PRELIMINARY DATA SHEET

NEC

MOS INTEGRATED CIRCUIT

μPD70423

V55SC™
16-BIT MICROPROCESSOR

DESCRIPTION

The μPD70423 (also called V55SC) is a microprocessor with a 16-bit CPU, RAM UART, timer, DMA controller, and interrupt controller all integrated on one chip.

The V55SC is one in the V55 family and is software compatible with the single-chip microcomputer mPD70322, and (also called V25™ and V35™). The V55 family is a high-order V25 model and is capable of higher functions, and higher performance and can be used in all application fields.

Especially, the serial data communication function in the V55SC is enhanced.

FEATURES

- Internal 16-bit architecture
- External 16/8-bit selectable data bus width
- Software compatible with the V20™, V30™ (native mode), V25, and V35 (There are additional instructions.)
- Minimum instruction cycle: 160 ns/12.5 MHz (for an external clock of 25 MHz)
- Memory space: Mainmemory space : 15M bytes
  Local memory space : 1M byte (mapping onto an area contiguous with the main memory space)
- Register file space (in on-chip RAM) : 512 bytes/16 register banks
- I/O space : 64K bytes
- Partitioning the memory in varisble sizes (maximum of 6 blocks) and automatic wait control
- I/O line (input port: 5-bit, input/output port: 51 bits)
- Universal asynchronous receiver/transmitter (UART): 1 channel
  - On-chip exclusive baudrate generator
  - Start-up transmission mode
- Multiprotocol serial controller (MPSC): 2 channels
  - μPD72001/72002 subset functions
  - On-chip exclusive baud rate generator
  - SYNC mode, ASYNC mode, HDLC mode
- DMA controller (DMAC): 2 channels
  - 4 types of DMA transmission modes (single transfer, demand release, single step, burst)
  - Intelligent DMA mode (ring buffer management operation)
  - DMA transmission rate: Maximum 4.1M word/sec (during I/O memory transmission in the burst mode)
  - DMA memory address register (linear): 24 bits
  - Terminal counter: 21 bits
- Local bus DMA controller (LDMAC): 4 channels
  - Block chain operation

The information contained in this document is being issued in advance of the production cycle for the device. The parameters for the device may change before final production or NEC Corporation, at its own discretion, may withdraw the device prior to its production.

The mark ★ shows major revised points.

Document No.
(O.D.No. ID-8206A)
Date Published March 1993 P
Printed in Japan

- Interrupt controller
  - Multiple interrupt serving control according to programmable priority (4 levels)
  - 3 types of interrupt response formats

    Vector interrupt function, register bank switching function, macro service function
- Dram and pseudo SRAM refresh function
- Wachdog timer function
- Standby function (STOP mode, HALT mode, IDLE mode)
- On-chip clock generation circuit
- 16-bit timer/counter: 4 channels
- Software interval timer (16 bits)
- Address block wait insertion function and RAS/CAS switching timing generation function

USES

- It can be used in sysyem control for data processing which uses serial communication

  (Data processing terminals, G4 facsimiles, switching devices, television telephones, etc.)

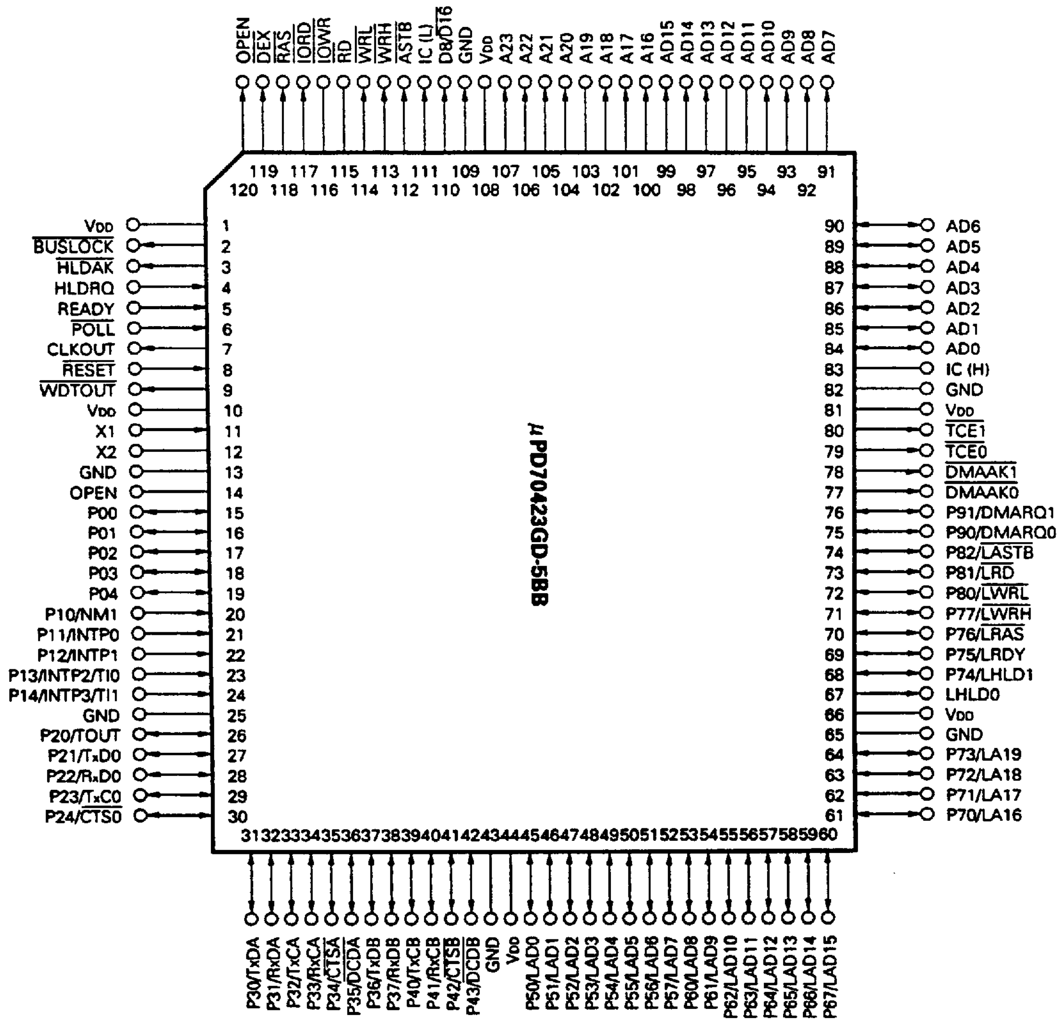
ORDING INFORMATION

Ordering Name	Package	Maximum Operating Frequency (MHz)	Quality Grade
μPD70423GD-5BB	120-pin plastic QFP (□28)	12.5	Standard
μPD70423SA	132-pin plastic PGA	12.5	Standard

Please refer to "Quality grade on NEC Semiconductor Devices" (Document number IEI-1209) published by NEC Corporation to know the specification of quality grade on the devices and its recommended applications.

PIN CONNECTION DIAGRAM (TOP VIEW)

(1) 120-Pin Plastic QFP



Remarks IC: Internally Connected

Note

1. **The IC (H) pin should be connected to VDD externally by way of a pull-up resister (1 to 10 kΩ).** ★
2. **The IC (L) pin should be connected to GND externally by way of a pull-up resister (1 to 10 kΩ).** ★
3. **The OPEN pin should not be connected to anything.**

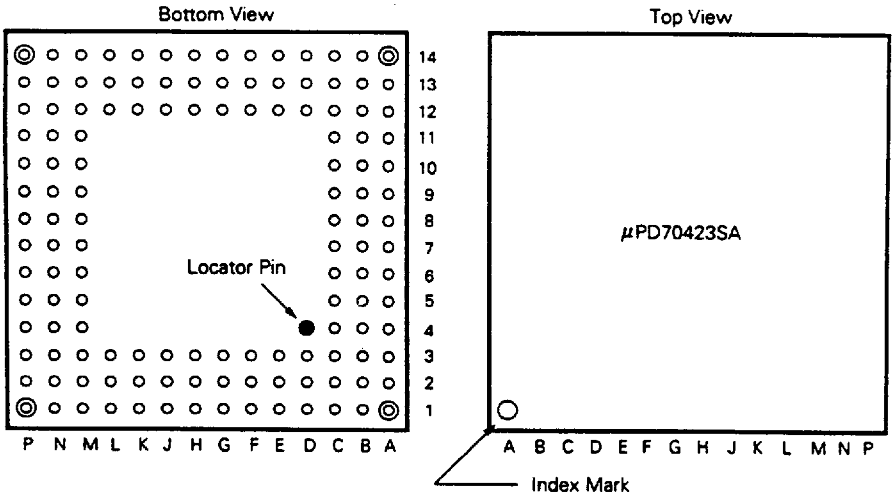
No.	Signal Nane	Port	No.	Signal Name	Port	No.	Signal Name	Port
F1	$\overline{LRD}$	P81	K3	AD2	—	N3	AD9	—
F2	$\overline{LWRL}$	P80	K12	$\overline{POLL}$	—	N4	AD11	—
F3	$\overline{LRAS}$	P76	K13	$\overline{WDTOUT}$	—	N5	AD14	—
F12	INTP1	P12	K14	X1	—	N6	A18	—
F13	NMI	P10	L1	AD0	—	N7	A21	—
F14	—	P04	L2	AD3	—	N8	A23	—
G1	NC	—	L3	AD6	—	N9	D8/$\overline{D16}$	—
G2	DMARQ0	P90	L12	$\overline{BUSLOCK}$	—	N10	$\overline{ASTB}$	—
G3	$\overline{LASTB}$	P82	L13	READY	—	N11	$\overline{IOWR}$	—
G12	—	P03	L14	$\overline{RESET}$	—	N12	$\overline{DEX}$	—
G13	—	P02	M1	AD1	—	N13	V_{DD}	—
G14	—	P01	M2	AD5	—	N14	HLDRQ	—
H1	DMARQ1	P91	M3	NC	—	P1	AD7	—
H2	$\overline{DMAAK0}$	—	M4	AD8	—	P2	AD10	—
H3	$\overline{DMAAK1}$	—	M5	AD12	—	P3	AD13	—
H12	OPEN	—	M6	A16	—	P4	AD15	—
H13	—	P00	M7	A20	—	P5	A17	—
H14	NC	—	M8	V_{DD}	—	P6	A19	—
J1	$\overline{TCE0}$	—	M9	$\overline{WRH}$	—	P7	NC	—
J2	$\overline{TCE1}$	—	M10	$\overline{IORD}$	—	P8	A22	—
J3	GND	—	M11	NC	—	P9	GND	—
J12	V_{DD}	—	M12	NC	—	P10	IC (L)	—
J13	X2	—	M13	$\overline{HLDAK}$	—	P11	$\overline{WRL}$	—
J14	GND	—	M14	CLKOUT	—	P12	$\overline{RD}$	—
K1	V_{DD}	—	N1	AD4	—	P13	$\overline{RAS}$	—
K2	IC (H)	—	N2	NC	—	P14	OPEN	—

