90	1	I/O	↑	Bit	TXD0
	P91	1	I/O	↑	Bit	RXD0
	P92	1	I/O	↑	Bit	$\overline{CTS0}$
	P93	1	I/O	↑	Bit	TXD1
	P94	1	I/O	↑	Bit	RXD1
	P95	1	I/O	↑	Bit	SCLK1

Resetting makes the port pins listed below function as general-purpose I/O ports.
I/O pins programmable for input or output function as input ports.
To set port pins for built-in functions, a program is required.

Since the TMP96C141 has an external ROM, some ports are permanently assigned to the CPU.

- P00~P07 → AD0~AD7
- P10~P17 → AD8~AD15
- P30 → $\overline{RD}$
- P31 → $\overline{WR}$

Setting the bus request signal $\overline{BUSRQ}$ to 0 sets the following pins to high impedance. The built-in programmable pull-up/pull-down resistors remain in operation.

Pin name	Conditions for high impedance
AD0~AD15 A0~A23	$\overline{BUSRQ}$ = 0 and pins are not set as I/O port pins.
P30 ($\overline{RD}$) P31 ($\overline{WR}$) P32 ($\overline{HWR}$) P36 (R/$\overline{W}$) P37($\overline{RAS}$) P40 ($\overline{CS0}$ / $\overline{CAS0}$) P41 ($\overline{CS1}$ / $\overline{CAS1}$) P42 ($\overline{CS2}$ / $\overline{CAS2}$)	[$\overline{BUSRQ}$ = 0 and even if pins are set as I/O port pins.]

Pins AD0 to 15 and A0 to 23 are also used as I/O ports. If they are set as I/O port pins, they will not be set to high impedance even if $\overline{BUSRQ} = 0$.

This device's built-in memory or I/Os cannot be accessed by the external DMA controller.

3.5.1 Port 0 (P00 - P07)

Port 0 is an 8-bit general-purpose I/O port. I/O can be set on a bit basis using the control register P0CR. Resetting resets all bits of P0CR to 0 and sets Port 0 to input mode.

In addition to functioning as a general-purpose I/O port, Port 0 also functions as an address data bus (AD0 to 7). To access external memory, Port 0 functions as an address data bus (AD0 to 7) and all bits of the control register P0CR are cleared to 0.

With the TMP96C141, which has an external ROM, Port 0 always functions as an address data bus (AD0 to 7) regardless of the value set in control register P0CR.

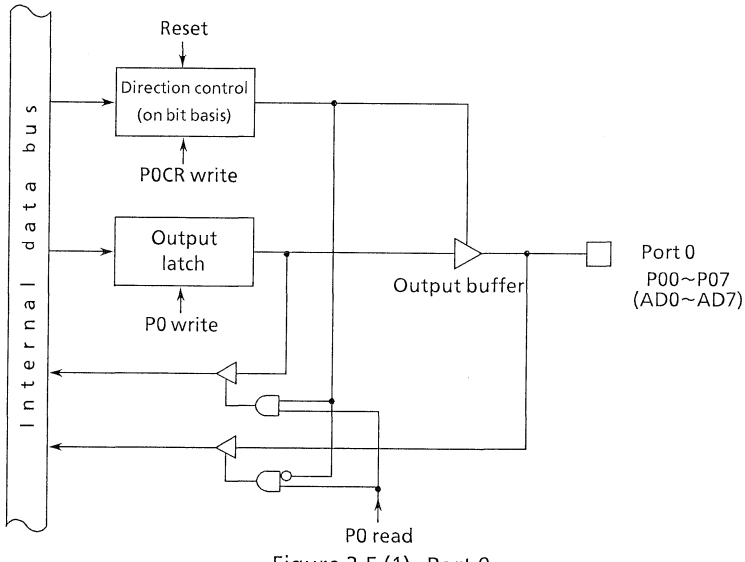


Figure 3.5 (1) Port 0

3.5.2 Port 1 (P10 - P17)

Port 1 is an 8-bit general-purpose I/O port. I/O can be set on a bit basis using control register P1CR and function register P1FC. Resetting resets all bits of output latch P1, control register P1CR, and function register P1FC to 0 and sets Port 1 to input mode.

In addition to functioning as a general-purpose I/O port, Port 1 also functions as an address data bus (AD8 to 15) or an address bus (A8 to 15).

With the TMP96C141, which comes with an external ROM, Port 1 always functions as an address data bus (AD8 to 15) regardless of the value set in control register P1CR.

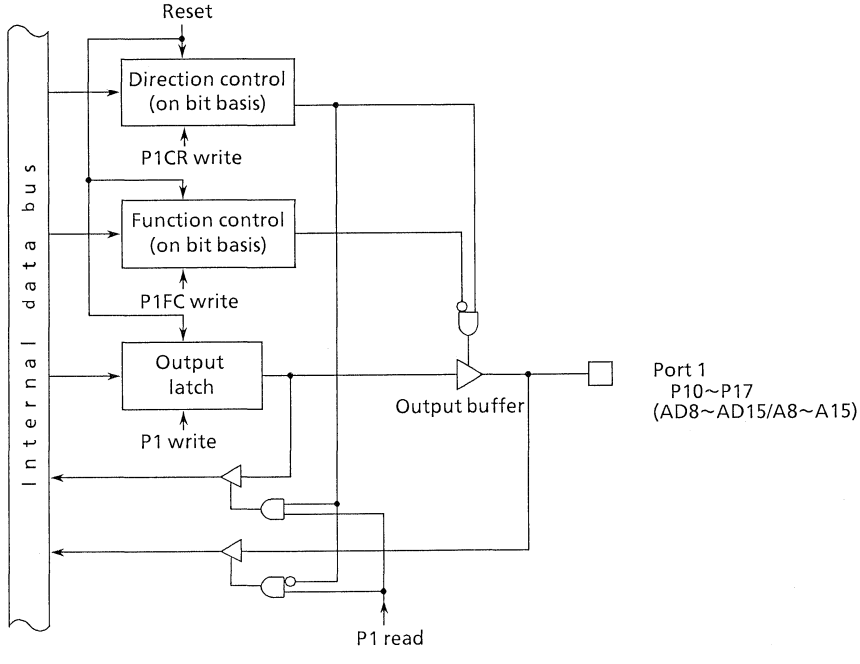


Figure 3.5 (2) Port 1

Port 0 Register

P0 (0000H)	7	6	5	4	3	2	1	0
bit Symbol	P07	P06	P05	P04	P03	P02	P01	P00
Read/Write	R / W							
After reset	Input mode (Output latch register becomes undefined.)							

Port 0 Control Register

P0CR (0002H)	7	6	5	4	3	2	1	0
bit Symbol	P07C	P06C	P05C	P04C	P03C	P02C	P01C	P00C
Read/Write	W							
After reset	0	0	0	0	0	0	0	0
Function	0:IN 1:OUT (At external access, Port 0 becomes AD7 - 0 and P0CR is cleared to 0.)							

→ Port 0 I/O setting

0	Input
1	Output

Port 1 Register

P1 (0001H)	7	6	5	4	3	2	1	0
bit Symbol	P17	P16	P15	P14	P13	P12	P11	P10
Read/Write	R / W							
After reset	Input mode (Output latch register is cleared to "0".)							

Port 1 Control Register

P1CR (0004H)	7	6	5	4	3	2	1	0
bit Symbol	P17C	P16C	P15C	P14C	P13C	P12C	P11C	P10C
Read/Write	W							
After reset	0	0	0	0	0	0	0	0
Function	<<See P1FC below.>>							

Port 1 Function Register

P1FC (0005H)	7	6	5	4	3	2	1	0
bit Symbol	P17F	P16F	P15F	P14F	P13F	P12F	P11F	P10F
Read/Write	W							
After reset	0	0	0	0	0	0	0	0
Function	P1FC/P1CR = 00 : IN, 01 : OUT, 10 : AD15-8, 11 : A15-8							

Read-modify-write is prohibited for registers P0CR, P1CR, and P1FC.

→ Port 1 function setting

P1CR <P1XC> \ P1FC <P1XF>	0	1
0	Input port	Address data bus (AD15-8)
1	Output port	Address bus (A15-8)

Note : <P1XF> is bit X in register P1FC; <P1XC>, in register P1CR.

Figure 3.5 (3) Registers for Ports 0 and 1

3.5.3 Port 2 (P20 - P27)

Port 2 is an 8-bit general-purpose I/O port. I/O can be set on bit basis using the control register P2CR and function register P2FC. Resetting resets all bits of output latch P2, control register P2CR and function register P2FC to 0. It also sets Port 2 to input mode and connects a pull-down resistor. To disconnect the pull-down resistor, write 1 in the output latch.

In addition to functioning as a general-purpose I/O port, Port 2 also functions as an address data bus (A0 to 7) and an address bus (A16 to 23).

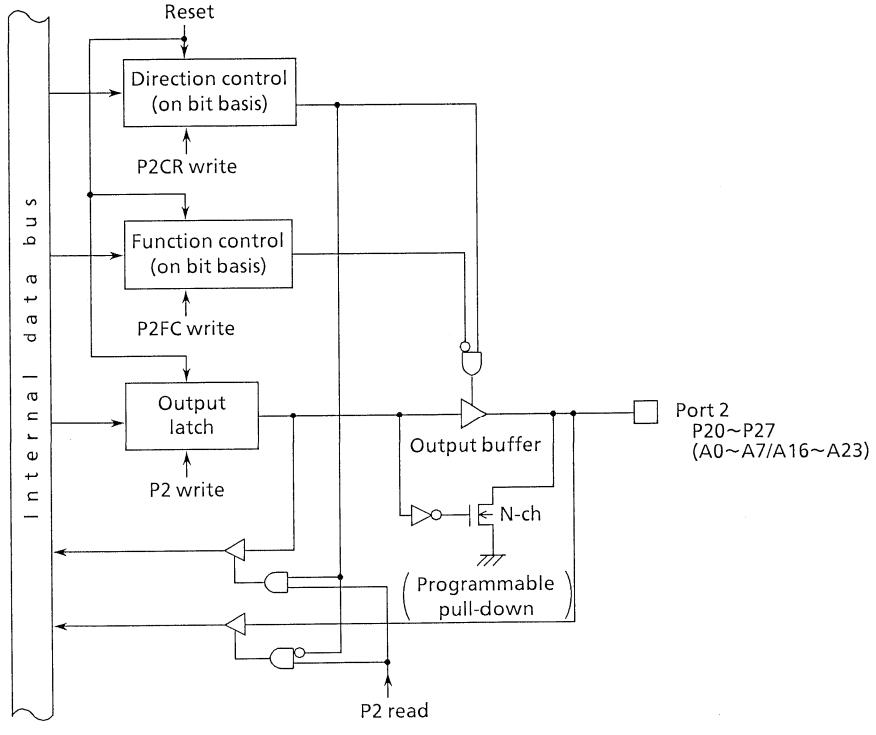


図 3.5 (4) Port 2

Port 2 Register

P2 (0006H)

	7	6	5	4	3	2	1	0
bit Symbol	P27	P26	P25	P24	P23	P22	P21	P20
Read/Write	R / W							
After reset	Input mode (Output latch register is cleared to "0".)							

Port 2 Control Register

P2CR (0008H)

	7	6	5	4	3	2	1	0
bit Symbol	P27C	P26C	P25C	P24C	P23C	P22C	P21C	P20C
Read/Write	W							
After reset	0	0	0	0	0	0	0	0
Function	<<See P2FC below.>>							

Port 2 Function Register

P2FC (0009H)

	7	6	5	4	3	2	1	0
bit Symbol	P27F	P26F	P25F	P24F	P23F	P22F	P21F	P20F
Read/Write	W							
After reset	0	0	0	0	0	0	0	0
Function	P2FC/P2CR = 00 : IN, 01 : OUT, 10 : A7-0, 11 : A23-16							

Read-modify-write is prohibited for registers P2CR and P2FC.

Port 2 function setting

P2CR <P2XC> \ P2FC<P2XF>	0	1
0	Input	address bus (A7-0)
1	Output	address bus (A23-16)

Note : <P2XF> is bit X in register P2FC; <P2XC>; in register P2CR.

Figure 3.5 (5) Registers for Port 2

3.5.4 Port 3 (P30 - P37)

Port 3 is an 8-bit general-purpose I/O port.

I/O can be set on a bit basis, but note that P30 and P31 are used for output only. I/O is set using control register P3CR and function register P3FC. Resetting resets all bits of output latch P3, control register P3CR (bits 0 and 1 are unused), and function register P3FC to 0. Resetting also outputs 1 from P30 and P31, sets P32 to P37 to input mode, and connects a pull-up resistor. To disconnect the pull-up resistor, write 0 in the output latch.

In addition to functioning as a general-purpose I/O port, Port 3 also functions as an I/O for the CPU's control/status signal.

With the TMP96CM40/TMP96PM40, when P30 pin is defined as $\overline{RD}$ signal output mode (<P30F> = 1), clearing the output latch register <P30> to 0 outputs the $\overline{RD}$ strobe (used for the pseudo static RAM) from the P30 pin even when the internal address area is accessed.

If the output latch register <P30> remains 1, the $\overline{RD}$ strobe signal is output only when the external address area is accessed.

With the TMP96C141, which comes with an external ROM, P30 outputs the $\overline{RD}$ signal; P31, the $\overline{WR}$ signal, regardless of the values set in function registers P30F and P31F.

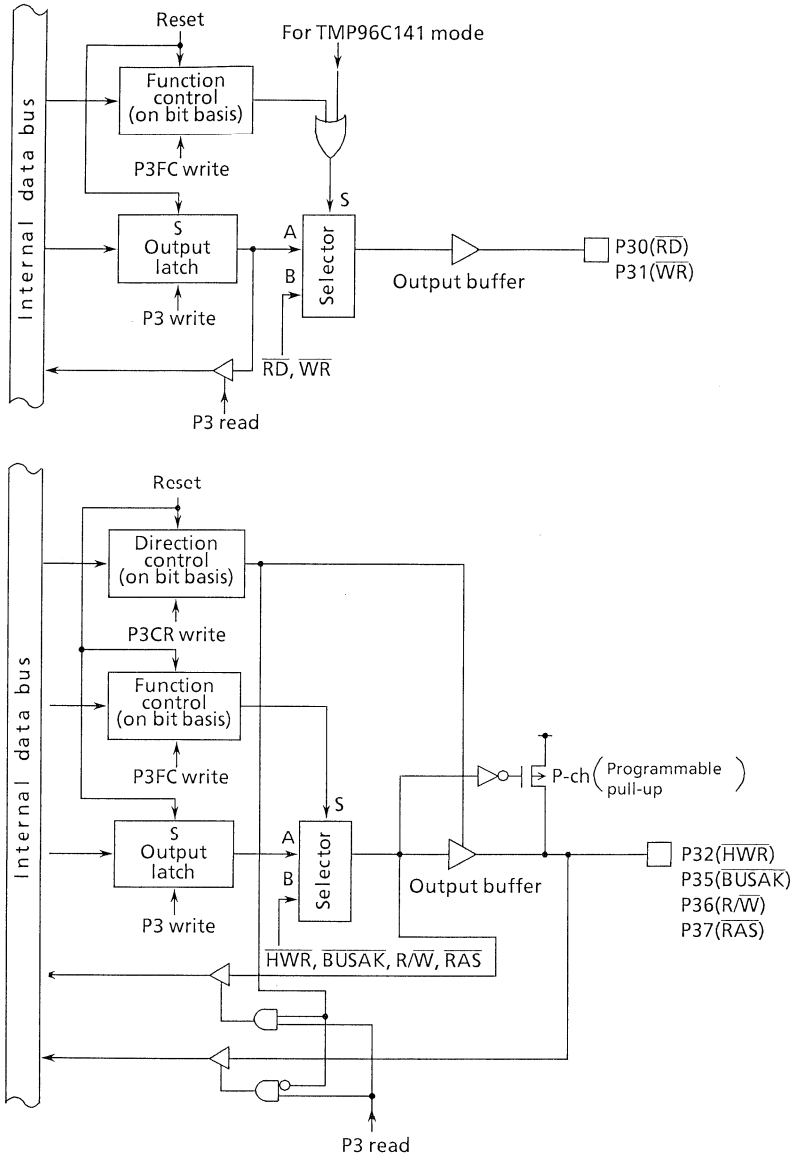


Figure 3.5 (6) Port 3 (P30, P31, P32, P35, P36, P37)

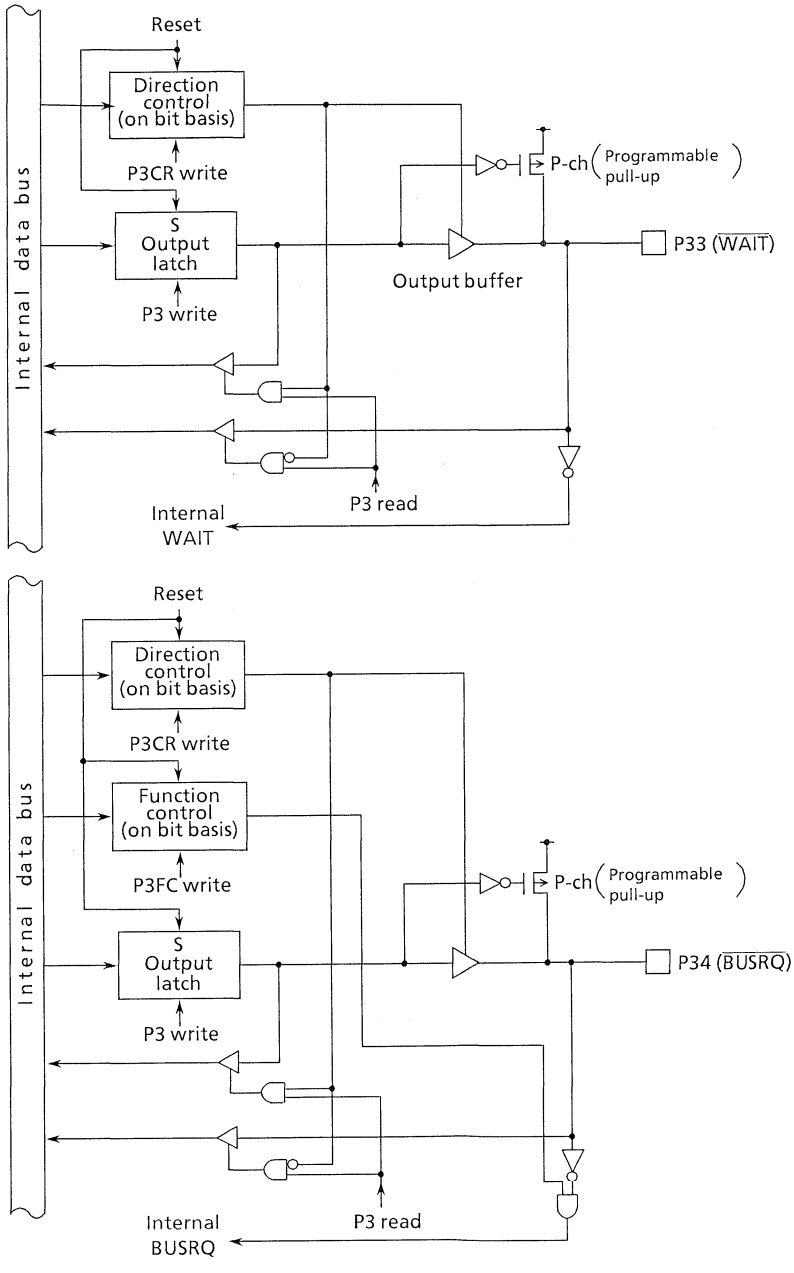


Figure 3.5 (7) Port3 (P33, P34)

Port 3 Register

P3 (0007H)	7	6	5	4	3	2	1	0
bit Symbol	P37	P36	P35	P34	P33	P32	P31	P30
Read/Write	R/W							
After reset	Input mode (pulled-up)						Output mode	
	1	1	1	1	1	1	1	1

Port 3 Control Register

P3CR (000AH)	7	6	5	4	3	2	1	0
bit Symbol	P37C	P36C	P35C	P34C	P33C	P32C		
Read/Write	W							
After reset	0	0	0	0	0	0		
	0 : IN 1 : OUT							

I/O setting

0	Input
1	Output

Port 3 Function Register

P3FC (000BH)	7	6	5	4	3	2	1	0
bit Symbol	P37F	P36F	P35F	P34F		P32F	P31F	P30F
Read/Write	W							
After reset	0	0	0	0		0	0	0
Function	0 : PORT 1 : $\overline{RAS}$	0 : PORT 1 : R/$\overline{W}$	0 : PORT 1 : $\overline{BUSAK}$	0 : PORT 1 : $\overline{BUSRQ}$		0 : PORT 1 : $\overline{HWR}$	0 : PORT 1 : $\overline{WR}$	0 : PORT 1 : $\overline{RD}$

Read-modify-write is prohibited for registers P3CR and P3FC.

$\overline{BUSRQ}$ setting

P3FC <P34F>	1
P3CR <P34C>	0

$\overline{BUSAK}$ setting

P3FC <P35F>	1
P3CR <P35C>	1

R/$\overline{W}$ setting

P3FC <P36F>	1
P3CR <P36C>	1

$\overline{RAS}$ setting

P3FC <P37F>	1
P3CR <P37C>	1

P30 ($\overline{RD}$) function setting

<P30F> \ <P30>	0	1
0	"0" output	"1" output
1	Always $\overline{RD}$ output (for pseudo SRAM)	$\overline{RD}$ output only for external access

P31 ($\overline{WR}$) function setting

<P31F> \ <P31>	0	1
0	"0" output	"1" output
1	$\overline{WR}$ output only for external access	

$\overline{HWR}$ setting

P3FC <P32F>	1
P3CR <P32C>	1

Figure3.5 (8) Registers for Port 3

3.5.5 Port 4 (P40 - P42)

Port 4 is a 3-bit general-purpose I/O port. I/O can be set on a bit basis using control register P4CR and function register P4FC. Resetting does the following:

- Sets the P40 and P42 output latch registers to 1.
- Resets all bits of the P42 output latch register, the control register P4CR, and the function register P4FC to 0.
- Sets P40 and P41 to input mode and connects a pull-up resistor.
- Sets P42 to input mode and connects a pull-down resistor.

To disconnect the resistors, write 0 in the output latch (for pull-up), or 1 (for pull-down).

In addition to functioning as a general-purpose I/O port, Port 4 also functions as a chip select output signal ($\overline{CS0}$ to $\overline{CS2}$ or $\overline{CAS0}$ to $\overline{CAS2}$).

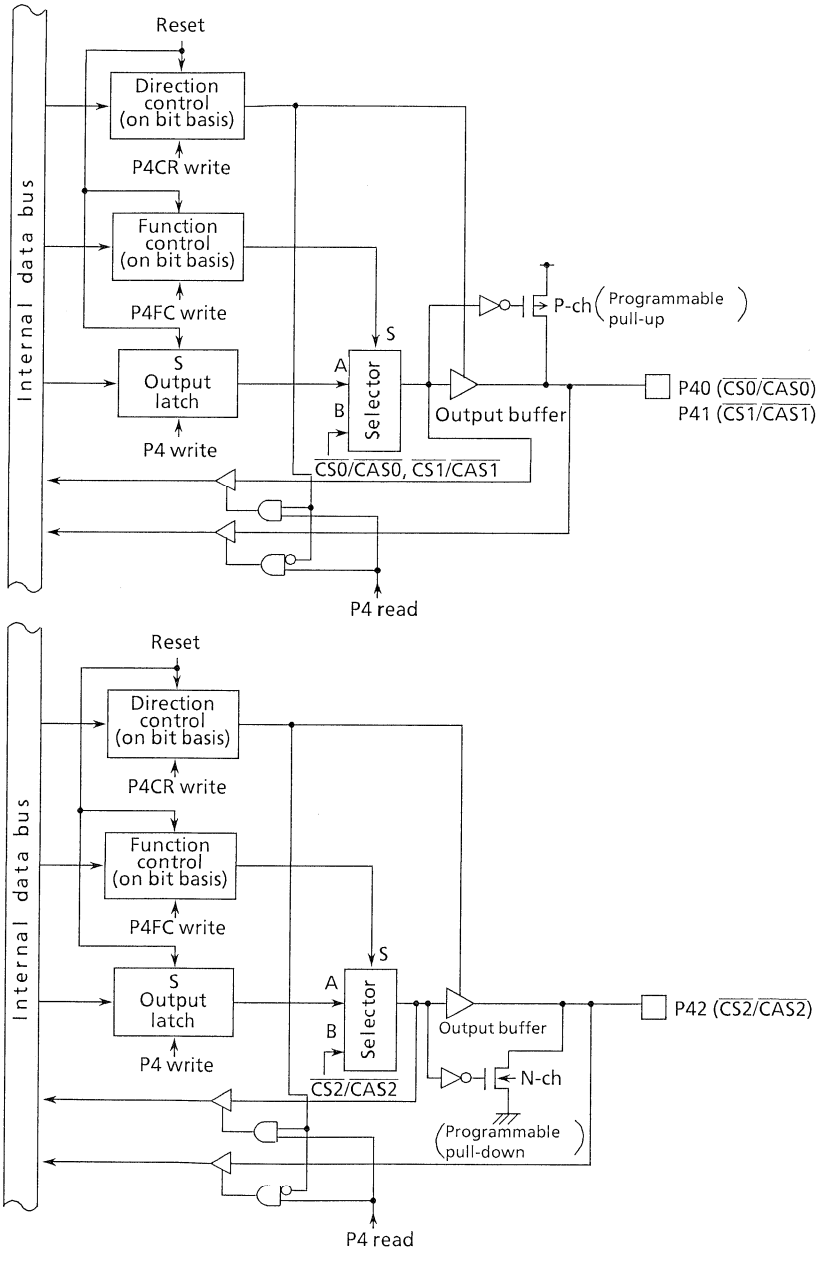


Figure 3.5 (9) Port 4

Port 4 Register

P4 (000CH)

	7	6	5	4	3	2	1	0
bit Symbol						P42C	P41C	P40C
Read/Write						R/W		
After reset						Input mode		
						0 (Pull-down)	1 (Pull-up)	1 (Pull-up)

Port 4 Control Register

P4CR (000EH)

	7	6	5	4	3	2	1	0
bit Symbol						P42C	P41C	P40C
Read/Write						W		
After reset						0	0	0
						0 : IN	1 : OUT	

I/O setting

0	Input
1	Output

Port 4 Function Register

P4FC (0010H)

	7	6	5	4	3	2	1	0
bit Symbol						P42F	P41F	P40F
Read/Write						W		
After reset						0	0	0
Function						0 : PORT	1 : $\overline{CS}/\overline{CAS}$	

Read-modify-write is not possible for P4CR or P4FC.

0	Port (P40)
1	$\overline{CS0}$ / $\overline{CAS0}$

0	Port (P41)
1	$\overline{CS1}$ / $\overline{CAS1}$

0	Port (P42)
1	$\overline{CS2}$ / $\overline{CAS2}$

Note : To output chip select signal ($\overline{CS0}/\overline{CAS0}$ to $\overline{CS2}/\overline{CAS2}$), set the corresponding bits of the control register P4CR and the function register P4FC.
The B0CS, B1CS, and B2CS registers of the chip select / wait controller are used to select the $\overline{CS}/\overline{CAS}$ function.

290591

Figure 3.5 (10) Registers for Port 4

3.5.6 Port 5 (P50 - P53)

Port 5 is a 4-bit input port, also used as an analog input pin.

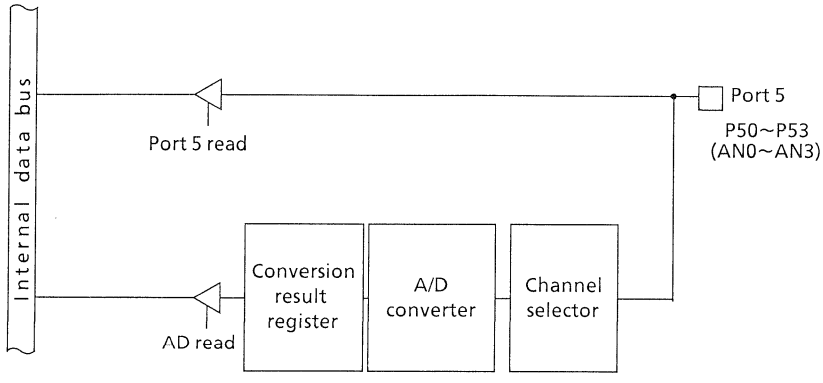


Figure3.5 (11) Port 5

Port 5 Register

		7	6	5	4	3	2	1	0
P5 (000DH)	bit Symbol					P53	P52	P51	P50
	Read/Write					R			
	After reset					Input mode			

Figure 3.5 (12) Registers for Port 5

3.5.7 Port 6 (P60 - P67)

Port 6 is an 8-bit general-purpose I/O port. I/O can be set on bit basis. Resetting sets Port 6 as an input port and connects a pull-up resistor. It also sets all bits of the output latch to 1. In addition to functioning as a general-purpose I/O port, Port 6 also functions as a pattern generator PG0/PG1 output. PG0 is assigned to P60 to P63; PG1, to P64 to P67. Writing 1 in the corresponding bit of the port 6 function register (P6FC) enables PG output. Resetting resets the function register P6FC value to 0, and sets all bits to ports.

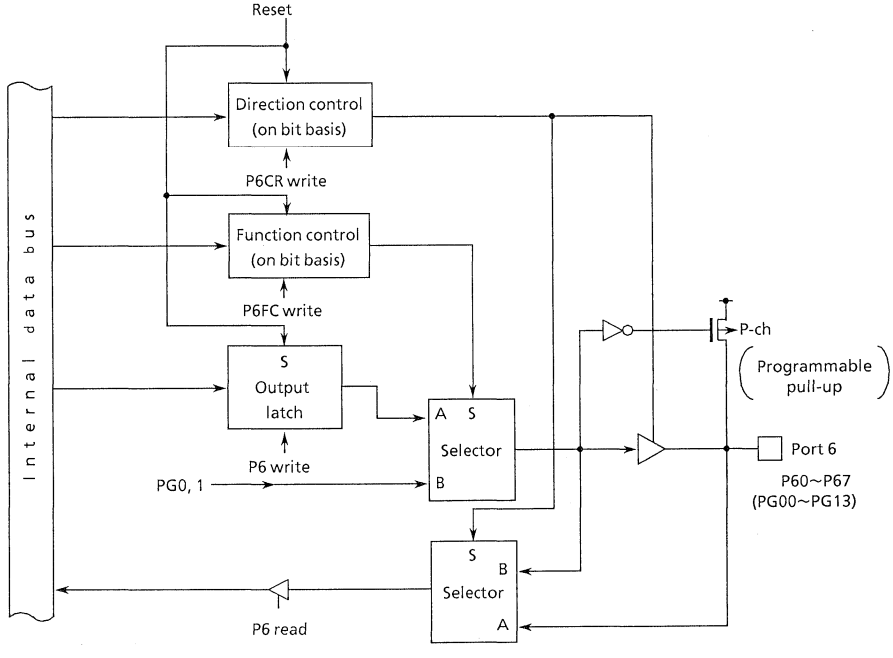


Figure 3.5 (13) Port 6

Port 6 Register

P6 (0012H)

	7	6	5	4	3	2	1	0
bit Symbol	P67	P66	P65	P64	P63	P62	P61	P60
Read/Write	R/W							
After reset	Input mode (Pull-up)							
	1	1	1	1	1	1	1	1

Port 6 Control Register

P6CR (0014H)

	7	6	5	4	3	2	1	0
bit Symbol	P67C	P66C	P65C	P64C	P63C	P62C	P61C	P60C
Read/Write	W							
After reset	0	0	0	0	0	0	0	0
Function	0 : IN 1 : OUT							

Port 6 I/O setting

0	Input
1	Output

Port 6 Function Register

P6FC (0016H)

	7	6	5	4	3	2	1	0
bit Symbol	P67F	P66F	P65F	P64F	P63F	P62F	P61F	P60F
Read/Write	W							
After reset	0	0	0	0	0	0	0	0
Function	0 : PORT		1 : PG1-OUT		0 : PORT		1 : PG0-OUT	

Read-modify-write is prohibited for registers P6CR and P6FC.

Port 6 function setting

0	General-purpose port
1	Stepping motor control / Pattern generation port

Figure3.5 (14) Registers for Port 6

3.5.8 Port 7 (P70 - P73)

Port 7 is a 4-bit general-purpose I/O port. I/O can be set on bit basis. Resetting sets Port 7 as an input port and connects a pull-up resistor. In addition to functioning as a general-purpose I/O port, Port 70 also functions as an input clock pin TI0; Port 71 as an 8-bit timer output (TO1), Port 72 as a PWM0 output (TO2), and Port 73 as a PWM1 output (TO3) pin. Writing 1 in the corresponding bit of the Port 7 function register (P7FC) enables output of the timer. Resetting resets the function register P7FC value to 0, and sets all bits to ports.

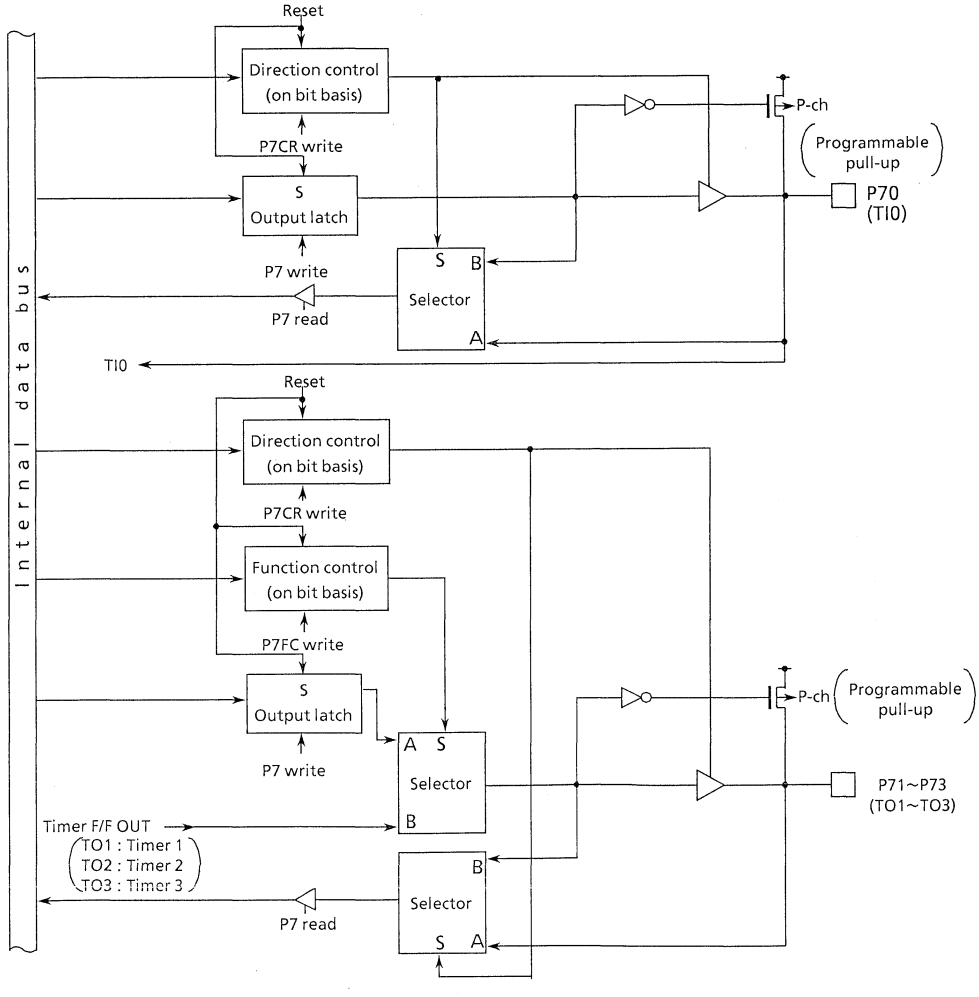


Figure 3.5 (15) Port 7

Port 7 Register

P7 (0013H)

	7	6	5	4	3	2	1	0
bit Symbol					P73	P72	P71	P70
Read/Write					R/W			
After reset					Input mode (Pull-up)			
					1	1	1	1

Port 7 Control Register

P7CR (0015H)

	7	6	5	4	3	2	1	0
bit Symbol					P73C	P72C	P71C	P70C
Read/Write					W			
After reset					0	0	0	0
Function					0 : IN		1 : OUT	

Port 7 I/O setting

0	Input
1	Output

Port 7 Function Register

P7FC (0017H)

	7	6	5	4	3	2	1	0
bit Symbol					P73F	P72F	P71F	
Read/Write					W			
After reset					0	0	0	
Function					0 : PORT 1 : TO3	0 : PORT 1 : TO2	0 : PORT 1 : TO1	

Read-modify-write is prohibited for registers P7CR and P7FC.

Setting P71 as TO1

P7FC <P71F>	1
P7CR <P71C>	1

Setting P72 as TO2

P7FC <P72F>	1
P7CR <P72C>	1

Setting P73 as TO3

P7FC <P73F>	1
P7CR <P73C>	1

Figure3.5 (16) Registers for Port 7

3.5.9 Port 8 (P80 - P83)

Port 8 is an 8-bit general-purpose I/O port. I/O can be set on a bit basis. Resetting sets Port 8 as an input port and connects a pull-up resistor . It also sets all bits of the output latch register P8 to 1. In addition to functioning as a general-purpose I/O port, Port 8 also functions as an input for 16-bit timer 4 & 5 clocks, an output for 16-bit timer F/F 4, 5, & 6 output, and an input for INT0. Writing 1 in the corresponding bit of the Port 8 function register (P8FC) enables those functions. Resetting resets the function register P8FC value to 0 and sets all bits to ports.

(1) P80~P86

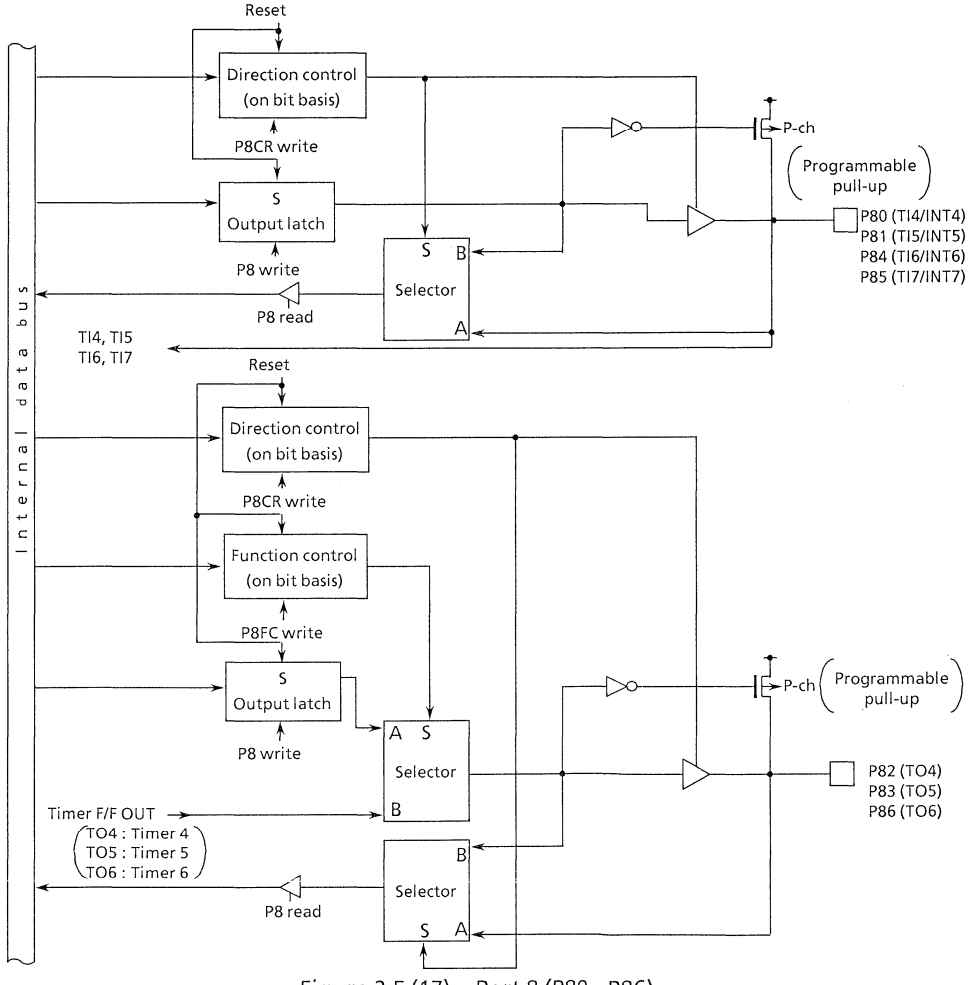


Figure 3.5 (17) Port 8 (P80 - P86)

(2) P87 (INT0)

Port 87 is a general-purpose I/O port, and also used as an INT0 pin for external interrupt request input.

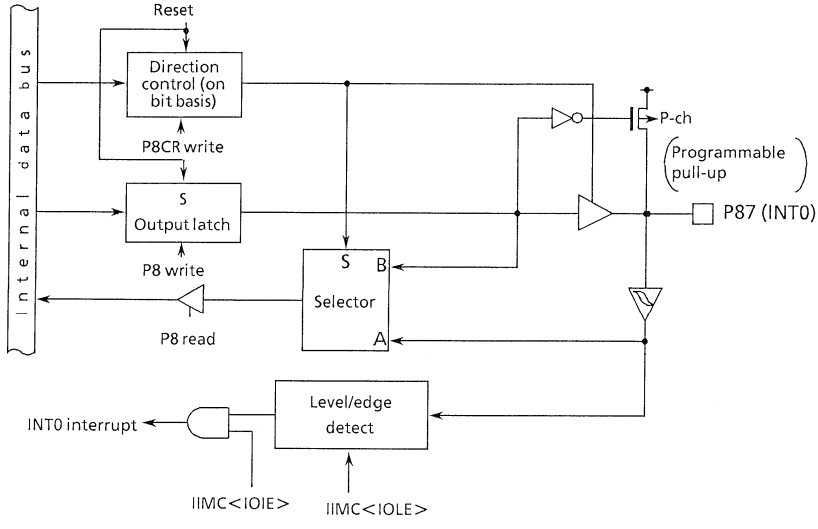


Figure 3.5 (18) Port 87

Port 8 Register

P8 (0018H)

	7	6	5	4	3	2	1	0
bit Symbol	P87	P86	P85	P84	P83	P82	P81	P80
Read/Write	R/W							
After reset	Input mode							
	1	1	1	1	1	1	1	1

Port 8 Control Register

P8CR (001AH)

	7	6	5	4	3	2	1	0
bit Symbol	P87C	P86C	P85C	P84C	P83C	P82C	P81C	P80C
Read/Write	W							
After reset	0	0	0	0	0	0	0	0
Function	0 : IN				1 : OUT			

Port 8 I/O setting

0	Input
1	Output

Port 8 Function Register

P8FC (001CH)

	7	6	5	4	3	2	1	0
bit Symbol		P86F			P83F	P82F		
Read/Write		W			W	W		
After reset		0			0	0		
Function		0 : PORT 1 : TO6			0 : PORT 1 : TO5	0 : PORT 1 : TO4		

Read-modify-write is prohibited for registers P8CR and P8FC.

Setting P82 as TO4

P8FC <P82F>	1
P8CR <P82C>	1

Setting P83 as TO5

P8FC <P83F>	1
P8CR <P83C>	1

Setting P84 as TO6

P8FC <P86F>	1
P8CR <P86C>	1

Figure3.5 (19) Registers for Port 8

3.5.10 Port 9 (P90 - P95)

Port 9 is a 6-bit general-purpose I/O port. I/Os can be set on a bit basis.

Resetting sets Port 9 to an input port and connects a pull-up resistor.

It also sets all bits of the output latch register to 1.

In addition to functioning as a general-purpose I/O port, Port 9 can also function as an I/O for serial channels 0 and1. Writing 1 in the corresponding bit of the port 9 function register (P9FC) enables this function.

Resetting resets the function register value to 0 and sets all bits to ports.

(1) Port 90 & 93 (TXD0/TXD1)

Ports 90 and 93 also function as serial channel TXD output pins in addition to I/O ports.

They have a programmable open drain function.

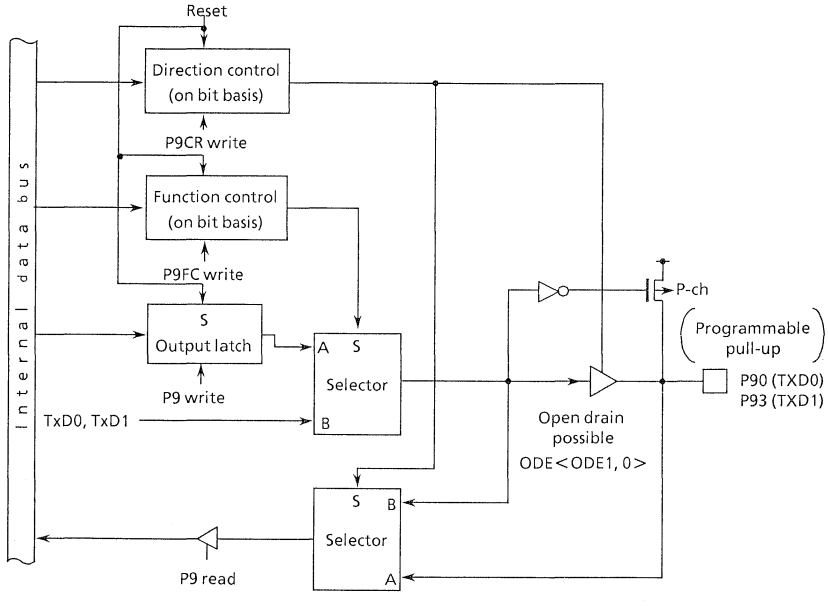


Figure 3.5 (20) Ports 90 and 93

(2) Ports 91 and 94 (RXD0, 1)

Ports 91 and 94 are I/O ports, and also used as RXD input pins for serial channels.

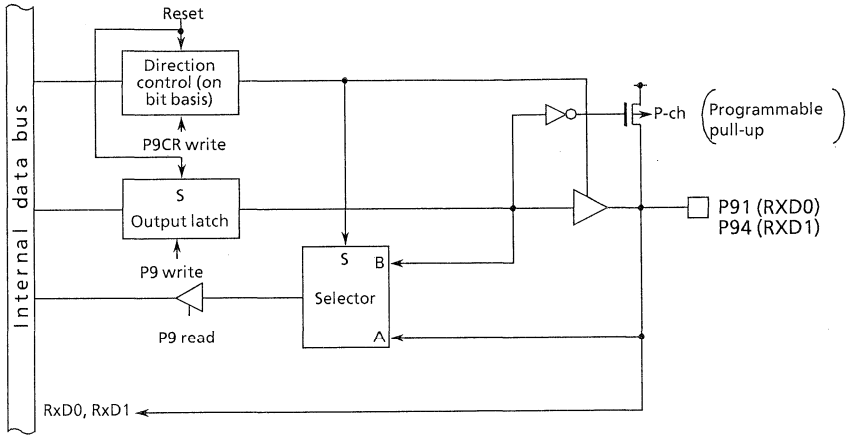


Figure3.5 (21) Ports 91 and 94

(3) Port 92 ($\overline{\text{CTS}}$)

Port 92 is an I/O port, and also used as a $\overline{\text{CTS}}$ input pin for serial channels.

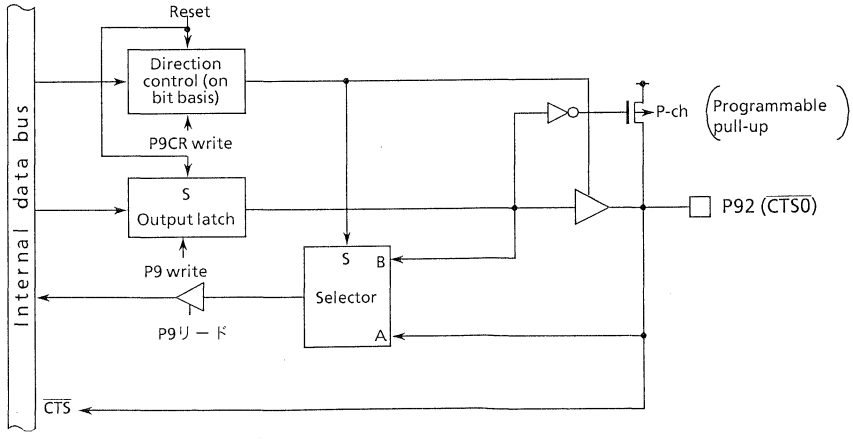


Figure3.5 (22) Port 92

(4) Port 95 (SCLK)

Port 95 is a general-purpose I/O port. It is also used as an SCLK I/O pin for serial channel 1.

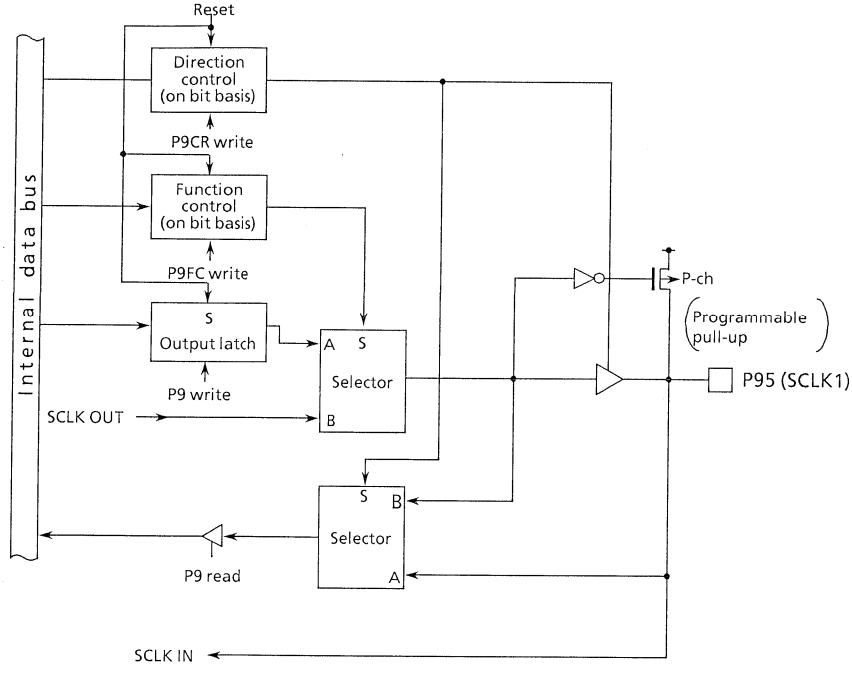


Figure3.5 (23) Port 95

Port 9 Register

P9 (0019H)

	7	6	5	4	3	2	1	0
bit Symbol			P95	P94	P93	P92	P91	P90
Read/Write			R/W					
After reset			Input mode					
			1	1	1	1	1	1

Port 9 Control Register

P9CR (001BH)

	7	6	5	4	3	2	1	0
bit Symbol			P95C	P94C	P93C	P92C	P91C	P90C
Read/Write			W					
After reset			0	0	0	0	0	0
Function			0 : IN			1 : OUT		

Port 9 I/O setting

0	Input
1	Output

Port 9 Function Register

P9FC (001DH)

	7	6	5	4	3	2	1	0
bit Symbol			P95F		P93F	P92F		P90F
Read/Write			W		W	W		W
After reset			0		0	0		0
Function			0 : PORT 1 : SCLK1		0 : PORT 1 : TxD1	0 : PORT 1 : SCLK0		0 : PORT 1 : TxD0

Note : Only the TMP96CM40 and TMP96PM40 have register P92F. That is, SCLK0 cannot be specified for the TMP96C141.

P90 TxD0 output setting (Note)

P9FC <P90F>	1
P9CR <P90C>	1

P93 TxD1 output setting (Note)

P9FC <P93F>	1
P9CR <P93C>	1

P95 SCLK output setting

P9FC <P95F>	1
P9CR <P95C>	1

Note : To set the TxD pin to open drain, write 1 in bit 0 (for TxD0 pin) or bit 1 (for TxD1 pin) of the ODE register.

Figure 3.5 (24) Registers for Port 9

3.6 Chip Select / Wait Control

TMP96C141/TMP96CM40/TMP96PM40 has a built-in chip select / wait controller used to control chip select ($\overline{CS0}$ - $\overline{CS2}$ pins), wait ($\overline{WAIT}$ pin), and data bus size (8 or 16 bits) for any of the three block address areas.

3.6.1 Control Registers

Table 3.6.(1) shows control registers.

One block address areas are controlled by 1-byte CS/WAIT control registers (B0CS, B1CS, and B2CS). Registers can be written to only when the CPU is in system mode (there are two CPU modes: system and normal). The reason is that the settings of these registers have an important effect on the system.

(1) Enable

Control register bit 7 (B0E, B1E, and B2E) is a master bit used to specify enable (1) / disable (0) of the setting.

Resetting sets B0E and B1E to disable (0) and B2E to enable (1).

(2) System only specification

Control resgister bit 6 (B0SYS, B1SYS, and B2SYS) is used to specify enable / disable of the setting depending on the CPU operating mode (system or normal). Setting this bit to 0 enables setting (Address space for $\overline{CS}$, Wait state, Bus size, etc.) regardless of the CPU operating mode; setting it to 1 enables setting in system mode but disables setting in normal mode.

Resetting clears bit 6 to 0.

Bit 6 is mainly used when external memory data should not be accessed in normal mode (ie, for system mode only memory data for the operating system).

(3) CS/CAS Waveform select

Control register bit 5 (B0CAS, B1CAS, and B2CAS) is used to specify waveform mode output from the chip select pin ($\overline{CS0}/\overline{CAS0}$ - $\overline{CS2}/\overline{CAS2}$). Setting this bit to 0 specifies $\overline{CS0}$ to $\overline{CS2}$ waveforms; setting it to 1 specifies CAS0 to CAS2 waveforms.

Resetting clears bit 5 to 0.

(4) Data bus size select

Bit 4 (B0BUS, B1BUS, and B2BUS) of the control register is used to specify data bus size. Setting this bit to 0 accesses the memory in 16-bit data bus mode; setting it to 1 accesses the memory in 8-bit data bus mode.

Changing data bus size depending on the access address is called dynamic bus sizing. Table 3.6 (2) shows the details of the bus operation.

(5) Wait control

Control register bits 3 and 2 (B0W1,0; B1W1,0; B2W1,0) are used to specify the number of waits. Setting these bits to 00 inserts a 2-state wait regardless of the $\overline{\text{WAIT}}$ pin status. Setting them to 01 inserts a 1-state wait regardless of the $\overline{\text{WAIT}}$ status. Setting them to 10 inserts a 1-state wait and samples the $\overline{\text{WAIT}}$ pin status. If the pin is low, inserting the wait maintains the bus cycle until the pin goes high. Setting them to 11 completes the bus cycle without a wait regardless of the $\overline{\text{WAIT}}$ pin status.

Resetting sets these bits to 00 (2-state wait mode).

(6) Address area specification

Control register bits 1 and 0 (B0C1,0; B1C1,0; B2C1,0) are used to specify the target address area. Setting these bits to 00 enables settings ($\overline{\text{CS}}$ output, Wait state, Bus size, etc.) as follows:

* CS0 setting enabled when 7F00H to 7FFFH is accessed.
* CS1 setting enabled when 480H to 7FFFH is accessed.
* CS2 setting enabled when 8000H to 3FFFFFFH is accessed, for the TMP96C141, which does not have a built-in ROM.
  CS2 setting enabled when 10000H to 3FFFFFH is accessed for the TMP96CM40/ TMP96PM40, which has built-in ROM/PROM

Setting bits to 01 enables setting for all CS's blocks and outputs a low strobe signal ($\overline{\text{CS0}}/\overline{\text{CAS0}}$~$\overline{\text{CS2}}/\overline{\text{CAS2}}$) from chip select pins when 400000H to 7FFFFFH is accessed. Setting bits to 10 enables them 800000H to BFFFFFH is accessed. Setting bits to 11 enables them when C00000H to FFFFFFH is accessed.

Table 3.6 (1) Chip select / wait control register

Code	Name	Address	7	6	5	4	3	2	1	0
B0CS	Block0 CS/WAIT control register	0068H	B0E	B0SYS	B0CAS	B0BUS	B0W1	B0W0	B0C1	B0C0
			W	W	W	W	W	W	W	W
			0	0	0	0	0	0	0	0
			1: CS/CAS Enable	1: SYSTEM only	0: $\overline{CS0}$ 1: $\overline{CAS0}$	0:16bit Bus 1:8bit Bus	00: 2WAIT 01: 1WAIT 10: 1WAIT + n 11: 0WAIT		00: 7F00H~7FFFH 01: 400000H~ 10: 800000H~ 11: C00000H~	
B1CS	Block1 CS/WAIT control register	0069H	B1E	B1SYS	B1CAS	B1BUS	B1W1	B1W0	B1C1	B1C0
			W	W	W	W	W	W	W	W
			0	0	0	0	0	0	0	0
			1: CS/CAS Enable	1: SYSTEM only	0: $\overline{CS1}$ 1: $\overline{CAS1}$	0:16bit Bus 1:8bit Bus	00: 2WAIT 01: 1WAIT 10: 1WAIT + n 11: 0WAIT		00: 480H~7FFFH 01: 400000H~ 10: 800000H~ 11: C00000H~	
B2CS	Block2 CS/WAIT control register	006AH	B2E	B2SYS	B2CAS	B2BUS	B2W1	B2W0	B2C1	B2C0
			W	W	W	W	W	W	W	W
			1	0	0	0	0	0	0	0
			1: CS/CAS Enable	1: SYSTEM only	0: $\overline{CS2}$ 1: $\overline{CAS2}$	0:16bit Bus 1:8bit Bus	00: 2WAIT 01: 1WAIT 10: 1WAIT + n 11: 0WAIT		00: 8000H~ 01: 400000H~ 10: 800000H~ 11: C00000H~	

Note : With only block 2, enable (16-bit data bus, 2-wait mode) after reset.

Table 3.6 (2) Dynamic bus sizing

Operand data size	Operand start address	Memory data size	CPU address	CPU data	
				D15 – D8	D7 – D0
8 bits	2n + 0 (even number)	8 bits	2n + 0	xxxxx	b7 – b0
		16 bits	2n + 0	xxxxx	b7 – b0
	2n + 1 (odd number)	8 bits	2n + 1	xxxxx	b7 – b0
		16 bits	2n + 1	b7 – b0	xxxxx
16 bits	2n + 0 (even number)	8 bits	2n + 0	xxxxx	b7 – b0
			2n + 1	xxxxx	b15 – b8
		16 bits	2n + 0	b15 – b8	b7 – b0
	2n + 1 (odd number)	8 bits	2n + 1	xxxxx	b7 – b0
			2n + 2	xxxxx	b15 – b8
		16 bits	2n + 1	b7 – b0	xxxxx
			2n + 2	xxxxx	b15 – b8
32 bits	2n + 0 (even number)	8 bits	2n + 0	xxxxx	b7 – b0
			2n + 1	xxxxx	b15 – b8
			2n + 2	xxxxx	b23 – b16
			2n + 3	xxxxx	b31 – b24
		16 bits	2n + 0	b15 – b8	b7 – b0
			2n + 2	b31 – b24	b23 – b16
	2n + 1 (odd number)	8 bits	2n + 1	xxxxx	b7 – b0
			2n + 2	xxxxx	b15 – b8
			2n + 3	xxxxx	b23 – b16
			2n + 4	xxxxx	b31 – b24
		16 bits	2n + 1	b7 – b0	xxxxx
			2n + 2	b23 – b16	b15 – b8
			2n + 4	xxxxx	b31 – b24

xxxxx : During a read, data input to the bus is ignored. At write, the bus is at high impedance and the write strobe signal remains non-active.

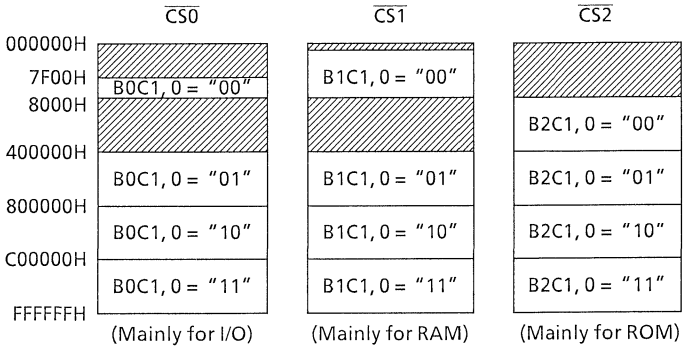
3.6.2 Chip Select Image

An image of the actual chip select is shown below. Out of the whole memory area, address areas that can be specified are divided into four parts. Addresses from 000000H to 3FFFFFH are divided differently: 7F00H to 7FFFH is specified for CS0; 480H to 7FFFH, for CS1; and 8000H to 3FFFFFH, for CS2. The reason is that a device other than ROM (ie, RAM or I/O) might be connected externally.

7F00 to 7FFFH (256 bytes) for CS0 are mapped mainly for possible expansions to external I/O.

480H to 7FFFH (approx. 31K bytes) for CS1 are mapped there mainly for possible extensions to external RAM.

8000H to 3FFFFFFH (approx. 4M bytes) for CS2 are mapped mainly for possible extensions to external ROM. After reset, CS2 is enabled in 16-bit bus and 2-wait. With the TMP96C141, which does not have a built-in ROM, the program is externally read at address 8000II in this setting (16-bit bus, 2-wait). With the TMP96CM40/TMP96PM40, which has a built-in ROM, addresses from 8000H to FFFFFH are used as the internal ROM area; CS2 is disabled in this area. After reset, the CPU reads the program from the built-in ROM in 16-bit bus, 0-wait mode.



Note 1 : Access priority is highest for built-in I/O, then built-in memory, and lowest for the chip select/wait controller.

Note 2 : External areas other than $\overline{CS0}$ to $\overline{CS2}$ are accessed in 16-bit data bus (0 wait) mode.
When using the chip select/wait controller, do not specify the same address area more than once. (However, when addresses 7F00H - 7FFFH for CS0 and 480H - 7FFFH for CS1 are specified, in other words, specifications overlap, only the CS0 setting/pin is active.)

3.6.3 Example of Usage

Figure 3.6 (1) is an example in which an external memory is connected to the TMP96C141. In this example, a ROM is connected using 16 bit Bus; a RAM is connected using 8 bit Bus.

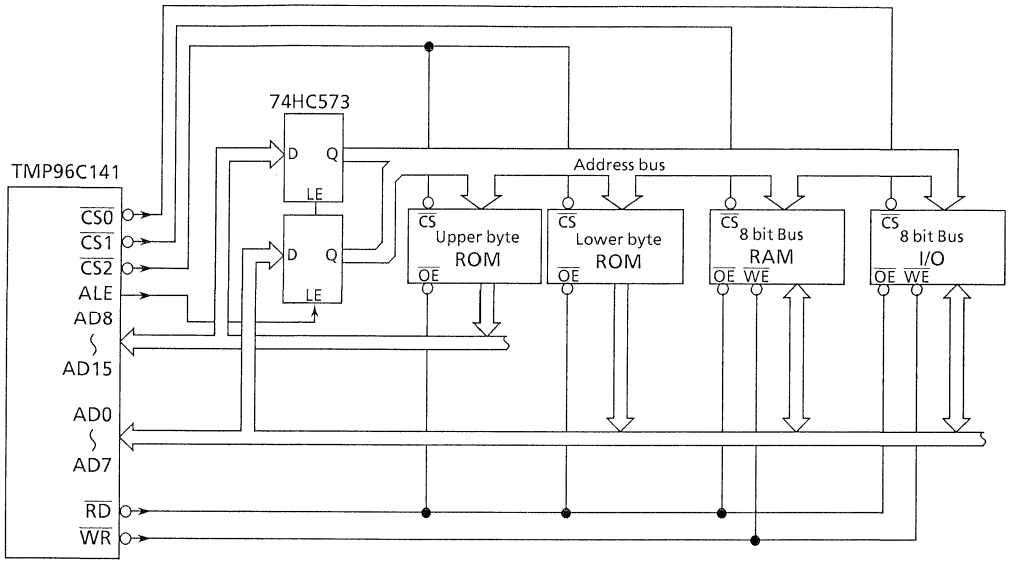


Figure 3.6 (1) Example of External Memory Connection (ROM = 16 bits, RAM & I/O = 8 bits)

Resetting sets pins $\overline{CS0}$ to $\overline{CS2}$ to input port mode. $\overline{CS0}$ and $\overline{CS1}$ are set high due to an internal pull-up resistor; $\overline{CS2}$, low due to an internal pull-down resistor. The program used to set these pins is as follows.

```
P4CR    EQU     0EH
P4FC    EQU     10H
B0CS    EQU     68H
B1CS    EQU     69H
B2CS    EQU     6AH
LD      (B0CS),90H      ; CS0 = 8 bits, 2WAIT, 7F00H~7FFFH
LD      (B1CS),9CH      ; CS1 = 8 bits, 0WAIT, 480H~7EFFH
LD      (B2CS),84H      ; CS2 = 16 bits, 1WAIT, 8000H~3FFFFFH
LD      (P4CR),07H      } CS0, CS1, CS2 output mode setting
LD      (P4FC),07H      }
```

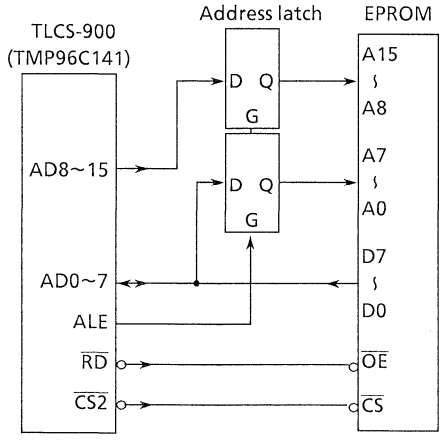
3.6.4 How to Start with an 8-bit Data Bus (with TMP96C141)

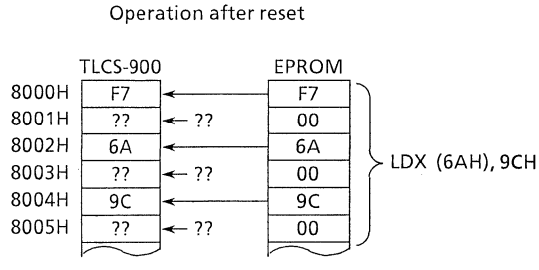
Resetting sets the $\overline{CS2}$ pin low due to an internal pull-down resistor; memory access starts in 16-bit data bus (2-wait) mode. To start in 8-bit data bus mode, a special operation is required. Operation is as described in the example below.

```
B2CS    EQU    6AH              ; CS2 register address
        ORG    8000H            ; RESET address
        LDX    (B2CS), 9CH      ; CS2 8bit, 0WAIT, 8000H~
```

After reset, the program reads the LDX(B2CS),9CH instruction in 16-bit data bus mode. LDX is a 6-byte instruction: the 2nd, 4th, and 6th bytes are handled as dummies (ie, only codes in the 1st, 3rd, and 5th bytes are actually used). Even if starting in 8-bit data bus mode, it is possible to program so that the LDX instruction is executed and the block 2 area (8000H - 3FFFFFH) is accessed in 8-bit data bus mode without any problem.

The above program does not include setting the P42/$\overline{CS2}$ pin to output; add a program to set the P4CR and P4FC registers as required.





3.7 8-bit Timers

TMP96C141/TMP96CM40/TMP96PM40 contains two 8-bit timers (timers 0 and 1), each of which can be operated independently. The cascade connection allows these timers to be used as 16-bit timer. The following four operating modes are provided for the 8-bit timers.

- 8-bit interval timer mode (2 timers)
- 16-bit interval timer mode (1 timer)
- 8-bit programmable square wave pulse generation (PPG: variable duty with variable cycle) output mode (1 timer)
- 8-bit pulse width modulation (PWM: variable duty with constant cycle) output mode (1 timer)

Figure 3.7 (1) shows the block diagram of 8-bit timer (timer 0 and timer 1).

Each interval timer consists of an 8-bit up-counter, 8-bit comparator, and 8-bit timer register. Besides, one timer flip-flop (TFF1) is provided for pair of timer 0 and timer 1.

Among the input clock sources for the interval timers, the internal clocks of ϕT1, ϕT4, ϕT16, and ϕT256 are obtained from the 9-bit prescaler shown in Figure 3.7 (2).

The operation modes and timer flip-flops of the 8-bit timer are controlled by three control registers TMOD, TFFCR, and TRUN.

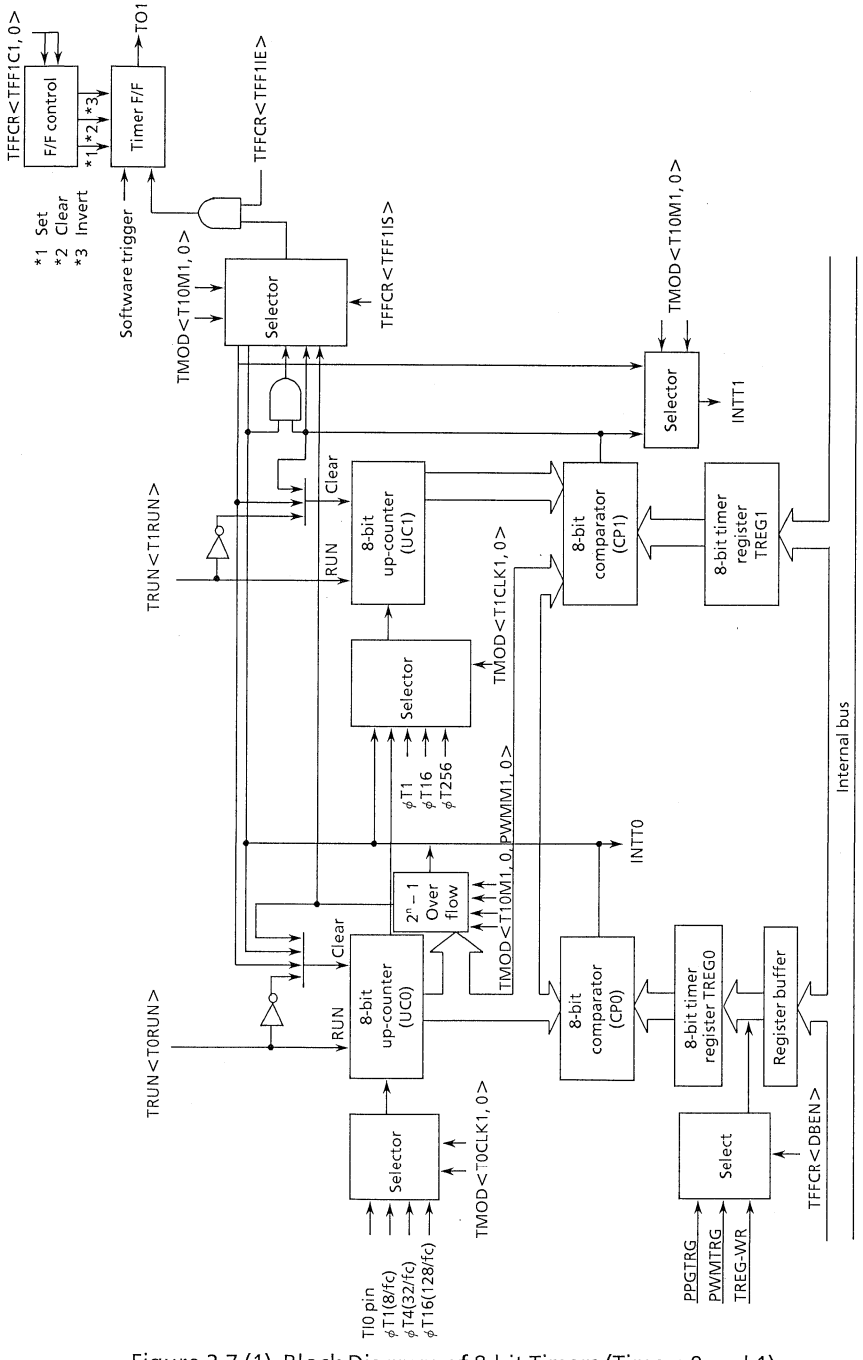


Figure 3.7 (1) Block Diagram of 8-bit Timers (Timers 0 and 1)

① Prescaler

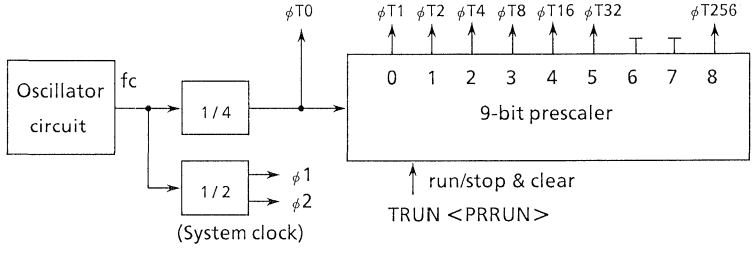
This 9-bit prescaler generates the clock input to the 8-bit timers, 16-bit timer/event counters, and baud rate generators by further dividing the fundamental clock (fc) after it has been divided by 4 (fc/4).

Among them, 8-bit timer uses 4 types of clock: ϕT1, ϕT4, ϕT16, and ϕT256.

This prescaler can be run or stopped by the timer operation control register TRUN<PRRUN>. Counting starts when <PRRUN> is set to "1", while the prescaler is cleared to zero and stops operation when <PRRUN> is set to "0". Resetting clears <PRRUN> to "0", which clears and stops the prescaler.

Cycle

Input clock \ fc	16MHz	20MHz
ϕT1 (8/fc)	0.5μs	0.4μs
ϕT4 (32/fc)	2.0μs	1.6μs
ϕT16 (128/fc)	8.0μs	6.4μs
ϕT256 (2048/fc)	128μs	102μs



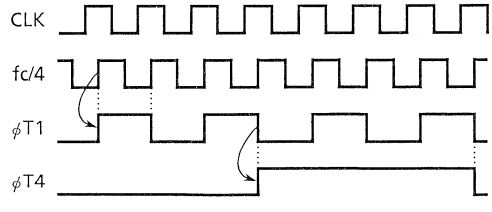


Figure 3.7 (2) Prescaler

② Up-counter

This is an 8-bit binary counter which counts up by the input clock pulse specified by TMOD.

The input clock of timer 0 is selected from the external clock from TI0 pin and the three internal clocks ϕT1 (8/fc), ϕT4 (32/fc), and ϕT16 (128/fc), according to the set value of TMOD register.

The input clock of timer 1 differs depending on the operation mode. When set to 16-bit timer mode, the overflow output of timer 0 is used as the input clock. When set to any other mode than 16-bit timer mode, the input clock is selected from the internal clocks ϕT1 (8/fc), ϕT16 (128/fc), and ϕT256 (2048/fc) as well as the comparator output (match detection signal) of timer 0 according to the set value of TMOD register.

Example : When TMOD<T10M1,0>=01, the overflow output of timer 0 becomes the input clock of timer 1 (16-bit timer mode).
When TMOD<T10M1,0>=00 and TMOD<T1CLK1,0>=01, ϕT1 (8/fc) becomes the input of timer 1 (8bit timer mode).

Operation mode is also set by TMOD register. When reset, it is initialized to TMOD<T01M1, 0>=00 whereby the up-counter is placed in the 8-bit timer mode.

The counting and stop & clear of up-counter can be controlled for each interval timer by the timer operation control register TRUN. When reset, all up-counters will be cleared to stop the timers.

③ Timer register

This is an 8-bit register for setting an interval time. When the set value of timer registers TREG0, TREG1, matches the value of up-counter, the comparator match detect signal becomes active. If the set value is 00H, this signal becomes active when the up-counter overflows.

Timer register TREG0 is of double buffer structure, each of which makes a pair with register buffer.

The timer flip-flop controll register TFFCR<DBEN> bit controls whether the double buffer structure in the TREG0 should be enabled or disabled. It is disabled when <DBEN>=0 and enabled when they are set to 1.

In the condition of double buffer enable state, the data is transfered from the register buffer to the timer register when the 2^n-1 overflow occurs in PWM mode, or at the PPG cycle in PPG mode. Therefore, during timer mode, the double buffer can not be used.

When reset, it will be initialized to <DBEN>=0 to disable the double buffer. To use the double buffer, write data in the timer register, set <DBEN> to 1, and

write the following data in the register buffer.

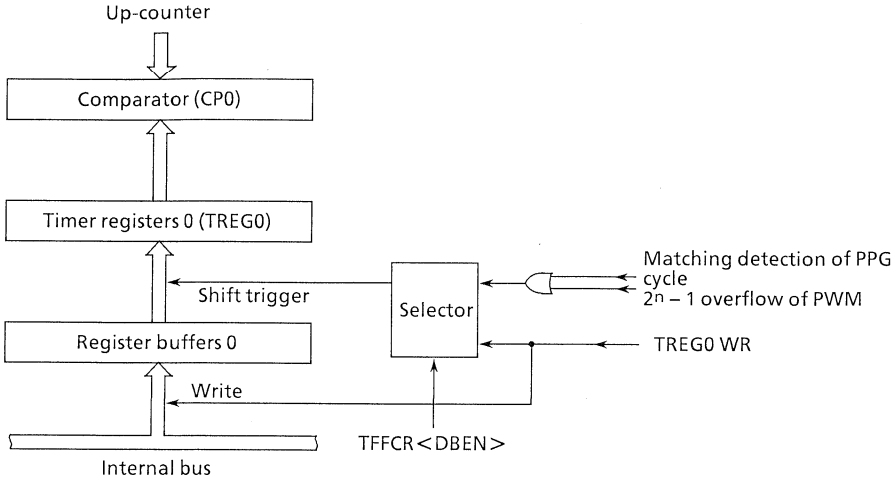


Figure 3.7 (3) Configuration of Timer Register 0

Note : Timer register and the register buffer are allocated to the same memory address. When <DBEN> = 0, the same value is written in the register buffer as well as the timer register, while when <DBEN> = 1 only the register buffer is written.

The memory address of each timer register is as follows.

TREG0: 000022H

TREG1: 000023H

All the registers are write-only and cannot be read.

④ Comparator

A comparator compares the value in the up-counter with the values to which the timer register is set. When they match, the up-counter is cleared to zero and an interrupt signal (INTT0, INTT1) is generated. If the timer flip-flop inversion is enabled, the timer flip-flop is inverted at the same time.

⑤ Timer flip-flop (timer F/F : TFF1)

The status of the timer flip-flop is inverted by the match detect signal (comparator output) of each interval timer and the value can be output to the timer output pins TO1 (also used as P71).

A timer F/F is provided for a pair of timer 0 and timer 1 and is called TFF1. TFF1 is output to TO1 pin.

TRUN (0020H)

	7	6	5	4	3	2	1	0
bit Symbol	PRRUN		T5RUN	T4RUN	P1RUN	P0RUN	T1RUN	T0RUN
Read/Write	R/W		R/W					
After reset	0		0	0	0	0	0	0
Function	Prescaler & Timer Run/Stop CONTROL 0 : Stop & Clear 1 : Run (Count up)							

Count Operation

0	Stop and clear
1	Count

PRRUN : Operation of prescaler
T5RUN : Operation of 16-bit timer (timer5)
T4RUN : Operation of 16-bit timer (timer4)
P1RUN : Operation of PWM timer (PWM1/timer3)
P0RUN : Operation of PWM timer (PWM0/timer2)
T1RUN : Operation of 8-bit timer (timer1)
T0RUN : Operation of 8-bit timer (timer0)

Figure 3.7 (4) Timer Operation Control Register (TRUN)

TMOD (0024H)

	7	6	5	4	3	2	1	0
bit Symbol	T10M1	T10M0	PWMM1	PWMM0	T1CLK1	T1CLK0	T0CLK1	T0CLK0
Read/Write	W							
After reset	0	0	0	0	0	0	0	0
Function	Operation mode 00: 8bit Timer 01: 16bit Timer 10: 8bit PPG 11: 8bit PWM		PWM cycle 00: – 01: 2^6-1 10: 2^7-1 11: 2^8-1		Source clock of timer1 00: TO0TRG 01: ϕT1 10: ϕT16 11: ϕT256		Source clock of timer0 00: TI0 01: ϕT1 10: ϕT4 11: ϕT16	

Prohibit Read Modify Write

Input clock of timer 0

00	External Input (TI0)
01	ϕT1 (Prescaler)
10	ϕT4 (Prescaler)
11	ϕT16 (Prescaler)

Input clock of timer 1

	TMOD<T10M1,0> ≠ 01	TMOD <T10M1,0> = 01
00	Comparator output of timer 0	Overflow output of timer 0
01	Internal clock ϕT1	
10	Internal clock ϕT16	(16-bit timer mode)
11	Internal clock ϕT256	

Select PWM cycle

00	——
01	$2^6 - 1$
10	$2^7 - 1$
11	$2^8 - 1$

Set the operation mode of timer 0 and 1.

00	Two 8-bit timers (timer 0 and timer 1)
01	16-bit timer
10	8-bit PPG output
11	8-bit PWM output (timer 0) + 8-bit timer (timer 1)

Figure 3.7 (5) Timer Mode control Register (TMOD)

TFFCR (0025H)

	7	6	5	4	3	2	1	0
bit Symbol				DBEN	TFF1C1	TFF1C0	TFF1IE	TFF1IS
Read/Write				R/W	W		R/W	
After reset				0	0	0	0	0
Function				Double buffer 0: Disable 1: Enable	00: Invert TFF1 01: Set TFF1 10: Clear TFF1 11: don't care ※ Always read as "11"		TFF1 Inversion trigger 0: Disable 1: Enable	TFF1 Inversion source 0: Timer 0 1: Timer 1

Select inverse signal of timer F/F1
("Don't care" except in 8-bit timer mode)

	TMOD<T10M1,0> = 00	01	10	11
0	Inversion by timer0 match signal	16-bit timer mode Inversion by match signal	PPG mode Inversion by match signal of each timer 0 and timer 1	PWM mode Inversion by match and overflow signal of timer 0
1	Inversion by timer1 match signal			

Inversion of Timer F/F1 (TFF1)

0	Disable invert
1	Enable invert

Control of Timer F/F1 (TFF1)

00	Invert the value of TFF1 (software inversion)
01	Set TFF1 to "1".
10	Clear TFF1 to "0".
11	Don't care

Double Buffer Control of TREG0

0	Disable double Buffer
1	Enable double Buffer

Figure 3.7 (6) Timer Flip-flop Control Register (TFFCR)

The operation of 8-bit timers will be described below:

(1) 8-bit timer mode

Two interval timers 0, 1, can be used independently as 8-bit interval timer. All interval timers operate in the same manner, and thus only the operation of timer 1 will be explained below.

① Generating interrupts in a fixed cycle

To generate timer 1 interrupt at constant intervals using timer 1 (INTT1), first stop timer 1 then set the operation mode, input clock, and a cycle to TMOD and TREG1 register, respectively. Then, enable interrupt INTT1 and start the counting of timer 1.

Example : To generate timer 1 interrupt every 40 microseconds at fc=16 MHz, set each register in the following manner.

```
          MSB          LSB
            7 6 5 4 3 2 1 0
TRUN      ← - X - - - - 0 -   Stop timer 1, and clear it to "0".
TMOD      ← 0 0 X X 0 1 - -   Set the 8-bit timer mode, and select φT1 (0.5 μs @ fc = 16
                              MHz) as the input clock.
TREG1     ← 0 1 0 1 0 0 0 0   Set the timer register at 40 μs φT1 = 50H.
INTET10   ← 1 1 0 1 - - - -   Enable INTT1, and set it to "Level 5".
TRUN      ← 1 X - - - - 1 -   Start timer 1 counting.
```

Note : ×: don't care –; no change

Use the following table for selecting the input clock.

Table 3.7 (1) 8-Bit Timer Interrupt Cycle and Input Clock

Input clock	Interrupt cycle (at fc = 16MHz)	Resolution	Interrupt cycle (at fc = 20MHz)	Resolution
φT1 (8/fc)	0.5 μs ~ 128 μs	0.5 μs	0.4 μs~102.4 μs	0.4 μs
φT4 (32/fc)	2 μs ~ 512 μs	2 μs	1.6 μs~409.6 μs	1.6 μs
φT16 (128/fc)	8 μs ~ 2.048 ms	8 μs	6.4 μs~1.638 ms	6.4 μs
φT256 (2048/fc)	128 μs~32.708 ms	128 μs	102.4 μs~2.621 ms	102.4 μs

Note : The input clock of timer 0 and timer 1 are different from as follows.

Timer 0 : TI0 input, φT1, φT4, φT16

Timer 1 : Match Output of Timer 0, φT1, φT16, φT256

② Generating a 50% duty square wave pulse

The timer flip-flop (TFF1) is inverted at constant intervals, and its status is output to timer output pin (TO1).

Example : To output a 3.0 μs square wave pulse from TO1 pin at fc = 16 MHz, set each register in the following procedures. Either timer 0 or timer 1 may be used, but this example uses timer 1.

```
          7 6 5 4 3 2 1 0
TRUN  ← - X - - - - 0 -     Stop timer 1, and clear it to "0".
TMOD  ← 0 0 X X 0 1 - -     Set the 8-bit timer mode, and select φT1 (0.5 μs @ fc = 16 MHz) as
                            the input clock.
TREG1 ← 0 0 0 0 0 0 1 1     Set the timer register at 3.0 μs ÷ φT1 ÷ 2 = 3.
TFFCR ← - - - - 1 0 1 1     Clear TFF1 to "0", and set to invert by the match detect signal
                            from timer 1.
P7CR  ← X X X X - - 1 -  }  Select P71 as TO1 pin.
P7FC  ← X X X X - - 1 X  }
TRUN  ← 1 X - - - - 1 -     Start timer 1 counting.
```

Note : X ; don't care - ; no change

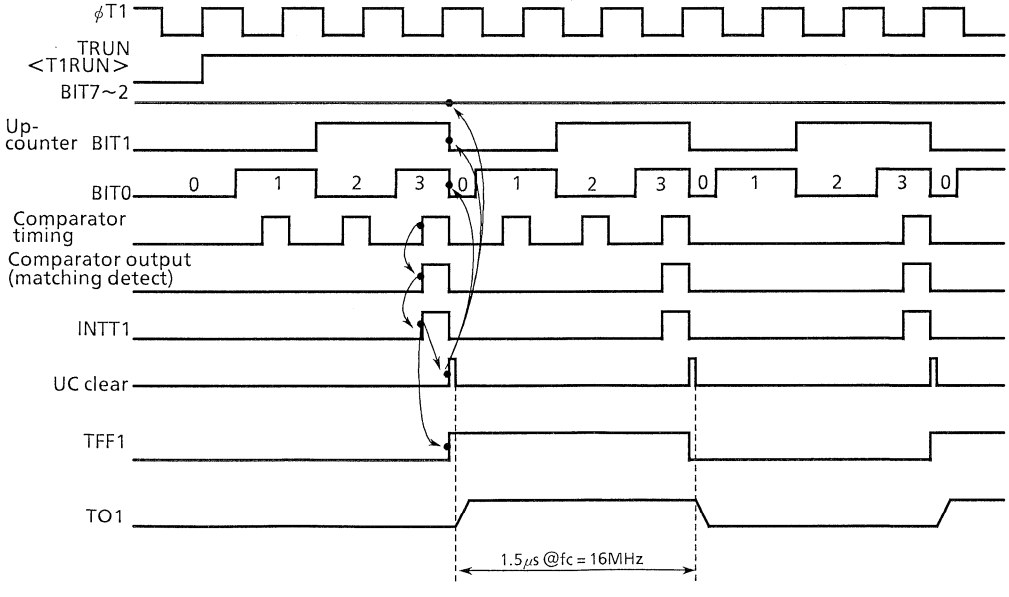


Figure 3.7 (7) Square Wave (50% Duty) Output Timing Chart

③ Making timer 1 count up by match signal from timer 0 comparator

Set the 8-bit timer mode, and set the comparator output of timer 0 as the input clock to timer 1.