Remarks IC : Internally Connected
NC : Non-Connection

Note

1. **The IC (H) pin should be connected to V_{DD} externally by way of a pull-up resister (1 to 10 kΩ).** ★
2. **The IC (L) pin should be connected to GND externally by way of a pull-up resister (1 to 10 kΩ).** ★
3. **The OPEN pin should not be connected to anything.**

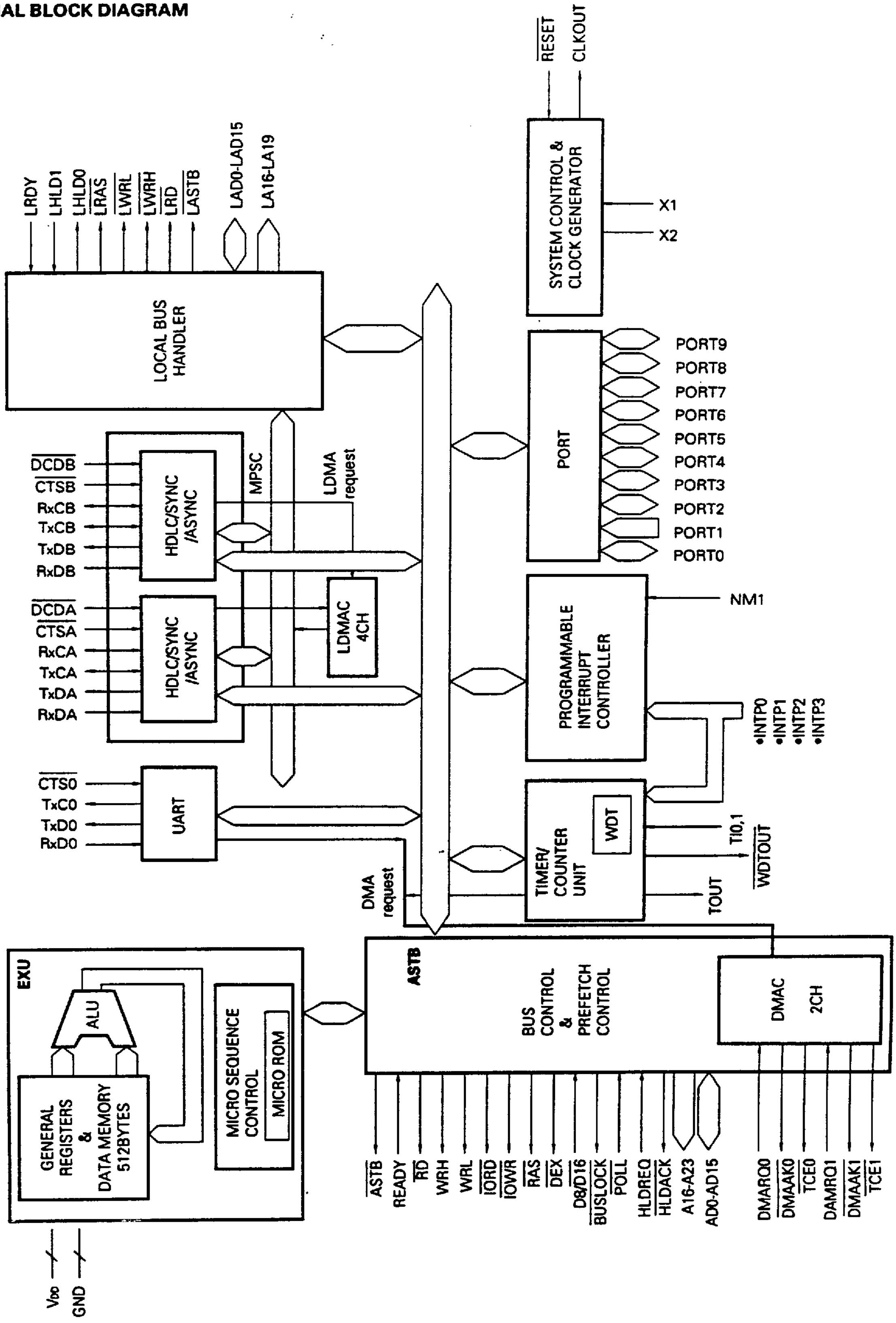
(2) 132-Pin Plastic PGA



Remarks The locator pin is not included in the pin numbers.

No.	Signal Nane	Port	No.	Signal Name	Port	No.	Signal Name	Port
Ai	LAD15	P67	B5	LAD7	P57	C9	RxCB	P41
A2	LAD13	P65	B6	LAD5	P55	C10	$\overline{\text{DCDA}}$	P35
A3	LAD10	P62	B7	LAD2	P52	C11	RxDA	P31
A4	LAD9	P61	B8	LAD0	P50	C12	NC	—
A5	LAD6	P56	B9	$\overline{\text{DCDB}}$	P43	C13	TxC0	P23
A6	LAD4	P54	B10	RxDB	P37	C14	GND	—
A7	LAD1	P51	B11	$\overline{\text{CTSA}}$	P34	D1	LHLD1	P74
A8	NC	—	B12	TxCA	P32	D2	GND	—
A9	GND	—	B13	NC	—	D3	LA17	P71
A10	$\overline{\text{CTSB}}$	P42	B14	RxD0	P22	D12	$\overline{\text{CTS0}}$	P24
A11	TxCB	P40	C1	LHLD0	—	D13	TxD0	P21
A12	TxDB	P36	C2	LA18	P72	D14	INTP3/TI1	P14
A13	RxCA	P33	C3	NC	—	E1	$\overline{\text{LWRH}}$	P77
A14	TxDA	P30	C4	NC	—	E2	LRDY	P75
B1	LA19	P73	C5	LAD12	P64	E3	V_{DD}	—
B2	LA16	P70	C6	LAD8	P60	E12	TOUT	P20
B3	LAD14	P66	C7	LAD3	P53	E13	INTP2/TI0	P13
B4	LAD11	P63	C8	V_{DD}	—	E14	INTP0	P11

INTERNAL BLOCK DIAGRAM



CONTENTS

★ 1. PIN FUNCTIONS

1.1 PIN FUNCTION LIST

1.1.1 Port Functions

Pin Name	Input/Output	Function	Dual-Function Pin
P00 to P04	Input/output	Port 0 Input/output specifiable bit-wise 5-bit input/output port	—
P10 *	Input	Port 1 5-bit input/output port	NM1
P11			INTP0
P12			INTP2
P13			INTP2/TI0
P14			INTP3/TI1
P20	Input/output	Port 2 Input/output specifiable bit-wise 5-bit input/output port	TOUT
P21			TxD0
P22			RxD0
P23			TxC0
P24			$\overline{\text{CTS0}}$
P30		Port 3 Input/output specifiable bit-wise 8-bit input/output port	TxDA
P31			RxDA
P32			TxCA
P33			RxCA
P34			$\overline{\text{CTSA}}$
P35			$\overline{\text{DCDA}}$
P36			TxDB
P37			RxDB
P40		Port 4 Input/output specifiable bit-wise 4-bit input/output port	TxCB
P41			RxCB
P42			$\overline{\text{CTSB}}$
P43			$\overline{\text{DCDB}}$
P50 to P57		Port 5 Input/output specifiable in 8-bit units 8-bit input/output port	LAD0 to LAD7
P60 to P67		Port 6 Input/output specifiable in 8-bit units 8-bit input/output port	LAD8 to LAD15

* Cannot be used as a general-purpose port (non-maskable interrupt)

Pin Name	Input/Output	Function	Dual-Function Pin
P70 to P73	Input/output	Port 7 Input/output specifiable bit-wise 8-bit input/output port	LA16 to LA19
P74			LHLD1
P75			LRDY
P76			$\overline{\text{LRAS}}$
P77			$\overline{\text{LWRH}}$
P80		Port 8 Input/output specifiable bit-wise 3-bit input/output port	$\overline{\text{LRWL}}$
P81			$\overline{\text{LRD}}$
P82			$\overline{\text{LASTB}}$
P90		Port 9 Input/output specifiable bit-wise 2-bit input/output port	DMARQ0
P91			DMARQ1