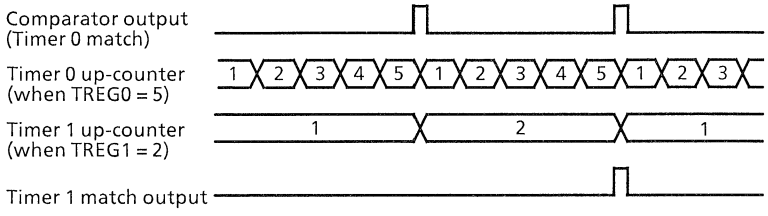


Figure 3.7 (8) Timer 1 count up by timer 0

④ Output inversion with software

The value of timer flip-flop (TFF1) can be inverted, independent of timer operation.

Writing "00" into TFFCR<TFF1C1, 0> (memory address : 000025h of bit 3 and bit 2) inverts the value of TFF1.

⑤ Initial setting of timer flip-flop (TFF1)

The value of TFF1 can be initialized to "0" or "1", independent of timer operation.

For example, write "10" in TFFCR<TFF1C1,0> to clear TFF1 to "0", while write "01" in TFFCR<TFF1C1,0> to set TFF1 to "1".

Note: The value of timer register cannot be read.

(2) 16-bit timer mode

A 16-bit interval timer is configured by using the pair of timer 0 and timer 1.

To make a 16-bit interval timer by cascade connecting timer 0 and timer 1, set timer 0/timer 1 mode register TMOD<T10M1,0> to "0, 1".

When set in 16-bit timer mode, the overflow output of timer 0 will become the input clock of timer 1, regardless of the set value of TMOD<T1CLK1,0>. Table 3.7 (2) shows the relation between the cycle of timer (interrupt) and the selection of input clock.

Table 3.7 (2) 16-Bit Timer (Interrupt) and Input Clock

Input clock	Interrupt cycle (fc = 16MHz)	Resolution	Interrupt cycle (fc = 20MHz)	Resolution
ϕT1 (8/fc)	0.5 μs ~ 32.786ms	0.5 μs	0.4 μs ~ 26.214ms	0.4 μs
ϕT4 (32/fc)	2 μs ~ 131.072ms	2 μs	1.6 μs ~ 104.857ms	1.6 μs
ϕT16 (128/fc)	8 μs ~ 524.288ms	8 μs	6.4 μs ~ 419.430ms	6.4 μs

The lower 8 bits of the timer (interrupt) cycle are set by the timer register TREG0, and the upper 8 bits are set by TREG1. Note that TREG0 always must be set first. (Writing data into TREG0 disables the comparator temporarily, and the comparator is restarted by writing data into TREG1.)

Setting example: To generate an interrupt INTT1 every 0.5 seconds at fc=16 MHz, set the following values for timer registers TREG0 and TREG1.

When counting with input clock of ϕT16 (8μs @ 16 MHz)

0.5sec÷8μs=62500=F424H

Therefore, set TREG1=F4H and TREG0=24H, respectively.

The comparator match signal is output from timer 0 each time the up-counter UC0 matches TREG0, where the up-counter UC0 is not be cleared.

With the timer 1 comparator, the match detect signal is output at each comparator timing when up-counter UC1 and TREG1 values match. When the match detect signal is output simultaneously from both comparators of timer 0 and timer 1, the up-counters UC0 and UC1 are cleared to "0", and the interrupt INTT1 is generated. If inversion is enabled, the value of the timer flip-flop TFF1 is inverted.

Example : When TREG1=04H and TREG0=80H

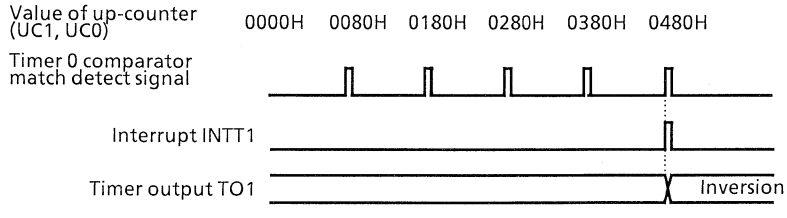


Figure 3.7 (9) Output timer by 16-bit timer mode

(3) 8-bit PPG (Programmable Pulse Generation) Output mode

Square wave pulse can be generated at any frequency and duty by timer 0 and timer 1. The output pulse may be either low-active or high-active. In this mode, timer 1 cannot be used.

Timer 0 outputs pulse to TO1 pin (also used as P70).

In this mode, a programmable square wave is generated by inverting timer output each time the 8-bit up-counter (UC0) matches the timer registers TREG0 and TREG1.

However, it is required that the set value of TREG0 is smaller than that of TREG1.

Though the up-counter (UC1) of timer 1 is not used in this mode, UC1 should be set for counting by setting TRUN<T1RUN>to 1.

Figure 3.7 (11) shows the block diagram for this mode.

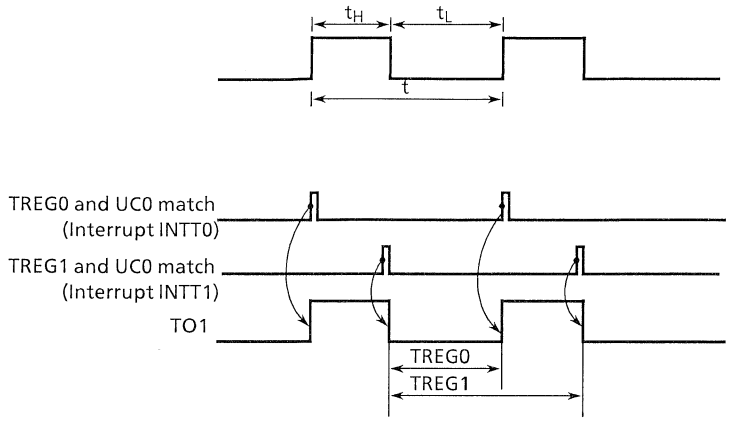


Figure 3.7 (10) 8bit PPG output waveforms

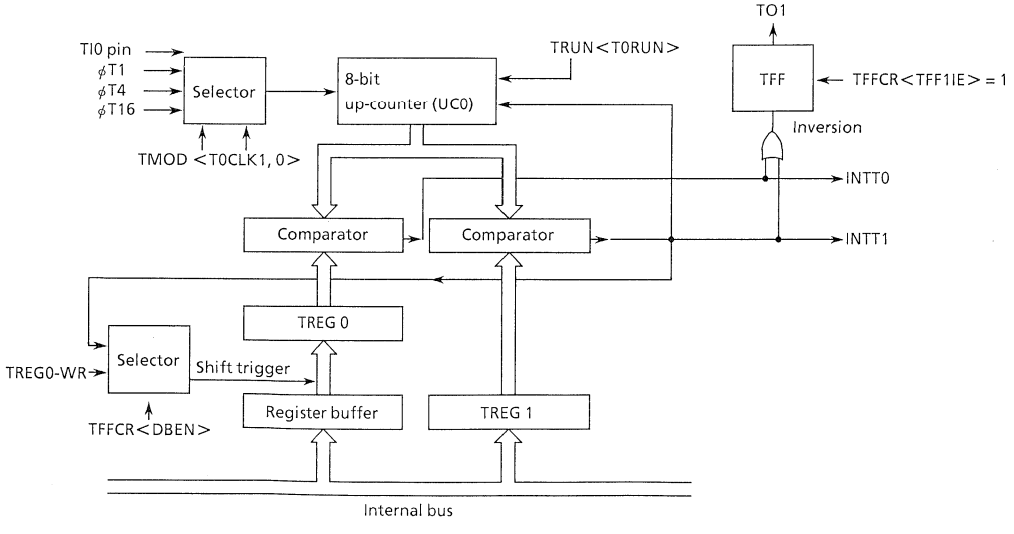


Figure 3.7 (11) Block Diagram of 8-Bit PPG Output Mode

When the double buffer of TREG0 is enabled in this mode, the value of register buffer will be shifted in TREG0 each time TREG1 matches UC0.

Use of the double buffer makes easy the handling of low duty waves (when duty is varied).

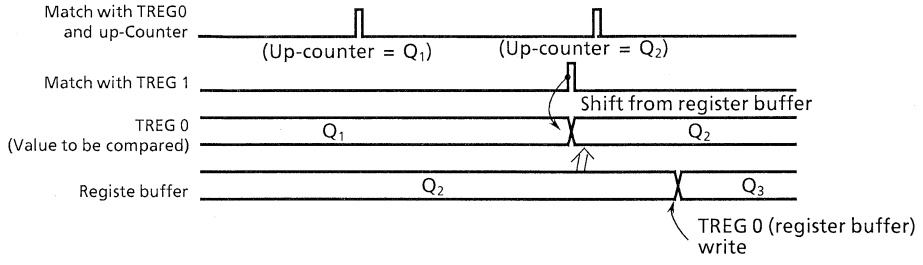
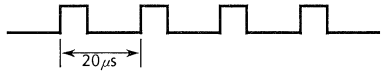


Figure 3.7 (12) Operation of Register buffer

Example : Generating 1/4 duty 50 KHz pulse (@ fc=16 MHz)



- Calculate the value to be set for timer register.
  To obtain the frequency 50 KHz, the pulse cycle t should be : t=1/50 KHz = 20 μs.
  Given φT1=0.5μs (@ 16 Hz),
  20μs÷0.5μs=40
  Consequently, to set the timer register 1 (TREG1) to TREG1=40=28H
  and then duty to 1/4, t×1/4 = 20μs×1/4=5μs
  5μs÷0.5μs=10
  Therefore, set timer register 0 (TREG0) to TREG0=10=0AH.

```
          7 6 5 4 3 2 1 0
TRUN  ← - X - - - - 0 0     Stop timer 0, and clear it to "0".
TMOD  ← 1 0 X X X X 0 1     Set the 8-bit PPG mode, and select φT1 as input clock.
TREG0 ← 0 0 0 0 1 0 1 0     Write "0AH".
TREG1 ← 0 0 1 0 1 0 0 0     Write "28H".
TFFCR ← - - - 1 0 1 1 X     Sets TFF1 and enable the inversion and double buffer
                            enable.
                            Writing "10" provides negative logic pulse.
P7CR  ← X X X X - - 1 -     Set P71 as the TO1 pin.
P7FC  ← X X X X - - 1 X
TRUN  ← 1 X - - - - 1 1     Start timer 0 and timer 1 counting.
```

Note : X ; don't care - ; no change

(4) 8-bit PWM Output mode

This mode is valid only for timer 0. In this mode, maximum 8-bit resolution of PWM pulse can be output.

PWM pulse is output to TO1 pin (also used as P71) when using timer 0. Timer 1 can also be used as 8-bit timer.

Timer output is inverted when up-counter (UC0) matches the set value of timer register TREG0 or when $2n-1$ (n=6, 7, or 8; specified by T01MOD<PWM01,0>) counter overflow occurs. Up-counter UC0 is cleared when $2n-1$ counter overflow occurs. For example, when n=6, 6-bit PWM will be outputted, while when n=7, 7-bit PWM will be outputted.

To use this PWM mode, the following conditions must be satisfied.

(Set value of timer register) < (Set value of 2^n-1 counter overflow)
(Set value of timer register) ≠ 0

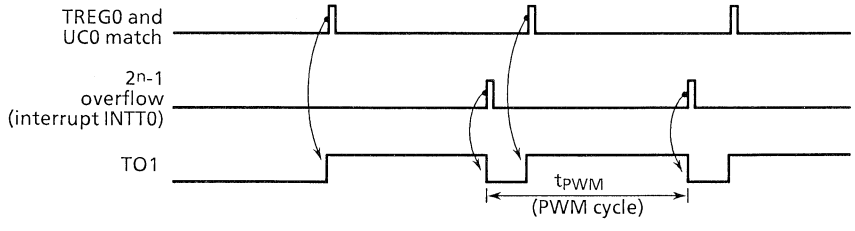


Figure 3.7(13) 8-bit PWM waveforms

Figure 3.7 (14) shows the block diagram of this mode.

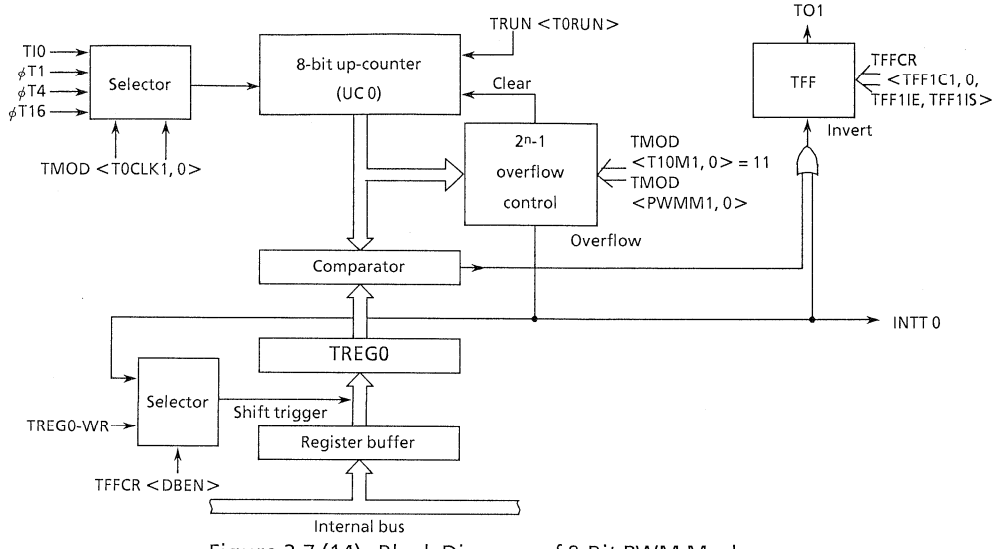


Figure 3.7 (14) Block Diagram of 8-Bit PWM Mode

In this mode, the value of register buffer will be shifted in TREG0 if 2^n-1 overflow is detected when the double buffer of TREG0 is enabled.

Use of the double buffer makes easy the handling of small duty waves.

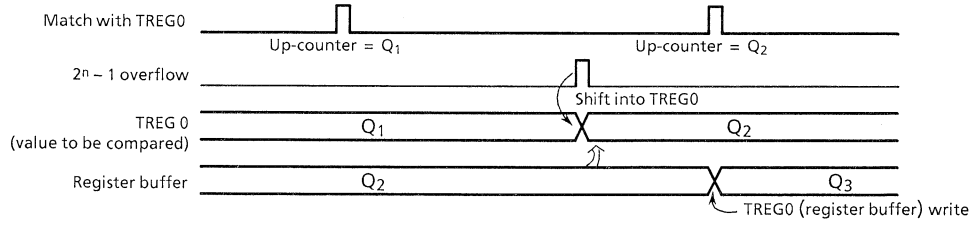
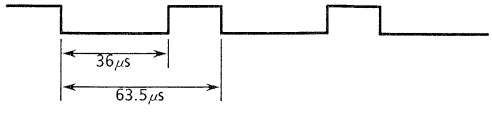


Figure 3.7 (15) Operation of Register buffer

Example : To output the following PWM waves to TO1 pin at fc = 16 MHz.



To realize 63.5μs of PWM cycle by ϕT1 = 0.5μs (@fc = 16 MHz),

$63.5\mu s \div 0.5\mu s = 127 = 2^7 - 1$

Consequently, n should be set to 7.

As the period of low level is 36μs, for ϕT1 = 0.5μs,

set the following value for TREG0.

$36\mu s \div 0.5\mu s = 72 = 48H$

```
          MSB              LSB
            7 6 5 4 3 2 1 0
TRUN  ← – X – – – – – 0      Stop timer 0, and clear it to "0".
TMOD  ← 1 1 1 0 – – 0 1      Set 8-bit PWM mode (cycle: 2^7 – 1) and select φT1 as the input
                             clock.
TREG0 ← 0 1 0 0 1 0 0 0      Writes "48H".
TFFCR ← X X X X 1 0 1 X      Clears TFF1, enable the inversion and double buffer.

P7CR  ← X X X X – – 1 –  }
P7FC  ← X X X X – – 1 X  }   Set P71 as the TO1 pin.
TRUN  ← 1 X – – – – – 1      Start timer 0 counting.
```

Note : X ; don't care – ; no change

Table 3.7 (3) PWM Cycle and the Setting of 2^n – 1 Counter

	PWM cycle (@ fc = 16MHz)			PWM cycle (@ fc = 20MHz)		
	φT1	φT4	φT16	φT1	φT4	φT16
2^6-1	31.5μsec (31.7KHz)	126msec (7.9KHz)	0.50μsec (1.9KHz)	25.2μsec (39.0KHz)	100μsec (10.0KHz)	0.40msec (2.4KHz)
2^7-1	63.5μsec (15.7KHz)	254msec (3.9KHz)	1.01μsec (0.98KHz)	50.8μsec (19.7KHz)	203μsec (4.9KHz)	0.81msec (1.2KHz)
2^8-1	127μsec (7.8KHz)	510msec (1.9KHz)	2.04μsec (0.49KHz)	102μsec (9.80KHz)	408μsec (2.4KHz)	1.63msec (0.61KHz)

(5) Table 3.7 (4) shows the list of 8-bit timer modes.

Table 3.7 (4) Timer Mode Setting Registers

Register name	TMOD				TFFCR
Name of function in	T10M	PWMM	T1CLK	T0CLK	TFF1IS
Function	Timer mode	PWM0 cycle	Upper timer input clock	Lower timer input clock	Timer F/F invert signal select
16-bit timer mode	01	–	–	External clock, φT1, φT4, φT16 (00, 01, 10, 11)	–
8-bit timer × 2 channels	00	–	Lower timer match: φT1, 16, 256 (00, 01, 10, 11)	External clock, φT1, φT4, φT16 (00, 01, 10, 11)	0: Lower timer output 1: Upper timer output
8-bit PPG x 1channel	10	–	–	External clock, φT1, φT4, φT16 (00, 01, 10, 11)	–
8-bit PWM x 1channel	11	2^6-1, 2^7-1, 2^8-1 (01, 10, 11)	–	External clock, φT1, φT4, φT16 (00, 01, 10, 11)	–
8-bit timer x 1channel	11	–	φT1, φT16, φT256 (01, 10, 11)	–	Output disabled

Note : - ; Don't care

3.8 8-bit PWM Timer

The TMP96C141/TMP96CM40/TMP96PM40 has two built-in 8-bit PWM timers (timers 2 & 3).

They have two operating modes.

- 8-bit PWM (pulse width modulation: variable duty at fixed interval) output mode
- 8-bit interval timer mode

Figure 3.8 (1) is a block diagram of 8-bit PWM timer (timers 2 & 3).

PWM timers consist of an 8-bit up-counter, 8-bit comparator, and 8-bit timer register. Two timer flip-flops (TFF2 for timer 2 and TFF3 for timer 3) are provided.

Input clocks ϕP1, ϕP4, and ϕP16 for the PWM timers can be obtained using the built-in prescaler.

PWM timer operating mode and timer flip-flops are controlled by four control registers (P0MOD, P1MOD, PFFCR, and TRUN).

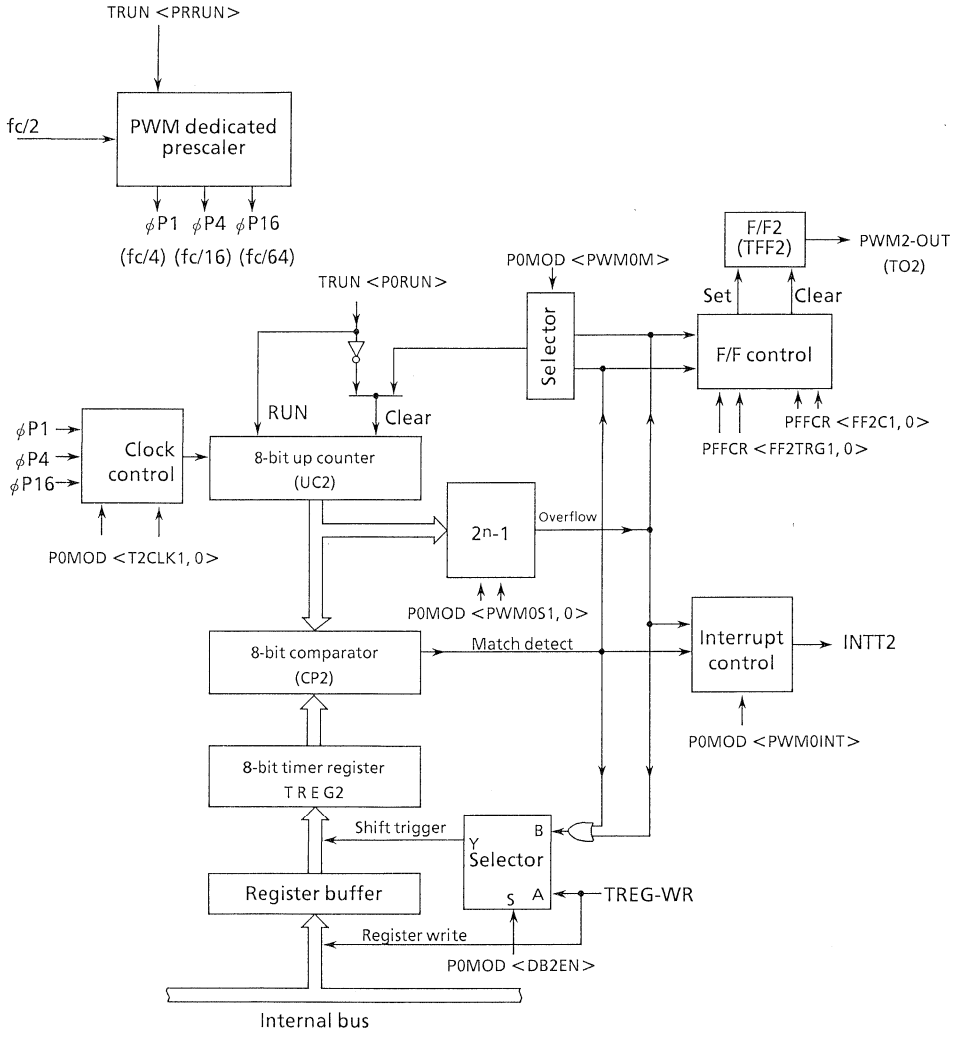


Figure3.8 (1) Block diagram of 8-bit PWM timer 0 (timer 2)

Note: Block diagram for 8-bit PWM timer 1 (timer 3) is the same as the above diagram.

① Prescaler

Generates input clocks dedicated to PWM timers by further dividing the fundamental clock (fc) after it has been divided by 2 (fc/2). Since the register used to control the prescaler is the same as the one for other timers, the prescaler cannot be operated independently.

The PWM timer uses three input clocks: ϕ/P1, ϕ/P4, and ϕ/P16.

Like the 9-bit prescaler described in the 8-bit timer section, this prescaler can be counted/stopped using bit 7 <PRRUN> of the timer operation control register TRUN. Setting <PRRUN> to 1 starts counting; setting it to 0 zero-clears and stops counting. Resetting clears <PRRUN> to 0, which clears and stops the prescaler.

Dedicated prescaler cycle

	16MHz	20MHz
ϕP1 (4/fc)	250nSec	200nSec
ϕP4 (16/fc)	1μSec	800nSec
ϕP16 (64/fc)	4μSec	3.2μSec

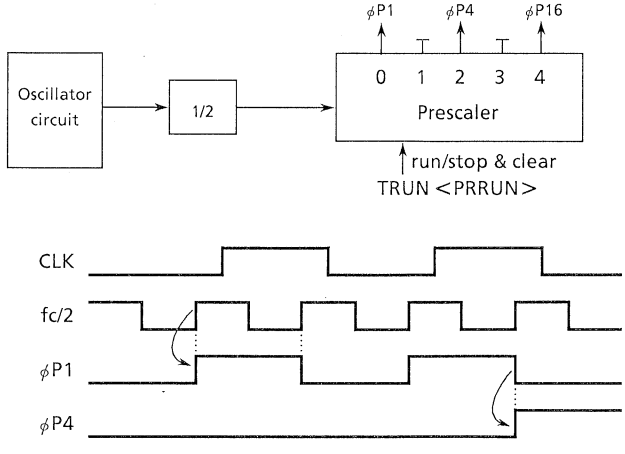


Figure3.8 (2) Prescaler

② Up-counter

An 8-bit binary counter which counts up using the input clock specified by PWM mode register (P0MOD or P1MOD).

The input clock for the PWM0/PWM1 is selected from the internal clocks ϕP1, ϕP4, and ϕP16 (PWM dedicated prescaler output) depending on the value set in the P0MOD/P1MOD register.

Operating mode is also set by P0MOD and P1MOD registers. At reset, they are initialized to P0MOD<PWM0M> = 0 and P1MOD<PWM1M> = 0, thus, the up-counter is in PWM mode. In PWM mode, the up-counter is cleared when a 2^n-1 overflow occurs; in timer mode, the up-counter is cleared at compare and match.

Count/stop & clear of the up-counter can be controlled for each PWM timer using the timer operation control register TRUN. Resetting clears all up-counters and stops timers.

③ Timer registers

Two 8-bit registers used for setting an interval time. When the value set in the timer registers (TREG2 & 3) matches the value in the up-counter, the match detect signal of the comarator becomes active.

Timer registers TREG2 and TREG3 are each paired with register buffer to make a double buffer structure.

TREG2 and TREG3 are controlled double buffer enable/disable by P0MOD <DB2EN> and P1MOD <DB3EN> : disabled when <DB2EN> / <DB3EN> = 0, enabled when <DB2EN> / <DB3EN> = 1.

Data is transferred from register buffer to timer register when a 2^n-1 overflow occurs in PWM mode, or when compare and match occurs in 8-bit timer mode. That is, with a PWM timer, the timer mode can be operated in double buffer enable state, unlike timer mode for timers 0 and 1.

At reset, <DB2EN>/<DB3EN> is initialized to 0 to disable double buffer. To use double buffer, write the data in the timer register at first, then set <DB2EN> / <DB3EN> to 1, and write the following data in the register buffer.

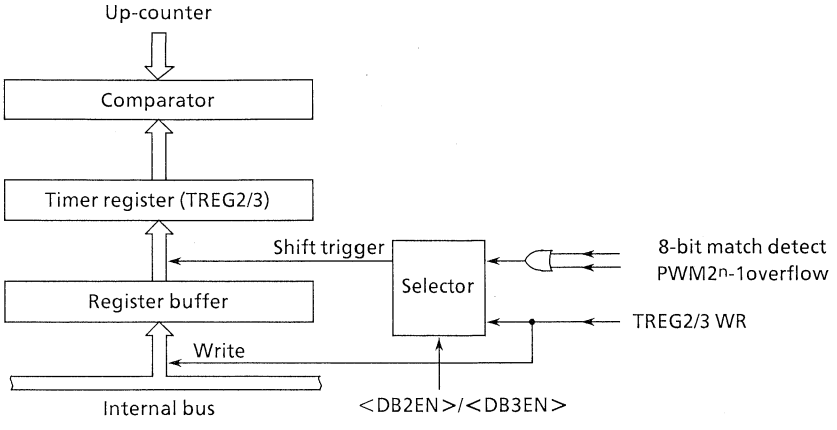


Figure3.8 (3) Structure of Timer Registers 2 &3

Note : The timer register and register buffer are allocated to the same memory address. When <DB2EN>/<DB3EN> = 0, the same value is written to both register buffer and timer register. When <DB2EN>/<DB3EN> = 1, the value is written to the register buffer only.

Memory addresses of the timer registers are as follows:

TREG2 : 000026H
TREG3 : 000027H

Both timer registers are write only; however, register buffer values can be read when reading the above addresses.

④ Comparator

Compares the value in the up-counter with the value in the timer register (TREG2/TREG3). When they match, the comparator outputs the match detect signal. A timer interrupt (INTT2/INTT3) is generated at compare and match if the interrupt select bit <PWM0INT>/<PWM1NT> of the mode register (P0MOD/P1MOD) is set to 1. In timer mode, the comparator clears the up-counter to 0 at compare and match. It also inverts the value of the timer flip-flop if timer flip-flop invert is enabled.

⑤ Timer flip-flop

The value of the timer flip-flop is inverted by the match detect signal (comparator output) of each interval timer or 2^n-1 overflow. The value can be output to the timer output pin TO2/TO3 (also used as P72/P73).

P0MOD (0028H)

	7	6	5	4	3	2	1	0
bit Symbol	FF2RD	DB2EN	PWM0INT	PWM0M	T2CLK1	T2CLK0	PWM0S1	PWM0S0
Read/Write	R	W						
After reset	–	0	0	0	0	0	0	0
Function	Flip-flop (F/F2) output data	1: Double Buffer2 Enable	0: 2^{n-1} overflow interrupt 1: compare & match interrupt	0: PWM mode 1: Timer mode	00: ϕP1 (fc/4) 01: ϕP4 (fc/16) 10: ϕP16 (fc/64) 11: Don't care		00: 2^6-1 01: 2^7-1 10: 2^8-1 11: Don't care	

Read-modify-write is prohibited.

Select PWM0 cycle

00	2^6-1
01	2^7-1
10	2^8-1
11	Don't care

Select PWM0 input clock

00	ϕ P1 (fc/4)
01	ϕ P4 (fc/16)
10	ϕ P16 (fc/64)
11	Don't care

select PWM0 mode

0	PWM mode
1	8-bit timer mode

select PWM0 interrupt

0	Overflow interrupt
1	Compare and match interrupt

control double buffer

0	Disable
1	Enable

PWM timer Flip-flop2 (TFF2) output value (TO2)

Figure3.8 (4) 8-bit PWM0 mode control register

P1MOD (0029H)

	7	6	5	4	3	2	1	0
bit Symbol	FF3RD	DB3EN	PWM1INT	PWM1M	T3CLK1	T3CLK0	PWM1S1	PWM1S0
Read/Write	R	W						
After reset	–	0	0	0	0	0	0	0
Function	Flip-flop (F/F3) output data	1: Double Buffer3 Enable	0: 2^n-1 overflow interrupt 1: Compare and match interrupt	0: PWM mode 1: Timer mode	00: ϕP1 (fc/4) 01: ϕP4 (fc/16) 10: ϕP16 (fc/64) 11: Don't care		00: 2^6-1 01: 2^7-1 10: 2^8-1 11: Don't care	

Read-modify-write is prohibited.

Select PWM1 cycle

00	2^6-1
01	2^7-1
10	2^8-1
11	Don't care

Select PWM1 input clock

00	ϕ P1 (fc/4)
01	ϕ P4 (fc/16)
10	ϕ P16 (fc/64)
11	Don't care

Select PWM1 mode

0	PWMmode
1	8-bit timer mode

Select PWM1 interrupt

0	Overflow interrupy
1	Compare and match interrupt

Control double buffer

0	Disable
1	enable

PWM timer Flip-flop3 (TFF3) output value (TO3)

Figure 3.8 (5) 8-bit PWM1 mode control register

PFFCR (002AH)

	7	6	5	4	3	2	1	0
bit Symbol	FF3C1	FF3C0	FF3TRG1	FF3TRG0	FF2C1	FF2C0	FF2TRG1	FF2TRG0
Read/Write	W		R/W		W		R/W	
After reset	0	0	0	0	0	0	0	0
Function	00 : Don't care 01 : Set TFF3 10 : Clear TFF3 11 : Don't care		00 : Disable TFF3 inverted. 01 : Invert by match. 10 : Set by match ; clear by overflow, 11 : Clear by match ; set by overflow.		00 : Don't care 01 : Set TFF2 10 : Clear TFF2 11 : Don't care		00 : Disable TFF2 inverted. 01 : Invert by match. 10 : Set by match ; clear by overflow, 11 : Clear by match ; set by overflow.	

Select PWM timer F/F2 (TFF2) trigger

00	Disable TFF2 trigger.
01	Invert by compare and match.
10	Set by compare and match. Clear by 2^{n-1} overflow.
11	Clear by compare and match. Set by 2^{n-1} overflow.

Control PWM timerF/F2 (TFF2)

00	Don't care
01	Set TFF2 to "1".
10	Clear TFF2 to "0".
11	Don't care

Select PWM timer F/F3 (TFF3) trigger

00	Disable TFF3 trigger.
01	Invert by compare and match.
10	Set by compare and match. Clear by 2^{n-1} overflow.
11	Clear by compare and match. Set by 2^{n-1} overflow.

Control PWM timerF/F2 (TFF2)

00	Don't care
01	Set TFF3 to "1".
10	Clear TFF3 to "0".
11	Don't care

Figure 3.8 (6) 8-bit PWM F/F control register

TRUN (0020H)

	7	6	5	4	3	2	1	0
bit Symbol	PRRUN		T5RUN	T4RUN	P1RUN	P0RUN	T1RUN	T0RUN
Read/Write	R/W		R/W					
After reset	0		0	0	0	0	0	0
Function	Prescaler & Timer Run/Stop CONTROL 0: Stop & Clear 1: Run (Count up)							

8-bit timer (timer0) operation

0	Stop & clear
1	Count

8-bit timer (timer1) operation

0	Stop & clear
1	Count

8-bit PWM timer (PWM0/timer2) operation

0	Stop & clear
1	Count

8-bit PWM timer (PWM0/timer2) operation

0	Stop & clear
1	Count

16-bit timer (timer4) operation

0	Stop & clear
1	Count

16-bit timer (timer5) operation

0	Stop & clear
1	Count

Prescaler operation

0	Stop & clear
1	Count

Figure 3.8 (7) Timer operation control register (TRUN)

The following explains PWM timer operations.

(1) PWM timer mode

Both PWM timers can output 8-bit resolution PWM independently. Since both timers operate in exactly the same way, PWM0 is used for the purposes of explanation.

PWM output changes under the following two conditions.

Condition 1:

- TFF2 is cleared to 0 when the value in the up-counter (UC2) and the value set in the TREG2 match.
- TFF2 is set to 1 when a 2^n-1 counter overflow (n = 6, 7, or 8) occurs.

Condition 2:

- TFF2 is set to 1 when the value in the up-counter (UC2) and the value set in TREG2 match.
- TFF2 is cleared to 0 when a 2^n-1 counter overflow (n = 6, 7, or 8) occurs.

The up-counter (UC2) is cleared by a 2^n-1 counter overflow.

The PWM timer can output 0%-100% duty pulses because a 2^n-1 counter overflow has a higher priority. That is, to obtain 0% output (always low), the mode used to set TFF2 to 0 due to overflow (PFFCR<FF2TRG1,0> = 1,0) must be set and 2^n-1 (Value for overflow) must be set in TREG2. To obtain 100% output (always high), the mode must be changed: PFFCR<FF2TRG1,0> = 1,1 then the same operation is required.

PWM timing

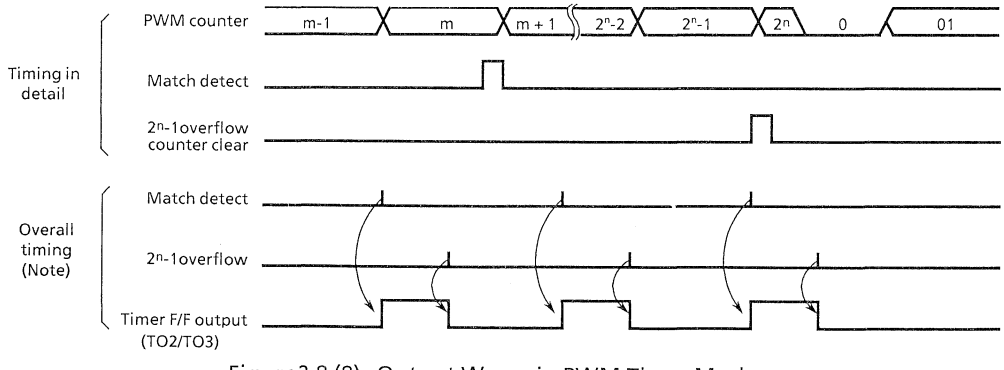


Figure3.8 (8) Output Waves in PWM Timer Mode

Note : The above waves are obtained in a mode where the F/F is set by a match with the timer register (TREG) and reset by an overflow.

Figure 3.8 (9) is a block diagram of this mode.

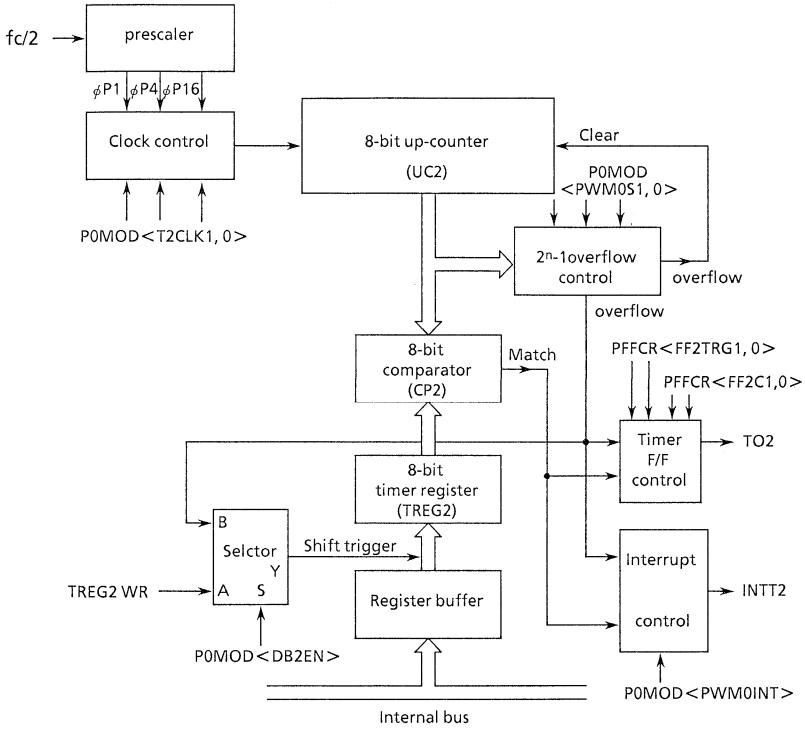


Figure 3.8 (9) Block diagram of PWM Timer Mode (PWM0)

In this mode, enabling double buffer is very useful. The register buffer value shifts into TREG2 when a 2^n-1 overflow is detected, when double buffer is enabled.

Using double buffer makes handling small duty waves easy.

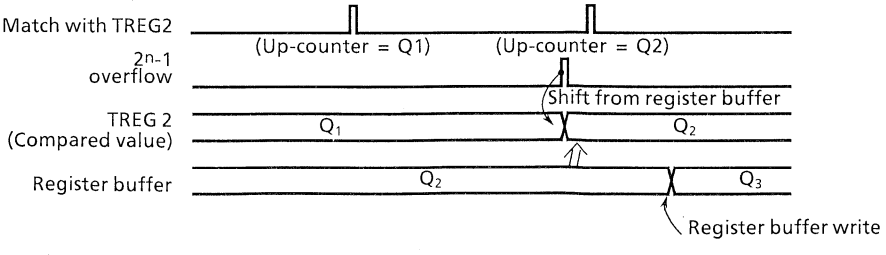
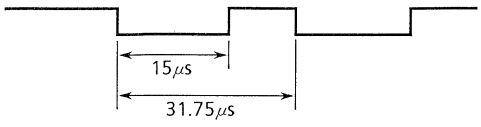


Figure3.8 (10) Register Buffer Operation

Example: To output the following PWM waves to TO2 pin using PWM0 at fc = 16 MHz



To implement 31.75μs PWM cycle by φP1 = 0.25 μs (@fc = 16 MHz)

$31.75\mu s \div 0.25\mu s = 127 = 2^7\text{-}1$.

Consequently, set n to 7.

Since the low level cycle = 15 μs; for φP1 = 0.25 μs

$15\mu s \div 0.25 = 60 = 3CH$

set the 3CH in TREG2.

```
        7 6 5 4 3 2 1 0
TRUN  ← - X - - - 0 - -    Stops PWM0 and clears it to 0.
POMOD ← - 0 0 0 0 0 0 1    Sets PWM (27-1) mode, input clock φP1, overflow
                           interrupt, and disables double buffer.
TREG2 ← 0 0 1 1 1 1 0 0    Writes 3CH.
POMOD ← - 1 0 0 0 0 0 1    Enables double buffer.
PFFCR ← - - - - 0 1 1 1    Sets TFF2 and a mode where TFF2 is set by compare and
                           match, and cleared by overflow.
P7CR  ← X X X X - 1 - -  }
P7FC  ← X X X X - 1 - X  }  Sets P72 as TO2 pin
TRUN  ← 1 X - - - 1 - -    Starts PWM0 counting.
```

Note: X; don't care -; no change

Table3.8 (1) PWM Cycle and 2^n-1 Counter Setting

	Formula	16MHz			20MHz		
		φP1	φP4	φP16	φP1	φP4	φP16
2^6-1	$2^6\text{-}1 \times \phi Pn$	15.8μsec (63KHz)	63.0μsec (16KHz)	252μsec (3.9KHz)	12.6μsec (79KHz)	50.4μsec (20KHz)	201μsec (4.9KHz)
2^7-1	$2^7\text{-}1 \times \phi Pn$	31.8μsec (31KHz)	127.0μsec (7.9KHz)	508μsec (1.9KHz)	25.4μsec (39KHz)	101.6μsec (9.8KHz)	406μsec (2.5KHz)
2^8-1	$2^8\text{-}1 \times \phi Pn$	63.8μsec (16KHz)	255.0μsec (3.9KHz)	1020μsec (0.98KHz)	51.0μsec (20KHz)	204.0μsec (4.9KHz)	816μsec (1.2KHz)

(2) 8-bit timer mode

Both PWM timers can be used independently as 8-bit interval timers. Since both timers operate in exactly the same way, PWM0 (timer 2) is used for the purposes of explanation.

① Generating interrupts at a fixed interval

To generate timer 2 interrupt (INTT2) at a fixed interval using PWM0 timer, first stop PWM0, then set the operating mode, input clock, and interval in the P0MOD and TREG2 registers. Next, enable INTT2 and start counting PWM0.

Example: To generate a timer 2 interrupt every 40μs at fc = 16MHz, set registers as follows:

```
                7 6 5 4 3 2 1 0
TRUN        ← - X - - - 0 - -     Stops PWM0 and clears it to 0.
P0MOD       ← X 0 1 1 0 0 X X     Sets 8-bit timer mode and selects φP1 (0.25 us) and
                                  compare interrupt.
TREG2       ← 1 0 1 0 0 0 0 0     Sets 40μs/0.25μs = A0H in timer register.
INTEPW10    ← - - - - 1 1 0 0     Enables INTT2 and sets interrupt level 4.
TRUN        ← 1 X - - - 1 - -     Starts counting PWM0.
```

Note : X ; don't care - ; no change

Select an input clock using the table below.

Table3.8 (2) Interrupt Cycle and Input Clock Selection using 8-bit timer mode

Input clock	Interrupt cycle (@fc = 16MHz)	Resolution	Interrupt cycle (@fc = 20MHz)	Resolution
φP1 (4/fc)	0.25 μs ~ 64 μs	0.25 μs	0.2 μs~51.2 μs	0.2 μs
φP4 (16/fc)	1 μs ~ 256 μs	1 μs	0.8 μs~204.8 μs	0.8 μs
φP16 (64/fc)	4 μs ~ 1024μs	4 μs	3.2 μs~819.2 μs	3.2 μs

Note : To generate interrupts in 8-bit timer mode, bit 5 (interrupt control bit <PWM0INT> / <PWM1NT> of P0MOD/P1MOD) must be set to 1.