1.1.2 Non-Port Functions

(1) Pin function for main bus control

Pin Name	Input/Output	Function	Dual-Function Pin
$\overline{\text{ASTB}}$	Output	Main bus external bus cycle address stobe signal output	—
$\overline{\text{RD}}$		Main bus external memory cycle address stobe signal output	
$\overline{\text{WRL}}$		Main bus external memory cycle lower byte data write strobe signal output	
$\overline{\text{WRH}}$		Main bus external memory cycle upperr byte data write strobe signal output	
READY	Input	Main bus external bus cycle ready signal input	
$\overline{\text{DEX}}$	Output	External bus cycle upper byte data enable signal output	
$\overline{\text{RAS}}$		DRAM row address latch timing signal output	
D8/$\overline{\text{D16}}$	Input	External main bus data bus width selection signal input	
$\overline{\text{BUSLOCK}}$	Output	External main bus bus lock signal output	
$\overline{\text{POLL}}$	Input	$\overline{\text{POLL}}$ signal (sampling in excution of POLL instruction) input	
HLDRQ		Main bus hold request signal input	
$\overline{\text{HLDAK}}$	Output	Main bus hold acknowledge signal output	
AD0 to AD15	3-state input/output	Main bus external bus cycle address/data multiplex signal input /output	
A16 to A23	3-state output	Main bus external bus cycle address signal output	
$\overline{\text{IORD}}$	Output	External I/O cycle data read strobe signal output	
$\overline{\text{IOWR}}$		External I/O cycle data write strobe signal output	
DMARQ0	Input	DMA request signal output (channel 0)	P90
DMARQ1		DMA request signal output (channel 1)	P91
$\overline{\text{DMAAK0}}$	Output	DMA acknowledge signal output (channel 0)	—
$\overline{\text{DMAAK1}}$		DMA acknowledge signal output (channel 1)	
$\overline{\text{TCE0}}$		DMA end signal output (channel 0)	
$\overline{\text{TCE1}}$		DMA end signal output (channel 1)	

(2) Pin function for local bus control

Pin Name	Input/Output	Function	Dual-Function Pin
LA16 to LA19	Output	Local bus cycle address signal output	P70 to P73
LAD0 to LAD15	Input/output	Local bus cycle address/data multiplex signal input/output	P50 to P57, P60 to P67
$\overline{\text{LRD}}$	Output	Local bus cycle data read strobe signal output	P81
$\overline{\text{LWRL}}$	Output	Local bus cycle lower byte data write strobe signal output	P80
$\overline{\text{LWRH}}$	Output	Local bus cycle upper byte data write strobe signal output	P77
$\overline{\text{LASTB}}$	Output	Local bus cycle address strobe signal output	P82
LRDY	Input	Local bus cycle ready signal input	P75
$\overline{\text{LRAS}}$	Output	Local bus cycle DRAM row address latch timing signal output	P76
LHLD1	Input	Local bus cycle hold request/acknowledge signal input	P74
LHLD0	Output	Local bus cycle hold request/acknowledge signal output	—

(3) Other pin f unctions

Pin Name	Input/Output	Function	Dual-Function Pin
GND	—	GND potential	—
V_{DD}		Positive power supply	
$\overline{\text{RESET}}$	Input	System reset signal input	
X1	Input	System clock generation crystal resonator/ceramic resonator connection pin. When an external clock is supplied, input to X1 and leave X2 open.	
X2	—		
CLKOUT	Output	Internal system clock ϕ output	
$\overline{\text{WDTOUT}}$	Output	Watchdog timer overflow signal output	
NMI	—	Non-maskable interrupt request input *1	P10
INTP0		External interrupt request input *2	P11
INTP1			P12
INTP2			P13/TI0
INTP3			P14/TI1
TI0		External event clock input	P13/TI0
TI1			P14/TI1
TOUT	Output	Timer unit output	P20
TxDo		UART transmit data output	P21

* 1. SInce an NMI interrupt cannotb be masked, an NMI interrupy is always started by valid edge detection (the pin level is read during the port 1 read operation).

2. Masking or disabling each interrupy (IE = 0) enables these pins to be used as a general-purpose input port.

Pin Name	Input/Output	Function	Dual-Function Pin
RxD0	Input	UART receive data input	P22
TxC0	Output	UART transmit clock output	P23
$\overline{\text{CTS0}}$	Input	UART transmit enable signal input	P24
TxDA	Output	MPSC channel A transmit data output	P30
RxDA	Input	MPSC channel A receive data input	P31
TxCA	Input/output	MPSC channel A transmit clock input/output	P32
RxCA		MPSC channel A receive clock input/output	P33
$\overline{\text{CTSA}}$	Input	MPSC channel A transmit enable signal input	P34
$\overline{\text{DCDA}}$		MPSC channel A receive enable signal input	P35
TxDB	Output	MPSC channel B transmit data output	P36
RxDB	Input	MPSC channel B receive data input	P37
TxCB	Input/output	MPSC channel B transmit clock input/output	P40
RxCB		MPSC channel B receive clock input/output	P41
$\overline{\text{CTSB}}$	Input	MPSC channel B transmit enable signal input	P42
$\overline{\text{DCDB}}$		MPSC channel A receive enable signal input	P43

2. BLOCK CONFIGURATION

2.1 BUS CONTROL UNIT (BCU)

The BCU controls the main bus. The necessary bus cycles are activated in the BCU based on the physical addresses obtained by the execution unit (EXU).

2.2 EXECUTION UNIT (EXU)

The EXU controls address calculation, arithmetic logical calculations, and data transfer using a microprogram (firm ware to control the micro sequencer based on the operation code decode calculations). Inside the EXU is an built-in 512-byte RAM (register file space).

2.3 LOCAL BUS HANDLER (LBH)

The LBH controls the local bus.

2.4 INTERRUPT CONTROLLER (INTC)

All kinds of hardware interrupts generated by the on-chip peripheral hardware or generated externally are processed by either switching register banks, vectored interrupts, of macro-services. The priority of programmable 4-level interrupts can be controlled and multiple servicing control of the interrupt sources is also possible.

2.5 DMA CONTROLLER (DMAC)

The DMAC is a general-purpose DMA controller, and can handle the memory space 16M bytes linearly. Besides the I/O memory transfer mode and memory-to-memory transfer mode, the operating modes consist of an intelligent DMA (ring buffer format) mode and next address specification mode.

2.6 LOCAL BUS DMA CONTROLLER (LDMAC)

The LDMAC is a MPSC-only DMA controller and performs DMA transfer between the local bus and MPSC. The operating modes consist of a normal transfer mode and a block-chain operating mode.

2.7 UNIVERSAL ASYNCHRONOUS RECEIVER TRANSMITTER (UART)

The UART obtains data synchronization from the start-stop bit of the serial data communication function and it supports the start-stop transmission format.

2.8 MULTIPROTOCOL SERIAL CONTROLLER (MPSC)

The MPSC supports the start-stop transmission format (ASYNC mode) of the serial data communication function, and it supports character oriented protocol (SYNC mode) and bit oriented protocol (HDLC mode). ASYNC of the MPSC supports the same communication protocol as the UART, however, the method for setting commands and some functions are different.

2.9 TIMER/COUNTER UNIT (TCU)

The TCUisanon-chip16-bittimer/counterandcan interval timer free running counter and eventcounter.

2.10 WATCHDOG TIMER (WDT)

The WDT is an on-chip 8-bit watchdog timer that detects inadvertent program loops and system abnormalities. It is equipped with a WDTOUT pin for external notification of the generation of a watchdog timer interrupt.

2.11 PORTS (PORT)

Port is provided with 56 port pins and includes dual-function pins not only as the external interrupt input but as the control pins.

2.12 CLOCK GENERATOR (CG)

The CG generates the 1/2, 1/4, 1/8 and 1/16 frequency clock of the crystal oscillator/ceramic oscillator connected to pins X1 and X2, and supplies the CPUoperating clock.

2.13 SOFTWARE INTERVAL TIMER (SIT)

The SIT is an on-chip 16-bit interval timer which is used as the software timer function or the clock function.

By selecting the input clock (count clock) and by setting the software timer compare register, the interval interrupt can be set.

3. CPU FUNCTION

The V55SC has as CPU which is software compatible with the V20 and V30 (native mode), and with the V25 and V35.

3.1 FEATURES

★

- Software upward compatible with the V20 and V30 (native mode), and with the V25 and V35. (has additional instructions)
- Minimum instruction cycle: 160 ns/12.5 MHz (using an external 25 MHz clock)
- Memory space: Main memory space: 16M bytes
  Local memory space: 1M byte (mapped onto an area contiguous with the main memory space)
- Register file space (in on-chip RAM): 512 bytes/16 register banks
- I/O space: 64K bytes
- Register configuration (comparison with V20/V30 or V25/V35)

Item		V20, V30	V25, V35	V55SC
Extended segment register		No	No	DS2. DS3
Register bank		No	8 banks (in memory space)	16 banks (in register file space)
PSW	Mode flag	MD	No	No
	Register bank flag	No	RB0 to RB2	RB0 to RB3
	Input/output instruction trapping flag	No	$\overline{\text{IBRK}}$	$\overline{\text{IBRK}}$
	User flag	No	F0, F1	No
Special function register area		No	240 bytes (memory mapping onto FFF00H to FFFEFH)	496 bytes (memory mapping onto FFE00H to FFFEFH)

- Internal 16-bit architecture, 16/8-bit selectable external data bus width
- Main memory is partitioned in variable sizes (maximum of 6 blocks) and automatic wait control
  - Programmable wait function
  - Ready pin wait function
- $\overline{\text{RAS}}$ pin function
  $\overline{\text{RAS}}$, $\overline{\text{LRAS}}$ pins →RAS timing of the DRAM
  $\overline{\text{RD}}$, $\overline{\text{WRH}}$, $\overline{\text{WRL}}$, LRD, $\overline{\text{LWRL}}$, $\overline{\text{LWRH}}$ pins →CAS timing of the DRAM
  ASTB, $\overline{\text{LASTB}}$ pins →Row/column address switch timing of the DRAM
- Refresh function
  - Automatic generation of refresh cycle (RAS only)

3.2 REGISTERS

The CPU of the V55SC has a general register set that is compatible with V20 and V30 (native mode) and with V25 and V35. Also, it has various special function registers for controlling the on-chip peripheral hardware. The general register set is mapped in the register file space. This general register set has dual-function as built-in RAM, and besides built-in RAM can have bank format using up to a maximum of 16 register sets. On the other hand, the special function registers are mapped in the main memory space 0FFE00H to 0FFFEFH.

3.2.1 Register Banks

The general register set is mapped in the register file space (in the built-in RAM). The general register set take bank format and 1 bank uses 32 bytes and it is possible to set up to 16 banks. Of these 16 banks, bank 0 to bank 7 can also be used for macro service. Also, by attaching an exclusive prefix (IRAM:) to the memory transfer instruction, they can be used for accessing the data memory.

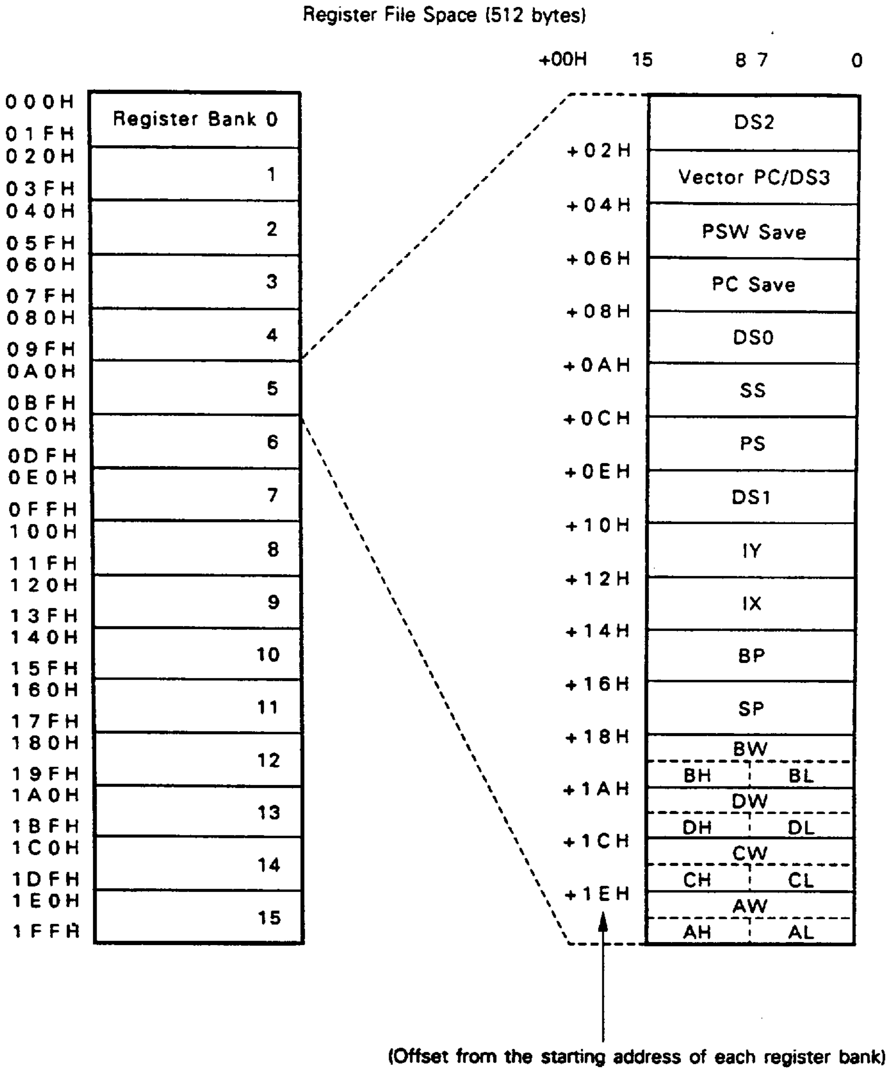
During program execution, it is possible to automatically switch to another register bank using maskable hardware interrupts or software interrupts (BRKCS instruction). To return to the original register bank from the register bank switched to using the interrupt, the instruction to return from an interrupt (RETRBI) is used.

The register bank configuration is as shown in Figure 3-1. The general register set is mapped in the area (+0.8H) to (+1FH) of the offset from the start address of each register bank. The word area from the start of a register bank becomes the expansion segment register (DS2) area. The vector PC/DS3 area is the value loaded into the PC when the register bank is switched, or in other words, it is the area for setting the offset value of the starting address of the interrupt process routine. This area can also be used as an expansion segment register (DS3) area. The PSW save area is an area used for saving the PSW when the register bank is switched. The PC save area is an area for saving the PC when the register bank is switched.

After reset, register bank 15 is selected automatically. Also, initialization of the segment register after reset is performed only for register bank 15.

In addition, the macro channels (parameter and work areas for macro service) are assigned in duplication to bank 0 and bank 1 of the 16 sets of register bank.

Fig. 3-1 Register bank Configuration



3.2.2 General Registers (AW, BW, CW, DW)

The general registers are made up of four 16-bit registers. These registers can of course access as 16-bit registers, and can also access as 8-bit registers (AH, AL, BH, BL, CH, CL, DH, DL) by dividing the upper and lower 8 bits of each registers.

These registers can be used as 8-bit registers or 16-bit registers for a wide range of instructions such as transfer instructions, calculation instructions, and logical operation instructions.

Also, each register can be used as a default registers for processing the following specific instructions.

AW : Word multiplication/division, word input/output, data exchange
AL : Byte multiplication/division, byte input/output, BCD rotation, data exchange
AH : Byte multiplication/division BW: Translation
BW : Data conversion
CW : Loop control routine, repeat prefix
CL : Shift instruction, rotate instruction, BCD operation
DW : Word multiplication/division, indirect addressing input/output

These registers are mapped in the register file space (in the built-in RAM). The address is the value of the register bank number x 32 with the offset for each register added.

Table 3-1 General-Purpose Register Offset

Register	Offset	Register	Offset
AW	1EH	AL	1EH
		AH	1FH
BW	18H	BL	18H
		BH	19H
CW	1CH	CL	1CH
		CH	1DH
DW	1AH	DL	1AH
		DH	1BH

3.2.3 Pointers (SP, BP) and Index Registers (IX, IY)

These are 16-bit registers used as base pointers or index registers when the memory is accessed by based addressing (BP), indexed addressing (IX, IY), or based indexed addressing (BP, IX, IY). Also, SP can be used as a pointer during stack manipulation. They can also be used in the same way as the general registers for instructions such as transfer instructions or arithmetic operation instructions, however, in that case, they cannot be used as 8-bit registers. Also, each register is used as a fixed address pointer in the following specific processes.

SP : Stack manipulation
IX : Block transfer, address specification of BCD operation source
IY : Block transfer, address specification of BCD operation destination

These registers are mapped in the register file space (on the built-in RAM) and their address are the value of the register bank number x 32 with the offset for each register added.

Table 3-2 Pointer and Index Register Offsets

Register	Offset
SP	16H
BP	14H
IX	12H
IY	10H

3.2.4 Segment Registers (PS, SS, DS0, DS1)

The CPU divides and controls the 1M byte basic memory space into logical segments of 64K bytes. The CPU indicates the start address of each segment using a segment register, and indicates the relative address from the start address as the offset using a different register or the effective address.

The physical address is comprised as follows.

$$\begin{array}{rcccl} & \overbrace{\times\ \ \times\ \ \times\ \ \times}^{\text{Segment Register}} & \overbrace{0}^{\text{4-bit fixed}} & \text{H} & \ldots\ \text{Segment Start Address}\\ +) & 0\ \ \times\ \ \times\ \ \times & \cdot\times & \text{H} & \ldots\ \text{Offset Value}\\ \hline & \times\ \ \times\ \ \times\ \ \times & \times & \text{H} & \ldots\ \text{Physical Address}\end{array}$$

The segment registers are comprised of a PS (program segment), SS (stack segment), DS0 (data segment 0), and DS1 (data segment 1).

PS : Program fetch
SS : Stack manipulation instruction, addressing with BP as the base register.
DS0 : General variable access, access of source block data of the block transfer instruction.
DS1 : Access of destination block data of the block transfer instruction.

By using the segment override prefix instruction, it is possible for general variable access to change from DS0 to another register. Also, for addressing using BP as the base register, another segment register can be used in the same way as the SS register.

Example ★

```
MOV   AW, 1000H
MOV   DS1, AW
MOV   BL, DS1, BYTE PTR [IX]; Byte data read from DS1:IX
```

After reset, the PS of register bank 15 is initialized to FFFFH and SS, DS0 and DS1 are initialized to 0000H.

These registers are mapped in the register file space (in the built-in RAM) and the address has the value of (the register bank number x 32) with the offset for each register added.

Table 3-3 Segment Register Offsets

Register	Offset
DS0	08H
DS1	0EH
SS	0AH
PS	0CH

3.2.5 Expansion Segment Registers (DS2, DS3)

In the V55SC, besides the segment registers for accessing the 1M byte basic memory space, there is a 16M byte expansion memory space divided into 64M byte segments, and there is an expansion segment register for specifying the start address of each segment. In the expansion segment register there are DS2 (Data Segment 2) and DS3 (Data Segment 3) which are used as shown below.