② Generating a 50% square wave

To generate a 50% square wave, invert the timer flip-flop at a fixed interval and output the timer flip-flop value to the timer output pin (TO2).

Example: To output a 3.0 μs square wave at fc = 16 MHz from TO2 pin, set registers as follows.

```
          7 6 5 4 3 2 1 0
TRUN   ← - X - - - 0 - -      Stops PWM0 and clears it to 0.
P0MOD  ← X 0 1 1 0 0 X X      Sets 8-bit timer mode and selects φP1 (0.25 μs) as the
                              input clock.
TREG2  ← 0 0 0 0 0 1 1 0      Sets 3.0μs/0.25μs/2 = 6 in the timer register.
PFFCR  ← - - - - 1 0 0 1      Clears TFF2 to 0 and inverts using comparator output.
P7CR   ← X X X X - 1 - -  }
P7FC   ← X X X X - 1 - X  }   Sets P72 as TO2 pin.
TRUN   ← 1 X - - - 1 - -      Starts counting PWM0.
```

Note: X; don't care -; no change

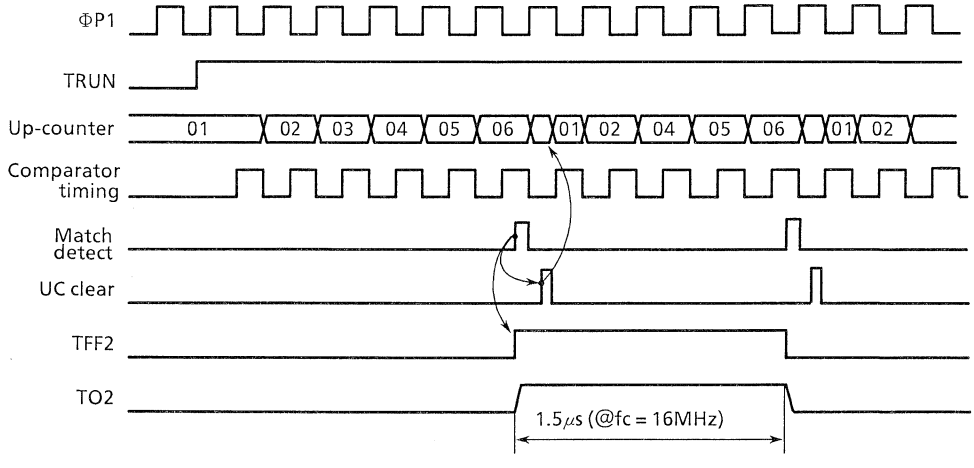


Figure3.8 (11) Square Wave (50% Duty) Output Timing Chart

This mode is as shown in Figure 3.8 (12) below.

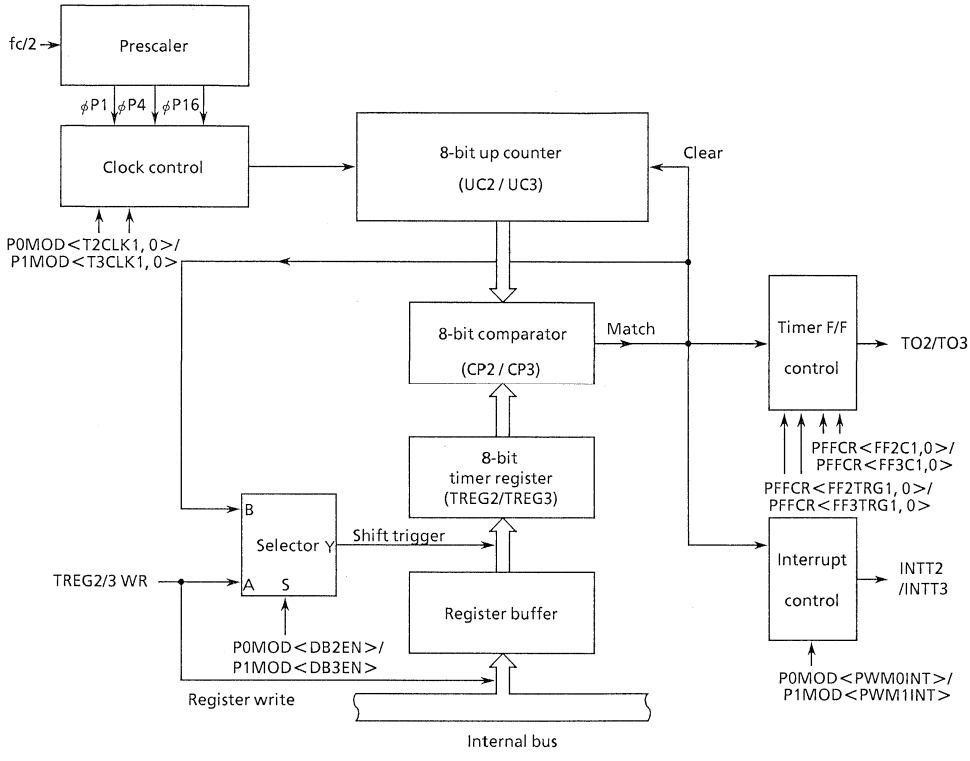


Figure3.8 (12) Block Diagram of 8-bit Timer Mode

3.9 16-bit Timer

TMP96C141/TMP96CM40/TMP96PM40 contains two (timer 4 and timer 5) multifunctional 16-bit timer/event counter with the following operation modes.

- 16-bit interval timer mode
- 16-bit event counter mode
- 16-bit programmable pulse generation (PPG) mode
- Frequency measurement mode
- Pulse width measurement mode
- Time differential measurement mode

Timer/event counter consists of 16-bit up-counter, two 16-bit timer registers, two 16-bit capture registers (One of them applies double-buffer), two comparators, capture input controller, and timer flip-flop and the control circuit.

Timer/event counter is controlled by 4 control registers: T4MOD/T5MOD, T4FFCR / T5FFCR, TRUN and T45CR.

Figure 3.9 (1), (2) shows the block diagram of 16-bit timer/event counter (timer 4 and timer 5) .

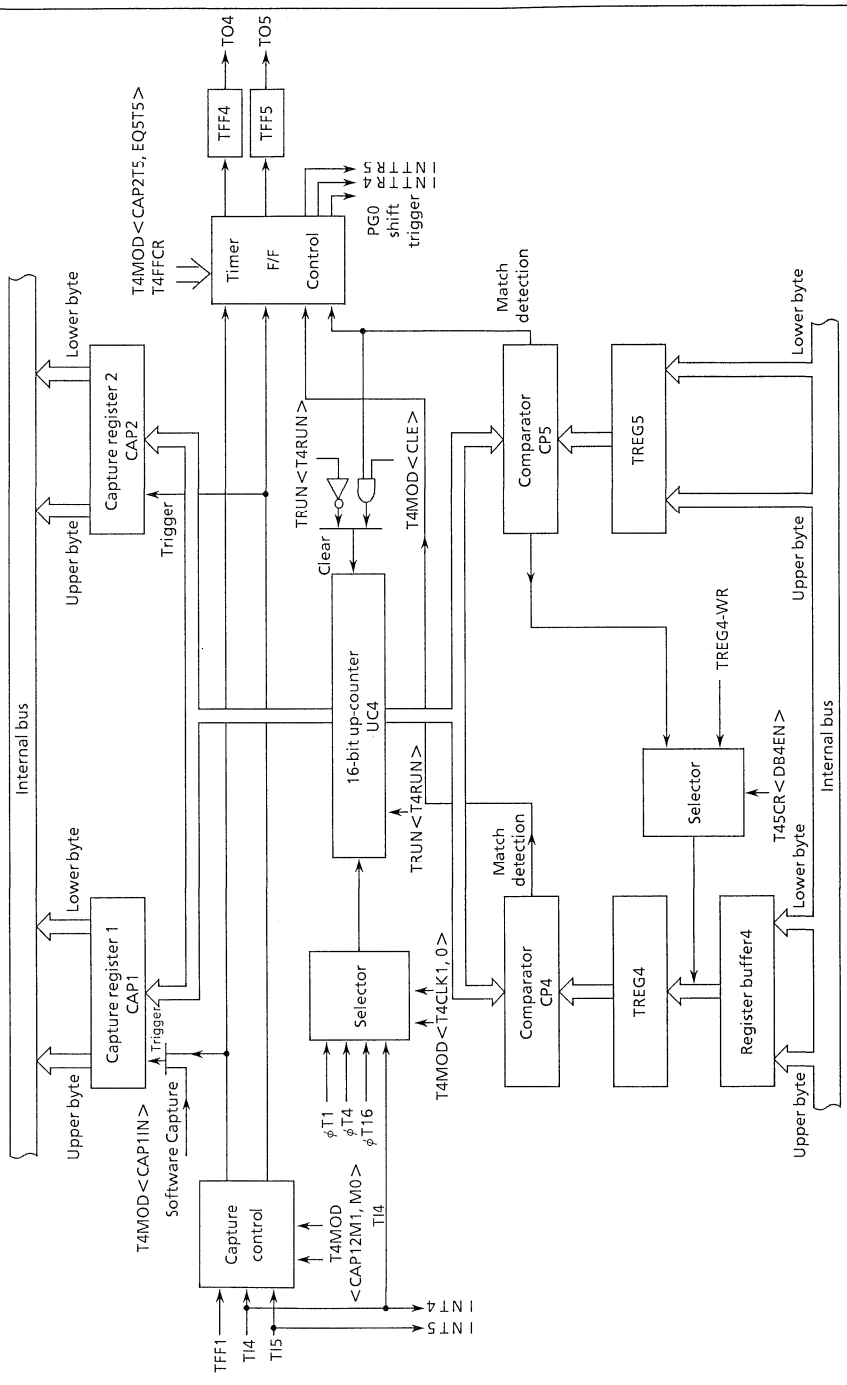


Figure 3.9 (1) Block Diagram of 16-Bit Timer (Timer 4)

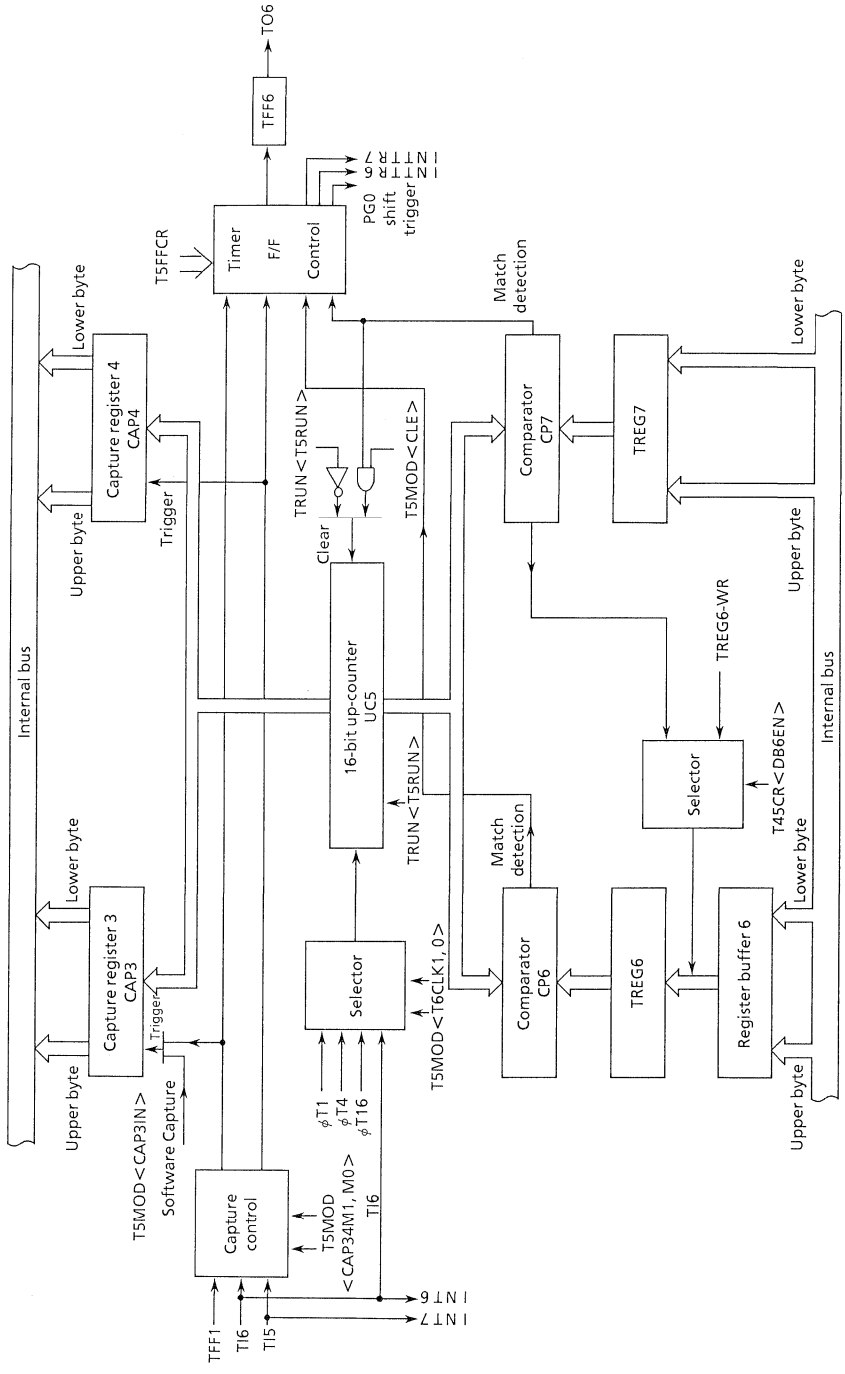


Figure 3.9 (2) Block Diagram of 16-Bit Timer (Timer 5)

T4MOD (0038H)

	7	6	5	4	3	2	1	0
bit Symbol	CAP2T5	EQ5T5	CAP1IN	CAP12M1	CAP12M0	CLE	T4CLK1	T4CLK0
Read/Write	R/W		W	R/W		R/W	R/W	
After reset	0	0	1	0	0	0	0	0
Function	TFF5 invert trigger 0: Disable trigger 1: Enable trigger		0: Soft-Capture 1: don't care	Capture timing 00: Disable INT4 occurs at rise edge. 01: TI4↑ TI5↑ INT4 occurs at rise edge. 10: TI4↑ TI4↓ INT4 occurs at fall edge. 11: TFF1↑ TFF1↓ INT4 occurs at rise edge.		1: UC4 Clear Enable	Timer 4 source clock 00: TI4 01: ϕT1 10: ϕT4 11: ϕT16	
	Invert when the UC value is loaded to CAP2	Invert when the up-counter matches TREG5						

Timer 4 input clock

00	External clock (TI4)
01	ϕT1 (8/fc)
10	ϕT4 (32/fc)
11	ϕT16 (128/fc)

Clearing the up-counter UC4

0	Clear disable
1	Clear by match with TREG5.

Figure 3.9 (3) 16-Bit Timer Mode Controller Register (T4MOD) (1/2)

T4MOD (0038H)

	7	6	5	4	3	2	1	0
bit Symbol	CAP2T5	EQ5T5	CAP1IN	CAP12M1	CAP12M0	CLE	T4CLK1	T4CLK0
Read/Write	R/W		W	R/W		R/W	R/W	
After reset	0	0	1	0	0	0	0	0
Function	TFF5 invert trigger 0: Disable trigger 1: Enable trigger		0: Soft-Capture 1: don't care	Capture timing 00: Disable INT4 occurs at rise edge. 01: TI4↑ TI5↑ INT4 occurs at rise edge. 10: TI4↑ TI4↓ INT4 occurs at fall edge. 11: TFF1↑ TFF1↓ INT4 occurs at rise edge.		1: UC4 Clear Enable	Timer 4 source clock 00: TI4 01: φT1 10: φT4 11: φT16	
	Invert when the UC value is loaded to CAP2	Invert when the up-counter matches TREG5						

Capture timing of timer4

	Capture control	INT4 control
00	Capture disable	Interrupt occurs at the rise edge of TI4 (INT1) input.
01	CAP1 at TI4 rise CAP2 at TI5 rise	
10	CAP1 at TI4 rise CAP2 at TI4 fall	Interrupt occurs at the fall edge of TI4 (INT1) input.
11	CAP1 at TFF1 rise CAP2 at TFF1 fall	Interrupt occurs at the rise edge of TI4 (INT1) input.

Software capture

0	The up-counter4 value is loaded to CAP1 (software capture).
1	Always read as "1".

Timer flip-flop 5 (TFF5) invert trigger

0	Trigger disable (Invert Prohibition)
1	Trigger enable (Invert permission)

CAP2T5 : Invert when the up-counter value is loaded to CAP2
EQ5T5 : Invert when the up-counter matches TREG5

Figure 3.9 (4) 16-Bit Timer Controller Register (T4MOD) (2/2)

T4FFCR (0039H)

	7	6	5	4	3	2	1	0
bit Symbol	TFF5C1	TFF5C0	CAP2T4	CAP1T4	EQ5T4	EQ4T4	TFF4C1	TFF4C0
Read/Write	W		R/W	R/W	R/W	R/W	W	
After reset	0	0	0	0	0	0	0	0
Function	00: Invert TFF5 01: Set TFF5 10: Clear TFF5 11: don't care Always read as "11".		TFF4 invert trigger 0: Disable trigger 1: Enable trigger				00: Invert TFF4 01: Set TFF4 10: Clear TFF4 11: don't care ※ Always read as "11"	
			Invert when the UC value is loaded to CAP2	Invert when the UC value is loaded to CAP1	Invert when the UC matches TREG5	Invert when the UC matches TREG4		

→ Timer flip-flop 4 (TFF4) control

00	Inverts the TFF4 value (software inversion).
01	Sets TFF4 to "1".
10	Clear TFF4 to "0".
11	Don't care (Always read as "11").

→ Timer flip-flop 4 (TFF4) invert trigger

0	Trigger disable (Invert prohibition)
1	Trigger enable (Invert permission)

CAP2T4 : Invert when the up-counter value is loaded to CAP2
CAP1T4 : Invert when the up-counter value is loaded to CAP1
EQ5T4 : Invert when up-counter matches TREG5
EQ4T4 : Invert when up-counter matches TREG4

→ Timer flip-flop 5 (TFF5) control

00	Inverts the TFF5 value (software inversion).
01	Set TFF5 to "1".
10	Clear TFF5 to "0".
11	Don't care (Always read as "11".)

Figure 3.9 (5) 16-Bit Timer 4 F/F Control (T4FFCR)

T5MOD (0048H)

	7	6	5	4	3	2	1	0
bit Symbol			CAP3IN	CAP34M1	CAP34M0	CLE	T5CLK1	T5CLK0
Read/Write			W	R/W		R/W	R/W	
After reset			1	0	0	0	0	0
Function			0: Soft-Capture 1: don't care	Capture timing 00: Disable INT6 occurs at rise edge. 01: TI6↑ TI7↑ INT6 occurs at rise edge. 10: TI6↑ TI6↓ INT6 occurs at fall edge. 11: TFF1↑ TFF1↓ INT4 occurs at rise edge.		1: UC5 Clear Enable	Timer 5 source clock 00: TI6 01: ϕT1 10: ϕT4 11: ϕT16	

Timer 5 input clock

00	External clock (TI4)
01	ϕT1 (8/fc)
10	ϕT4 (32/fc)
11	ϕT16 (128/fc)

Clearing the up-counter UC5

0	Clear disable
1	Clear by match with TREG7

Figure 3.9 (6) 16-bit Timer Mode Control Register (T5MOD) (1/2)

T5MOD (0048H)

	7	6	5	4	3	2	1	0
bit Symbol			CAP3IN	CAP34M1	CAP34M0	CLE	T5CLK1	T5CLK0
Read/Write			W	R/W		R/W	R/W	
After reset			1	0	0	0	0	0
			0: Soft-Capture 1: don't care	Capture timing 00: Disable INT6 occurs at rise edge. 01: TI6↑ TI7↑ INT6 occurs at rise edge. 10: TI6↑ TI6↓ INT6 occurs at fall edge. 11: TFF1↑ TFF1↓ INT4 occurs at rise edge.		1: UC5 Clear Enable	Timer 5 source clock 00: TI6 01: φT1 10: φT4 11: φT16	

Timer 5 Capture timing

	Capture control	INT4 Control
00	Capture disable	Interrupt occurs at the rise edge of TI6 (INT6) input.
01	CAP3 at TI6 rise CAP4 at TI7 rise	
10	CAP3 at TI6 rise CAP4 at TI6 fall	Interrupt occurs at the fall edge of TI6 (INT6) input.
11	CAP3 at TFF1 rise CAP4 at TFF1 fall	Interrupt occurs at the rise edge of TI6 (INT6) input.

Software capture

0	The up-counter 5 value is loaded to CAP3.
1	Always read as "1".

Figure 3.9 (7) 16-Bit Timer Control Register (T5MOD) (2/2)

T5FFCR (0049H)

	7	6	5	4	3	2	1	0
bit Symbol			CAP4T6	CAP3T6	EQ7T6	EQ6T6	TFF6C1	TFF6C0
Read/Write			R/W	R/W	R/W	R/W	W	
After reset			0	0	0	0	0	0
Function			TFF6 invert trigger 0: Disable trigger 1: Enable trigger				00: Invert TFF6 01: Set TFF6 10: Clear TFF6 11: don't care ※ Always read as "11"	
			Invert when the UC value is loaded to CAP4	Invert when the UC value is loaded to CAP3	Invert when the UC matches TREG7	Invert when the UC matches TREG6		

Timer flip-flop 6 (TFF6) control

00	Inverts the TFF4 value (software inversion).
01	Sets TFF6 to "1".
10	Clear TFF6 to "0".
11	Don't care (Always read as "11") .

Timer flip-flop 6 (TFF6) invert trigger

0	Trigger disable (Invert prohibition)
1	Trigger enable (Invert permission)

CAP4T6 : Invert when the up-counter value is loaded to CAP4
CAP3T6 : Invert when the up-counter value is loaded to CAP3
EQ7T6 : Invert when up-counter matches TREG7
EQ6T6 : Invert when up-counter matches TREG6

Figure 3.9 (8) 16-Bit Timer5 F/F Control (T5FFCR)

T45CR (003AH)

	7	6	5	4	3	2	1	0
bit Symbol	—				PG1T	PG0T	DB6EN	DB4EN
Read/Write	R/W				R/W			
After reset	0				0	0	0	0
	Fix at "0"				PG1 shift Trigger 0: 8 bit Timer Trigger (Timer 0, 1) 1: 16 bit Timer Trigger (Timer 5)	PG0 shift Trigger 0: 8 bit Timer Trigger (Timer 0, 1) 1: 16 bit Timer Trigger (Timer 4)	0: Disable 1: Enable	
							Double buffer of TREG6	Double buffer of TREG4

Double buffer countrol

0	Disable
1	Enable

DB6EN : Double buffer of TREG6
DB4EN : Double buffer of TREG4

Figure 3.9 (9) 16-Bit Timer (Timer 4, 5) Control Register (T45CR)

TRUN (0020H)

	7	6	5	4	3	2	1	0
bit Symbol	PRRUN		T5RUN	T4RUN	P1RUN	P0RUN	T1RUN	T0RUN
Read/Write	R/W		R/W					
After reset	0		0	0	0	0	0	0
	Prescaler & Timer Run/Stop CONTROL 0: Stop & Clear 1: Run (Count up)							

Operation of 16-bit timer (timer4)

0	Stop and clear
1	Count

Operation of 16-bit timer (timer5)

0	Stop and clear
1	Count

Operation of prescaler

0	STOP and clear
1	Count

Figure 3.9 (10) Timer Operation Contorl Register (TRUN)

① Up-counter (UC4/UC5)

UC4/UC5 is a 16-bit binary counter which counts up according to the input clock specified by T4MOD<T4CLK1,0> or T5MOD<T5CLK1, 0> register.

As the input clock, one of the internal clocks ϕT1 (8/fc), ϕT4 (32/fc), and ϕT16 (128/fc) from 9-bit prescaler (also used for 8-bit timer), and external clock from TI4 pin (also used as P80/INT4 pin) or TI6 (also used as P84/INT6 pin) can be selected. When reset, it will be initialized to <T4CLK1,0> / <T5CLK1,0> = 00 to select TI4/TI6 input mode. Counting or stop & clear of the counter is controlled by timer operation control register TRUN<T4RUN, T5RUN>.

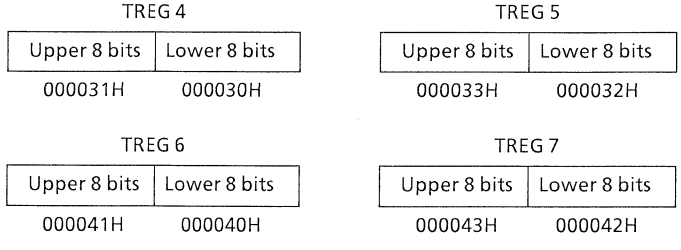
When clearing is enabled, up-counter UC4/UC5 will be cleared to zero each time it coincides matches the timer register TREG5, TREG7. The "clear enable/disable" is set by T4MOD<CLE>and T5MOD<CLE>.

If clearing is disabled, the counter operates as a free-running counter.

② Timer Registers

These two 16-bit registers are used to set the interval time. When the value of up-counter UC4/UC5 matches the set value of this timer register, the comparator match detect signal will be active.

Setting data for timer register (TREG4, TREG5, TREG6 and TREG7) is executed using 2 byte date transfer instruction or using 1 byte date transfer instruction twice for lower 8 bits and upper 1 bits in order.



TREG4 and TREG6 timer register is of double buffer structure, which is paired with register buffer. The timer control register T45CR<DB4EN, DB6EN> controls whether the double buffer structure should be enabled or disabled. : disabled when <DB4EN, DB6EN> = 0, while enabled when <DB4EN, DB6EN> = 1.

When the double buffer is enabled, the timing to transfer data from the register buffer to the timer register is at the match between the up-counter (UC4/UC5) and timer register TREG5/TREG7.

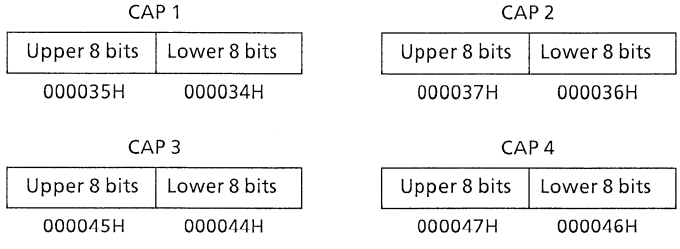
When reset, it will be initialized to <DB4EN, DB6EN> = 0, whereby the double buffer is disabled. To use the double buffer, write data in the timer register, set <DB4EN, DB6EN> = 1, and then write the following data in the register buffer.

TREG4, TREG6 and register buffer are allocated to the same memory addresses 000030H/000031H/000040H/000041H. When <DB4EN, DB6EN> = 0, same value will be written in both the timer register and register buffer. When <DB4EN, DB6EN> = 1, the value is written into only the register buffer.

③ Capture Register

These 16-bit registers are used to hold the values of the up-counter.

Data in the capture registers should be read by a 2-byte data load instruction or two 1-byte data load instruction, from the lower 8 bits followed by the upper 8 bits.



④ Capture Input Control

This circuit controls the timing to latch the value of up-counter UC4/UC5 into (CAP1, CAP2) / (CAP3, CAP4). The latch timing of capture register is controlled by register T4MOD<CAP12M 1, 0> / T5MOD <CAP34M1, 0>.

- When T4MOD<CAP12M 1, 0> / T5MOD <CAP34M1, 0> = 00

  Capture function is disabled. Disable is the default on reset.

- When T4MOD<CAP12M1, 0> / T5MOD <CAP34M1, 0> = 01

  Data is loaded to CAP1, CAP3 at the rise edge of TI4 pin (also used as P80/INT4) and TI6 pin (also used as P84/INT6) input, while data is loaded to CAP2, CAP4 at the rise edge of TI5 pin (also used as P81/INT5) and TI7 pin (also used as P85/INT7) input. (Time difference measurement)

- When T4MOD<CAP12M1, 0> / T5MOD <CAP34M1, 0> = 10

  Data is loaded to CAP1 at the rise edge of TI4 pin input and to CAP3 at the rise edge of TI6, while to CAP2, CAP4 at the fall edge. Only in this setting, interrupt INT4/INT6 occurs at fall edge. (Pulse width measurement)

- When T4MOD<CAP12M1, 0> / T5MOD <CAP34M1, 0> = 11

  Data is loaded to CAP1, CAP3 at the rise edge of timer flip-flop TFF1, while to CAP2, CAP4 at the fall edge.

  Besides, the value of up-counter can be loaded to capture registers by software. Whenever "0" is written in T4MOD<CAPIN>, T5MOD<CAP31N> the current value of up-counter will be loaded to capture register CAP1/CAP3. It is necessary to keep the prescaler in RUN mode (TRUN<PRRUN> to be "1").

⑤ Comparator

These are 16-bit comparators which compare the up-counter UC4/UC5 value with the set value of (TREG4, TREG5) / (TREG6, TREG7) to detect the match. When a match is detected, the comparators generate an interrupt (INTT4, INTT5) / (INTT6, INTT7) respectively. The up-counter UC4/UC5 is cleared only when UC4/UC5 matches TREG5/TREG7. (The clearing of up-counter UC4/UC5 can be disabled by setting T4MOD<CLE> / T5MOD<CLE> = 0.)

⑥ Timer Flip-flop (TFF4/TFF6)

This flip-flop is inverted by the match detect signal from the comparators and the latch signals to the capture registers. Disable/enable of inversion can be set for each element by T4FFCR<CAP2T4, CAP1T4, EQ5T4, EQ4T4> / T6FFCR <CAP4T6, CAP3T6, EQ7T6, EQ6T6>. TFF4/TFF6 will be inverted when "00" is written in T4FFCR<TFF4C1,0> / T6FFCR<TFF6C1, 0>. Also it is set to "1" when "10" is written, and cleared to "0" when "10" is written. The value of TFF4/TFF6 can be output to the timer output pin TO4 (also used as P82) and TO6 (also used as P86).

⑦ Timer Flip-flop (TFF5)

This flip-flop is inverted by the match detect signal from the comparator and the latch signal to the capture register CAP2. TFF5 will be inverted when "00" is written in T4FFCR<TFF5C1,0>/T6FFCR<TFF6C1, 0>. Also it is set to "1" when "10" is written, and cleared to "0" when "10" is written. The value of TFF5 can be output to the timer output pin TO5 (also used as P82).

Note : This flip-flop (TFF5) is contained only in the 16-bit timer 4

(1) 16-bit Timer Mode

Timers 4 and 5 operate independently.

Since both timers operate in exactly the same way, timer 4 is used for the purposes of explanation.

Generating interrupts at fixed intervals

In this example, the interval time is set in the timer register TREG5 to generate the interrupt INTTR5.

```
                7 6 5 4 3 2 1 0
TRUN      ← - X - 0 - - - -      Stop timer 4.
INTET54   ← 1 1 0 0 1 0 0 0      Enable INTTR5 and sets interrupt level 4. Disable
                                 INTTR4.
T4FFCR    ← 1 1 0 0 0 0 1 1      Disable trigger.
T4MOD     ← 0 0 1 0 0 1 * *      Select internal clock for input and
                (** = 01, 10, 11) disable the capture function.
TREG5     ← * * * * * * * *      Set the interval time (16 bits).
              * * * * * * * *
TRUN      ← 1 X - 1 - - - -      Start timer 4.
```

Note : X ; don't care - ; no change

(2) 16-bit Event Counter Mode

In 16-bit timer mode as described in above, the timer can be used as an event counter by selecting the external clock (TI4/TI6 pin input) as the input clock. To read the value of the counter, first perform "software capture" once and read the captured value.

The counter counts at the rise edge of TI4/TI6 pin input.

TI4/TI6 pin can also be used as P80/INT4 and P84/INT6.

Since both timers operate in exactly the same way, timer 4 is used for the purposes of explanation.

	7 6 5 4 3 2 1 0	
TRUN	← - X - 0 - - - -	Stop timer 4.
P8CR	← - - - - - - - 0	Set P80 to input mode
INTET54	← 1 1 0 0 1 0 0 0	Enable INTTR5 and sets interrupt level 4, while disables INTTR4.
T4FFCR	← 1 1 0 0 0 0 1 1	Disable trigger.
T4MOD	← 0 0 1 0 0 1 0 0	Select TI4 as the input clock.
TREG5	← * * * * * * * *	Set the number of counts (16 bits).
TRUN	← 1 X - 1 - - - -	Start timer 4.

Note : When used as an event counter, set the prescaler in RUN mode.

(3) 16-bit Programmable Pulse Generation (PPG) Output Mode

Since both timers operate in exactly the same way, timer 4 is used for the purposes of explanation.

The PPG mode is obtained by inversion of the timer flip-flop TFF4 that is to be enabled by the match of the up-counter UC4 with the timer register TREG4 or 5 and to be output to TO4 (also used as P82). In this mode, the following conditions must be satisfied.

(Set value of TREG4) < (Set value of TREG5)

	7 6 5 4 3 2 1 0	
TRUN	← - X - 0 - - - -	Stop timer 4.
TREG4	← * * * * * * * *	Set the duty. (16-Bit)
TREG5	← * * * * * * * *	Set the cycle. (16-Bit)
T45CR	← 0 X X X - - - 1	Double Buffer of TREG4 enable (Change the duty and cycle at the interrupt INTTR5)
T4FFCR	← 1 1 0 0 1 1 0 0	Set the mode to invert TFF4 at the match with TREG4/TREG5, and also set the TFF4 to "0".
T4MOD	← 0 0 1 0 0 1 * * (** = 01, 10, 11)	Select the internal clock for the input, and disable the capture function.
P8CR	← - - - - - - 1 - -	Assign P82 as TO4.
P8FC	← X - X X - 1 X X	
TRUN	← 1 X - 1 - - - -	Start timer 4.

Note : X ; don't care - ; no change

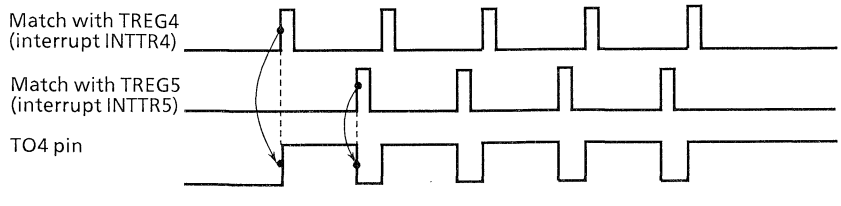


Figure 3.9 (11) Programmable Pulse Generation (PPG) Output Waveforms

When the double buffer of TREG4 is enabled in this mode, the value of register buffer 4 will be shifted in TREG4 at match with TREG5. This feature makes easy the handling of low duty waves.

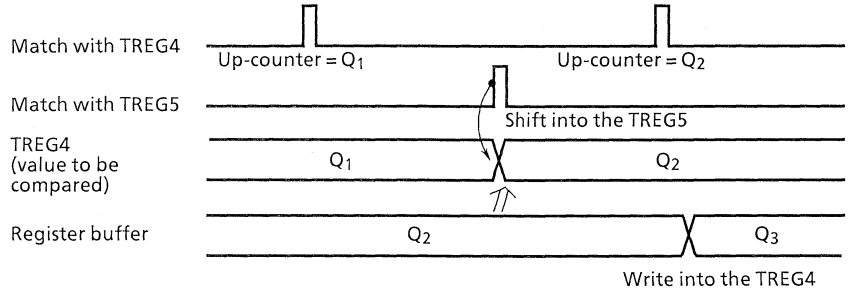


Figure 3.9 (12) Operation of Register Buffer

Shows the block diagram of this mode.

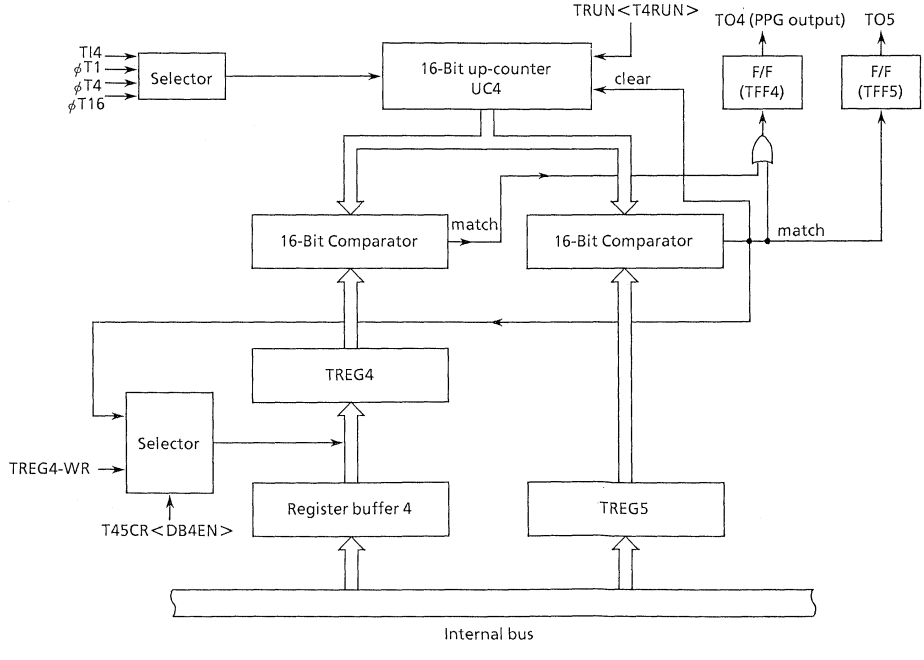


Figure 3.9 (13) Block Diagram of 16-Bit PPG Mode

(4) Application Examples of Capture Function

The loading of up-counter (UC4) values into the capture registers CAP1 and CAP2, the timer flip-flop TFF4 inversion due to the match detection by comparators CP4 and CP5, and the output of the TFF4 status to TO4 pin can be enabled or disabled. Combined with interrupt function, they can be applied in many ways, for example:

① One-shot pulse output from external trigger pulse

② Frequency measurement

③ Pulse width measurement

④ Time difference measurement

① One-shot Pulse Output from External Trigger Pulse

Set the up-counter UC4 in free-running mode with the internal input clock, input the external trigger pulse from TI4 pin, and load the value of up-counter into capture register CAP1 at the rise edge of the TI4 pin. Then set to T4MOD<CAP12M1, 0>=01.

When the interrupt INT4 is generated at the rise edge of TI4 input, set the CAP1 value (c) plus a delay time (d) to TREG4 (= c+d), and set the above set value (c+d) plus a one-shot pulse width (p) to TREG5 (= c+d+p). When the interrupt INT4 occurs the T4FFCR<EQ5T4, EQ4T4>register should be set that the TFF4 inversion is enabled only when the up-counter value matches TREG4 or TREG5. When interrupt INTTR5 occurs, this inversion will be disabled.

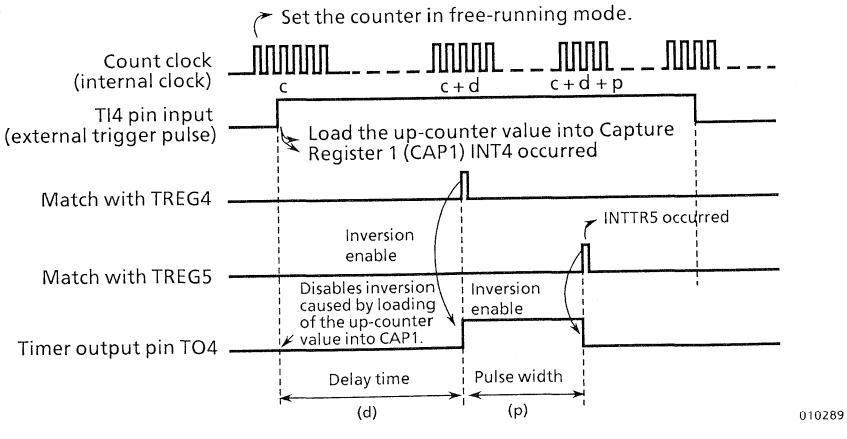
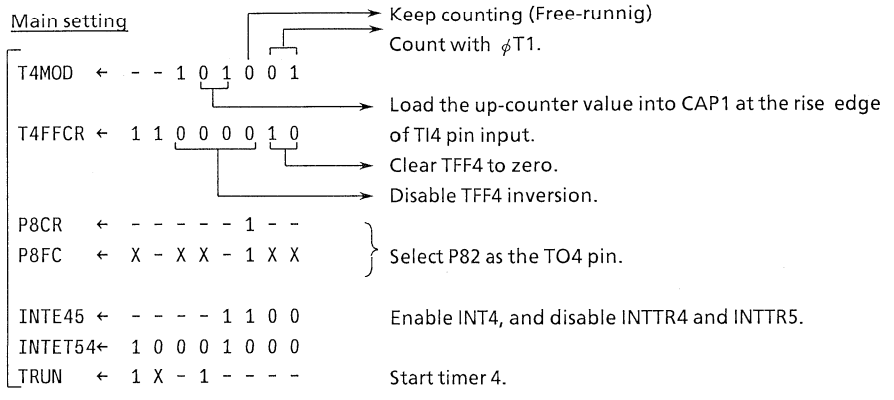
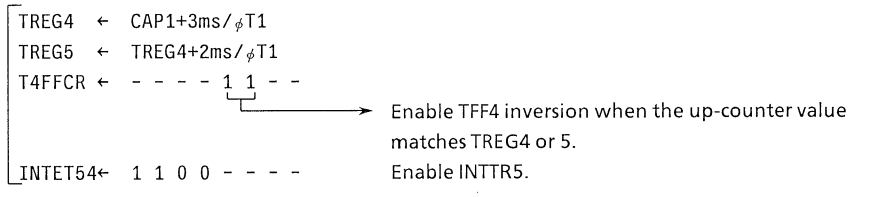


Figure 3.9 (14) One-Shot Pulse Output (with Delay)

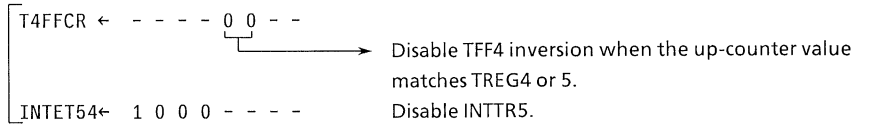
Setting example : To output 2ms one-shot pulse with 3ms delay to the external trigger pulse to TI4 pin



Setting of INT4



Setting of INT5



Note: X ; don't care - ; no change

When delay time is unnecessary, invert timer flip-flop TFF4 when the up-counter value is loaded into capture register 1 (CAP1), and set the CAP1 value (c) plus the one-shot pulse width (p) to TREG5 when the interrupt INT4 occurs. The TFF4 inversion should be enabled when the up-counter (UC4) value matches TREG5, and disabled when generating the interrupt INTTR5.

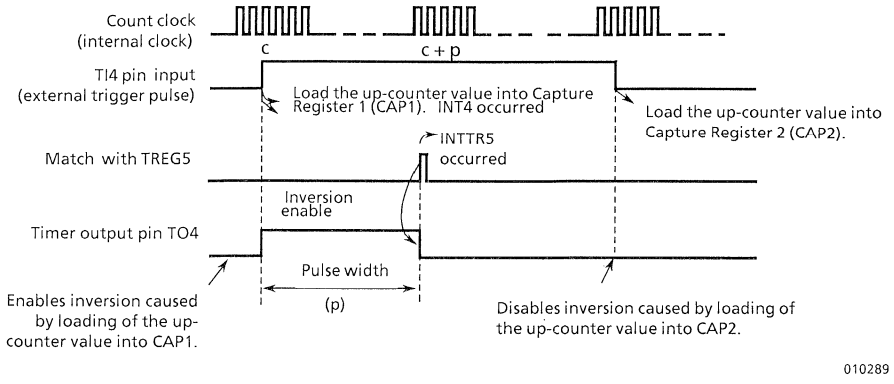


Figure 3.9 (15) One-Shot Pulse Output (without Delay)

② Frequency Measurement

The frequency of the external clock can be measured in this mode. The clock is input through the TI4 pin, and its frequency is measured by the 8-bit timers (Timer 0 and Timer 1) and the 16-bit timer/event counter (Timer 4).