DS2: Access of general variables of the expansion memory space, and access of the source block data of the expansion memory space block transfer instruction (using the segment override prefix)

DS3: Access of general variables of the expansion memory, and access of destination block data of the expansion memory space block transfer instruction (using the segment override prefix)

Data access using the extend segment register is made by using the segment override prefix. Especially in the block transfer instruction, DS2 and DS3 canbe specified in duplication by the segment override prefix (in this case, the specification order of DS2 and DS3 is arbitrary).

Example
```
REP
DS2:
DS3: MOVBKW ; word memory block transfer from DS2 : IX to DS3 : IY
```

The start address of each segment is indicated by the expansion segment register, and a relative address from the start address is taken to be the offset and is accessed by indicating it using another register or the effective address.

The physical address is constructed as shown below.

```
  ExpansionSegmentRegister  8-bit fixed
        ×   ×   ×   ×       0   0   H   ... Segment Start Address
 +)     0   0   ×   ×       ×   ×   H   ... Offset Value
 ------------------------------------
        ×   ×   ×   ×       ×   ×   H   ... Physical Address
```

After reset, DS2 and DS3 of register bank 15 are initialized to 0000H.

These registers are mapped in the register file space (in the built-in RAM area) and the address has the value of (the register bank number x 32) with the offset for each register added.

Table 3-4 Expansion Segment Register Offsets

Register	Offset
DS2	00H
DS3	02H (Dual-function as a vectored PC)

3.2.6 Special Function Registers (SFR)

In the V55SC there are register groups that have internal on-chip peripheral hardware control functions.

A few registers are prepared according to the control contents of each peripheral hardware, and the definite operation can be set using each bit in the registers. These register groups are mapped in the main memory space and can be read and written to in the same way as normal memory.

Example

```
MOV   AW, 0FFE0HH                                                   ★
MOV   DS1, AW
MOV   BL, DS1 : BYTE  PTR [1EFH]; 0FFE0H : 1EFH (PRC register) read
```

For the flag check operation, there is a BTCLR instruction which is valid for the the upper 240 bytes (0FFF00H to 0FFFEFH) of the special function registers. Also, the BTCLRL instruction is valid for the lower 256 bytes (0FFE00H to 0FFEFFH) of the special function registers.

3.3 PROGRAM COUNTER (PC)

This is a 16-bit binary counter which holds the offset value of the program memory address being executed by the CPU.

The PC is incremented each time an instruction code is fetched from the instruction queue. Also, during execution of the branch, call, return, and break instructions, a new location address value is loaded.

After reset, 0000H is loaded into the PC. PS is initialized to FFFFH at reset and so after reset, the CPU starts execution from the physical address 0FFFF0H.

3.4 PROGRAM STATUS WORDS (PSW)

The PSW (program status words) are made up of 6 kinds of status flags and 5 kinds of control flags.

- Status Flags
  - V (Overflow) ...Overflow detection flag
  - S (Sign) ...Sign bit detection flag
  - Z (Zero) ...All zero detection flag
  - AC (Auxiliary Carry) ...4-bit carry, borrow detection flag
  - P (Parity) ...Parity detection flag
  - CY (Carry) ...Carry, borrow detection flag
- Control Flags
  - RB0 to RB3 (Register Banks 0 to 3) ...Register bank indication flag
  - DIR (Direction) ...Block transfer/input-output instructions direction control flag
  - IE (Interrupt Enable) ...Interrupt enable control flag
  - BRK (Break) ...Single-step interrupt control flag
  - IBRK (I/O Break) ...Input-output instruction trap control flag

The status flags are automatically set (1) or reset (0) according to the results (data values) of execution of the various instructions. The CY flag can be directly set, reset, or inverted using an instruction.

The control flags are set or reset by instructions and control the operation of the CPU. The IE and BRK flags must be reset when interrupt processing is activated.

The contents of the PSW can be saved in or returned from a stack using the PUSH/POP instructions. However, when returning the contents using the POP PSW instruction, bits 12 to 15 (RB0 to RB3) do not return to the PSW.

Also, the lower 8 bits of the PSW can be saved in or returned from the AH register using the MOV instruction.

The bit configuration of the PSW is as follows.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RB3	RB2	RB1	RB0	V	DIR	IE	BRK	S	Z	0	AC	0	P	$\overline{\mathrm{IBRK}}$	CY

3.5 MEMORY SPACE

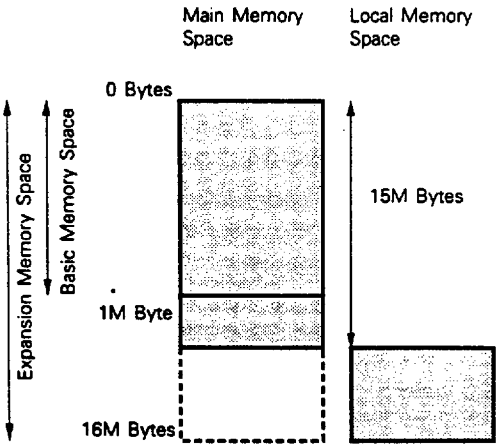
There are two kinds of memory space, main memory space and local memory space.

The main memory space is a 15M byte area and the lower 1M byte (000000H to 0FFFFFH) is the basic memory space, and the 15M bytes including the basic memory space (000000H to FFFFFFH) can be accessed as the expansion memory space. The basic memory space can be accessed using the segment registers (PS, DS0, DS1, SS), and the expansion memory space can be accessed by using the expansion segment registers (DS2, DS3).

The local memory is mapped in the 1M-byte area (F00000H to FFFFFFH) contiguous with the main memory space.The CPU accesses this local memory space as part of the extended memory space.

In addition, a 512-byte register file space (in the on-chip RAM) is separated from the main memory space and the local memory space. This register file space can be used not only as a register bank but as data memory , and can be accessed easily by attaching an exclusive override prefix instructions (IRAM:).

Fig. 3-2 Memory Space



3.5.1 Main Memory Space

In the main memory space is a 1M byte basic memory and a 16M byte expansion memory space which includes the basic memory space. The basic memory space is mapped in the lower 1M byte (000000H to 0FFFFFH) of the expansion memory space.

(1) Basic Memory Space

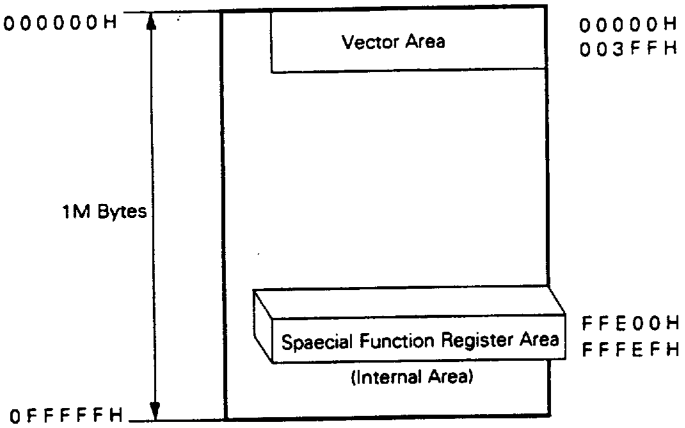
The 1M byte basic memory space is shown in Fig. 3-3.

The condition for accessing the basic memory space using the software is the same as for the V20 and V30 and the same as the V25 and V35.

The physical address of the basic memory space is specified by the segment start address indicated by the segment registers PS, SS, DS0 and DS1, and by the offset value from the segment start location indicated by another register or immediate data.

Also, the vector area for the vectored interrupt and the special function register area are mapped in the basic memory space. No data access to the external memory is possible in an area where the special function register is mapped (program fetch is possible).

Fig. 3-3 Basic Memory Space



Since area 0FFFF0H to 0FFFFFH is a system boot program area and PS and PC are set to 0FFFFH and 0H, respecively after reset release, execution of the program is started from 0FFFF0H

(2) **Expansion Memory Space**

The 16M byte expansion memory space is shown in Fig. 3-4.

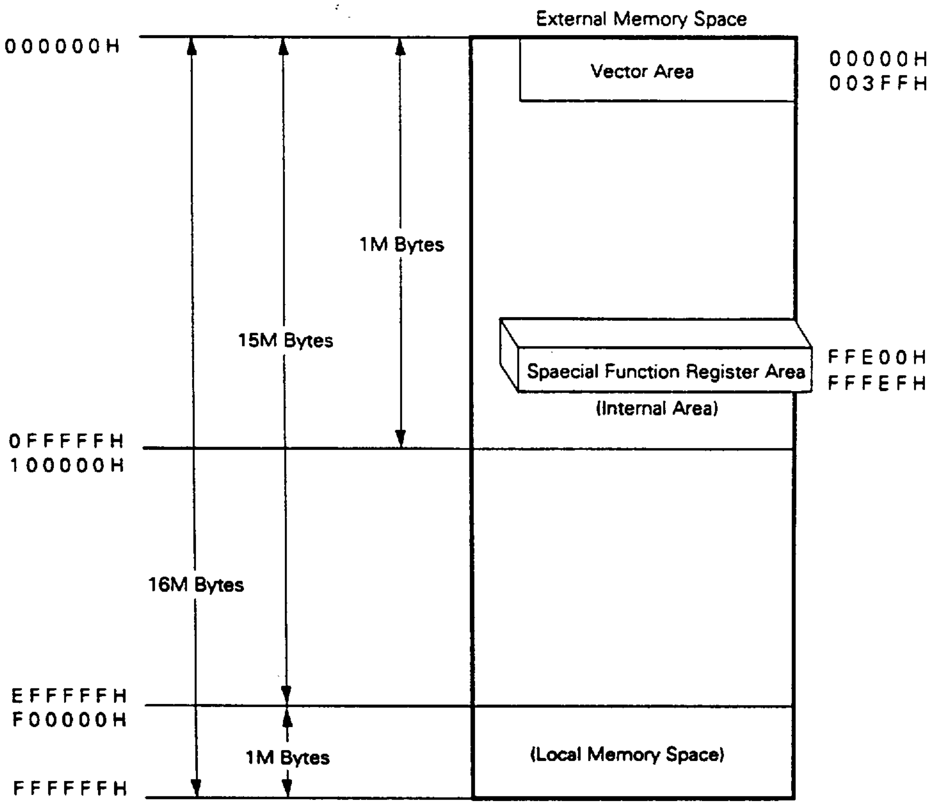
Only data access of the expansion memory space is possible using the software.