The TI4 pin input should be selected for the input clock of Timer 4. The value of the up-counter is loaded into the capture register CAP1 at the rise edge of the timer flip-flop TFF1 of 8-bit timers (Timer 0 and Timer 1), and into CAP2 at its fall edge.

The frequency is calculated by the difference between the loaded values in CAP1 and CAP2 when the interrupt (INTT0 or INTT1) is generated by either 8-bit timer.

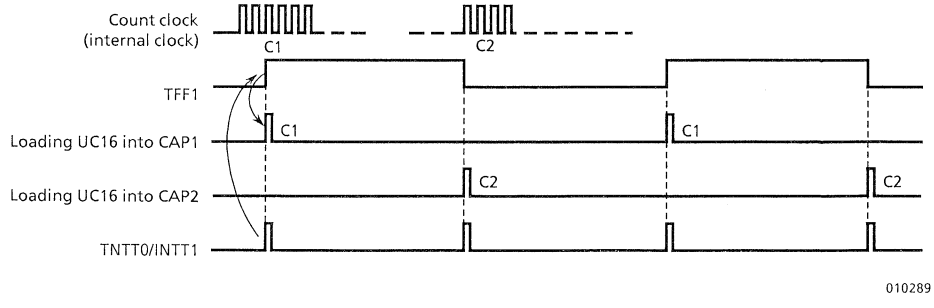


Figure 3.9 (16) Frequency Measurement

For example, if the value for the level "1" width of TFF1 of the 8-bit timer is set to 0.5 sec. and the difference between CAP1 and CAP2 is 100, the frequency will be 100/0.5 [sec.] = 200[Hz].

③ Pulse Width Measurement

This mode allows to measure the "H" level width of an external pulse. While keeping the 16-bit timer/event counter counting (free-running) with the internal clock input, the external pulse is input through the TI4 pin. Then the capture function is used to load the UC4 values into CAP1 and CAP2 at the rising edge and falling edge of the external trigger pulse respectively. The interrupt INT4 occurs at the falling edge of TI4.

The pulse width is obtained from the difference between the values of CAP1 and CAP2 and the internal clock cycle.

For example, if the internal clock is 0.8 microseconds and the difference between CAP1 and CAP2 is 100, the pulse width will be $100 \times 0.8 = 80$ microseconds.

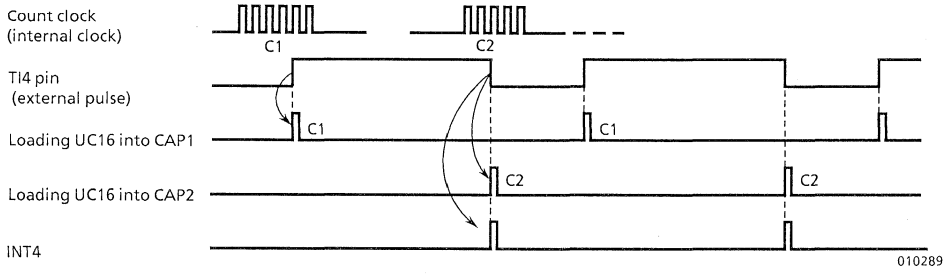


Figure 3.9 (17) Pulse Width Measurement

Note: Only in this pulse width measuring mode (T4MOD<CAP12M1, 0>=10), external interrupt INT4 occurs at the falling edge of TI4 pin input. In other modes, it occurs at the rising edge.

The width of "L" level can be measured from the difference between the first C2 and the second C1 at the second INT4 interrupt.

④ Time Difference Measurement

This mode is used to measure the difference in time between the rising edges of external pulses input through TI4 and TI5.

Keep the 16-bit timer/event counter (Timer 4) counting (free-running) with the internal clock, and load the UC4 value into CAP1 at the rising edge of the input pulse to TI4. Then the interrupt INT4 is generated.

Similarly, the UC4 value is loaded into CAP2 at the rising edge of the input pulse to TI5, generating the interrupt INT5.

The time difference between these pulses can be obtained from the difference between the time counts at which loading the up-counter value into CAP1 and CAP2 has been done.

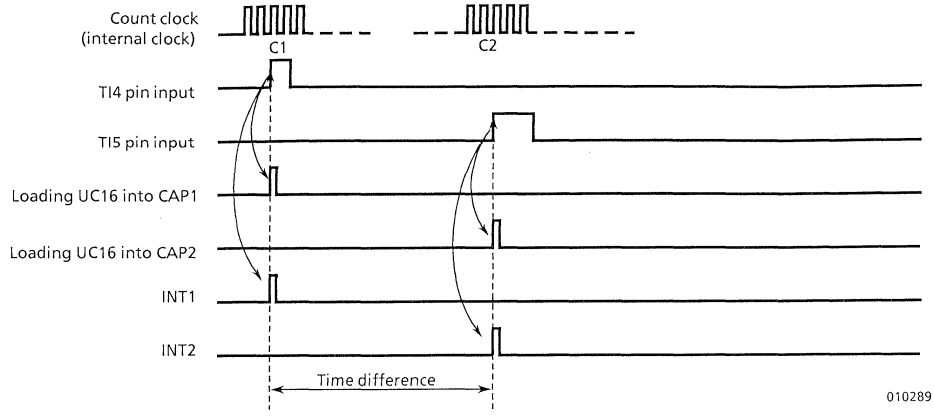


Figure 3.9 (18) Time Difference Measurement

(5) Different Phased Pulses Output Mode

In this mode, signals with any different phase can be outputted by free-running up-counter UC4.

When the value in up-counter UC4 and the value in TREG4 (TREG5) match, the value in TFF4 (TFF5) is inverted and output to TO4 (TO5).

This mode can only be used by 16-bit timer 4.

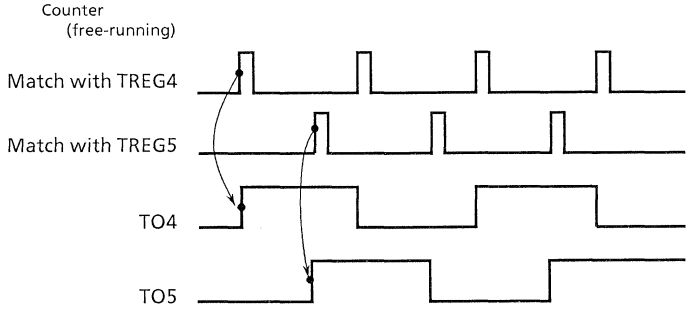


Figure 3.9 (19) Phase Output

Cycles (counter overflow time) of the above output waves are listed below.

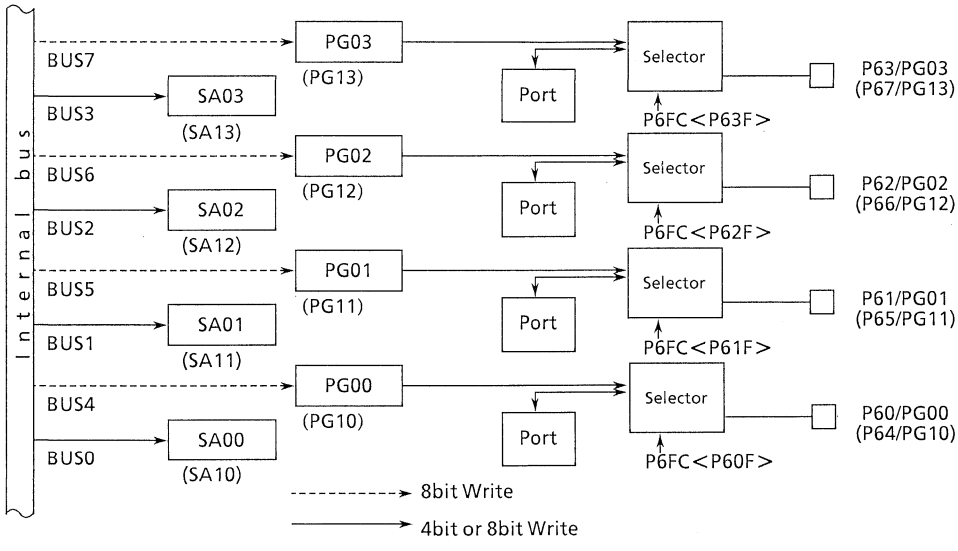
	16MHz	20MHz
ϕT1	1.024 msec	0.819 msec
ϕT4	4.096 msec	3.277 msec
ϕT16	16.38 msec	13.11 msec

3.10 Stepping Motor Control/Pattern Generation Port

TMP96C141/TMP96CM40/TMP96PM40 contains 2 channels (PG0 and PG1) of 4-bit hardware stepping motor control/pattern generation (herein after called PG) which actuate in synchronization with the (8-bit/16-bit) timers. The PG (PG0 and PG1) are shared in 8-bit I/O ports P6.

Channel 0 (PG0) is synchronous with 8-bit timer 0 or timer 1, 16-bit timer 5, to update the output.

The PG ports are controlled by control registers (PG01CR) and can select either stepping motor control mode or pattern generation mode. Each bit of the P6 can be used as the PG port.



Note: ( ) represents the case of channel 1.

Figre 3.10 (1) Port6/PG Circuit

PG01CR (004EH)

	7	6	5	4	3	2	1	0
bit Symbol	PAT1	CCW1	PG1M	PG1TE	PAT0	CCW0	PG0M	PG0TE
Read/Write	R/W				R/W			
After reset	0	0	0	0	0	0	0	0
Function	PG1 write mode 0: 8bit write 1: 4bit write	PG1 Rotaing direction 0: Normal rotation 1: Reverse rotation	PG1 mode (excitation) 0: 1excitation or 2excitation 1: 1-2 excitation	PG1 trigger input enable 0: disable 1: enable	PG0 write mode 0: 8bit write 1: 4bit write	PG0 Rotaing direction 0: Normal rotation 1: Reverse rotation	PG0 mode (excitation) 0: 1excitation or 2excitation 1: 1-2 excitation	PG0 trigger input enable 0: disable 1: enable

PG0 Trigger input enable

0	Trigger input disable to PG0
1	Trigger input enable to PG0

Set the operation mode of PG0

0	1 or 2 excitation (full step)
1	1-2 excitation (half step) / PG mode

PG0 (stepping motor control)
Rotaing direction control

0	Normal rotation / PG mode
1	Reverse rotation

Selecting PG0 write mode

0	8-bit write
1	4-bit write / PG mode (Only shifter alternate register can be written.)

Figure 3.10 (2 a) Pattern Generation Control Register (PG01CR)

PG01CR (004EH)

	7	6	5	4	3	2	1	0
bit Symbol	PAT1	CCW1	PG1M	PG1TE	PAT0	CCW0	PG0M	PG0TE
Read/Write	R/W				R/W			
After reset	0	0	0	0	0	0	0	0
Function	PG1 write mode 0: 8bit write 1: 4bit write	PG1 Rotaing direction 0: Normal rotation 1: Reverse rotation	PG1 mode (excitation) 0: 1excitation or 2excitation 1: 1-2 excitation	PG1 trigger input enable 0: disable 1: enable	PG0 write mode 0: 8bit write 1: 4bit write	PG0 Rotaing direction 0: Normal rotation 1: Reverse rotation	PG0 mode (excitation) 0: 1excitation or 2excitation 1: 1-2 excitation	PG0 trigger input enable 0: disable 1: enable

PG1 trigger input enable

0	Trigger input disable to PG1
1	Trigger input enable to PG1

Set the operation mode of PG1

0	1 or 2 excition (full step)
1	1-2 excitation (half step) / PG mode

PG1 (stepping motor control)
Rotaing direction control

0	Normal rotation / PG mode
1	Reverse rotation

Selecting PG1 write mode

0	8-bit write
1	4-bit write / PG mode (Only shifter alternate register can be written.)

Figure 3.10 (2 b) Pattern Generation Control Register (PG01CR)

PG0REG (004CH)

	7	6	5	4	3	2	1	0
bit Symbol	PG03	PG02	PG01	PG00	SA03	SA02	SA01	SA00
Read/Write	W				R / W			
After reset	0	0	0	0	Undefined			
Function	Pattern Generation 0 (PG0) output latch register (Reading the P6 that is set to the PG port allows to read-out.)				Shift alternate register 0 For the PG mode (4-bit write) register			

Prohibit Read modify write

Figure 3.10 (3) Pattern Generation 0 Register (PG0REG)

PG1REG (004DH)

	7	6	5	4	3	2	1	0
bit Symbol	PG13	PG12	PG11	PG10	SA13	SA12	SA11	SA10
Read/Write	W				R / W			
After reset	0	0	0	0	Undefined			
Function	Pattern Generation 1 (PG1) output latch register (Reading the P6 that is set to the PG port allows to read-out.)				Shift alternate register 1 For the PG mode (4-bit write) register			

Prohibit Read modify write

Figure 3.10 (4) Pattern Generation 1 Register (PG1REG)

T45CR (003AH)

	7	6	5	4	3	2	1	0
bit Symbol	–				PG1T	PG0T	DB6EN	DB4EN
Read/Write	R / W				R / W			
After reset	0				0	0	0	0
Function	Fix at "0" ※ Always read as "0"				PG1 Shift trigger 0: 8-bit timer trigger (timer0, 1) 1: 16-bit timer trigger (timer5)	PG0 Shift trigger 0: 8-bit timer trigger (timer0, 1) 1: 16bit timer trigger (timer4)	Double buffer 0: Disable 1: Enable Double buffer of TREG6	Double buffer of TREG4

Double buffer control

0	Disable
1	Enable

DB6EN : Double buffer of TREG6
DB4EN : Double buffer of TREG4

Selecting PG0 shift trigger

0	8-bit timer trigger (timer 0, 1)
1	16-bit timer trigger (timer 4)

Selecting PG1 shift trigger

0	8-bit timer trigger (timer 0, 1)
1	16-bit timer trigger (timer 5)

Write "0" to this bit, when this register is modified.

Figure 3.10 (5) 16-bit Timer Trigger Control Register (T45CR)

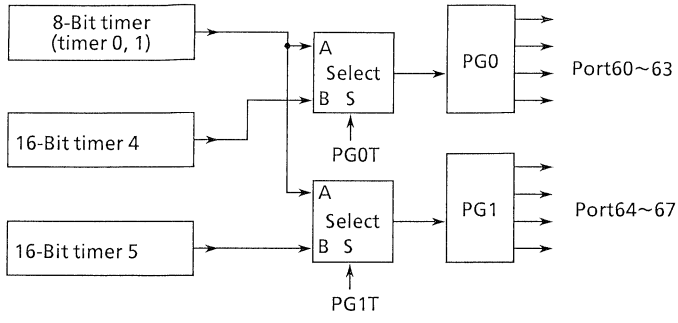


Figure 3.10 (6) Connection of Timer and Pattern Generator

(1) Pattern Generation Mode

PG functions as a pattern generation according to the setting of PG01CR <PAT1> / <PAT0>. In this mode, writing from CPU is executed only on the shifter alternate register. Writing a new data should be done during the interrupt operation of the timer for shift trigger and a pattern can be output, synchronous with the timer.

In this mode, set PG01CR<PG0M>and<PG1M>to 1, and PG01CR <CCW0> and <CCW1>to 0.

The output of this pattern generator is output to port 6 ; since port and functions can be switched on a bit basis using port function control register P6FC, any port pin can be assigned to pattern generator output.

Figure 3.10 (7) shows the block diagram of this mode.

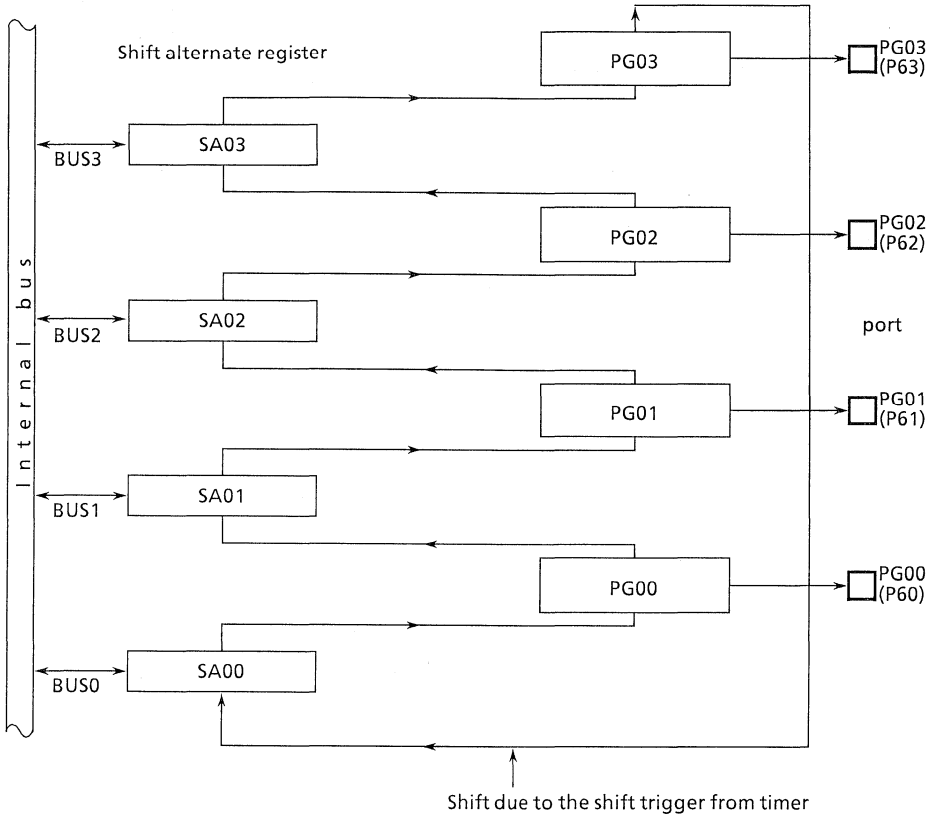


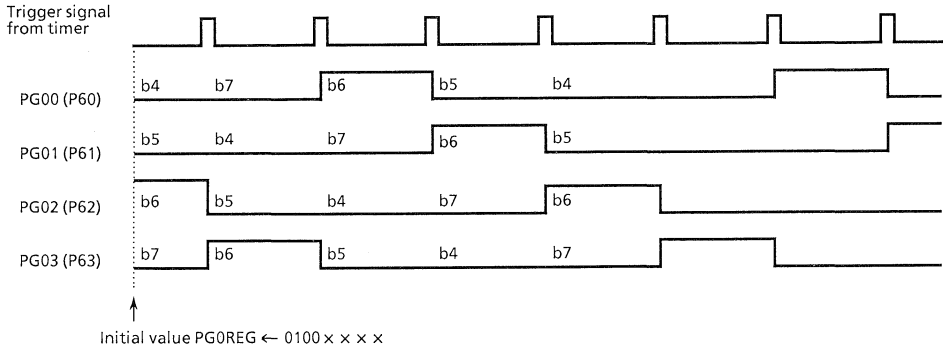
Figure 3.10 (7) Pattern Generation Mode Block Diagram (PG0)

In this pattern generation mode, only writing the output latch is disabled by hardware, but other functions do the same operation as 1-2 excitation in stepping motor control port mode. Accordingly, the data shifted by trigger signal from a timer must be written before the next trigger signal is output.

(2) Stepping Motor Control Mode

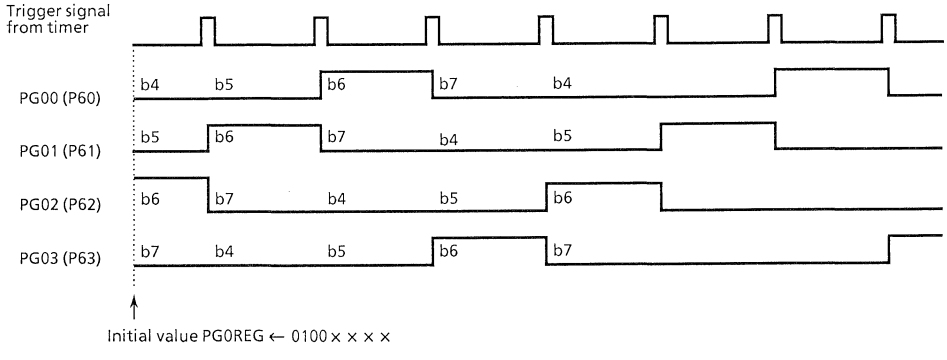
① 4-phase 1-Step/2-Step Excitation

Figure 3.10 (8) and Figure 3.10 (9) show the output waveforms of 4-phase 1 excitation and 4-phase 2 excitation, respectively when channel 0 (PG0) is selected.



Note : bn indicates the initial value of PG0REG ← b7 b6 b5 b4 × × × ×

① Normal Rotation



② Reverse Rotation

010289

Figure 3.10 (8) Output Waveforms of 4-Phase 1-step Excitation (Normal Rotation and Reverse Rotation)

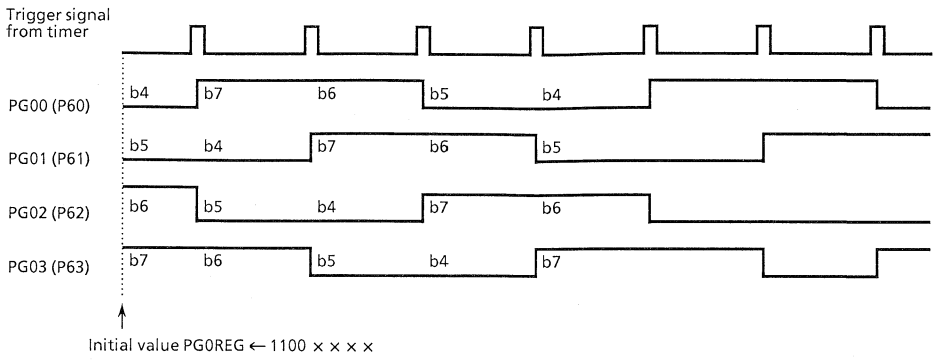


Figure 3.10 (9) Output Waveforms of 4-Phase 2-step Excitation (Normal Rotation)

The operation when channel 0 is selected is explained below.

The output latch of PG0 (also used as P6) is shifted at the rising edge of the trigger signal from the timer to be output to the port.

The direction of shift is specified by PG01CR<CCW0>: Normal rotation (PG00→PG01→PG02→PG03) when <CCW0> is set to "0"; reverse rotation (PG00←PG01←PG02←PG03) when "1". 4-phase 1-step excitation will be selected when only one bit is set to "1" during the initialization of PG, while 4-phase 2-step excitation will be selected when two consecutive bits are set to "1".

The value in the shift alternate registers are ignored when the 4-phase 1-step/2-step excitation mode is selected.

Figure 3.10 (10) shows the block diagram.

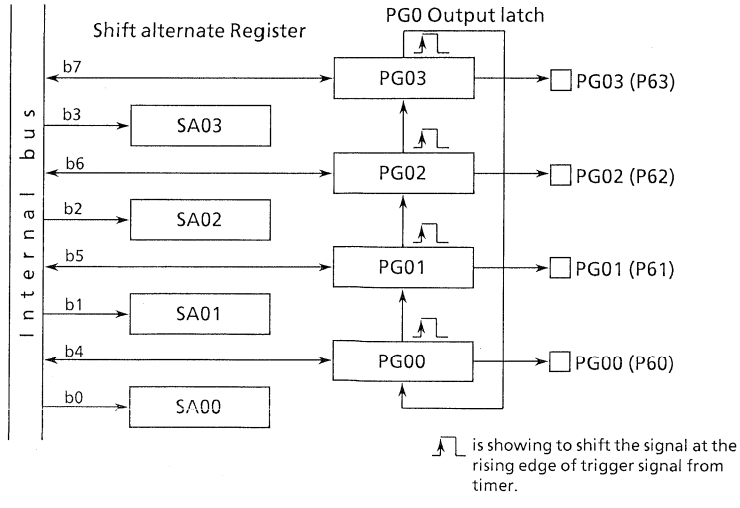
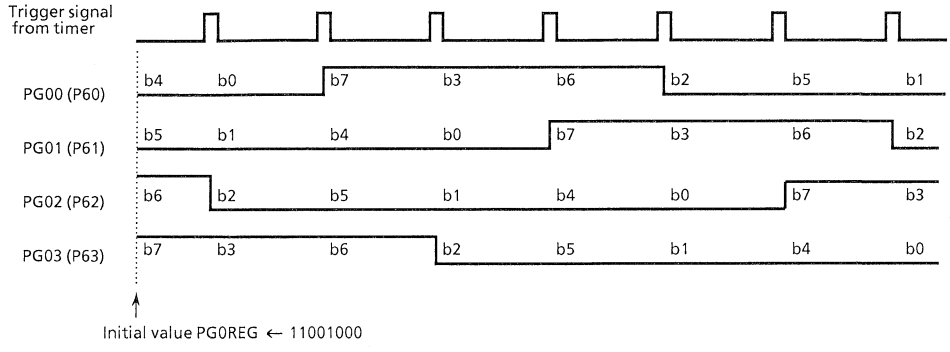


Figure 3.10 (10) Block Diagram of 4-Phase 1-step Excitation/2-step Excitation (Normal Rotation)

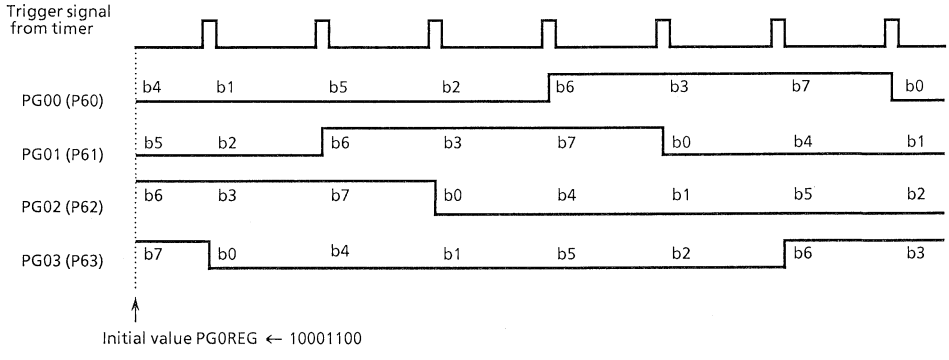
② 4-Phase 1-2 step Excitation

Figure 3.10 (11) shows the output waveforms of 4-phase 1-2 step excitation when channel 0 is selected.



Note : bn denotes the initial value PG0REG ← b7 b6 b5 b4 b3 b2 b1 b0

① Normal Rotation



② Reverse Rotation

Figure 3.10 (11) Output Waveforms of 4-Phase 1-2 step Excitation (Normal Rotation and Reverse Rotation)

The initialization for 4-phase 1-2 step excitation is as follows.

By rearranging the initial value "b7 b6 b5 b4 b3 b2 b1 b0" to "b7 b3 b6 b2 b5 b1 b4 b0", the consecutive 3 bits are set to "1" and other bits are set to "0" (positive logic).

For example, if b7, b3, and b6 are set to "1", the initial value becomes "11001000", obtaining the output waveforms as shown in Figure 3.10 (11).

To get an output waveform of negative logic, set values 1's and 0's of the initial value should be inverted. For example, to change the output waveform shown in Figure 3.10 (11) into negative logic, change the initial value to "00110111".

The operation will be explained below for channel 0.

The output latch of PG0 (shared by P6) and the shifter alternate register (SA0) for Pattern Generation are shifted at the rising edge of trigger signal from the timer to be output to the port. The direction of shift is set by PG01CR<CCW0>.

Figure 3.10 (12) shows the block diagram.

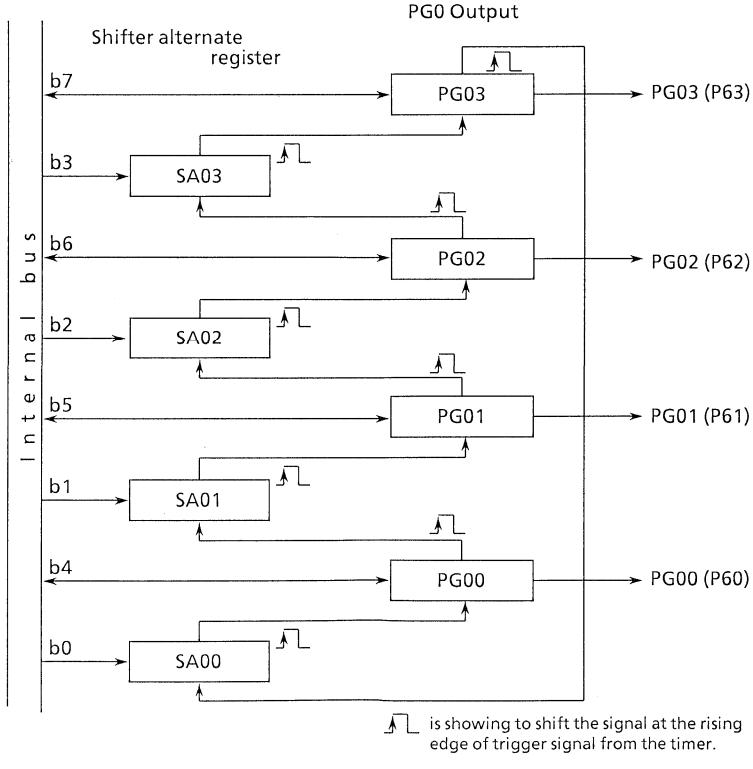


Figure 3.10 (12) Block Diagram of 4-Phase 1-2 step Excitation (Normal Rotation)

Setting example: To drive channel 0 (PG0) by 4-phase 1-2 step excitation (normal rotation) when timer 0 is selected, set each register as follows.

```
          7 6 5 4 3 2 1 0
TRUN    ← - X - - - - - 0     Stop timer 0, and clear it to zero.
TMOD    ← 0 0 X X - - 0 1     Set 8-bit timer mode and select φT1 as the input clock of timer 0.
TFFCR   ← X X X 0 1 0 1 0     Clear TFF1 to zero and enable the inversion trigger by timer 0.
TREG0   ← * * * * * * * *     Set the cycle in timer register.
P6CR    ← - - - - 1 1 1 1     Set P60~P63 bits to the output mode.
P6FC    ← - - - - 1 1 1 1     Set P60~P63 bits to the PG output.
PG01CR  ← - - - - 0 0 1 1     Select PG0 4-phase 1-2 step excitation mode and normal rotation .
PG0REG  ← 1 1 0 0 1 0 0 0     Set an initial value.
TRUN    ← 1 X - - - - - 1     Start timer 0.
```

Note : X ; don't care - ; no change

(3) Trigger Signal From Timer

The trigger signal from the timer which is used by PG is not equal to the trigger signal of timer flip-flop (TFF1, TFF4, TFF5, and TFF6) and differs as shown in Table 3.10 (1) depending on the operation mode of the timer.

Table 3.10 (1) Select of Trigger Signal

	TFF1 inversion	PG shift
8-bit timer mode	Selected by TFFCR <TFF1IS> when the up-counter value matches TREG0 or TREG1 value.	←
16-bit timer mode	When the up-counter value matches with both TREG0 and TREG1 values (The value of up-counter = TREG1*2^8 + TREG0)	←
PPG output mode	When the up-counter value matches with both TREG0 and TREG1	When the up-counter value matches TREG1 value (PPG cycle)
PWM output mode	When the up-counter value matches TREG0 value and PWM cycle.	Trigger signal for PG is not generated.

Note : To shift PG, TFFCR<TFF1IE> must be set to "1" to enable TFF1 inversion.

Channel 1 of PG can be synchronized with the 16-bit timer Timer4/Timer5. In this case, the PG shift trigger signal from the 16-bit timer is output only when the up-counter UC4 / UC5value matches TREG5/TREG7.

When using a trigger signal from Timer4, set either T4FFCR<EQ5T4> or T4MOD <EQ5T5> to "1" and a trigger is generated when the value in UC4 and the value in TREG5 match. When using a trigger signal from Timer5, set T5FFCR<EQ7T6>to 1. Generates a trigger when the value in UC5 and the value in TREG7 match.

(4) Application of PG and Timer Output

As explained "Trigger signal from timer", the timing to shift PG and invert TFF differs depending on the mode of timer. An application to operate PG while operating an 8-bit timer in PPG mode will be explained below.

To drive a stepping motor, in addition to the value of each phase (PG output), synchronizing signal is often required at the timing when excitation is changed over. In this application, port 6 is used as a stepping motor control port to output a synchronizing signal to the TO1 pin (shared by P71).

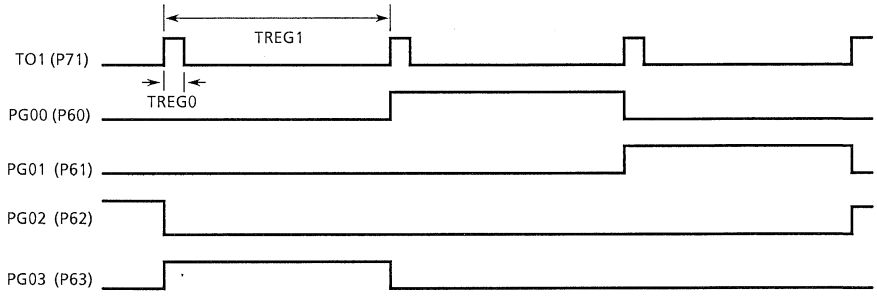


Figure 3.10 (13) Output Waveforms of 4-Phase 1-step Excitation

Setting example:

```
          7 6 5 4 3 2 1 0
TRUN    ← - X - - - - 0 0    Stop timer 0, and clear it to zero.
TMOD    ← 1 0 X X X X 0 1    Set timer 0 and timer 1 in PPG output mode and select
                             φT1 as the input clock.
TFFCR   ← X X X 0 0 1 1 X    Enable TFF1 inversion and set TFF1 to "1".
TREG0   ← * * * * * * * *    Set the duty of TO1 to TREG0.
TREG1   ← * * * * * * * *    Set the cycle of TO1 to TREG1.
P7CR    ← X X X X - - 1 -
P7FC    ← X X X X - - 1 X    } Assign P71 as TO1.
P6CR    ← - - - - 1 1 1 1
P6FC    ← - - - - 1 1 1 1    } Assign P60-63 as PG0.
PG01CR  ← - - - - 0 0 0 1    Set PG0 in 4-phase 1-step excitation mode.
PG0REG  ← * * * * * * * *    Set an initial value.
TRUN    ← 1 X - - - - 1 1    Start timer 0 and timer 1.
```

Note: X; don't care -; no change

3.11 Serial Channel

TMP96C141/TMP96CM40/TMP96PM40 contains 2 serial I/O channels for full duplex asynchronous transmission (UART) as well as for I/O extension.

The serial channel has the following operation modes.

- I/O interface mode (channel 1 only) — Mode 0: To transmit and receive I/O data as well as the synchronizing signal SCLK for extending I/O.
- Asynchronous transmission (UART) mode (channel 0 and 1)
  - Mode 1: 7-bit data
  - Mode 2: 8-bit data
  - Mode 3: 9-bit data

In mode 1 and mode 2, a parity bit can be added. Mode 3 has wake-up function for making the master controller start slave controllers in serial link (multi-controller system).

Figure 3.11 (1) shows the data format (for one frame) in each mode.

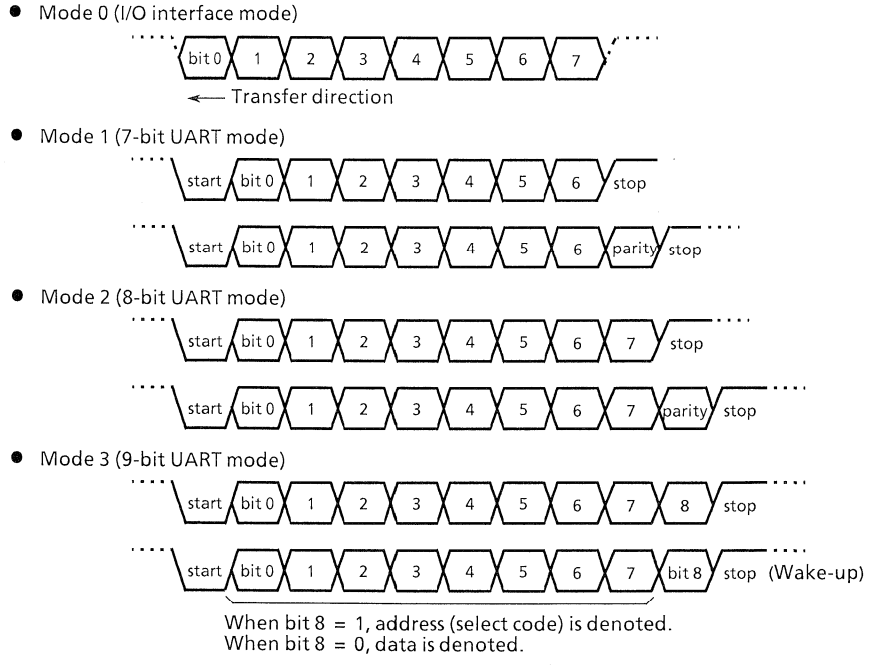


Figure 3.11 (1) Data Formats

The serial channel has a buffer register for transmitting and receiving operations, in order to temporarily store transmitted or received data, so that transmitting and receiving operations can be done independently (full duplex).

However, in I/O interface mode, SCLK (serial clock) pin is used for both transmission and receiving, the channel becomes half-duplex.

The receiving data register is of a double buffer structure to prevent the occurrence of overrun error and provides one frame of margin before CPU reads the received data. The receiving data register stores the already received data while the buffer register receives the next frame data.

By using CTS and RTS (there is no RTS pin, so any 1 port must be controlled by software), it is possible to halt data send until the CPU finishes reading receive data every time a frame is received. (Handshake function)

In the UART mode, a check function is added not to start the receiving operation by error start bits due to noise. The channel starts receiving data only when the start bit is detected to be normal at least twice in three samplings.

When the transmission buffer becomes empty and requests the CPU to send the next transmission data, or when data is stored in the receiving data register and the CPU is requested to read the data, INTTX or INTRX interrupt occurs. Besides, if an overrun error, parity error, or framing error occurs during receiving operation, flag SC0CR/SC1CR<OERR, PERR, FERR> will be set.

The serial channel 0/1 includes a special baud rate generator, which can set any baud rate by dividing the frequency of 4 clocks (ϕT0, ϕT2, ϕT8, and ϕT32) from the internal prescaler (shared by 8-bit/16-bit timer) by the value 2 to 16.

In I/O interface mode, it is possible to input synchronous signals as well as to transmit or receive data by external clock.

3.11.1 Control Registers

The serial channel is controlled by 3 control registers SC0CR, SC0MOD and BR0CR. Transmitted and received data are stored in register SC0BUF.

SC0MOD (0052H)

	7	6	5	4	3	2	1	0
bit Symbol	TB8	CTSE	RXE	WU	SM1	SM0	SC1	SC0
Read/Write	R/W							
After reset	0	0	0	0	0	0	0	0
Function	Transfer data Bit 8	Hand shake 0: CTS disable 1: CTS enable	Receiving Function 0: Receive disable 1: Receive enable	Wake up Function 0: disable 1: Enable	Serial transmission mode 00: Can not be used 01: 7-bit UART 10: 8-bit UART 11: 9-bit UART		Serial transmission clock (UART) 00: TO0 Trigger 01: baud rate generator 10: Internal clock Φ1 11: don't care	

Serial transmission clock (UART)

00	Timer 0 match detect signal
01	Baud rate generator
10	Internal clock Φ1
11	don't care

Serial transmission mode

00	Can not be used	
01	UART	7-bit length
10		8-bit length
11		9-bit length

Wake-up Function

	9-bit UART	Other mode
0	Interrupt when data are received	don't care
1	Interrupt only when RB8 = 1	

Receiving Function

0	Receive Disable
1	Receive Enable

Hand shake function ($\overline{CTS}$ Pin) enable

0	Disable (always Transferable)
1	Enable

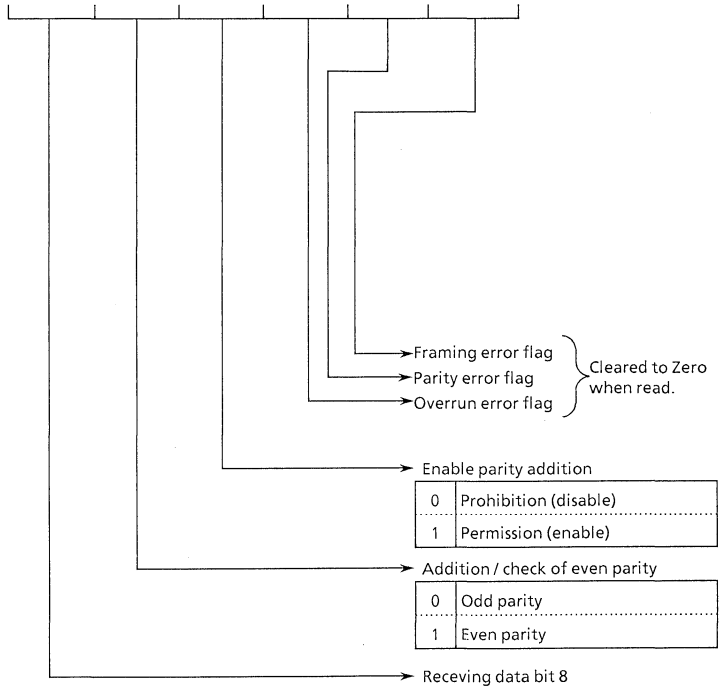
Transmission data bit 8

Note : There is SC1MOD (56H) in Channel1

Figure 3.11 (2) Serial Mode Control Register (channel 0, SC0MOD)

SC0CR (0051H)

	7	6	5	4	3	2	1	0
bit Symbol	RB8	EVEN	PE	OERR	PERR	FERR	—	—
Read/Write	R	R/W		R (cleared to Zero when read)			R/W	
After reset		0	0	0	0	0	0	0
Function	Received data Bit8	Parity 0: Odd 1: Even	Parity addition 0: Disable 1: Enable	1: error Overrun	Parity	Framing	Fix at "0"	Fix at "0"



Note : Serial control register for channel 1 is SC1CR (55H).

Note : As all error flags are cleared after reading do not test only a single bit with a bit-testing instruction.

Figure 3.11 (3) Serial Control Register (channel, SC0CR)

BR0CR (0053H)

	7	6	5	4	3	2	1	0
bit Symbol	—		BR0CK1	BR0CK0	BR0S3	BR0S2	BR0S1	BR0S0
Read/Write			R/W					
After reset	0		0	0	0	0	0	0
Function	Fix at "0"		00: ϕT0 (fc/4) 01: ϕT2 (fc/16) 10: ϕT8 (fc/64) 11: ϕT32 (fc/256)		Setting of the Divided frequency			

Setting of the divided frequency of baud rate generator

0000	16 divisions
0001	Don't set
0010 ∫ 1111	2 to 15 divisions

Selecting the input clock of baud rate generator

00	Internal clock ϕT0 (fc/4)
01	Internal clock ϕT2 (fc/16)
10	Internal clock ϕT8 (fc/64)
11	Internal clock ϕT32 (fc/256)

Note : As all error flags are cleared after reading, do not test only a single bit with a bit-testing instruction.

Figure 3.11 (4) Serial Channel Control (channel 0, BR0CR)

SC0BUF (50H)

7	6	5	4	3	2	1	0	
TB7	TB6	TB5	TB4	TB3	TB2	TB1	TB0	(Transmission)
RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	(Receiving)

Figure 3.11 (5) Serial Transmission / Receiving Buffer Registers (channel 0, SC0BUF)

SC1MOD (0056H)

	7	6	5	4	3	2	1	0
bit Symbol	TB8	—	RXE	WU	SM1	SM0	SC1	SC0
Read/Write	R/W							
After reset	0	0	0	0	0	0	0	0
Function	Transfered data Bit 8	Fix at "0"	Receiving Function 0: Receive disable 1: Receive Enable	Wake up Function 0: disable 1: enable	Serial Transmission mode 00: I/O interface mode 01: 7-Bit UART 10: 8-Bit UART 11: 9-Bit UART		Serial Transmission clock (UART) 00: TO0 Trigger 01: Baud rate generator 10: Internal clock Φ1 11: don't care	

Serial transmission clock (For UART)

00	Timer 0 match detect signal
01	Baud rate generator
10	Internal clock Φ1
11	don't care

Note: The clock selection for the I/O interface mode is controlled by the serial control register (SC1CR).

Serial transmission mode

00	I/O interface mode	
01	UART mode	7-Bit length
10		8-Bit length
11		9-Bit length

Wake up Function

	9-Bit UART	Other mode
0	Interrupt when data are received	don't care
1	Interrupt only when RB8 = 1	

Receiving control

0	Receive disable
1	Receive enable

Transmission data bit 8

Figure 3.11 (6) Serial Mode Control Register (Channel 1, SC1MOD)

SC1CR (0055H)

	7	6	5	4	3	2	1	0
bit Symbol	RB8	EVEN	PE	OERR	PERR	FERR	SCLKS	IOC
Read/Write	R	R/W		R (clear to Zero when read)			R/W	
After reset		0	0	0	0	0	0	0
Function	Received data Bit 8	Parity 0: Odd 1: Even	Parity addition 0: Disable 1: Enable	1: error / Overrun	Parity	Framing	0: SCLK1 (↑) / 1: SCLK1 (↓)	0: Baud rate generator 1: SCLK1 Pin input

Select I/O interface input clock (Note1)

0	Baud rate generate
1	SCLK1 Pin input

Note1 : For channel 0, fix this bit to "0"

Edge selection in SCLK pin input mode

0	Transmits and receives (↑) data at raise edge of SCLK
1	Transmits and receives (↓) data at fall edge of SCLK

Framing error flag
Parity error flag
Overrun error flag
} Cleared to Zero when read

Enable parity addition

0	Disable
1	Enable

Addition / check of even parity

0	Odd Parity
1	Even Parity

Receiving data bit 8

Note : As all error flags are cleared after reading, do not test only a single bit with a bit-testing instruction.

Figure 3.11 (7) Serial Control Register (Channel 1, SC1CR)

BR1CR (0057H)

	7	6	5	4	3	2	1	0
bit Symbol	—		BR1CK1	BR1CK0	BR1S3	BR1S2	BR1S1	BR1S0
Read/Write			R/W					
After reset	0		0	0	0	0	0	0
Function	Fix at "0"		00: ϕT0 (fc/4) 01: ϕT2 (fc/16) 10: ϕT8 (fc/64) 11: ϕT32 (fc/256)		Setting of the Divided frequency			

Setting of the divided frequency of baud rate generator

0000	16 divided
0001	Don't set
0010 ∫ 1111	2 to 5 divisions

Selecting the input clock of baud rate generator

00	Internal clock ϕT0 (fc/4)
01	Internal clock ϕT2 (fc/16)
10	Internal clock ϕT8 (fc/64)
11	Internal clock ϕT32 (fc/256)

Note : To use baud rate generator, set TRUN<PRRUN>to "1", putting the prescaler in RUN mode.

Figure 3.11 (8) Baud Rate Generator Control Register (channel 0, BR0CR)

SC1BUF (0054H)

7	6	5	4	3	2	1	0	
TB7	TB6	TB5	TB4	TB3	TB2	TB1	TB0	(Transmission)
RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	(Receiving)

Figure 3.11 (9) Serial Transmission / Receiving Buffer Registers (channel 1, SC1BUF)

P9FC (001DH)

	7	6	5	4	3	2	1	0
bit Symbol			P95F		P93F	P92F		P90F
Read/Write			W		W	W		W
			0		0	0		0
			0: PORT 1: SCLK1		0: PORT 1: TxD1	0: PORT 1: SCLK0		0: PORT 1: TxD0

Prohibit Read modify write

Note: The TMP96CM40 and TMP96PM40 have register P92F, The TMP96C141 does not. (In other wordes, SCLK 0 cannot be specified.)

Setting P90 as TxD0 output

0	Port output
1	TxD0 (channel 0) output

Setting P93 as TxD1 output

0	Port output
1	TxD1 (channel 1) output

Setting P95 as SCLK output

0	Port output
1	SCLK1 (channel 1) output

Figure 3.11 (10) Port 9 Function Register (P9FC)

ODE (0058H)

	7	6	5	4	3	2	1	0
bit Symbol							ODE1	ODE0
Read/Write							R/W	
							0	0
							P93 0: CMOS 1: OPEN Drain	P90 0: CMOS 1: OPEN Drain

Setting P90 as Open-drain output

0	CMOS output
1	Open-drain output

Setting P93 as Open-drain

0	CMOS output
1	Open-drain output

Figure 3.11 (11) Port 9 Open Drain Enable Register (ODE)

3.11.2 Configuration

Figure 3.11 (12) shows the block diagram of the serial channel 0.

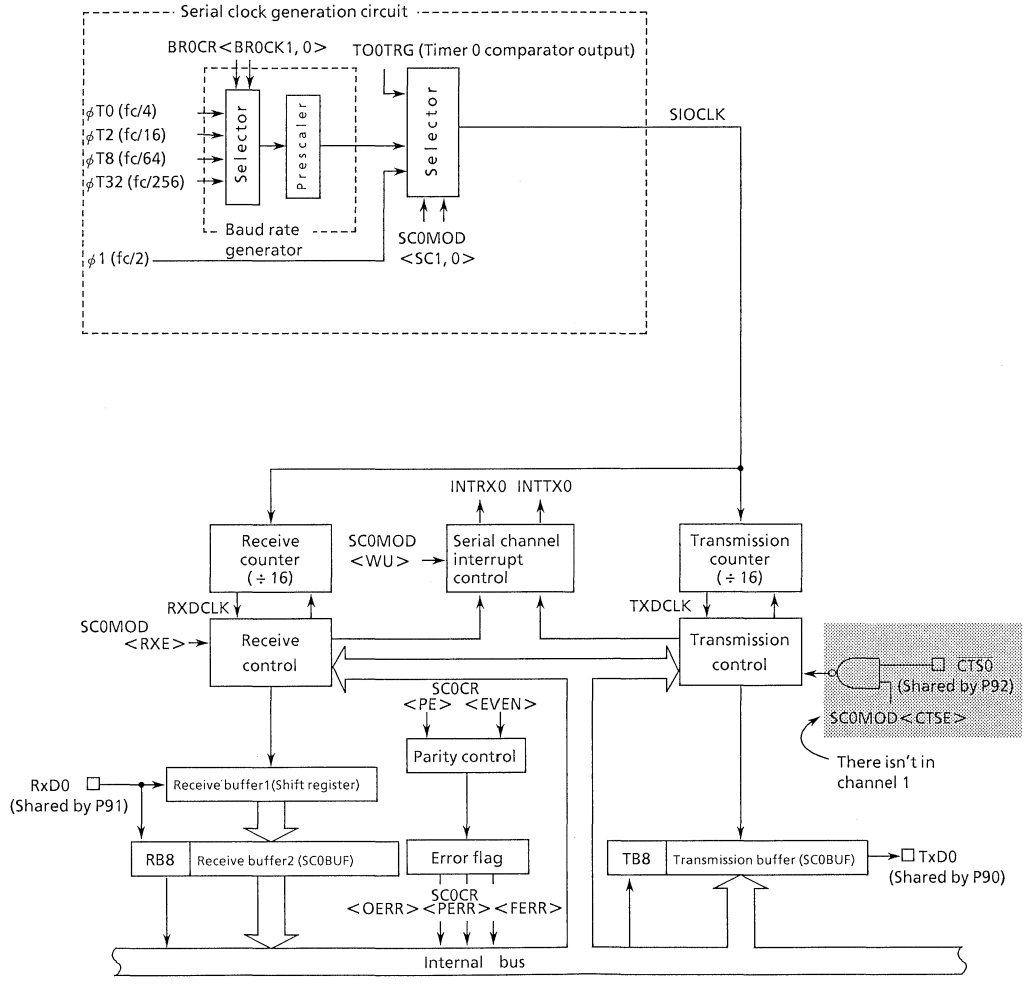


Figure 3.11 (12) Block Diagram of the Serial Channel 0

Figure 3.11 (13) shows the block diagram of the serial channel 1.

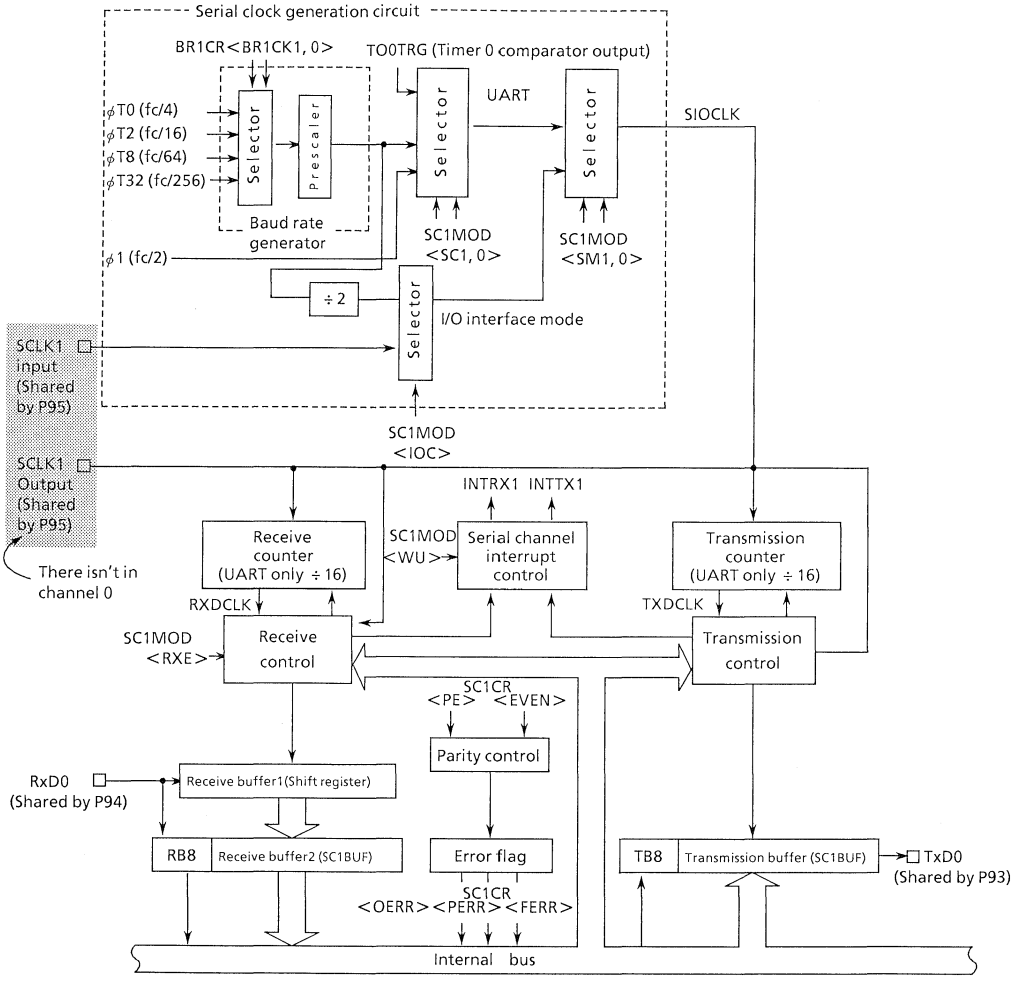


Figure 3.11 (13) Block Diagram of the Serial Channel 1

① Baud Rate Generator

Baud rate generator comprises a circuit that generates transmission and receiving clocks to determine the transfer rate of the serial channel.

The input clock to the baud rate generator, ϕT0 (fc/4), ϕT2 (fc/16), ϕT8 (fc/64), or ϕT32 (fc/256) is generated by the 9-bit prescaler which is shared by the timers. One of these input clocks is selected by the baud rate generator control register BR0CR/BR1CR<BR0CK1, 0/BR1CK1, 0>.

The baud rate generator includes a 4-bit frequency divider, which divides frequency by 2 to 16 values to determine the transfer rate.

How to calculate a transfer rate when the baud rate generator is used is explained below.

- UART mode

$$\text{Transfer rate} = \frac{\text{Input clock of baud rate generator}}{\text{Frequency divisor of baud rate generator}} \div 16$$

- I/O interface mode

$$\text{Transfer rate} = \frac{\text{Input clock of baud rate generator}}{\text{Frequency divisor of baud rate generator}} \div 2$$

The relation between the input clock and the source clock (fc) is as follows.

ϕT0 = fc/4
ϕT2 = fc/16
ϕT8 = fc/64
ϕT32 = fc/256

Accordingly, when source clock fc is 12.288 MHz, input clock is ϕT2 (fc/16), and frequency divisor is 5, the transfer rate in UART mode becomes as follows:

$$\text{Transfer rate} = \frac{\text{fc}/16}{5} \div 16$$

$$= 12.288 \times 10^6/16/5/16 = 9600 \text{ (bps)}$$

Table 3.11 (1) shows an example of the transfer rate in UART mode.

Also with 8-bit timer 0, the serial channel can get a transfer rate. Table 3.9 (2) shows an example of baud rate using timer 0.

Table 3.11 (1) Selection of Transfer Rate (1) (When Baud Rate Generator Is Used)

Unit (Kbps)

fc [MHz]	Input clock / Frequency divisor	ϕT0 (fc/4)	ϕT2 (fc/16)	ϕT8 (fc/64)	ϕT32 (fc/256)
9.830400	2	76.800	19.200	4.800	1.200
↑	4	38.400	9.600	2.400	0.600
↑	8	19.200	4.800	1.200	0.300
↑	0	9.600	2.400	0.600	0.150
12.288000	5	38.400	9.600	2.400	0.600
↑	A	19.200	4.800	1.200	0.300
14.745600	3	76.800	19.200	4.800	1.200
↑	6	38.400	9.600	2.400	0.600
↑	C	19.200	4.800	1.200	0.300

Note: Transfer rate in I/O interface mode is 8 times as fast as the values given in the above table.

Table 3.11 (2)Selection of Transfer Rate (1) (When timer 0 (input Clock ϕT1) is used

Unit (Kbps)

fc / TREG0	12.288 MHz	12 MHz	9.8304 MHz	8 MHz	6.144 MHz
1H	96		76.8	62.5	48
2H	48		38.4	31.25	24
3H	32	31.25			16
4H	24		19.2		12
5H	19.2				9.6
8H	12		9.6		6
AH	9.6				4.8
10H	6		4.8		3
14H	4.8				2.4

How to calculate the transfer rate (when timer 0 is used):

$$\text{Transfer rate} = \frac{fc}{TREG0 \times 8 \times 16}$$

(When Timer 0 (input clock ϕT1) is used)

Input clock of timer 0

ϕT1 = $fc/8$
ϕT4 = $fc/32$
ϕTI6 = $fc/128$

Note: Timer 0 match detect signal cannot be used as the transfer clock in I/O interface mode.

② Serial Clock Generation Circuit

This circuit generates the basic clock for transmitting and receiving data.

1) I/O interface mode (channel 1 only)

When in SCLK output mode with the setting of SC1CR<IOC> = "0", the basic clock will be generated by dividing by 2 the output of the baud rate generator described before. When in SCLK input mode with the setting of SC1CR<IOC> = "1", the rising edge or falling edge will be detected according to the setting of SC1CR<SCLKC> register to generate the basic clock.

2) Asynchronous Communication (UART) mode

According to the setting of SC0CR and SC1CR <SC1, 0>, the above baud rate generator clock, internal clock ϕ1 (500Kbps @ fc=16MHz), or the match detect signal from timer 0 will be selected to generate the basic clock SIOCLK.

③ Receiving Counter

The receiving counter is a 4-bit binary counter used in asynchronous communication (UART) mode and counts up by SIOCLK clock. 16 pulses of SIOCLK are used for receiving 1 bit of data, and the data bit is sampled three times at 7th, 8th and 9th clock.

With the three samples, the received data is evaluated by the rule of majority.

For example, if the sampled data bit is "1", "0" and "1" at 7th, 8th and 9th clock respectively, the received data is evaluated as "1". The sampled data "0" , "0" and "1" is evaluated that the received data is "0".

④ Receiving Control

1) I/O interface mode (channel 1 only)

When in SCLK1 output mode with the setting of SC1CR<IOC> = "0", RxD1 signal will be sampled at the rising edge of shift clock which is output to SCLK pin.

When in SCLK input mode with the setting SC1CR<IOC> = "1" RxD1 signal will be sampled at the rising edge or falling edge of SCLK input according to the setting of SC1CR<SCLKS> register.

2) Asynchronous communication (UART) mode

The receiving control has a circuit for detecting the start bit by the rule of majority. When two or more "0" are detected during 3 samples, it is recognized as start bit and the receiving operation is started.

Data being received are also evaluated by the rule of majority.

⑤ Receiving Buffer

To prevent overrun error, the receiving buffer has a double buffer structure.

Received data are stored one bit by one bit in the receiving buffer 1 (shift register type). When 7 bits or 8 bits of data is stored in the receiving buffer 1, the stored data are transferred to another receiving buffer 2 (SC0BUF/SC1BUF), generating an interrupt INTRX0/INTRX1. The CPU reads only receiving buffer 2 (SC0BUF/SC1BUF). Even before the CPU reads the receiving buffer 2 (SC0BUF/SC1BUF), the received data can be stored in the receiving buffer 1. However, unless the receiving buffer 2 (SC0BUF/SC1BUF) is read before all bits of the next data are received by the receiving buffer 1, an overrun error occurs. If an overrun error occurs, the contents of the receiving buffer 1 will be lost, although the contents of the receiving buffer 2 and SC0CR<RB8> SC1CR<RB8>is still preserved.

The parity bit added in 8-bit UART mode and the most significant bit (MSB) in 9-bit UART mode are stored in SC0CR<RB8>/ SC1CR<RB8>.

When in 9-bit UART mode, the wake-up function of the slave controllers is enabled by setting SC0MOD<WU>/SC1MOD<WU> to "1", and interrupt INTRX0/INTRX1 occurs only when SC0CR<RB8> / SC1CR<RB8>is set to "1".

⑥ Transmission Counter

Transmission counter is a 4-bit binary counter which is used in asynchronous communication (UART) mode and, like a receiving counter, counts by SIOCLK clock, generating TxDCLK every 16 clock pulses.

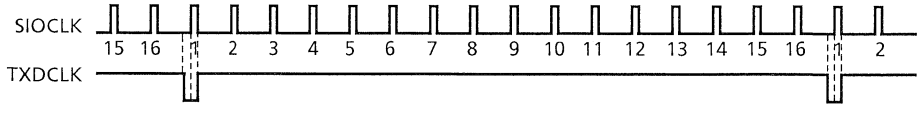


Figure 3.11 (14) Generation of Transmission Clock

⑦ Transmission Controller

1) I/O interface mode (channel 1 only)

In SCLK output mode with the setting of SC1CR<IOC> = "0", the data in the transmission buffer are output bit by bit to TxD1 pin at the rising edge of shift clock which is output from SCLK1 pin.

In SCLK input mode with the setting of SC1CR<IOC> = "1", the data in the transmission buffer are output bit by bit to TxD1 pin at the rising edge or falling edge of SCLK input according to the setting of SC1CR<SCLKC> register.

2) Asynchronous communication (UART) mode

When transmission data are written in the transmission buffer sent from the CPU, transmission starts at the rising edge of the next TxDCLK, generating a transmission shift clock TxDSFT.

Handshake function

Serial channel 0 has a $\overline{CTS0}$ pin. Using this pin, data can be sent in units of one frame ; thus, overrun errors can be avoided. The handshake function is enabled/ disabled by SC0MOD<CTSE>.

When the $\overline{CTS0}$ pin goes high, after completion of the current data send, data send is halted until the $\overline{CTS0}$ pin goes low again. The INTTX0 Interrupts are generated, requests the next send data to the CPU.

Though there is no $\overline{RTS}$ pin, a handshake function can be easily configured by setting any port assigned to the $\overline{RTS}$ function. The $\overline{RTS}$ should be output "High" to request data send halt after data receive is completed by a software in the RXD interrupt routine.

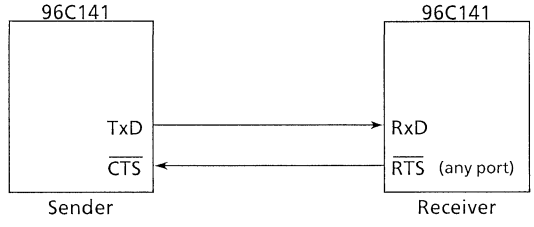
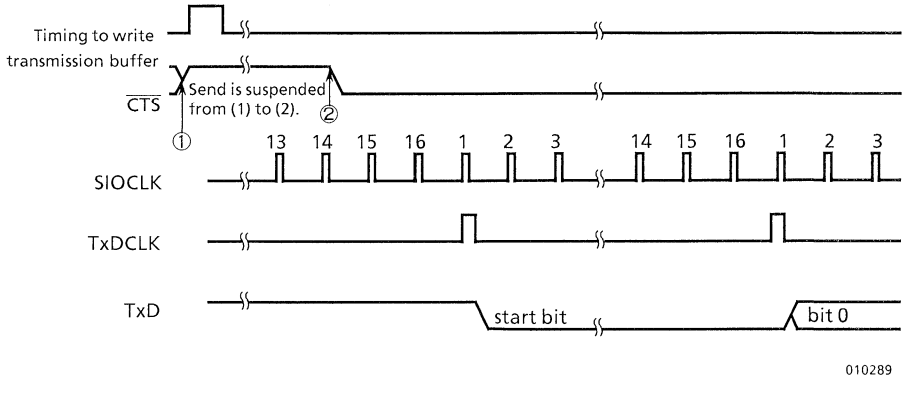


Figure 3.11 (15) Handshake Function



Note 1 : If the $\overline{CTS}$ signal falls during transmission, the next data is not sent after the completion of the current transmission.

Note 2 : Transmission starts at the first TxDCLK clock fall after the $\overline{CTS}$ signal falls.

Figure 3.11 (16) Timing of $\overline{CTS}$ (Clear to send)

⑧ Transmission Buffer

Transmission buffer (SC0BUF/SC1BUF) shifts out and sends the transmission data written from the CPU from the least significant bit (LSB) in order, using transmission shift clock TxDSFT which is generated by the transmission control. When all bits are shifted out, the transmission buffer becomes empty and generates INTTX0/INTTX1 interrupt.

⑨ Parity Control Circuit

When serial channel control register SC0CR<PE>/SC1CR<PE>is set to "1", it is possible to transmit and receive data with parity. However, parity can be added only in 7-bit UART or 8-bit UART mode. With SC0CR <EVEN> / SC1CR <EVEN> register, even (odd) parity can be selected.

For transmission, parity is automatically generated according to the data written in the transmission buffer SCBUF, and data are transmitted after being stored in SC0BUF<TB7>/SC1BUF<TB7> when in 7-bit UART mode while in SCMOD <TB8> / SCMOD <TB8> when in 8-bit UART mode. <PE> and <EVEN> must be set before transmission data are written in the transmission buffer.

For receiving, data are shifted in the receiving buffer 1, and parity is added after the data are transferred in the receiving buffer 2 (SC0BUF/SC1BUF), and then compared with SC0BUF<RB7>/SC1BUF<RB7> when in 7-bit UART mode and with SC0MOD<RB8>/SC1MOD<RB8> when in 8-bit UART mode. If they are not equal, a parity error occurs, and SC0CR<PERR>/SC1CR<PERR> flag is set.

⑩ Error Flag

Three error flags are provided to increase the reliability of receiving data.

1. Overrun error <OERR>

   If all bits of the next data are received in receiving buffer 1 while valid data are stored in receiving buffer 2 (SCBUF), an overrun error will occur.

2. Parity error <PERR>

   The parity generated for the data shifted in receiving buffer 2 (SCBUF) is compared with the parity bit received from RxD pin. If they are not equal, a parity error occurs.

3. Framing error <FERR>

   The stop bit of received data is sampled three times around the center. If the majority is "0", a framing error occurs.

⑪ Generating Timing

1) UART mode

Receiving

Mode	9 Bit	8 Bit + parity	8 Bit, 7 Bit + parity, 7 Bit
Interrupt timing	Center of last bit (Bit 8)	Center of last bit (parity bit)	Center of stop bit
Framing error timing	Center of stop bit	Center of stop bit	Center of stop bit
Parity error timing	Center of last bit (Bit 8)	Center of last bit (parity bit)	Center of stop bit
Overrun error timing	Center of last bit (Bit 8)	Center of last bit (parity bit)	Center of stop bit

Note: Framing error occurs after an interrupt has occurred. Therefore, to check for framing error during interrupt operation, it is necessary to wait for 1 bit period of transfer rate.

Transmitting

Mode	9 Bit	8 Bit + parity	8 Bit, 7 Bit + parity, 7 Bit
Interrupt timing	Just before last bit is transmitted.	←	←

2) I/O interface mode

Transmission Interrupt timing	SCLK output mode	Immediately after rise of last SCLK signal. (See figure 3.11 (19).)
	SCLK input mode	Immediately after rise of last SCLK signal (rising mode), or immediately after fall in falling mode. (See figure 3.11 (20).)
Receiving Interrupt timing	SCLK output mode	Timing used to transfer received data to data receive buffer 2 (SC1BUF) (that is, immediately after last SCLK). (See figure 3.11 (21).)
	SCLK input mode	Timing used to transfer received data to data receive buffer 2 (SC1BUF) (that is, immediately after last SCLK). (See figure 3.11 (22).)

3.11.3 Operational Description

(1) Mode 0 (I/O interface mode)

This mode is used to increase the number of I/O pins of for transmitting or receiving data to or from the external shifter register.

This mode includes SCLK output mode to output synchronous clock SCLK and SCLK input mode to input external synchronous clock SCLK.

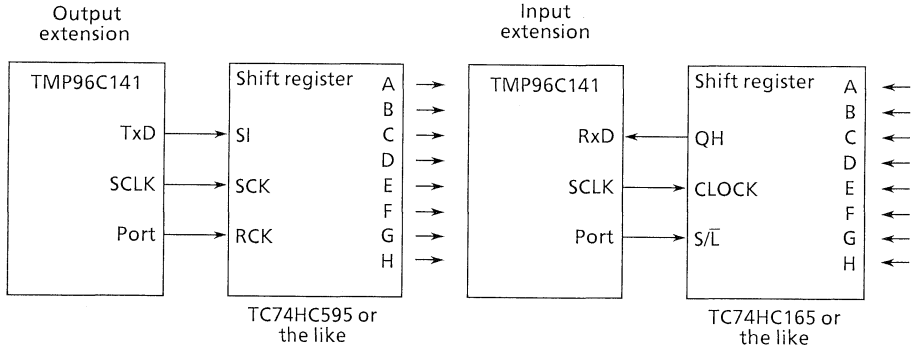


Figure 3.11 (17) Example of SCLK Output Mode Connection

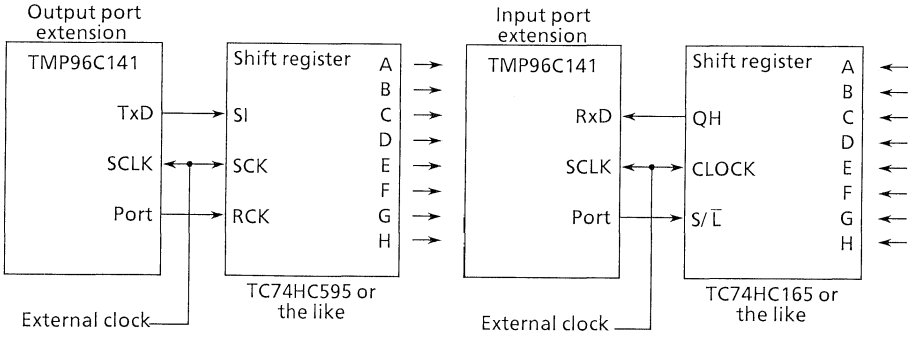


Figure 3.11 (18) Example of SCLK Input Mode Connection

① Transmission

In SCLK output mode, 8-bit data and synchronous clock are output from TxD pin and SCLK pin, respectively, each time the CPU writes data in the transmission buffer. When all data is output, INTES1<ITX1C> will be set to generate INTTX1 interrupt.

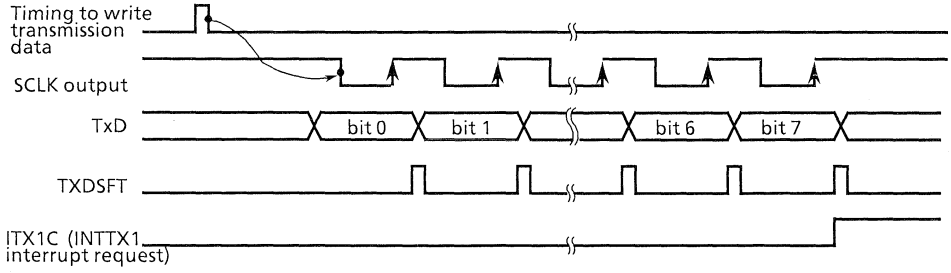


Figure 3.11 (19) Transmitting Operation in I/O Interface Mode (SCLK Output Mode)

In SCLK output mode, 8-bit data are output from TxD1 pin when SCLK input becomes active while data are written in the transmission buffer by CPU.

When all data are output, INTES1<ITXIC> will be set to generate INTTX1 interrupt.

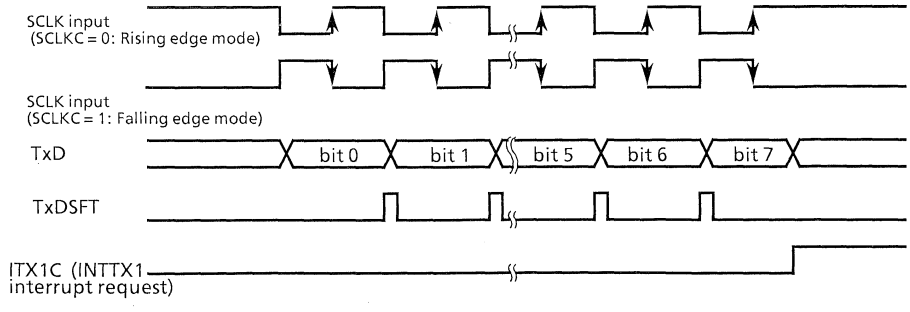


Figure 3.11 (20) Transmitting Operation in I/O Interface Mode (SCLK Input Mode)

② Receiving

In SCLK output mode, synchronous clock is outputted from SCLK pin and the data are shifted in the receiving buffer 1 whenever the receive interrupt flag INTES1<IRX1C> is cleared by reading the received data. When 8-bit data are received, the data will be transferred in the receiving buffer 2 (SC1BUF) at the timing shown below, and INTES1<IRX1C> will be set again to generate INTRX1 interrupt.

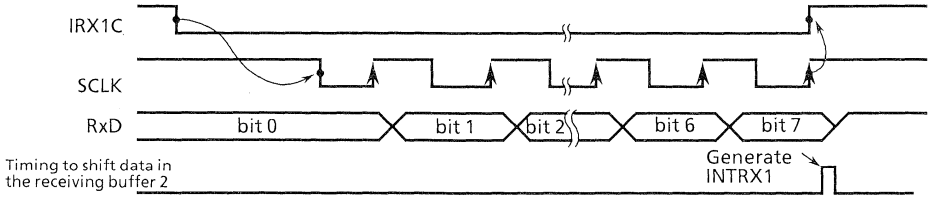


Figure 3.11 (21) Receiving Operation in I/O Interface Mode (SCLK Output Mode)

In SCLK input mode, the data is shifted in the receiving buffer 1 when SCLK input becomes active while the receive interrupt flag INTES1 <IRX1C> is cleared by reading the received data. When 8-bit data is received, the data will be shifted in the receiving buffer 2 (SC1BUF) at the timing shown below, and INTES1 <IRX1C> will be set again to generate INTRX interrupt.

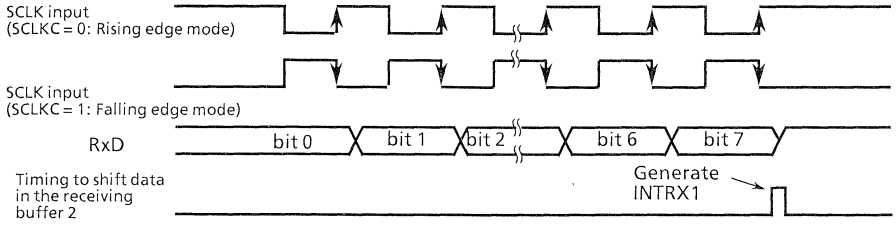


Figure 3.11 (22) Receiving Operation in I/O Interface Mode (SCLK Input Mode)

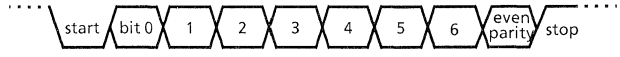
Note : For data receiving, the system must be placed in the receive enable state (SCMOD<RXE> = "1")

(2) Mode 1 (7-bit UART Mode)

7-bit mode can be set by setting serial channel mode register SC0MOD <SM1,0> / SC1MOD<SM1, 0> to "01".

In this mode, a parity bit can be added, and the addition of a parity bit can be enabled or disabled by serial channel control register SC0CR<PE>/SC1CR<PE>, and even parity or odd parity is selected by SC0CR <EVEN>/SC1CR<EVEN> when <PE> is set to "1" (enable).

Setting example: When transmitting data with the following format, the control registers should be set as described below. Channel 0 is explained here.



← Direction of transmission (transmission rate: 2400 bps @ fc = 12.288 MHz)

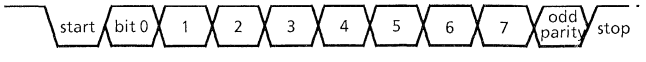
```
          7 6 5 4 3 2 1 0
P9CR    ← X X - - - - - 1   } Select P90 as the TxD pin.
P9FC    ← X X - X - X X 1
SC0MOD  ← X 0 - X 0 1 0 1     Set 7-bit UART mode.
SC0CR   ← X 1 1 X X X 0 0     Add an even parity.
BR0CR   ← 0 X 1 0 0 1 0 1     Set transfer rate at 2400 bps.
TRUN    ← 1 X - - - - - -     Start the prescaler for the baud rate generator.
INTES0  ← 1 1 0 0 - - - -     Enable INTTX0 interrupt and set interrupt level 4.
SC0BUF  ← * * * * * * * *     Set data for transmission.
```

Note: X; don't care - ; no change

(3) Mode 2 (8-bit UART Mode)

8-bit UART mode can be specified by setting SC0MOD<SM1,0>/SC1MOD<SM1, 0> to "10". In this mode, parity bit can be added, the addition of a parity bit is enabled or disabled by SC0CR<PE>/SC1CR<PE>, and even parity or odd parity is selected by SC0CR<EVEN>/SC1CR<EVEN> when <PE> is set to "1" (enable).

Setting example: When receiving data with the following format, the control register should be set as described below.



← Direction of transmission (transmission rate: 9600 bps @ fc = 12.288 MHz)

Main setting

```
                7 6 5 4 3 2 1 0
P9CR     ← X X - - - - 0 -      Select P91 (RxD) as the input pin.
SC0MOD   ← - 0 1 X 1 0 0 1      Enable receiving in 8-bit UART mode.
SC0CR    ← X 0 1 X X X 0 0      Add an odd parity.
BR0CR    ← 0 X 0 1 0 1 0 1      Set transfer rate at 9600 bps.
TRUN     ← 1 X - - - - - -      Start the prescaler for the baud rate generator.
INTES0   ← - - - - 1 1 0 0      Enable INTTX0 interrupt and set interrupt level 4.
```

Interrupt processing

```
Acc ← SC0CR AND 00011100    } Check for error.
if Acc ≠ 0 then ERROR
Acc ← SC0BUF                  Read the received data.
```

Note : X ; don't care - ; no change

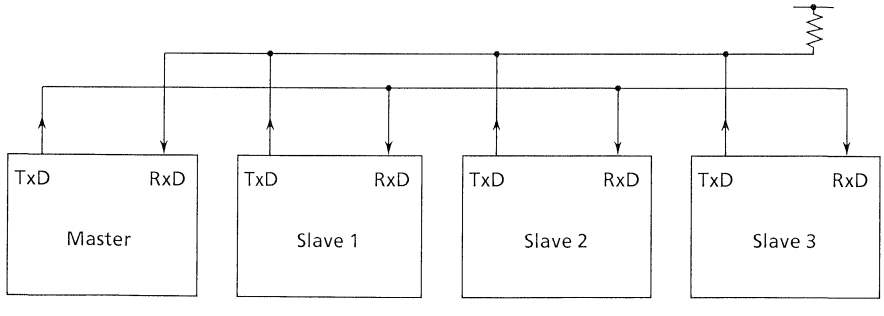
(4) Mode 3 (9-bit UART Mode)

9-bit UART mode can be specified by setting SC0MOD<SM1,0>/SC1MOD<SM 1, 0> to "11". In this mode, parity bit cannot be added.

For transmission, the MSB (9th bit) is written in SCMOD <TB8>, while in receiving it is stored in SCCR<RB8>. For writing and reading the buffer, the MSB is read or written first then SC0BUF/SC1BUF.

Wake-up function

In 9-bit UART mode, the wake-up function of slave controllers is enabled by setting SC0MOD<WU>/SC1MOD<WU> to "1". The interrupt INTRX1/INTRX0 occurs only when<RB8> = 1.



Note : TxD pin of the slave controllers must be in open drain output mode.

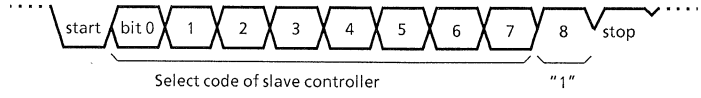
Figure 3.11 (23) Serial Link Using Wake-Up Function

Protocol

① Select the 9-bit UART mode for the master and slave controllers.

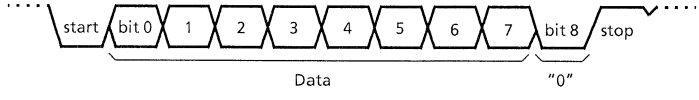
② Set SC0MOD<WU>/SC1MOD<WU> bit of each slave controller to "1" to enable data receiving.

③ The master controller transmits one-frame data including the 8-bit select code for the slave controllers. The MSB (bit 8)<TB8> is set to "1".



④ Each slave controller receives the above frame, and clears WU bit to "0" if the above select code matches its own select code.

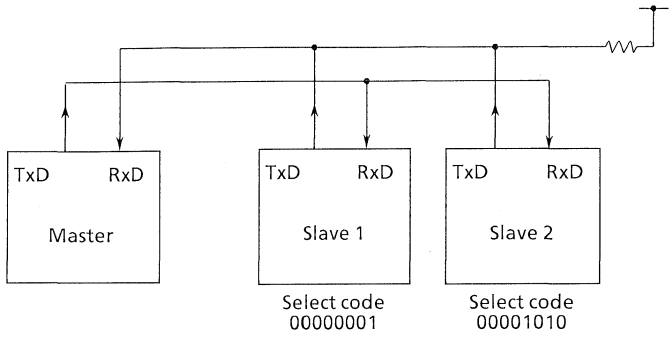
⑤ The master controller transmits data to the specified slave controller whose SC0MOD<WU>/SC1MOD<WU> bit is cleared to "0". The MSB (bit 8)<TB8> is cleared to "0".



⑥ The other slave controllers (with the <WU> bit remaining at "1") ignore the receiving data because their MSBs (bit 8 or <RB8>) are set to "0" to disable the interrupt INTRX0/INTRX1.

The slave controllers (WU = 0) can transmit data to the master controller, and it is possible to indicate the end of data receiving to the master controller by this transmission.

Setting example: To link two slave controllers serially with the master controller, and use the internal clock ϕ1 (fc/2) as the transfer clock.



Since serial channels 0 and 1 operate in exactly the same way, channel 0 is used for the purposes of explanation.

- Setting the master controller

Main

```
P9CR   ← X X - - - - 0 1   } Select P90 as TxD pin and P91 as RxD pin.
P9FC   ← X X - X - X X 1   }
INTES0 ← 1 1 0 0 1 1 0 1     Enable INTTX0 and set the interrupt level 4.
                             Enable INTRX0 and set the interrupt level 5.
SC0MOD ← 1 0 1 0 1 1 1 0     Set φ1 (fc/2) as the transmission clock in 9-bit UART mode.
SC0BUF ← 0 0 0 0 0 0 0 1     Set the select code for slave controller 1.
```

INTTX0 interrupt

```
SC0MOD ← 0 - - - - - - -     Sets TB8 to "0".
SC0BUF ← * * * * * * * *     Set data for transmission.
```

- Setting the slave controller 2

Main

```
P9CR   ← X X - - - - 0 1   } Select P91 as RxD pin and P90 as TxD pin (open drain
P9FC   ← X X - X - X X 1   } output).
ODE    ← X X X X X X - 1   }
INTES0 ← 1 1 0 1 1 1 1 0     Enable INTRX0 and INTTX0.
SC0MOD ← 0 0 1 1 1 1 1 0     Set <WU> to "1" in the 9-bit UART transmission mode
                             with transfer clock φ1 (fc/2).
```

INTRX0 interrupt

```
Acc ← SC0BUF
if Acc = Select code
Then SC0MOD4 ← - - - 0 - - - -   Clear <WU> to "0".
```

3.12 Analog/Digital Converter

TMP96C141/TMP96CM40/TMP96PM40 contains a high-speed analog / digital converter (A/D converter) with 4-channel analog input that features 10-bit successive approximation.

Figure 3.12 (1) shows the block diagram of the A/D converter. 4-channel analog input pins (AN3 to AN0) are shared by input-only port P5 and so can be used as input port.

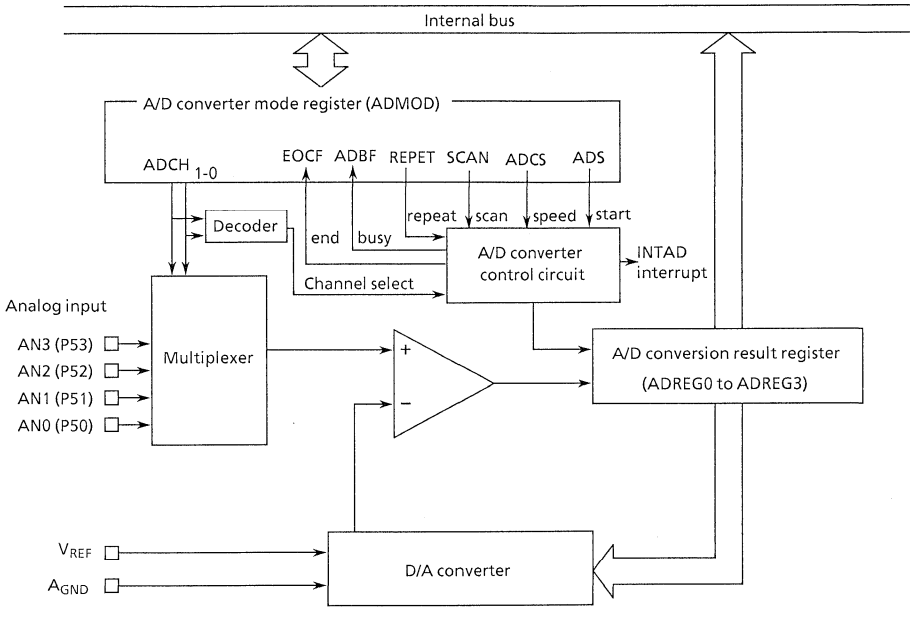


Figure 3.12 (1) Block Diagram of A/D Converter

Note : This A/D converter does not have a built-in sample and hold circuit.
Therefore, when A/D converting high-frequency signals, connect a sample and hold circuit externally.