However, the basic memory is mapped in the lower 1M byte (000000H to 0FFFFFH) and can be accessed by using the segment registers PS, SS, DS0 and DS1.

Data access is possible in the expansion memory space by using the expansion segment registers DS2 and DS3. DS2 and DS3 specify attaching as the expansion segment override prefix instruction to the memory manipulation instruction.

The physical address of the expansion memory space is specified by the segment start address indicated by the expansion segment registers, and by the off values from the segment start location indicated by another register or immediate data. When the generated address indicates the lowest 1M byte (000000H to 0FFFFFH), the basic memory space is accessed.

Fig. 3-4 Expansion Memory Space ★



3.5.2 Special Function Register Area

The register groups that have been assigned functions such as specifying the operating mode of the on-chip peripheral hardware, and monitoring the status, are mapped in the 496-byte space 0FFE00H to 0FFFEFH.

Program fetch cannot be performed from these areas (a progam fetch is perfomed on the external memory).

The special function registers are controlled by accessing using the memory manipulation instruction.

A list of special function registers is given in Table 3-5. The meaning of the items in the table are as follows.

- Symbols ... These are codes which show the name of the special function register. They correspond to the operand entry format (Symbol name) of the memory manipulation instruction.
- R/W ... This indicates whether read/write of the special function register is possible.
  R/W : Read/write possible
  R : Read only possible
  W : Write only possible
- Manipulation Method ... Shows whether 1-bit manipulation, 8-bit manipulation, 16-bit manipulation, or 32-bit manipulation for each register is possible.
- RESET ... Shows the status of each register when $\overline{\text{RESET}}$ is input. "X" indicates that it is undefined.

Note Since the portions of the address which is not listed is reserved, do not access it by a user program.

Table 3-5 List of Special Function Registers (1/12)

Adress	Special Function Register Name	Symbol	R/W	Manipulatable Bit Units				After Reset	
				1 bit	8-bit	18-bit	32-bit		
0FFE00H	MPSC send buffer A	TxB_A	W		○			Undefined	★
0FFE01H	MPSC receive buffer A	RxB_A	R		○			Undefined	★
0FFE02H	MPSC control register 00A	CR00A	W		○			xxxx0xxxB	
0FFE03H	MPSC control register 01A	CR01A	R/W	○	○			0x000x00B	
0FFE04H	MPSC control register 02A	CR02A	R/W	○	○			xxx0xxxB	
0FFE05H	MPSC control register 03A	CR03A	R/W	○	○			Undefined	
0FFE06H	MPSC control register 04A	CR04A	R/W	○	○			0xx0000xB	
0FFE07H	MPSC control register 05A	CR05A	W		○			Undefined	
0FFE08H	MPSC control register 06A	CR06A	W	○	○			Undefined	
0FFE09H	MPSC control register 07A	CR07A	R/W	○	○			00H	
0FFE0AH	MPSC control register 08A	CR08A	R/W	○	○			FEH	
0FFE0BH	MPSC control register 09A	CR09A	R/W		○			00H	
0FFE0CH	MPSC control register 10A	CR10A	W		○			01100x00B	
0FFE0DH	MPSC control register 11A	CR11A	R/W	○	○			00H	
0FFE0EH	MPSC control register 12A	CR12A	W		○			Undefined	
0FFE10H	MPSC status register 0A	SR0A	R		○			04H	
0FFE11H	MPSC status register 1A	SR1A	R		○			x1xxx000B	
0FFE12H	MPSC status register 2A	SR2A	R		○			00000xxxB	
0FFE20H	MPSC send buffer B	TxB_B	W		○			Undefined	★
0FFE21H	MPSC receive buffer B	RxB_B	R		○			Undefined	★
0FFE22H	MPSC control register 00B	CR00B	W		○			xxxx0xxxB	
0FFE23H	MPSC control register 01B	CR01B	R/W	○	○			0x000x00B	
0FFE24H	MPSC control register 02B	CR02B	R/W	○	○			xxx0xxx0B	
0FFE25H	MPSC control register 03B	CR03B	R/W	○	○			Undefined	

Table 3-5 List of Special Function Registers (2/12)

Adress	Special Function Register Name	Symbol		R/W	Manipulatable Bit Units				After Reset
					1 bit	8-bit	18-bit	32-bit	
0FFE26H	MPSC control register 04B	CR04B		R/W	○	○			0××0000×B
0FFE27H	MPSC control register 05B	CR05B		W		○			Undefined
0FFE28H	MPSC control register 06B	CR06B		W		○			Undefined
0FFE29H	MPSC control register 07B	CR07B		R/W	○	○			00H
0FFE2AH	MPSC control register 08B	CR08B		R/W	○	○			FEH
0FFE2BH	MPSC control register 09B	CR09B		R/W	○	○			00H
0FFE2CH	MPSC control register 10B	CR10B		W		○			01100×00B
0FFE2DH	MPSC control register 11B	CR11B		R/W	○	○			00H
0FFE2EH	MPSC control register 12B	CR12B		W		○			Undefined
0FFE30H	MPSC status register 0B	SR0B		R	○	○			04H
0FFE31H	MPSC status register 1B	SR1B		R	○	○			×1×××000B
0FFE32H	MPSC status register 2B	SR2B		R		○			00000×××B
0FFE40H	Next block terminal counter 2 (low-order)	TCN2	TCN2L	W			○	○*	Undefined
0FFE42H	Next block terminal counter 2 (high-order)		TCN2H	R/W		○			Undefined
0FFE43H	Next block LDMA control register 2	LDMAN2		R/W	○	○			00H
0FFE44H	Next block tmemory address register 2 (low-order)	NMAR2	NMAR2L	R/W			○	○*	Undefined
0FFE46H	Next block tmemory address register 2 (high-order)		NMAR2H	R/W		○			Undefined
0FFE47H	MPSC receive end-of-block status register A	REOBS_A		R	○	○			0000×0×××B

•* When the SFR registerⒶis access using 32 bits in the configuration as shown in the example below,Ⓑof the upper 8 bits is not effected.

Example

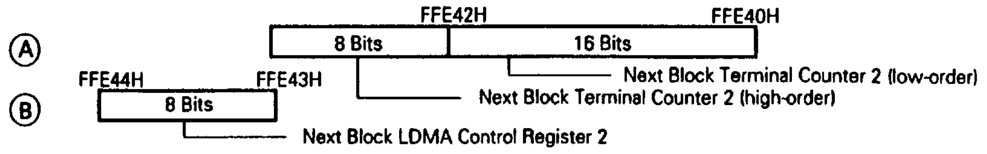

Ⓐ
FFE42H
FFE40H
8 Bits
16 Bits
Next Block Terminal Counter 2 (low-order)
Next Block Terminal Counter 2 (high-order)
Ⓑ
FFE44H
FFE43H
8 Bits
Next Block LDMA Control Register 2

Table 3-5 List of Special Function Registers (3/12)

Adress	Special Function Register Name	Symbol		R/W	Manipulatable Bit Units				After Reset
					1 bit	8-bit	18-bit	32-bit	
0FFE48H	Terminal counter save register 2 (low-level)	TCS2	TCS2L	R			○	○*	Undefined
0FFE4AH	Terminal counter save register 2 (high-level)		TCS2H	W		○			Undefined
0FFE4BH	LDMA control save register 2	LDMAS2		R		○			00H
0FFE4CH	LDMA mode register	LDMAM		R/W	○	○			00H
0FFE50H	Current block terminal counter 2 (low-order)	TCC2	TCC2L	R/W			○	○*	Undefined
0FFE52H	Current block terminal counter 2 (high-order)		TCC2H	R/W		○			Undefined
0FFE53H	LDMA control register 2	LDMAC2		R/W	○	○			00H
0FFE54H	Current block memory address register 2 (low-order)	MAR2	MAR2L	R/W			○	○	Undefined
0FFE56H	Current block memory address register 2 (high-order)		MAR2H	R/W		○	○		Undefined
0FFE5EH	Local bus programmable wait control register	LPWC		R/W	○	○			42H
0FFE5FH	Local bus refresh mode register	LRFM		R/W	○	○			47H
0FFE60H	Next block terminal counter 3 (low-order)	TCN3	TCN3L	R/W			○	○*	Undefined
0FFE62H	Next block terminal counter 3 (high-order)		TCN3H	R/W		○			Undefined
0FFE63H	Next block LDMA control register 3	LDMAN3		R/W	○	○			00H
0FFE64H	Next block tmemory address register 3 (low-order)	NMAR3	NMAR3L	R/W			○	○*	Undefined
0FFE66H	Next block tmemory address register 3 (high-order)		NMAR3H	R/W		○			Undefined
0FFE67H	MPSC receive end-of-block status register B	REOBS_B		R	○	○			000×0×××B

* When the SFR registerⒶis access using 32 bits in the configuration as shown in the example below,Ⓑof the upper 8 bits is not effected.