ADMOD (005EH)

	7	6	5	4	3	2	1	0
bit Symbol	EOCF	ADBF	REPET	SCAN	ADCS	ADS	ADCH1	ADCH0
Read/Write	R		R/W					
After reset	0	0	0	0	0	0	0	0
Function	A/D conversion End Flag 1: END	A/D conversion BUSY Flag 1: BUSY	Repeat mode 0: Single mode 1: Repeat mode	Scan mode 0: Fixed channel mode 1: Channel Scan mode	A/D conversion Speed 0: High speed mode 1: Low speed mode	A/D conversion Start 1: conversion Start Always read as "0"	Analog Input Channel Select	

Analog input channel select

	Normal	Scan mode
00	AN0	AN0
01	AN1	AN0→AN1
10	AN2	AN0→AN1→AN2
11	AN3	AN0→AN1→AN2→AN3

A/D conversion start

0	—
1	Start A/D conversion.

Note) Always "0" when data is read.

A/D conversion speed selection

0	A/D High speed conversion mode: 160 states = 20 μs (@16MHz)
1	A/D Low speed conversion mode : 320 states = 40 μs (@16MHz)

Specification of A/D scan mode

0	Fixed A/D conversion channel mode
1	A/D conversion channel scan mode

Selecting A/D repeat mode

0	A/D conversion single mode
1	A/D conversion repeat mode

A/D conversion busy flag

0	A/D conversion not busy
1	A/D conversion busy

A/D conversion end flag

0	A/D conversion not ended nor started
1	A/D conversion ended

Figure 3.12 (2) A/D Control Register

ADREG0L (0060H)

	7	6	5	4	3	2	1	0
bit Symbol	ADR01	ADR00						
Read/Write	R							
After reset	Undefined		1	1	1	1	1	1
Function	Lower 2 bits of A/D result for AN0 are stored.							

ADREG0H (0061H)

	7	6	5	4	3	2	1	0
bit Symbol	ADR09	ADR08	ADR07	ADR06	ADR05	ADR04	ADR03	ADR02
Read/Write	R							
After reset	Undefined							
Function	Upper 8 bits of A/D result for AN0 are stored.							

ADREG1L (0062H)

	7	6	5	4	3	2	1	0
bit Symbol	ADR11	ADR10						
Read/Write	R							
After reset	Undefined		1	1	1	1	1	1
Function	Lower 2 bits of A/D result for AN1 are stored.							

ADREG1H (0063H)

	7	6	5	4	3	2	1	0
bit Symbol	ADR19	ADR18	ADR17	ADR16	ADR15	ADR14	ADR13	ADR12
Read/Write	R							
After reset	Undefined							
Function	Upper 8 bits of A/D result for AN1 are stored.							

Figure 3.12 (3-1) A/D Conversion Result Register (ADREG0, 1)

ADREG2L (0064H)

	7	6	5	4	3	2	1	0
bit Symbol	ADR21	ADR20						
Read/Write	R							
After reset	Undefined		1	1	1	1	1	1
Function	Lower 2 bits of A/D result for AN2 are stored.							

ADREG2H (0065H)

	7	6	5	4	3	2	1	0
bit Symbol	ADR29	ADR28	ADR27	ADR26	ADR25	ADR24	ADR23	ADR22
Read/Write	R							
After reset	Undefined							
Function	Upper 8 bits of A/D result for AN2 are stored.							

ADREG3L (0066H)

	7	6	5	4	3	2	1	0
bit Symbol	ADR31	ADR30						
Read/Write	R							
After reset	Undefined		1	1	1	1	1	1
Function	Lower 2 bits of A/D result for AN3 are stored.							

ADREG3H (0067H)

	7	6	5	4	3	2	1	0
bit Symbol	ADR39	ADR38	ADR37	ADR36	ADR35	ADR34	ADR33	ADR32
Read/Write	R							
After reset	Undefined							
Function	Upper 8 bits of A/D result for AN3 are stored.							

Figure 3.12 (3-2) A/D Conversion Result Register (ADREG2, 3)

3.12.1 Operation

(1) Analog Reference Voltage

High analog reference voltage is applied to the VREF pin, and the low analog reference voltage is applied to AGND pin.

The reference voltage between VREF and AGND is divided by 1024 using ladder resistance, and compared with the analog input voltage for A/D conversion.

(2) Analog Input Channels

Analog input channel is selected by ADMOD<ADCH1,0>. However, which channel to select depends on the operation mode of the A/D converter.

In fixed analog input mode, one channel is selected by ADMOD<ADCH1,0> among four pins: AN0 to AN3.

In analog input channel scan mode, the number of channels to be scanned from AN0 is specified by ADMOD<ADCH1,0>, such as AN0→AN1, AN0→AN1→AN2, and AN0→AN1→AN2→AN3.

When reset, A/D conversion channel register will be initialized to ADMOD<ADCH1,0> = 00, so that AN0 pin will be selected.

The pins which are not used as analog input channel can be used as ordinary input port P5.

(3) Starting A/D Conversion

A/D conversion starts when A/D conversion register ADMOD<ADS> is written "1". When A/D conversion starts, A/D conversion busy flag ADMOD<ADBF> which indicates "A/D conversion is in progress" will be set to "1".

(4) A/D Conversion Mode

Both fixed A/D conversion channel mode and A/D conversion channel scan mode have two conversion modes, i.e., single and repeat conversion modes.

In fixed channel repeat mode, conversion of specified one channel is executed repeatedly.

In scan repeat mode, scanning from AN0, ··· →AN3 is executed repeatedly.

A/D conversion mode is selected by ADMOD<REPET, SCAN>.

(5) A/D Conversion Speed Selection

There are two A/D conversion speed modes: high speed mode and low speed mode. The selection is executed by ADMOD<ADCS> register.

When reset, ADMOD<ADCS> will be initialized to "0", so that high speed conversion mode will be selected.

(6) A/D Conversion End and Interrupt

- A/D conversion single mode

ADMOD<EOCF> for A/D conversion end will be set to "1", ADMOD <ADBF> flag will be reset to "0", and INTAD interrupt will be enabled when A/D conversion of specified channel ends in fixed conversion channel mode or when A/D conversion of the last channel ends in channel scan mode.

- A/D conversion repeat mode

  For both fixed conversion channel mode and conversion channel scan mode, INTAD should be disabled when in repeat mode. Always set the INTE0AD at "000", that disables the interrupt request.

  Write "0" to ADMOD<REPET> to end the repeat mode. Then, the repeat mode will be exited as soon as the conversion in progress is completed.

(7) Storing the A/D Conversion Result

The results of A/D conversion are stored in ADREG0 to ADREG3 registers for each channel. In repeat mode, the registers are updated whenever conversion ends.

ADREG0 to ADREG3 are read-only registers.

(8) Reading the A/D Conversion Result

The results of A/D conversion are stored in ADREG0 to ADREG3 registers. When the contents of one of ADREG0 to ADREG3 registers are read, ADMOD<EOCF> will be cleared to "0".

Setting example: ① When the analog input voltage of the AN3 pin is A/D converted and the result is stored in the memory address FF10H by A/D interrupt INTAD routine

Main setting

```
INTE0AD  ← 1 1 0 0 - - - -    Enable INTAD and set interrupt level 4.
ADMOD    ← X X 0 0 0 1 1 1    Specify AN3 pin as an analog input channel and
                              starts A/D conversion in high speed mode.
```

INTAD routine

```
WA          ← ADREG3          Read ADREG3L and ADREG3H values and write to
                              WA (16 bit)
WA          >> 6              Right-shifts WA six times and writes 0 in upper
                              bits.
(00FF10H)←  WA                Writes contents of WA in memory at FF10H
```

When the analog input voltage of AN0~AN2 pins is A/D converted in high speed conversion channel scan repeat mode

```
INTE0AD  ← 1 0 0 0 - - - -    Disable INTAD.
ADMOD    ← X X 1 1 0 1 1 0    Start the A/D conversion of analog input channels
                              AN0~AN2 in the high-speed scan repeat mode.
```

Note: X; don't care -; no change

3.13 Watchdog Timer (Runaway Detecting Timer)

TMP96C141/TMP96CM40/TMP96PM40 are containing watchdog timer of Runaway detcting.

The watchdog timer (WDT) is used to return the CPU to the normal state when it detects that the CPU has started to malfunction (runaway) due to causes such as noise. When the watchdog timer detects a malfunction, it generates a non-maskable interrupt to notify the CPU of the malfunction, and outputs 0 externally from watchdog timer out pin WDTOUT to notify the peripheral devices of the malfunction.

Connecting the watchdog timer output to the reset pin internally forces a reset.

3.13.1 Configuration

Figure 3.13 (1) shows the block diagram of the watchdog timer (WDT).

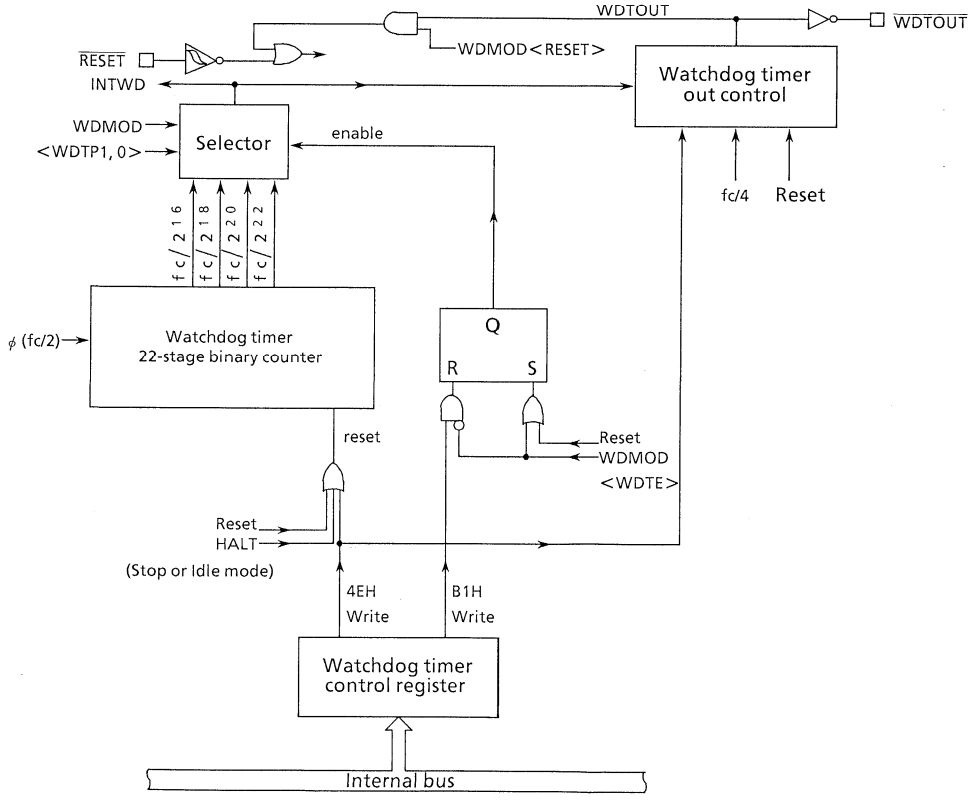


Figure 3.13 (1) Block Diagram of Watchdog Timer

The watchdog timer is a 22-stage binary counter which uses ϕ(fc/2) as the input clock. There are four outputs from the binary counter: 2^{16}/fc, 2^{18}fc, 2^{20}/fc, and 2^{22}/fc. Selecting one of the outputs with the WDMOD register generates a watchdog interrupt, and outputs watchdog timer out when an overflow occurs.

Since the watchdog timer out pin ($\overline{\text{WDTOUT}}$) outputs "0" due to a watchdog timer overflow, the peripheral devices can be reset. The watchdog timer out pin is set to 1 by clearing the watchdog timer (by writing a clear code 4EH in the WDCR register). In other words, the $\overline{\text{WDTOUT}}$ keeps outputting "0" until the clear code is written.

The watchdog timer out pin can also be connected to the reset pin internally. In this case, the watchdog timer out pin ($\overline{\text{WDTOUT}}$) outputs 0 at 32/fc=2.0 us (fc: 16MHz) and resets itself.

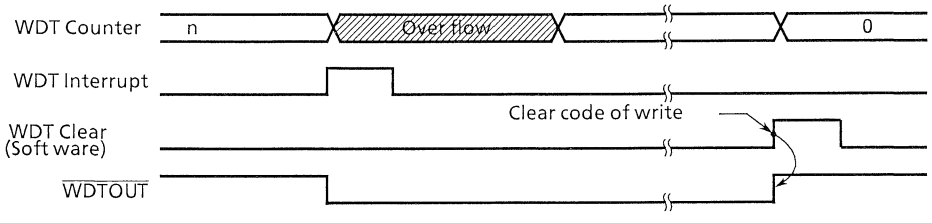


Figure 3.13 (2) Normal Mode

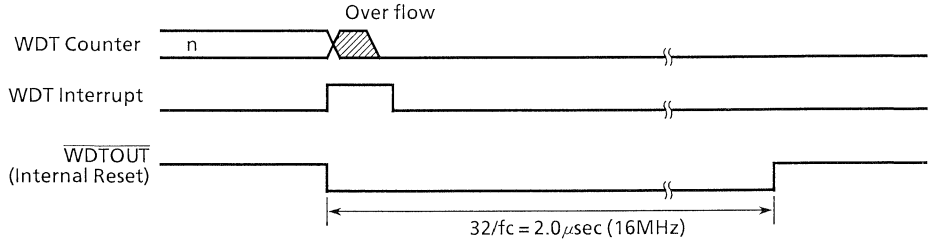


Figure 3.13 (3) Reset Mode

3.13.2 Control Registers

Watchdog timer WDT is controlled by two control registers WDMOD and WDCR.

(1) Watchdog Timer Mode Register (WDMOD)

① Setting the detecting time of watchdog timer <WDTP>

This 2-bit register is used to set the watchdog timer interrupt time for detecting the runaway. This register is initialized to WDMOD<WDTP1, 0>=00 when reset, and therefore 2^{16}/fc is set. (The number of states is approx. 32,768.)

② Watchdog timer enable/disable control register <WDTE>

When reset, WDMOD<WDTE> is initialized to "1" enable the watchdog timer.

To disable, it is necessary to clear this bit to "0" and write the disable code (B1H) in the watchdog timer control register WDCR. This makes it difficult for the watchdog timer to be disabled by runaway.

However, it is possible to return from the disable state to enable state by merely setting <WDTE> to "1".

③ Watchdog timer out reset connection <RESCR>

This register is used to connect the output of the watchdog timer with $\overline{\text{RESET}}$ terminal, internally. Since WDMOD<RESCR> is initialized to 0 at reset, a reset by the watchdog timer will not be performed.

(2) Watchdog Timer Control Register (WDCR)

This register is used to disable and clear of binary counter the watchdog timer function.

- Disable control

  By writting the disable code (B1H) in this WDCR register after clearing WDMOD<WDTE> to "0", the watchdog timer can be disabled.

```
WDMOD ← 0 - - - - - X X     Clear WDMOD<WDTE> to "0".
WDCR  ← 1 0 1 1 0 0 0 1     Write the disable code (B1H).
```

- Enable control

  Set WDMOD<WDTE> to "1".

- Watchdog timer clear control

  The binary counter can be cleared and resume counting by writing clear code (4EH) into the WDCR register.

```
WDCR  ← 0 1 0 0 1 1 1 0     Write the clear code (4EH).
```

WDMOD (005CH)

	7	6	5	4	3	2	1	0
bit Symbol	WDTE	WDTP1	WDTP0	WARM	HALTM0	HALTM0	RESCR	DRVE
Read/Write	R/W							
After reset	1	0	0	0	0	0	0	0
Function	WDT control 1: Enable	Select detecting time 00: 2^{16}/fc 01: 2^{18}/fc 10: 2^{20}/fc 11: 2^{22}/fc		Warming Up time 0: 2^{14}/fc 1: 2^{16}/fc	Standby mode 00: RUN mode 01: STOP mode 10: IDLE mode 11: Don't care		1: Internally connects WDT out to the reset pin	1: Drive the pin even in STOP mode

DRVE (explanation by stop mode)

Watchdog timer out control

0	—
1	Connects WDT out to a reset

Select the standby mode HALT instruction

00	RUN mode (Only the CPU stops)
01	STOP mode (All circuits stop)
10	IDLE mode (Only the oscillator operates)
11	Don't care

Select the detecting period of watchdog timer

0	2^{14}/fc (approx. 1.0ms @16MHz)
1	2^{16}/fc (approx. 4.1ms @16MHz)

Select the detecting time of watchdog timer

00	2^{16}/fc (approx. 4.1ms @16MHz)
01	2^{18}/fc (approx. 16ms @16MHz)
10	2^{20}/fc (approx. 66ms @16MHz)
11	2^{22}/fc (approx. 262ms @16MHz)

Watchdog timer Enable / Disable control

0	Disable
1	Enable

Figure 3.13 (4) Watchdog Timer Mode Register

WDCR (005DH)

	7	6	5	4	3	2	1	0
bit Symbol	–							
Read/Write	W							
After reset	–							
Function	B1H : WDT disable code 4EH : WDT clear code							

→ Disable/clear WDT

B1H	Disable code
4EH	Clear code
Others	—

Figure 3.13 (5) Watchdog Timer Control Register

3.13.3 Operation

The watchdog timer generates interrupt INTWD after the detecting time set in the WDMOD<WDTP1, 0>register and outputs a low level signal. The watchdog timer must be zero-cleared by software before an INTWD interrupt is generated. If the CPU malfunctions (runaway) due to causes such as noise, but does not execute the instruction used to clear the binary counter, the binary counter overflows and an INTWD interrupt is generated. The CPU detects malfunction (runaway) due to the INTWD Interrupt and it is possible to return to normal operation by an anti-mulfunction program. By connecting the watchdog timer out pin to peripheral devices' resets, a CPU malfunction can also be acknowledged to other devices.

The watchdog timer restarts operation immediately after resetting is released.

The watchdog timer stops its operation in the IDLE and STOP modes. In the RUN mode, the watchdog timer is enabled.

However, the function can be disabled when entering the RUN mode.

Example : ① Clear the binary counter

```
WDCR  ← 0 1 0 0 1 1 1 0        Write clear code (4EH).
```

② Set the watchdog timer detecting time to 2^{18}/fc

```
WDMOD ← 1 0 1 - - - X X
```

③ Disable the watchdog timer.

```
WDMOD ← 0 - - - - - X X        Clear WDTE to "0".
WDCR  ← 1 0 1 1 0 0 0 1        Write disable code (B1H).
```

④ Set IDLE mode.

```
WDMOD ← 0 - - - 1 0 X X        Disables WDT and sets IDLE mode.
WDCR  ← 1 0 1 1 0 0 0 1
Executes HALT command          Set the standby mode
```

⑤ Set the STOP mode (warming up time: 2^{16}/fc)

```
WDMOD ← - - - 1 0 1 X X        Set the STOP mode.
Executes HALT command.         Execute HALT instruction. Set the standby
                               mode.
```

4. ELECTRICAL CHARACTERISTICS

4.1 Absolute Maximum (TMP96C141F-16)

Symbol	Parameter	Rating	Unit
V cc	Power Supply voltage	− 0.5~6.5	V
V IN	Input voltage	− 0.5~Vcc + 0.5	V
Σ IOL	Output Current (total)	100	mA
Σ IOH	Output Current (total)	− 100	mA
P D	Power Dissipation (Ta = 70℃)	600	mW
T SOLDER	Soldering Temperature (10sec)	260	℃
T STG	Storage temperature	− 65~150	℃
T OPR	Operating temperature	− 20~70	℃

4.2 DC Characteristics (TMP96C141F-16)

Vcc = 5V ± 10%, Ta = − 20~70℃ (Typical values are for Ta = 25℃ and Vcc = 5V.)

Symbol	Parameter	Min	Max	Unit	Test Condition
V IL V IL1 V IL2 V IL3 V IL4	Input Low Voltage (AD0 − 15) P2, P3, P4, P5, P6, P7, P8, P9 RESET,NMI,INT0(P87) EA X1	−0.3 −0.3 −0.3 −0.3 −0.3	0.8 0.3Vcc 0.25Vcc 0.3 0.2Vcc	V V V V V	
V IH V IH1 V IH2 V IH3 V IH4	Input High Voltage (AD0 − 15) P2, P3, P4, P5, P6, P7, P8, P9 RESET, NMI, INT0 (P87) EA X1	2.2 0.7Vcc 0.75Vcc Vcc − 0.3 0.8Vcc	Vcc + 0.3 Vcc + 0.3 Vcc + 0.3 Vcc + 0.3 Vcc + 0.3	V V V V V	
V OL	Output Low Voltage		0.45	V	I OL = 1.6mA
V OH V OH1 V OH2	Output High Voltage	2.4 0.75Vcc 0.9Vcc		V V V	I OH = − 400μA I OH = − 100μA I OH = − 20μA
I DAR	Darlington Drive Current (8 Output Pins max.)	− 1.0	− 3.5	mA	V EXT = 1.5V R EXT = 1.1KΩ
I LI I LO	Input Leakage Current Output Leakage Current	0.02 (Typ) 0.05 (Typ)	± 5 ± 10	μA μA	0.0 ≦ Vin ≦ Vcc 0.2 ≦ Vin ≦ Vcc − 0.2
I cc	Operating Current (RUN) IDLE STOP (Ta = − 20~70℃) STOP (Ta = 0~50℃)	26 (Typ) 1.7 (Typ) 0.2 (Typ)	50 10 50 10	mA mA μA μA	f osc = 16MHz 0.2 ≦ Vin ≦ Vcc − 0.2 0.2 ≦ Vin ≦ Vcc − 0.2
V STOP	Power Down Voltage (@STOP, RAM Back up)	2.0	6.0	V	V IL2 = 0.2Vcc, V IH2 = 0.8Vcc
R RST	RESET Pull Up Registor	50	150	KΩ	
C IO	Pin Capacitance		10	pF	tosc = 1MHz
V TH	Schmitt Width RESET, NMI, INT0 (P87)	0.4	1.0 (Typ)	V	
R K	Pull Down/Up Registor	50	150	KΩ	

Note : I-DAR is guaranteed for a total of up to 8 ports.

4.3 AC Electrical Characteristics (TMP96C141F-16)

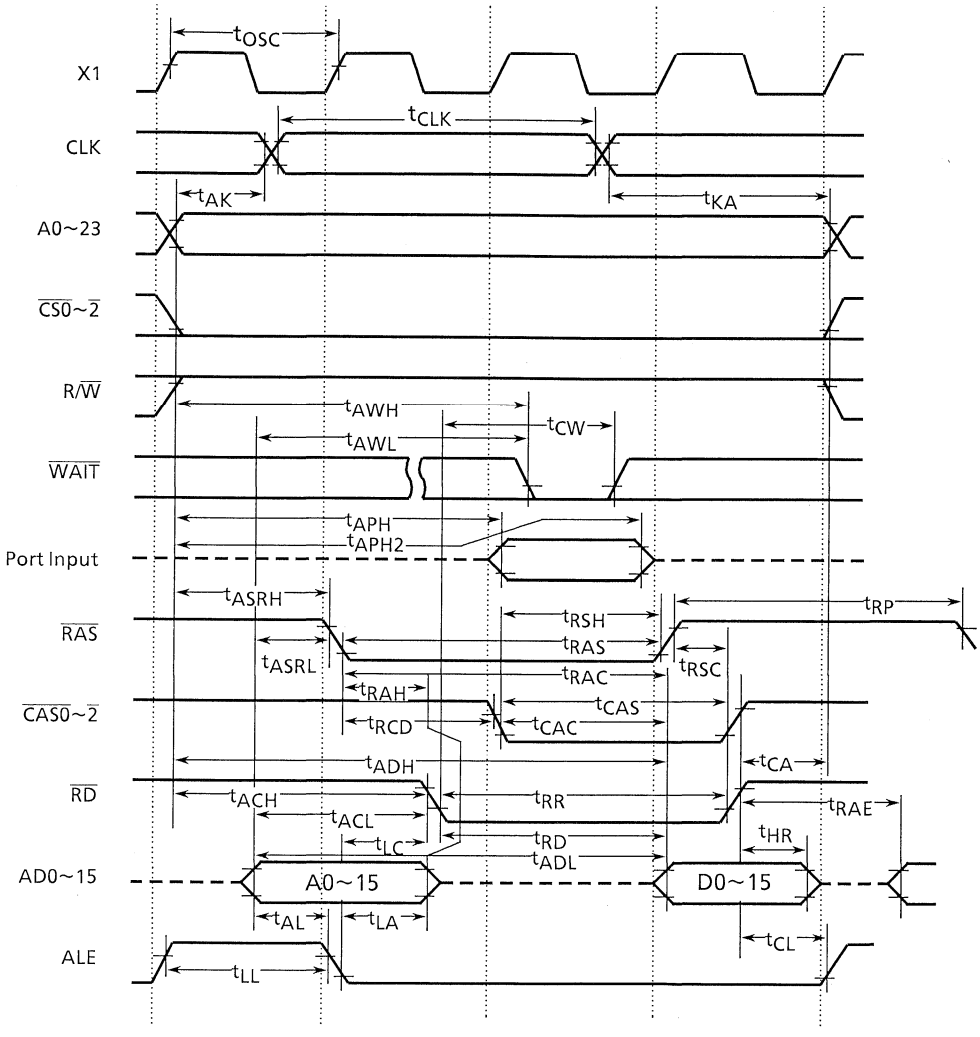
Vcc = 5V ± 10% , TA = −20~70℃
(4MHz~16MHz)

No.	Symbol	Paramerer	Voriable		12.5MHz		16MHz		Unit
			Min	Max	Min	Max	Min	Max	
1	t_{OSC}	Osc. Period (= x)	62.5	250	80		62.5		ns
2	t_{CLK}	CLK width	2x − 40		120		85		ns
3	t_{AK}	A0-23 Valid→ CLK Hold	0.5x − 20		20		11		ns
4	t_{KA}	CLK Valid→ A0-23 Hold	1.5x − 70		50		24		ns
5	t_{AL}	A0-15 Valid→ ALE fall	0.5x − 15		25		16		ns
6	t_{LA}	ALE fall→ A0-15 Hold	0.5x − 15		25		16		ns
7	t_{LL}	ALE High width	x − 40		40		23		ns
8	t_{LC}	ALE fall→ $\overline{RD}/\overline{WR}$ fall	0.5x − 30		10		1		ns
9	t_{CL}	$\overline{RD}/\overline{WR}$ rise→ ALE rise	0.5x − 20		20		11		ns
10	t_{ACL}	A0-15 Valid→ $\overline{RD}/\overline{WR}$ fall	x − 25		55		38		ns
11	t_{ACH}	A0-23 Valid→ $\overline{RD}/\overline{WR}$ fall	1.5x − 50		70		44		ns
12	t_{CA}	$\overline{RD}/\overline{WR}$ rise→ A0-23 Hold	0.5x − 20		20		11		ns
13	t_{ADL}	A0-15 Valid→ D0-15 input		3.0x − 45		195		143	ns
14	t_{ADH}	A0-23 Valid→ D0-15 input		3.5x − 65		215		154	ns
15	t_{RD}	$\overline{RD}$ fall → D0-15 input		2.0x − 50		110		75	ns
16	t_{RR}	$\overline{RD}$ Low width	2.0x − 40		120		85		ns
17	t_{HR}	$\overline{RD}$ rise→ D0-15 Hold	0		0		0		ns
18	t_{RAE}	$\overline{RD}$ rise→ A0-15output	x − 15		65		48		ns
19	t_{WW}	$\overline{WR}$ Low width	2.0x − 40		120		85		ns
20	t_{DW}	D0-15 Valid→ $\overline{WR}$ rise	2.0x − 50		110		75		ns
21	t_{WD}	$\overline{WR}$ rise →D0-15 Hold	0.5x − 10		30		21		ns
22	t_{AEH}	A0-23 Valid→ $\overline{WAIT}$ input (1WAIT + n mode)		3.5x − 90		190		129	ns
23	t_{AWL}	A0-15 Valid→ $\overline{WAIT}$ input (1WAIT + n mode)		3.0x − 80		160		108	ns
24	t_{CW}	$\overline{RD}/\overline{WR}$ fall→ $\overline{WAIT}$ Hold (1WAIT + n mode)	2.0x + 0		160		125		ns
25	t_{APH}	A0-23 Valid→ PORT input		2.5x − 120		80		36	ns
26	t_{APH2}	A0-23 Valid→ PORT Hold	2.5x + 50		250		206		ns
27	t_{CP}	$\overline{WR}$ rise→ PORT Valid		200		200		200	ns
28	t_{ASRH}	A0-23 Valid→ $\overline{RAS}$ fall	1.0x − 40		40		23		ns
29	t_{ASRL}	A0-15 Valid→ $\overline{RAS}$ fall	0.5x − 15		25		16		ns
30	t_{RAC}	$\overline{RAS}$ fall→ D0-15 input		2.5x − 70		130		86	ns
31	t_{RAH}	$\overline{RAS}$ fall→ A0-15 Hold	0.5x − 15		25		16		ns
32	t_{RAS}	$\overline{RAS}$ Low width	2.0x − 40		120		85		ns
33	t_{RP}	$\overline{RAS}$ High width	2.0x − 40		120		85		ns
34	t_{RSH}	$\overline{CAS}$ fall→ $\overline{RAS}$ rise	1.0x − 35		45		28		ns
35	t_{RSC}	$\overline{RAS}$ rise→ $\overline{CAS}$ rise	0.5x − 25		15		6		ns
36	t_{RCD}	$\overline{RAS}$ fall→ $\overline{CAS}$ fall	1.0x − 40		40		23		ns
37	t_{CAC}	$\overline{CAS}$ fall→ D0-15 input		1.5x − 65		55		29	ns
38	t_{CAS}	$\overline{CAS}$ Low width	1.5x − 30		90		64		ns

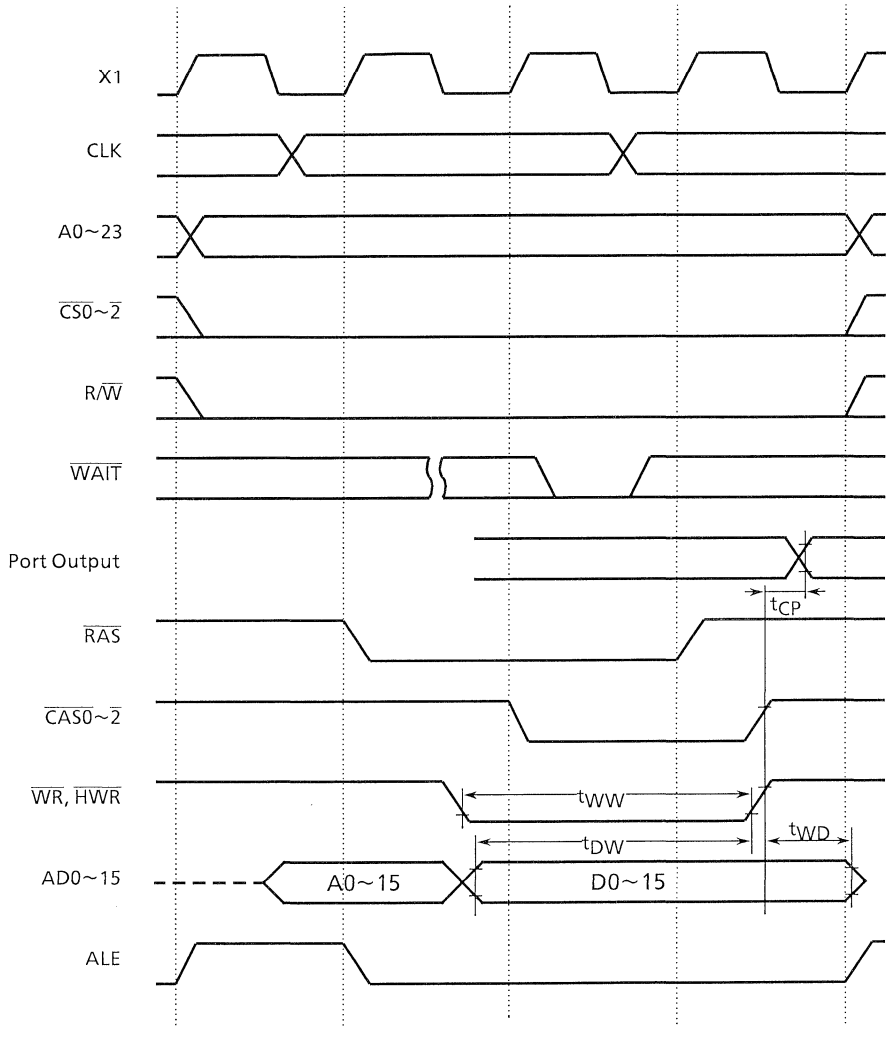
AC Measuring Conditions

- Output Level : High 2.2V /Low 0.8V , CL50pF
  (However CL = 100pF for AD0~AD15, AD0~AD23, ALE, $\overline{RD}$, $\overline{WR}$, $\overline{HWR}$, R/$\overline{W}$, CLK, $\overline{RAS}$, $\overline{CAS0}$~$\overline{CAS2}$)
- Input Level : High 2.4V /Low 0.45V (AD0~AD15)
  High 0.8Vcc /Low 0.2Vcc (Except for AD0~AD15)

(1) Read Cycle


tOSC
X1
tCLK
CLK
tAK
tKA
A0~23
CS0~2
R/W
tAWH
tCW
tAWL
WAIT
tAPH
tAPH2
Port Input
tASRH
tRP
tRSH
RAS
tRAS
tASRL
tRSC
tRAC
tRAH
tCAS
CAS0~2
tRCD
tCAC
tADH
tCA
RD
tACH
tRR
tRAE
tACL
tLC
tRD
tHR
tADL
AD0~15
A0~15
D0~15
tAL
tLA
tCL
ALE
tLL

(2) Write Cycle



4.4 A/D Conversion Characteristics (TMP96C141F-16)

Vcc = 5V ± 10% TA = − 20~70 °C

Symbol	Parameter	Min	Typ	Max	Unit
V_{REF}	Analog reference voltage	Vcc − 1.5	Vcc	Vcc	V
A_{GND}	Analog reference voltage	Vss	Vss	Vss	
V_{AIN}	Analog input voltage range	Vss		Vcc	
I_{REF}	Anlog current for analog reference voltage		0.5	1.5	mA
Error(Quantize error of ± 0.5 LSB not included)	Total error (TA = 25 °C, Vcc = VREF = 5.0V)			± 4.0	LSB
	Total error			± 6.0	

4.5 Serial Channel Timing − I/O Interface Mode

(1) SCLK Input Mode

Vcc = 5V ± 10% TA = − 20~70 °C

Symbol	Parameter	Variable		12.5MHz		16MHz		Unit
		Min	Max	Min	Max	Min	Max	
t_{SCY}	SCLK cycle	16X		1.28		1		μs
t_{OSS}	Output Data → Rising edge of SCLK	t_{SCY}/2 − 5X − 50		190		137		ns
t_{OHS}	SCLK rising edge→ Output Data hold	5X − 100		300		212		ns
t_{HSR}	SCLK rising edge→ Input Data hold	0		0		0		ns
t_{SRD}	SCLK rising edge→ effective data input		t_{SCY} − 5X − 100		780		587	ns

(2) SCLK Output Mode

Symbol	Parameter	Variable		12.5MHz		16MHz		Unit
		Min	Max	Min	Max	Min	Max	
t_{SCY}	SCLK cycle (programmable)	16X	8192X	1.28	655.4	1	512	μs
t_{OSS}	Output Data → SCLK rising edge	t_{SCY} − 2X − 150		970		725		ns
t_{OHS}	SCLK rising edge→ Output Data hold	2X − 80		80		45		ns
t_{HSR}	SCLK rising edge→ Input Data hold	0		0		0		ns
t_{SRD}	SCLK rising edge→ effective data input		t_{SCY} − 2X − 150		970		725	ns

4.6 Timer/Counter Input Clock (TI0, TI4, TI5, TI6, TI7)

Vcc = 5V ± 10% TA = − 20~70 °C

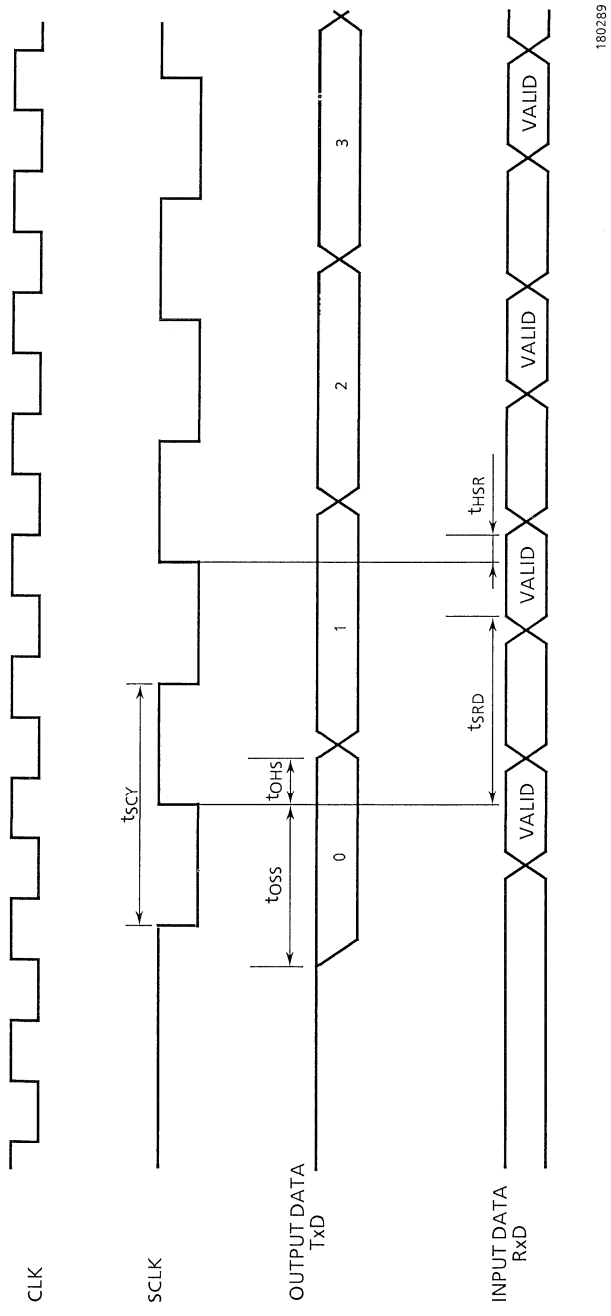
Symbol	Parameter	Variable		12.5MHz		16MHz		Unit
		Min	Max	Min	Max	Min	Max	
t_{VCK}	Clock Cycle	8X + 100		740		600		ns
t_{VCKL}	Low level clock Pulse width	4X + 40		360		290		ns
t_{VCKH}	High level clock Pulse width	4X + 40		360		290		ns

4.7 Interrupt Operation

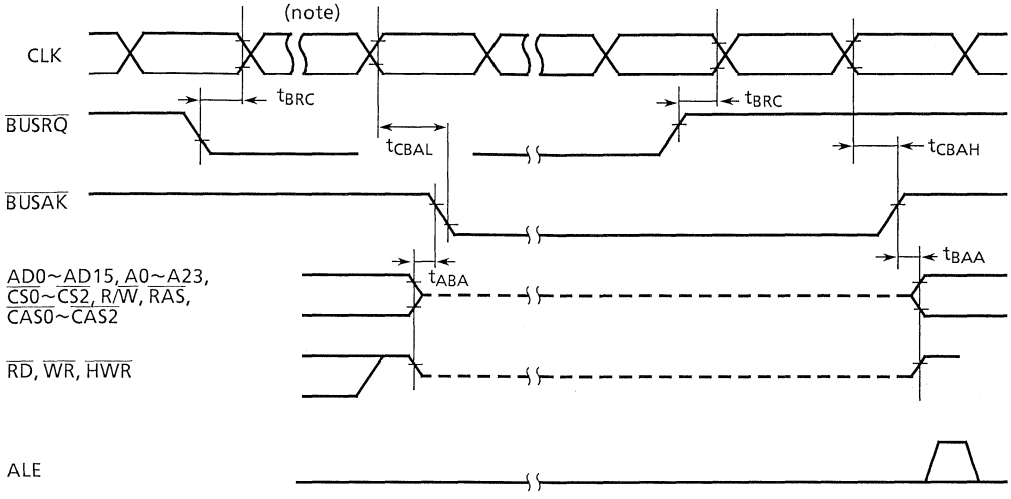
Vcc = 5V ± 10% TA = − 20~70 °C

Symbol	Parameter	Variable		12.5MHz		16MHz		Unit
		Min	Max	Min	Max	Min	Max	
t_{INTAL}	$\overline{NMI}$, INT0 Low level Pulse width	4X		320		250		ns
t_{INTAH}	$\overline{NMI}$, INT0 High level Pulse width	4X		320		250		ns
t_{INTBL}	INT4~INT7 Low level Pulse width	8X + 100		740		600		ns
t_{INTBH}	INT4~INT7 High level Pulse width	8X + 100		740		600		ns

4.8 Timing Chart for I/O Interface Mode



4.9 Timing Chart for Bus Request ($\overline{BUSRQ}$) / BUS Acknowledge ($\overline{BUSAK}$)



Symbol	Parameter	Variable		12.5MHz		16MHz		Unit
		Min	Max	Min	Max	Min	Max	
t_{BRC}	$\overline{BUSRQ}$ set-up time for CLK	120		120		120		ns
t_{CBAL}	CLK→ $\overline{BUSAK}$ falling edge		1.5x + 120		240		214	ns
t_{CBAH}	CLK→ $\overline{BUSAK}$ rising edge		0.5x + 40		80		71	ns
t_{ABA}	Floating time to $\overline{BUSAK}$ fall	0	80	0	80	0	80	ns
t_{BAA}	Floating time to $\overline{BUSAK}$ rise	0	80	0	80	0	80	ns

Note: The Bus will be released after the $\overline{WAIT}$ request is inactive, when the $\overline{BUSRQ}$ is set to "0" during "Wait" cycle.

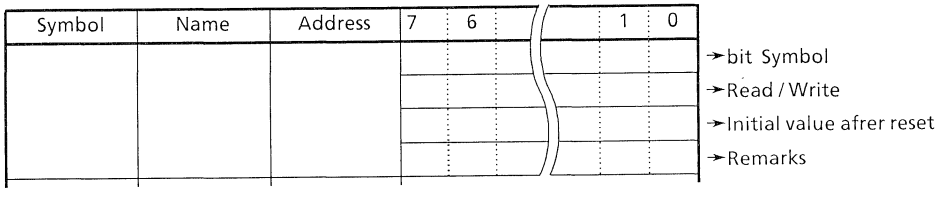
5. TABLE OF SPECIAL FUNCTION REGISTERS (SFRs)

(SFR ; Special Function Register)

The special function registers (SFRs) include the I/O ports and peripheral control registers allocated to the 128-byte addresses from 000000H to 00007FH.