Example

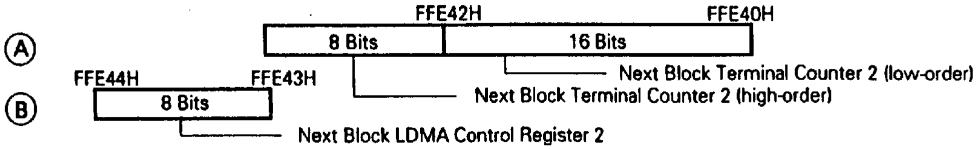

Ⓐ
FFE42H
FFE40H
8 Bits
16 Bits
Next Block Terminal Counter 2 (low-order)
Next Block Terminal Counter 2 (high-order)
Ⓑ
FFE44H
FFE43H
8 Bits
Next Block LDMA Control Register 2

Table 3-5 List of Special Function Registers (4/12)

Adress	Special Function Register Name	Symbol		R/W	Manipulatable Bit Units				After Reset
					1 bit	8-bit	18-bit	32-bit	
0FFE68H	Terminal counter save register 3 (low-level)	TC32	TCS3L	R			○	○*	Undefined
0FFE6AH	Terminal counter save register 3 (high-level)		TCS3H	R		○			Undefined
0FFE6BH	LDMA control save register 3	LDMAS3		R		○			00H
0FFE70H	Current block terminal counter 3 (low-order)	TCC3	TCC3L	R/W	○	○		○*	Undefined
0FFE72H	Current block terminal counter 3 (high-order)		TCC2H	R/W			○		Undefined
0FFE73H	LDMA control register 3	LDMAC3		R/W		○			00H
0FFE74H	Current block memory address register 3 (low-order)	MAR3	MAR3L	R/W	○	○		○	Undefined
0FFE76H	Current block memory address register 3 (high-order)		MAR3H	R/W			○		Undefined
0FFE80H	Next block terminal counter 4 (low-order)	TCN4	TCN4L	R/W		○	○	○*	Undefined
0FFE82H	Next block terminal counter 4 (high-order)		TCN4H	R/W	○	○			42H
0FFE83H	Next block LDMA control register 4	LDMAN4		R/W	○	○			00H
0FFE84H	Next block tmemory address register 4 (low-order)	NMAR4	NMAR4L	R/W			○	○*	Undefined
0FFE86H	Next block tmemory address register 4 (high-order)		NMAR4H	R/W		○			Undefined
0FFE87H	MPSC receive end-of-block status register A	TEOBS_B		R	○	○			00H
0FFE88H	Terminal counter save register 4 (low-order)	TCS4	TCS4L	R/W			○	○*	Undefined
0FFE8AH	Terminal counter save register 4 (high-order)		TCS4H	R/W		○			Undefined
0FFE8BH	LDMA control save register 4	LDMAS4		R	○	○			00H

* When the SFR registerⒶis access using 32 bits in the configuration as shown in the example below,Ⓑof the upper 8 bits is not effected.

Example

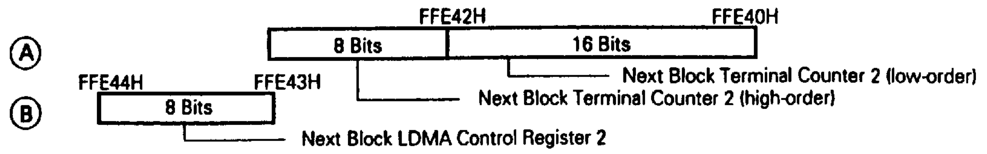

Ⓐ
FFE42H
FFE40H
8 Bits
16 Bits
Next Block Terminal Counter 2 (low-order)
Next Block Terminal Counter 2 (high-order)
Ⓑ
FFE44H
FFE43H
8 Bits
Next Block LDMA Control Register 2

Table 3-5 List of Special Function Registers (5/12)

Adress	Special Function Register Name	Symbol		R/W	Manipulatable Bit Units				After Reset
					1 bit	8-bit	18-bit	32-bit	
0FFE90H	Current block terminal counter 4 (low-order)	TCC4	TCC4L	R/W			○	○*	Undefined
0FFE92H	Current block terminal counter 4 (high-order)		TCC4H	R/W		○			Undefined
0FFE93H	LDMA control save register 4	LDMAC4		R/W	○	○			00H
0FFE94H	Current block t memory address register 4 (low-order)	MAR4	MAR4L	R/W			○	○	Undefined
0FFE96H	Current block memory address register 4 (high-order)		MAR4H	R/W		○	○		Undefined
0FFEA0H	Next block terminal counter 5 (low-order)	TCN5	TCN5L	R/W			○	○*	Undefined
0FFEA2H	Next block terminal counter 5 (high-order)		TCN5H	R/W		○			Undefined
0FFEA3H	Next block LDMA control register 5	LDMAN5		R/W	○	○			Undefined
0FFEA4H	Next block tmemory address register 5 (low-order)	NMAR5	NMAR5L	R/W			○	○*	Undefined
0FFEA6H	Next block memory address register 5 (high-order)		NMAR5H	R/W		○			Undefined
0FFEA7H	MPSC receive end-of-block status register B	TEOBS_B		R	○	○			00H
0FFEA8H	Terminal counter save register 5 (low-order)	TCS5	TCS5L	R/W			○	○*	Undefined
0FFEAAH	Terminal counter save register 5 (high-order)		TCS5H	R/W		○			Undefined
0FFEABH	LDMA control save register 5	LDMAS5		R		○			00H
0FFEB0H	Current block terminal counter 5 (low-order)	TCC5	TCC5L	R/W			○	○*	Undefined
0FFEB2H	Current block terminal counter 5 (high-order)		TCC5H	R/W		○			Undefined

* When the SFR registerⒶis access using 32 bits in the configuration as shown in the example below,Ⓑof the upper 8 bits is not effected.

Example

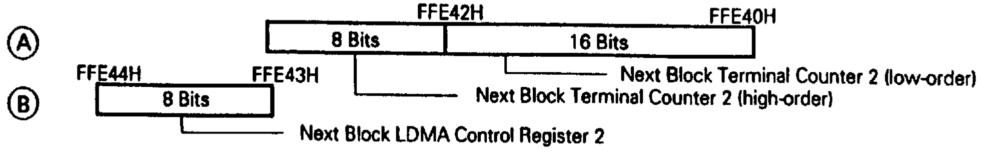

Ⓐ
FFE42H
FFE40H
8 Bits
16 Bits
Ⓑ
FFE44H
FFE43H
8 Bits
Next Block Terminal Counter 2 (low-order)
Next Block Terminal Counter 2 (high-order)
Next Block LDMA Control Register 2

Table 3-5 List of Special Function Registers (6/12)

Adress	Special Function Register Name	Symbol		R/W	Manipulatable Bit Units				After Reset
					1 bit	8-bit	18-bit	32-bit	
0FFEB3H	LDMA control register 5	LDMAC5		R/W	○	○			00H
0FFEB4H	Current block memory address register 5 (low-order)	MAR5	MAR5L	R/W			○	○	Undefined
0FFEB6H	Current block memory address register 5 (low-order)		MAR5L	R/W		○	○		Undefined
0FFEC0H	Interrupt mask flag register 0 (low-order)	MK0	MK0L	R/W	○	○	○		FFH
0FFEC1H	Interrupt mask flag register 0 (high-order)		MK0H	R/W	○	○			FEH
0FFEC2H	Interrupt mask flag register 1 (low-order)	MK1	MK1L	R/W	○	○	○		FEH
0FFEC3H	Interrupt mask flag register 1 (high-order)		MK1H	R/W	○	○			FFH
0FFEC4H	Inservice priority register	ISPR		R/W	○	○			00H
0FFEC5H	Interrupt mode control register	IMC		R/W		○			80H
0FFEC9H	Interrupt request control register 09	IC09		R/W	○	○			43H
0FFECAH	Interrupt request control register 10	IC10		R/W	○	○			43H
0FFECBH	Interrupt request control register 11	IC11		R/W	○	○			43H
0FFECCH	Interrupt request control register 12	IC12		R/W	○	○			43H
0FFECDH	Interrupt request control register 13	IC13		R/W	○	○			43H
0FFECEH	Interrupt request control register 14	IC14		R/W	○	○			43H
0FFECFH	Interrupt request control register 15	IC15		R/W	○	○			43H
0FFED0H	Interrupt request control register 16	IC16		R/W	○	○			43H
0FFED1H	Interrupt request control register 17	IC17		R/W	○	○			43H

Table 3-5 List of Special Function Registers (7/12)