(1) I/O port
(2) I/O port control
(3) Timer control
(4) Pattern Generator control
(5) Watch Dog Timer control
(6) Serial Channel control
(7) A / D converter control
(8) Interrupt control
(9) Chip Select / Wait control

Configuration of the table



020289

Table5 I/O register address map

ADDRESS	NAME	ADDRESS	NAME	ADDRESS	NAME	ADDRESS	NAME
000000H	P0	20H	TRUN	40H	TREG6L	60H	ADREG0L
1H	P1	21H		41H	TREG6H	61H	ADREG0H
2H	P0CR	22H	TREG0	42H	TREG7L	62H	ADREG1L
3H		23H	TREG1	43H	TREG7H	63H	ADREG1H
4H	P1CR	24H	TMOD	44H	CAP3L	64H	ADREG2L
5H	P1FC	25H	TFFCR	45H	CAP3H	65H	ADREG2H
6H	P2	26H	TREG2	46H	CAP4L	66H	ADREG3L
7H	P3	27H	TREG3	47H	CAP4H	67H	ADREG3H
8H	P2CR	28H	P0MOD	48H	T5MOD	68H	B0CS
9H	P2FC	29H	P1MOD	49H	T5FFCR	69H	B1CS
AH	P3CR	2AH	PFFCR	4AH		6AH	B2CS
BH	P3FC	2BH		4BH		6BH	
CH	P4	2CH		4CH	PG0REG	6CH	
DH	P5	2DH		4DH	PG1REG	6DH	
EH	P4CR	2EH		4EH	PG01CR	6EH	
FH		2FH		4FH		6FH	
10H	P4FC	30H	TREG4L	50H	SC0BUF	70H	INTE0AD
11H		31H	TREG4H	51H	SC0CR	71H	INTE45
12H	P6	32H	TREG5L	52H	SC0MOD	72H	INTE67
13H	P7	33H	TREG5H	53H	BR0CR	73H	INTET10
14H	P6CR	34H	CAP1L	54H	SC1BUF	74H	INTEPW10
15H	P7CR	35H	CAP1H	55H	SC1CR	75H	INTET54
16H	P6FC	36H	CAP2L	56H	SC1MOD	76H	INTET76
17H	P7FC	37H	CAP2H	57H	BR1CR	77H	INTES0
18H	P8	38H	T4MOD	58H	ODE	78H	INTES1
19H	P9	39H	TFF4CR	59H		79H	
1AH	P8CR	3AH	T45CR	5AH		7AH	
1BH	P9CR	3BH		5BH		7BH	IIMC
1CH	P8FC	3CH		5CH	WDMOD	7CH	DMA0V
1DH	P9FC	3DH		5DH	WDCR	7DH	DMA1V
1EH		3EH		5EH	ADMOD	7EH	DMA2V
1FH		3FH		5FH		7FH	DMA3V

(1) I/O Port

Symbol	Name	Address	7	6	5	4	3	2	1	0
P0	PORT0	00H	P07	P06	P05	P04	P03	P02	P01	P00
			R/W							
			Input mode							
			Undefined							
P1	PORT1	01H	P17	P16	P15	P14	P13	P12	P11	P10
			R/W							
			Input mode							
			0	0	0	0	0	0	0	0
P2	PORT2	06H	P27	P26	P25	P24	P23	P22	P21	P20
			R/W							
			Input mode							
			0	0	0	0	0	0	0	0
P3	PORT3	07H	P37	P36	P35	P34	P33	P32	P31	P30
			R/W							
			Input mode						Output mode	
			1	1	1	1	1	1	1	1
P4	PORT4	0CH						P42	P41	P40
								R/W		
								Input mode		
								0	1	1
P5	PORT5	0DH					P53	P52	P51	P50
							R			
							Input mode			
P6	PORT6	12H	P67	P66	P65	P64	P63	P62	P61	P60
			R/W							
			Input mode							
			1	1	1	1	1	1	1	1
P7	PORT7	13H					P73	P72	P71	P70
							R/W			
							Input mode			
							1	1	1	1
P8	PORT8	18H	P87	P86	P85	P84	P83	P82	P81	P80
			R/W							
			Input mode							
			1	1	1	1	1	1	1	1
P9	PORT9	19H			P95	P94	P93	P92	P91	P90
					R/W					
					Input mode					
					1	1	1	1	1	1

Note : When P30 pin is defined as $\overline{RD}$ signal output mode (P30F = 1), clearing the output latch register P30 to "0" outputs the $\overline{RD}$ strobe from P30 pin for PSRAM, even when the internal address is accessed. If the output latch register P30 remains "1", the $\overline{RD}$ strobe is output only when the external address is accessed.

Read/Write

R/W ; Either read or write is possible
R ; Only read is possible
W ; Only write is possible
Prohibit RMW ; Prohibit Read Modify Write. (Prohibit RES/SET/TSET/CHG/STCF/ANDCF/ORCF/XORCF Instruction)

(2) I/O Port Control (1/2)

Symbol	Name	Address	7	6	5	4	3	2	1	0
P0CR	PORT0 Control	02H (Prohibit RMW)	P07C	P06C	P05C	P04C	P03C	P02C	P01C	P00C
			W							
			0	0	0	0	0	0	0	0
			0 : IN 1 : OUT (When external access, set as AD7-0 and cleared to "0".)							
P1CR	PORT1 Control	04H (Prohibit RMW)	P17C	P16C	P15C	P14C	P13C	P12C	P11C	P10C
			W							
			0	0	0	0	0	0	0	0
			<<Refer to the "P1FC">>							
P1FC	PORT1 Function	05H (Prohibit RMW)	P17F	P16F	P15F	P14F	P13F	P12F	P11F	P10F
			W							
			0	0	0	0	0	0	0	0
			P1FC/P1CR = 00 : IN, 01 : OUT, 10 : AD15-8, 11 : A15-8							
P2CR	PORT2 Control	08H (Prohibit RMW)	P27C	P26C	P25C	P24C	P23C	P22C	P21C	P20C
			W							
			0	0	0	0	0	0	0	0
			<<Refer to the "P2FC">>							
P2FC	PORT2 Function	09H (Prohibit RMW)	P27F	P26F	P25F	P24F	P23F	P22F	P21F	P20F
			W							
			0	0	0	0	0	0	0	0
			P2FC/P2CR = 00 : IN, 01 : OUT, 10 : A7-0, 11 : A23-16							
P3CR	PORT3 Control	0AH (Prohibit RMW)	P37C	P36C	P35C	P34C	P33C	P32C		
			W							
			0	0	0	0	0	0		
			0 : IN			1 : OUT				
P3FC	PORT3 Function	0BH (Prohibit RMW)	P37F	P36F	P35F	P34F		P32F	P31F	P30F
			W							
			0	0	0	0		0	0	0
			0 : PORT 1 : $\overline{RAS}$	0 : PORT 1 : R/$\overline{W}$	0 : PORT 1 : $\overline{BUSAK}$	0 : PORT 1 : $\overline{BUSRQ}$		0 : PORT 1 : $\overline{HWR}$	0 : PORT 1 : $\overline{WR}$	0 : PORT 1 : $\overline{RD}$
P4CR	PORT4 Control	0EH (Prohibit RMW)						P42C	P41C	P40C
									W	
								0	0	0
								0 : IN	1 : OUT	
P4FC	PORT4 Function	10H (Prohibit RMW)						P42F	P41F	P40F
									W	
								0	0	0
								0 : PORT	1 : $\overline{CS}$/$\overline{CAS}$	

Note : With the TMP96C141, which requires an external ROM, PORT0 functions as AD0 to AD7; PORT1, AD8 to AD15; P30, the $\overline{RD}$ signal; P31, the $\overline{WR}$ signal, regardless of the values set in P0CR, P1CR, P1FC, P30F and P31F.

I/O Port Control (2/2)

Symbol	Name	Address	7	6	5	4	3	2	1	0
P6CR	PORT6 Control	14H (Prohibit RMW)	P67C	P66C	P65C	P64C	P63C	P62C	P61C	P60C
			W							
			0	0	0	0	0	0	0	0
			0 : IN				1 : OUT			
P7CR	PORT7 Control	15H (Prohibit RMW)					P73C	P72C	P71C	P70C
							W			
							0	0	0	0
							0 : IN		1 : OUT	
P6FC	PORT6 Function	16H (Prohibit RMW)	P67F	P66F	P65F	P64F	P63F	P62F	P61F	P60F
			W							
			0	0	0	0	0	0	0	0
			0 : PORT		1 : PG1-OUT		0 : PORT		1 : PG0-OUT	
P7FC	PORT7 Function	17H (Prohibit RMW)					P73F	P72F	P71F	
							W			
							0	0	0	
							0 : PORT 1 : TO3	0 : PORT 1 : TO2	0 : PORT 1 : TO1	
P8CR	PORT8 Control	1AH (Prohibit RMW)	P87C	P86C	P85C	P84C	P83C	P82C	P81C	P80C
			W							
			0	0	0	0	0	0	0	0
					0 : IN		1 : OUT			
P9CR	PORT9 Control	1BH (Prohibit RMW)			P95C	P94C	P93C	P92C	P91C	P90C
					W					
					0	0	0	0	0	0
					0 : IN		1 : OUT			
P8FC	PORT8 Function	1CH (Prohibit RMW)		P86F			P83F	P82F		
				W			W	W		
				0			0	0		
				0 : PORT 1 : TO6			0 : PORT 1 : TO5	0 : PORT 1 : TO4		
P9FC	PORT9 Function	1DH (Prohibit RMW)			P95F		P93F	P92F		P90F
					W		W	W		W
					0		0	0		0
					0 : PORT 1 : SCLK1		0 : PORT 1 : TxD1	0 : PORT 1 : SCLK0		0 : PORT 1 : TxD0

Note : The register P92F is used only in the TMP96CM40 and TMP96PM40, and not available in the TMP96C141. (It cannot be defined as SCLK0).

(3) Timer Control (1/3)

Symbol	Name	Address	7	6	5	4	3	2	1	0
TRUN	Timer Control	20H	PRRUN		T5RUN	T4RUN	P1RUN	P0RUN	T1RUN	T0RUN
			R/W		R/W					
			0		0	0	0	0	0	0
			Prescaler & Timer Run/Stop CONTROL 0 : Stop & Clear 1 : Run (Count up)							
TREG0	8bit Timer Register 0	22H (Prohibit RMW)	–							
			W							
			Undefined							
TREG1	8bit Timer Register 1	23H (Prohibit RMW)	–							
			W							
			Undefined							
TMOD	8bit Timer Source CLK & MODE	24H (Prohibit RMW)	T10M1	T10M0	PWMM1	PWMM0	T1CLK1	T1CLK0	T0CLK1	T0CLK0
			W							
			0	0	0	0	0	0	0	0
			00 : 8bit Timer 01 : 16bit Timer 10 : 8bit PPG 11 : 8bit PWM		00 : – 01 : 2^6-1 10 : 2^7-1 11 : 2^8-1	PWM	00 : TO0TRG 01 : ϕT1 10 : ϕT16 11 : ϕT256		00 : TI0 Input 01 : ϕT1 10 : ϕT4 11 : ϕT16	
TFFCR	8bit Timer Flip-Flop Control	25H				DBEN	TFF1C1	TFF1C0	TFF1IE	TFF1IS
						R/W	W		R/W	
						0	0	0	0	0
						1 : Double Buffer Enable	00 : Invert TFF1 01 : Set TFF1 10 : Clear TFF1 11 : Don't care		1 : TFF1 Invert Enable	0 : Inverted by Timer 0
TREG2	PWM Timer Register 2	26H	–							
			(R)/W (Can read double buffer values.)							
			Undefined							
TREG3	PWM Timer Register 3	27H	–							
			(R)/W (Can read double buffer values.)							
			Undefined							
P0MOD	PWM0 Mode	28H (Prohibit RMW)	FF2RD	DB2EN	PWM0INT	PWM0M	T2CLK1	T2CLK0	PWM0S1	PWM0S0
			R		W					
			–	0	0	0	0	0	0	0
			TFF2 output value	1 : Double Buffer Enable	0 : Overflow interrupt 1 : Compare/match interrupt	0 : PWM Mode 1 : Timer Mode	00 : ϕP1 (fc/4) 01 : ϕP4 (fc/16) 10 : ϕP16 (fc/64) 11 : Don't care		00 : 2^6-1 01 : 2^7-1 10 : 2^8-1 11 : Don't care	
P1MOD	PWM1 Mode	29H (Prohibit RMW)	FF3RD	DB3EN	PWM1INT	PWM1M	T3CLK1	T3CLK0	PWM1S1	PWM1S0
			R		W					
			–	0	0	0	0	0	0	0
			TFF3 output value	1 : Double Buffer Enable	0 : Overflow interrupt 1 : Compare/match interrupt	0 : PWM Mode 1 : Timer Mode	00 : ϕP1 (fc/4) 01 : ϕP4 (fc/16) 10 : ϕP16 (fc/64) 11 : Don't care		00 : 2^6-1 01 : 2^7-1 10 : 2^8-1 11 : Don't care	

Timer Control (2/3)

Symbol	Name	Address	7	6	5	4	3	2	1	0
PFFCR	PWM Flip-Flop Control	2AH	FF3C1	FF3C0	FF3TRG1	FF3TRG0	FF2C1	FF2C0	FF2TRG1	FF2TRG0
			W		R/W		W		R/W	
			0	0	0	0	0	0	0	0
			00 : Don't care 01 : Set TFF3 10 : Clear TFF3 11 : Don't care		00 : Prohibit TFF3 inverted 01 : Invert if matched 10 : Set if matched; clear if overflowed 11 : Clear if matched; set if overflowed		00 : Don't care 01 : Set TFF2 10 : Clear TFF2 11 : Don't care		00 : Prohibit TFF2 inverted 01 : Invert if matched 10 : Set if matched; clear if overflowed 11 : Clear if matched; set if overflowed	
TREG4L	16bit Timer Register4L	30H (Prohibit RMW)	–							
			W							
			Undefined							
TREG4H	16bit Timer Register4H	31H (Prohibit RMW)	–							
			W							
			Undefined							
TREG5L	16bit Timer Register5L	32H (Prohibit RMW)	–							
			W							
			Undefined							
TREG5H	16bit Timer Register5H	33H (Prohibit RMW)	–							
			W							
			Undefined							
CAP1L	Capture Register1L	34H	–							
			R							
			Undefined							
CAP1H	Capture Register1H	35H	–							
			R							
			Undefined							
CAP2L	Capture Register2L	36H	–							
			R							
			Undefined							
CAP2H	Capture Register2H	37H	–							
			R							
			Undefined							
T4MOD	16bit Timer 4 Source CLK & MODE	38H	CAP2T5	EQ5T5	CAP1IN	CAP12M1	CAP12M0	CLE	T4CLK1	T4CLK0
			R/W		W			R/W		
			0	0	0	0	0	0	0	0
			TFF5 INV TRG 0 : TRG Disable 1 : TRG Enable		0 : Soft-Capture 1 : Don't care		Capture Timming 00 : Disable 01 : TI4 ↑ TI5 ↑ 10 : TI4 ↑ TI4 ↓ 11 : TFF1 ↑ TFF1 ↓	1 : UC4 Clear Enable	Source Clock 00 : TI4 01 : ϕT1 10 : ϕT4 11 : ϕT16	
T4FFCR	16bit Timer 4 Flip-Flop Control	39H	TFF5C1	TFF5C0	CAP2T4	CAP1T4	EQ5T4	EQ4T4	TFF4C1	TFF4C0
			W		R/W				W	
			0	0	0	0	0	0	0	0
			00 : Invert TFF5 01 : Set TFF5 10 : Clear TFF5 11 : Don't care		TFF4 Invert Trigger 0 : Trigger Disable 1 : Trigger Enable				00 : Invert TFF4 01 : Set TFF4 10 : Clear TFF4 11 : Don't care	

Timer Control (3/3)

Symbol	Name	Address	7	6	5	4	3	2	1	0
T45CR	T4, T5 Control	3AH	–				PG1T	PG0T	DB6EN	DB4EN
			R/W				R/W			
			0				0	0	0	0
			Fix at "0"				PG1 shift trigger 0 : Timer 0, 1 1 : Timer 5	PG0 shift trigger 0 : Timer 0, 1 1 : Timer 4	1 : Double Buffer Enable	
TREG6L	16bit Timer Register6L	40H (Prohibit RMW)	–							
			W							
			Undefined							
TREG6H	16bit Timer Register6H	41H (Prohibit RMW)	–							
			W							
			Undefined							
TREG7L	16bit Timer Register7L	42H (Prohibit RMW)	–							
			W							
			Undefined							
TREG7H	16bit Timer Register7H	43H (Prohibit RMW)	–							
			W							
			Undefined							
CAP3L	Capture Register3L	44H	–							
			R							
			Undefined							
CAP3H	Capture Register3H	45H	–							
			R							
			Undefined							
CAP4L	Capture Register4L	46H	–							
			R							
			Undefined							
CAP4H	Capture Register4H	47H	–							
			R							
			Undefined							
T5MOD	16bit Timer 5 Source CLK & MODE	48H			CAP3IN	CAP34M1	CAP34M0	CLE	T5CLK1	T5CLK0
					W	R/W				
					0	0	0	0	0	0
					0 : Soft-Capture 1 : Don't care	Capture Timming 00 : Disable 01 : TI6 ↑ TI7 ↑ 10 : TI6 ↑ TI6 ↓ 11 : TFF1 ↑ TFF1 ↓		1 : UC5 Clear Enable	Source Clock 00 : TI6 01 : ϕT1 10 : ϕT4 11 : ϕT16	
T5FFCR	16bit Timer 5 Flip-Flop Control	49H			CAP4T6	CAP3T6	EQ7T6	EQ6T6	TFF6C1	TFF6C0
					R/W				W	
					0	0	0	0	0	0
					TFF6 Invert Trigger 0 : Trigger Disable 1 : Trigger Enable				00 : Invert TFF6 01 : Set TFF6 10 : Clear TFF6 11 : Don't care	

(4) Pattern Generator

Symbol	Name	Address	7	6	5	4	3	2	1	0
PG0REG	PG0 Register	4CH (Prohibit RMW)	PG03	PG02	PG01	PG00	SA03	SA02	SA01	SA00
			W				R/W			
			0	0	0	0	Undefined			
PG1REG	PG1 Register	4DH (Prohibit RMW)	PG13	PG12	PG11	PG10	SA13	SA12	SA11	SA10
			W				R/W			
			0	0	0	0	Undefined			
PG01CR	PG0, 1 Contorol	4EH	PAT1	CCW1	PG1M	PG1TE	PAT0	CCW0	PG0M	PG0TE
			R/W							
			0	0	0	0	0	0	0	0
			0: 8bit write 1: 4bit write	0: Normal Rotation 1: Reverse Rotation	0: 4bit Step 1: 8bit Step	PG1 trigger input enable 1: Enable	0: 8bit write 1: 4bit write	0: Normal Rotation 1: Reverse Rotation	0: 4bit Step 1: 8bit Step	PG0 trigger input enable 1: Enable

(5) Watch Dog Timer

Symbol	Name	Address	7	6	5	4	3	2	1	0
WD-MOD	Watch Dog Timer Mode	5CH	WDTE	WDTP1	WDTP0	WARM	HALTM1	HALTM0	RESCR	DRVE
			R/W							
			1	0	0	0	0	0	0	0
			1: WDT Enable	00: 2^{16}/fc 01: 2^{18}/fc 10: 2^{20}/fc 11: 2^{22}/fc		Warming up Time 0: 2^{14}/fc 1: 2^{16}/fc	Standby Mode 00: RUN Mode 01: STOP Mode 10: IDLE Mode 11: Don't care		1: Connect internally WDT out pin to Reset Pin	1: Drive the pin in STOP mode
WDCR	Watch Dog Timer Control Register	5DH	–							
			W							
			–							
			B1H: WDT Disable Code 4EH: WDT Clear Code							

(6) Serial Channel

Symbol	Name	Address	7	6	5	4	3	2	1	0
SC0BUF	Serial Channel 0 Buffer	50H	RB7 TB7	RB6 TB6	RB5 TB5	RB4 TB4	RB3 TB3	RB2 TB2	RB1 RB1	RB0 TB0
			R (Receiving)/W (Transmission)							
			Undefined							
SC0CR	Serial Channel 0 Control	51H	RB8	EVEN	PE	OERR	PERR	FERR	–	–
			R	R/W		R (Cleared to 0 by reading)			R/W	
				0	0	0	0	0	0	0
			Receiving data bit 8	Parity 0: Odd 1: Even	1: Parity Enable	1: Error Overrun	1: Error Parity	1: Error Framing	Fix at "0"	Fix at "0"
SC0-MOD	Serial Channel 0 Mode	52H	TB8	CTSE	RXE	WU	SM1	SM0	SC1	SC0
			R/W							
			0	0	0	0	0	0	0	0
			Transmission data bit 8	1: CTS Enable	1: Receive Enable	1: Wake up Enable	00: Unused 01: UART 7bit 10: UART 8bit 11: UART 9bit		00: TO0 Trigger 01: Baud rate generator 10: Internal clock ϕ1 11: Don't care	
BR0CR	Baud Rate Control	53H	–		BR0CK1	BR0CK0	BR0S3	BR0S2	BR0S1	BR0S0
			R/W		R/W					
			0		0	0	0	0	0	0
			Fix at "0"		00: ϕT0 (fc/4) 01: ϕT2 (fc/16) 10: ϕT8 (fc/64) 11: ϕT32 (fc/256)		Set frequency divisor 0~F ("1" prohibited)			
SC1BUF	Serial Channel 1 Buffen	54H	RB7 TB7	RB6 TB6	RB5 TB5	RB4 TB4	RB3 TB3	RB2 TB2	RB1 RB1	RB0 TB0
			R (Receiving) /W (Transmission)							
			Undefined							
SC1CR	Serial Channel 1 Control	55H	RB8	EVEN	PE	OERR	PERR	FERR	SCLKS	IOC
			R	R/W		R (Cleared to 0 by reading)			R/W	
				0	0	0	0	0	0	0
			Receiving data bit 8	Parity 0: Odd 1: Even	1: Parity Enable	1: Error Overrun	1: Error Parity	1: Error Framing	0: SCLK1 (↑) 1: SCLK1 (↓)	1: Input SCLK1 pin
SC1-MOD	Serial Channel 1 Mode	56H	TB8	–	RXE	WU	SM1	SM0	SC1	SC0
			R/W							
			0	0	0	0	0	0	0	0
			Transmission data bit 8	Fix at "0"	1: Receive Enable	1: Wake up Enable	00: I/O Interface 01: UART 7bit 10: UART 8bit 11: UART 9bit		00: TO0 Trigger 01: Baud rate generator 10: Internal clock ϕ1 11: Don't care	
BR1CR	Baud Rate Control	57H	–		BR1CK1	BR1CK0	BR1S3	BR1S2	BR1S1	BR1S0
			R/W		R/W					
			0		0	0	0	0	0	0
			Fix at "0"		00: ϕT0 (fc/4) 01: ϕT2 (fc/16) 10: ϕT8 (fc/64) 11: ϕT32 (fc/256)		Set frequency divisor 0~F ("1" prohibited)			
ODE	Serial Open Drain Enable	58H							ODE1	ODE0
									R/W	
									0	0
									1:P93 Open-drain	1:P90 Open-drain

(7) A/D Converter Control

Symbol	Name	Address	7	6	5	4	3	2	1	0
ADMOD	A/D Converter Mode reg	5EH	EOCF	ADBF	REPET	SCAN	ADCS	ADS	ADCH1	ADCH0
			R		R/W					
			0	0	0	0	0	0	0	0
			1: End	1: Busy	1: Repeat mode	1: Scan mode	1: Slow mode	1: START	Analog Input Channel Select	
ADREG0L *1)	AD Result Reg 0 low	60H	ADR01	ADR00						
			R							
			Undefined		1	1	1	1	1	1
ADREG0H	AD Result Reg 0 high	61H	ADR09	ADR08	ADR07	ADR06	ADR05	ADR04	ADR03	ADR02
			R							
			Undefined							
ADREG1L *1)	AD Result Reg 1 low	62H	ADR11	ADR10						
			R							
			Undefined		1	1	1	1	1	1
ADREG1H	AD Result Reg 1 high	63H	ADR19	ADR18	ADR17	ADR16	ADR15	ADR14	ADR13	ADR12
			R							
			Undefined							
ADREG2L *1)	AD Result Reg 2 low	64H	ADR21	ADR20						
			R							
			Undefined		1	1	1	1	1	1
ADREG2H	AD Result Reg 2 high	65H	ADR29	ADR28	ADR27	ADR26	ADR25	ADR24	ADR23	ADR22
			R							
			Undefined							
ADREG3L *1)	AD Result Reg 3 low	66H	ADR31	ADR30						
			R							
			Undefined		1	1	1	1	1	1
ADREG3H	AD Result Reg 3 high	67H	ADR39	ADR38	ADR37	ADR36	ADR35	ADR34	ADR33	ADR32
			R							
			Undefined							

*1 : Data to be stored in A/D Conversion Result Reg Low are the lower 2 bits of the conversion result. The contents of the lower 6 bits of this register are always read as "1".

(8) Interrupt Control (1/2)

Symbol	Name	Address	7	6	5	4	3	2	1	0
INTE-0AD	INTerrupt Enable 0 & A/D	70H (Prohibit RMW)	INTAD				INT0			
			IADC	IADM2	IADM1	IADM0	I0C	I0M2	I0M1	I0M0
			R/W		W		R/W		W	
			0	0	0	0	0	0	0	0
INTE45	INTerrupt Enable 4/5	71H (Prohibit RMW)	INT5				INT4			
			I5C	I5M2	I5M1	I5M0	I4C	I4M2	I4M1	I4M0
			R/W		W		R/W		W	
			0	0	0	0	0	0	0	0
INTE67	INTerrupt Enable 6/7	72H (Prohibit RMW)	INT7				INT6			
			I7C	I7M2	I7M1	I7M0	I6C	I6M2	I6M1	I6M0
			R/W		W		R/W		W	
			0	0	0	0	0	0	0	0
INTET10	INTerrupt Enable Timer 1/0	73H (Prohibit RMW)	INTT1 (Timer 1)				INTT0 (Timer 0)			
			IT1C	IT1M2	IT1M1	IT1M0	IT0C	IT0M2	IT0M1	IT0M0
			R/W		W		R/W		W	
			0	0	0	0	0	0	0	0
INTE-PW10	INTerrupt Enable PWm 1/0	74H (Prohibit RMW)	INTT3 (Timer 3/PWM1)				INTT2 (Timer 2/PWM0)			
			IPW1C	IPW1M2	IPW1M1	IPW1M0	IPW0C	IPW0M2	IPW0M1	IPW0M0
			R/W		W		R/W		W	
			0	0	0	0	0	0	0	0
INTET54	INTerrupt Enable Treg 5/4	75H (Prohibit RMW)	INTTR5 (TREG5)				INTTR4 (TREG4)			
			IT5C	IT5M2	IT5M1	IT5M0	IT4C	IT4M2	IT4M1	IT4M0
			R/W		W		R/W		W	
			0	0	0	0	0	0	0	0
INTET76	INTerrupt Enable Treg 7/6	76H (Prohibit RMW)	INTTR7 (TREG7)				INTTR6 (TREG6)			
			IT7C	IT7M2	IT7M1	IT7M0	IT6C	IT6M2	IT6M1	IT6M0
			R/W		W		R/W		W	
			0	0	0	0	0	0	0	
INTES0	INTerrupt Enable Serial 0	77H (Prohibit RMW)	INTTX0				INTRX0			
			ITX0C	ITX0M2	ITX0M1	ITX0M0	IRX0C	IRX0M2	IRX0M1	IRX0M0
			R/W		W		R/W		W	
			0	0	0	0	0	0	0	0
INTES1	INTerrupt Enable Serial 1	78H (Prohibit RMW)	INTTX1				INTRX1			
			ITX1C	ITX1M2	ITX1M1	ITX1M0	IRX1C	IRX1M2	IRX1M1	IRX1M0
			R/W		W		R/W		W	
			0	0	0	0	0	0	0	0

IxxM2	IxxM1	IxxM0	Function (Write)
0	0	0	Prohibit interrupt request.
0	0	1	Set interrupt request level to "1".
0	1	0	Set interrupt request level to "2".
0	1	1	Set interrupt request level to "3".
1	0	0	Set interrupt request level to "4".
1	0	1	Set interrupt request level to "5".
1	1	0	Set interrupt request level to "6".
1	1	1	Prohibit interrupt request.

IxxC	Function (Read)	Function (Write)
0	Indicate no interrupt request.	Clear interrupt request flag.
1	Indicate interrupt request.	----- Don't care -----

Interrupt Control (2/2)

Symbol	Name	Address	7	6	5	4	3	2	1	0
DMA0V	DMA 0 request Vector	7CH (Prohibit RMW)				μDMA0 start vector				
						DMA0V8	DMA0V7	DMA0V6	DMA0V5	DMA0V4
						W				
						0	0	0	0	0
DMA1V	DMA 1 request Vector	7DH (Prohibit RMW)				μDMA1 start vector				
						DMA1V8	DMA1V7	DMA1V6	DMA1V5	DMA1V4
						W				
						0	0	0	0	0
DMA2V	DMA 2 request Vector	7EH (Prohibit RMW)				μDMA2 start vector				
						DMA2V8	DMA2V7	DMA2V6	DMA2V5	DMA2V4
						W				
						0	0	0	0	0
DMA3V	DMA 3 request Vector	7FH (Prohibit RMW)				μDMA3 start vector				
						DMA3V8	DMA3V7	DMA3V6	DMA3V5	DMA3V4
						W				
						0	0	0	0	0
IIMC	Interrupt Input Mode Contorol	7BH (Prohibit RMW)						I0IE	I0LE	NMIREE
								W	W	W
								0	0	0
								1: INT0 input enable	0: INT0 edge mode 1: INT0 level mode	1: Operate even at $\overline{NMI}$ rise edge

(9) Chip Select / Wait Controller

Symbol	Name	Address	7	6	5	4	3	2	1	0
B0CS	Block 0 CS/WAIT control register	68H (Prohibit RMW)	B0E	B0SYS	B0CAS	B0BUS	B0W1	B0W0	B0C1	B0C0
			W	W	W	W	W	W	W	W
			0	0	0	0	0	0	0	0
			1: CS Enable	1: SYSTEM only	0: $\overline{CS0}$ 1: $\overline{CAS0}$	0: 16bit Bus 1: 8bit Bus	00: 2WAIT 01: 1WAIT 10: 1WAIT + n 11: 0WAIT		00: 7F00H~7FFFH 01: 400000H~ 10: 800000H~ 11: C00000H~	
B1CS	Block 1 CS/WAIT control register	69H (Prohibit RMW)	B1E	B1SYS	B1CAS	B1BUS	B1W1	B1W0	B1C1	B1C0
			W	W	W	W	W	W	W	W
			0	0	0	0	0	0	0	0
			1: CS Enable	1: SYSTEM only	0: $\overline{CS1}$ 1: $\overline{CAS1}$	0: 16bit Bus 1: 8bit Bus	00: 2WAIT 01: 1WAIT 10: 1WAIT + n 11: 0WAIT		00: 480H~7FFFH 01: 400000H~ 10: 800000H~ 11: C00000H~	
B2CS	Block 2 CS/WAIT control register	6AH (Prohibit RMW)	B2E	B2SYS	B2CAS	B2BUS	B2W1	B2W0	B2C1	B2C0
			W	W	W	W	W	W	W	W
			1	0	0	0	0	0	0	0
			1: CS Enable	1: SYSTEM only	0: $\overline{CS2}$ 1: $\overline{CAS2}$	0: 16bit Bus 1: 8bit Bus	00: 2WAIT 01: 1WAIT 10: 1WAIT + n 11: 0WAIT		00: 8000H~ 01: 400000H~ 10: 800000H~ 11: C00000H~	

Note 1 : After reset, only "Block 2" is set to enable.
→ After reset, the program starts in 16-bit data bus, 2-wait state.

Note 2 : These registers can be accessed only in system mode.

TOSHIBA

CHAPTER 3 TLCS-900 APPLICATION CIRCUIT

TOSHIBA CORPORATION

APPLICATION CIRCUIT

(1) Resonator

The TLCS-900 incorporates the oscillation circuit, so that it can easily obtain the required clock by connecting the resonator to pins (X1 and X2).

Table 1 Ceramic Resonator (Murata Manufacturing Co., Ltd)

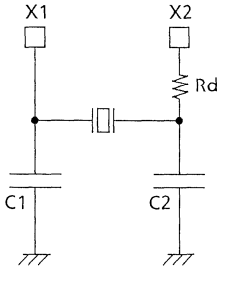
Frequency	Product No.	$C_1 = C_2$	Rd
4MHz	CSA 4.00MG	30PF	0 KΩ
4MHz	CST 4.00MGW	(Built-in 30PF)	0 KΩ
8MHz	CSA 8.0MT	30PF	0 KΩ
8MHz	CST 8.0MT	(Built-in 30PF)	0 KΩ
12MHz	CSA 12.0MT	30PF	0 KΩ
12MHz	CST 12.0MT	(Built-in 30PF)	0 KΩ
16MHz	CSA 16.00MX040	5PF	0 KΩ

220192

Table 2 Ceramic Resonator (KYOCERA CORPOLATION)

Frequency	Product No.	C_1	C_2	Rd
4MHz	KBR-4.0MSA	33PF	82PF	0 KΩ
10MHz	KBR-10.0M	33PF	33PF	0 KΩ
12MHz	KBR-12.0M	33PF	33PF	0 KΩ

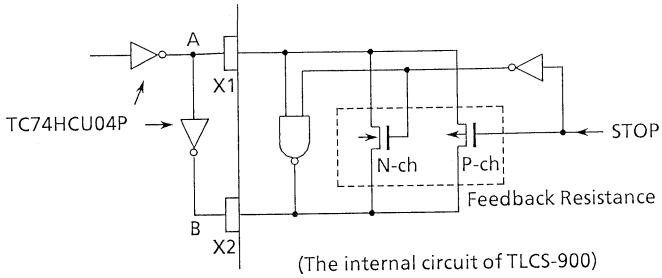
220192



030289

Figure1. Resonator Connection

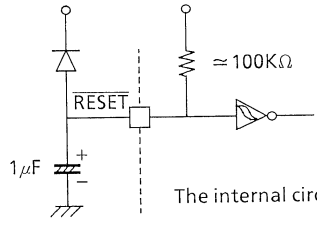
(3) Recommended Circuit for External Clock Input



(The internal circuit of TLCS-900)

Condition : Duty of point A 50 ± 5% (@ Vcc / 2)
CL = 50PF (max) of point A, B
The single drive from X1 is recommended for application using STOP mode, because X2 is fixed to "H" level (X1:high-impedance) during stop.

(4) POWER ON RESET Circuit



The internal circuit of TLCS-900 180890

TOSHIBA

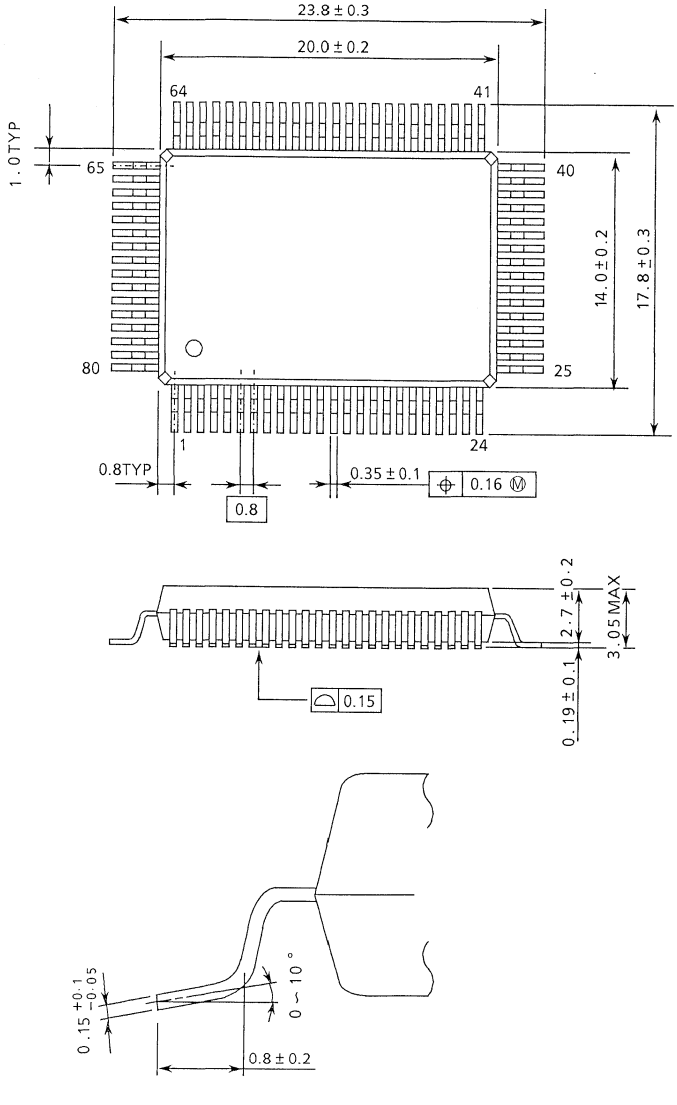
CHAPTER 4 PACKAGE

TOSHIBA CORPORATION

80 Pin FP (Flat Package)

PACKAGE NAME : QFP80-P-1420B

Unit : mm



170890

Postscript

This is a technical document that describes the operating functions and electrical specifications of the TLCS-900 (LSI) 16-bit microcontroller series.

This document is intended for use in designing system hardware. Software designers whould use the separate Assembly Language Reference Manual.

Toshiba provides a variety of development tools and basic software to enable efficient software development.

These development tools have specifications that support advances in microcomputer (LSI) hardware and can be used extensively. Both hardware and software are supported continuously with version updates.

Recent advances in CMOS LSI production technology have been phenomenal and microcomputer systems for LSI design are constantly being improved. The products described in this document may also be revised in the future. Be sure to check the latest specifications before using

Toshiba is developing highly integrated, high-performance microcomputers using advanced MOS production technology and especially well-established CMOS technology.

We are prepared to meet requests for custom packaging for a variety of application areas.

We are confident that out products can satisfy your application needs now and in the future.

OVERSEAS SUBSIDIARIES AND AFFILIATES

Toshiba America Electronic Components, Inc.

Irvine Head Office
Tel.: (714) 455-2000 Fax: (714) 859-3963
Burlington Office (Boston)
Tel.: (617) 272-4352, 5548 Fax: (617) 272-3089
Fishkill Office
Tel.: (914) 896-6500 Fax: (914) 297-6851
Marlton Office
Tel.: (609) 985-3737 Fax: (609) 985-6814
Deerfield Office (Chicago)
Tel.: (708) 945-1500 Fax: (708) 945-1044
Southfield Office (Detroit)
Tel.: (313) 827-7700 Fax: (313) 827-4444
Minneapolis Office
Tel.: (612) 921-8815 Fax: (612) 921-8818
Sunnyvale Office
Tel.: (408) 737-9844 Fax: (408) 737-9905
ASIC Marketing Office
Tel.: (408) 733-3223 Fax: (408) 733-4539
Beaverton Office (Portland)
Tel.: (503) 629-0818 Fax: (503) 629-0827
Bellevue Office
Tel.: (206) 646-6258 Fax: (206) 455-4699
Everett Office
Tel.: (206) 347-2993 Fax: (206) 347-4838
Norcross Office (Atlanta)
Tel.: (404) 368-0203 Fax: (404) 368-0075
North Carolina Office (Raleigh)
Tel.: (919) 859-2800 Fax: (919) 859-2898
Huntsville Office
Tel.: (205) 721-1467 Fax: (205) 721-1510
Altamonte Springs Office
Tel.: (407) 332-0966 Fax: (407) 339-3777
Boca Raton Office
Tel.: (407) 998-9204 Fax: (407) 998-9207
Maryland Office
Tel.: (301) 290-5918 Fax: (301) 290-5910
Richardson Office (Dallas)
Tel.: (214) 480-0470 Fax: (214) 235-4114
Houston Office
Tel.: (713) 320-9501 Fax: (713) 251-3257
Tustin Office
Tel.: (714) 259-0368 Fax: (714) 259-9439
Woodland Hills Office
Tel.: (818) 592-0216 Fax: (818) 592-0750
San Diego Office
Tel.: (619) 451-1869 Fax: (619) 451-9049
Microelectronics Center (Sunnyvale)
Tel.: (408) 739-0560 Fax: (408) 746-0577

Toshiba Display Devices Inc.

U.S.A.
Tel.: (607) 796-3500 Fax: (607) 796-3564

Vertex Semiconductor Corp.

U.S.A.
Tel.: (408) 456-8900 Fax: (408) 456-8910

Industria Mexicana Toshiba S.A. DE C.V.

Mexico
Tel.: 5-65-00-88 Telex: 017-72-560
Cable: Toshiba Mexico

Toshiba Electronics Europe GmbH

Düsseldorf Head Office
Tel.: (0211) 5296-0 Fax: (0211) 5296-400
Teletex: 2114689 = TOSHD
Stuttgart Office
Tel:(07152) 6045-0 Fax: (07152) 6045-45
Telex: 7245706 = TOSSD
München Office
Tel.: (089) 928091-0 Fax: 089-9280942
Telex: 5-213363 = TOSMD

Toshiba Electronics France SARL

France
Tel.: (1) 48 94 20 20 Fax: (1) 48 94 51 15
Telex: 232 030F

Toshiba Electronics Italiana S.R.L

Italy
Tel.: 039-6057234 Fax: 039-6057252
Telex: 326423 SIAVBC

Toshiba Electronics España, S.A.

Spain
Tel.: (1) 3143693 Fax: (1) 3143610

Toshiba Electronics (UK) Limited

U.K.
Tel.: 0276-694600 Fax: 0276-691583
Telex: 858803 = TOSHEC

Toshiba Electronics Scandinavia AB

Sweden
Tel.: 46-8-704 0900 Fax: 46-8-80 8459
Telex: 14169 TSBSTK S

Toshiba Semiconductor GmbH

F.R. Germany
Tel: (0531) 3199-0 Fax: (0531) 3199-299
Telex: 952368

Toshiba Electronics Asia (Singapore) Pte. Ltd.

Singapore Head Office
Tel.: 278-5252 Fax: 271-5155
Bangkok Office
Tel.: (2) 501-1635~6 Fax: (2) 501-1638

Toshiba Electronics Asia, Ltd.

Hong Kong Head Office
Tel.: 852-3756111 Fax: 852-3750969
Telex: 38501 TSBEH HX
Seoul Office
Tel.: (2) 757-2472~4 Fax: (2) 757-2475
Shanghai Office
Tel.: 86-21-4334077 86-21-4334131
Fax: 86-21-4333928

Toshiba Electronics Taiwan Corp.

Taipei Head Office
Tel.: 2-514-9988 Fax: 2-514-7892
Kaohsiung Office
Tel.: (07) 222-0826 Fax: (07) 223-0046

Toshiba Electronics Malaysia Sdn. Bhd.

Malaysia Head Office
Tel.: 03-352-6001-7 Fax: 03-352-6139
Telex: TOEL MA 39506
Penang Office
Tel.: 04-368523, 04-368529 Fax: 04-368515
Kuala Lumpur Office
Tel.: 03-7565561 Fax: 03-7565409

Toshiba Display Devices (Thailand) Co., Ltd.

Thailand
Tel.: (2) 501-1200 Fax: (2) 501-1209

Toshiba Semiconductor (Thailand) Co., Ltd.

Thailand
Tel.: (2) 513-9648 Fax: (2) 513-9653

9204

TOSHIBA

TOSHIBA CORPORATION
INTERNATIONAL OPERATIONS—ELECTRONIC COMPONENTS
1-1, SHIBAURA 1-CHOME, MINATO-KU, TOKYO, 105-01, JAPAN
Tel.: (03) 3457-3914 Fax: (03) 3451-0756 Telex: J22587

16-BIT MICROCONTROLLER TLCS-900 SERIES USERS MANUAL
TMP96C141/TMP96CM40/TMP96PM40

1992
4443-B

TOSHIBA CORPORATION

'92-5 So Printed in Japan