Adress	Special Function Register Name	Symbol	R/W	Manipulatable Bit Units				After Reset
				1 bit	8-bit	18-bit	32-bit	
0FFED3H	Interrupt request control register 19	IC19	R/W	○	○			43H
0FFED4H	Interrupt request control register 20	IC20	R/W	○	○			43H
0FFED5H	Interrupt request control register 21	IC21	R/W	○	○			43H
0FFED6H	Interrupt request control register 22	IC22	R/W	○	○			43H
0FFED7H	Interrupt request control register 23	IC23	R/W	○	○			43H
0FFED8H	Interrupt request control register 24	IC24	R/W	○	○			43H
0FFED9H	Interrupt request control register 25	IC25	R/W	○	○			43H
0FFEDAH	Interrupt request control register 26	IC26	R/W	○	○			43H
0FFEDBH	Interrupt request control register 27	IC27	R/W	○	○			43H
0FFEDCH	Interrupt request control register 28	IC28	R/W	○	○			43H
0FFEDDH	Interrupt request control register 29	IC29	R/W	○	○			43H
0FFEDEH	Interrupt request control register 30	IC30	R/W	○	○			43H
0FFEDFH	Interrupt request control register 31	IC31	R/W	○	○			43H
0FFEE0H	Interrupt request control register 32	IC32	R/W	○	○			43H
0FFEE1H	Interrupt request control register 33	IC33	R/W	○	○			43H
0FFEE2H	Interrupt request control register 34	IC34	R/W	○	○			43H
0FFEE3H	Interrupt request control register 35	IC35	R/W	○	○			43H
0FFEE4H	Interrupt request control register 36	IC36	R/W	○	○			43H
0FFEE5H	Interrupt request control register 37	IC37	R/W	○	○			43H
0FFF00H	Port 0	P0	R	○	○			Undefined
0FFF01H	Port 1	P1	R/W	○	○			Undefined
0FFF02H	Port 2	P2	R/W	○	○			Undefined
0FFF03H	Port 3	P3	R/W	○	○			Undefined
0FFF04H	Port 4	P4	R/W	○	○			Undefined

Table 3-5 List of Special Function Registers (8/12)

Adress	Special Function Register Name	Symbol	R/W	Manipulatable Bit Units				After Reset
				1 bit	8-bit	18-bit	32-bit	
0FFF05H	Port 5	P5	R/W	○	○			Undefined
0FFF06H	Port 6	P6	R/W	○	○			Undefined
0FFF07H	Port 7	P7	R/W	○	○			Undefined
0FFF08H	Port 8	P8	R/W	○	○			Undefined
0FFF09H	Port 9	P9	R/W	○	○			Undefined
0FFF0CH	Port read control register	PRDC	R/W	○	○			00H
0FFF10H	Port 0 mode register	PM0	R/W	○	○			FFH
0FFF12H	Port 2 mode register	PM2	R/W	○	○			FFH
0FFF13H	Port 3 mode register	PM3	R/W	○	○			FFH
0FFF14H	Port 4 mode register	PM4	R/W	○	○			FFH
0FFF15H	Port 5 mode register	PM5	R/W	○	○			FFH
0FFF16H	Port 6 mode register	PM6	R/W	○	○			FFH
0FFF17H	Port 7 mode register	PM7	R/W	○	○			FFH
0FFF18H	Port 8 mode register	PM8	R/W	○	○			FFH
0FFF19H	Port 9 mode register	PM9	R/W	○	○			FFH
0FFF22H	Port 2 mode control register	PMC2	R/W	○	○			00H
0FFF23H	Port 3 mode control register	PMC3	R/W	○	○			00H
0FFF24H	Port 4 mode control register	PMC4	R/W	○	○			00H
0FFF25H	Port 5 mode control register	PMC5	R/W	○	○			00H
0FFF26H	Port 6 mode control register	PMC6	R/W	○	○			00H
0FFF27H	Port 7 mode control register	PMC7	R/W	○	○			00H
0FFF28H	Port 8 mode control register	PMC8	R/W	○	○			00H
0FFF29H	Port 9 mode control register	PMC9	R/W	○	○			00H

★ ★ ★

Table 3-5 List of Special Function Registers (9/12)

Adress	Special Function Register Name	Symbol		R/W	Manipulatable Bit Units				After Reset
					1 bit	8-bit	18-bit	32-bit	
0 F F F 3 0 H	Timer control register 0	TMC0	TMC0	R/W	○	○	○		00H
0 F F F 3 1 H	Timer control register 1		TMC1	R/W	○	○			00H
0 F F F 3 2 H	Timer output control register	TOC		R/W	○	○			Undefined
0 F F F 3 4 H	External Interrupt mode register 0	INTM	INTM0	R/W	○	○	○		00H
0 F F F 3 5 H	External Interrupt mode register 1		INTM1	R/W	○	○			00H
0 F F F 4 0 H	Timer register 0	TM0		R			○		00H
0 F F F 4 2 H	Timer register 1	TM1		R			○		00H
0 F F F 4 4 H	Timer register 3	TM2		R			○		00H
0 F F F 4 6 H	Timer register 3	TM3		R			○		00H
0 F F F 4 8 H	Timer capture register 00	CTOO		R/W			○		Undefined
0 F F F 4 A H	Timer capture register 01	CT01		R/W			○		Undefined
0 F F F 4 C H	Timer campare register 00	CM00		R/W			○		Undefined
0 F F F 4 E H	Timer campare register 01	CM01		R/W			○		Undefined
0 F F F 5 0 H	Timer capture register 10	CT10		R/W			○		Undefined
0 F F F 5 2 H	Timer campare register 10	CM10		R/W			○		Undefined
0 F F F 5 4 H	Timer campare register 11	CM11		R/W			○		Undefined
0 F F F 5 6 H	Timer campare register 20	CM20		R/W			○		Undefined
0 F F F 5 8 H	Timer campare register 30	CM30		R/W			○		Undefined
0 F F F 6 0 H	Watchdog timer mode register	WDM		R/W*	○	○			00H
0 F F F 6 4 H	MPSC send baud rate generator register A	TxBRG_A		R/W		○			Undefined
0 F F F 6 5 H	MPSC receive baud rate generator register A	RxBRG_A		R/W		○			Undefined
0 F F F 6 6 H	MPSC send pre-scalar register A	TPRS_A		R/W	○	○			07H
0 F F F 6 7 H	MPSC receivepre-scalar register A	RPRS_A		R/W	○	○			07H

* A write to the WDM register is possible only by the RSTWDT instruction (in 8-bit units only).

Table 3-5 List of Special Function Registers (10/12)

Adress	Special Function Register Name	Symbol		R/W	Manipulatable Bit Units				After Reset	
					1 bit	8-bit	18-bit	32-bit		
0FFF68H	DPLL minus correction value setting register A	DPLM_A		W		○			Undefined	
0FFF69H	DPLL plus correction value setting register A	DPLP_A		W		○			Undefined	
0FFF6AH	MPSC send baud rate generator register B	TXBRG_B		R/W	○	○			Undefined	
0FFF6BH	MPSC receive baud rate generator register B	RXBRG_B		R/W	○	○			Undefined	
0FFF6CH	MPSC send pre-scalar register B	TPRS_B		R/W	○	○			07H	★
0FFF6DH	MPSC receive pre-scalar register B	RPRS_B		R/W	○	○			07H	★
0FFF6EH	DPLL minus correction value setting register B	DPLM_B		W		○			Undefined	
0FFF6FH	DPLL plus correction value setting register B	DPLP_B		W		○			Undefined	
0FFF70H	UART send baud rate generator register 0	TXBRG0		R/W	○	○			Undefined	
0FFF71H	UART receive baud rate generator register 0	RXBRG0		R/W	○	○			Undefined	
0FFF72H	UART pre-scalar register 0	PRS0		R/W	○	○			00H	
0FFF73H	UART mode register 0	UARTM		R/W	○	○			00H	
0FFF74H	UART status register 0	UARTS		R/W*	○	○			20H	★
0FFF75H	UART send register 0	TXB0		W		○			Undefined	★
0FFF76H	UART receive register 0	RXB0		R		○			Undefined	★
0FFF80H	Terminal counter 0 (low-order)	TC0	TC0L	R/W			○	○	Undefined	
0FFF82H	Terminal counter 0 (high-order)		TC0H	R/W		○	○		Undefined	
0FFF84H	Terminal counter module register 0 (low-order)	TCM0	TCM0L	R/W			○	○	Undefined	
0FFF86H	Terminal counter module register 0 (high-order)		TCM0H	R/W		○	○		Undefined	
0FFF88H	DMA up/down counter 0 (low-order)	UDC0	UDC0L	R/W			○	○	Undefined	
0FFF8AH	DMA up/down counter 0 (high-order)		UDC0H	R/W		○	○		Undefined	
0FFF8CH	DMA compare counter 0 (low-order)	DCM0	DCM0L	R/W			○	○	Undefined	
0FFF8EH	DMA compare counter 0 (high-order)		DCM0H	R/W		○	○		Undefined	

* Some bits can only be read.

Table 3-5 List of Special Function Registers (11/12)

Adress	Special Function Register Name	Symbol		R/W	Manipulatable Bit Units				After Reset
					1 bit	8-bit	18-bit	32-bit	
0FFF90H	DMA memory address register 0 (low-order)	MAR0	MAR0L	R/W			○	○	Undefined
0FFF92H	DMA memory address register 0 (high-order)		MAR0H	R/W		○	○		Undefined
0FFF94H	DMA read/write pointer 0 (low-order)	DPTC0	DPTC0L	R/W			○	○	Undefined
0FFF96H	DMA read/write pointer 0 (high-order)		DPTC0H	R/W		○	○		Undefined
0FFF9CH	DAM mode register 0	DMAM0		R/W	○	○			E0H
0FFF9DH	DAM control register 0	DMAC0		R/W	○	○			00H
0FFF9EH	DAM status register 0	DMAS		R/W	○	○			00H
0FFFA0H	Terminal counter 1 (low-order)	TCM1	TCM1L	R/W			○	○	Undefined
0FFFA2H	Terminal counter 1 (high-order)		TCM1H	R/W		○	○		Undefined
0FFFA4H	Terminal counter module register 1 (low-order)	TCM1	TCM1L	R/W			○	○	Undefined
0FFFA6H	Terminal counter module register 1 (high-order)		TCM1H	R/W		○	○		Undefined
0FFFA8H	DMA up/down counter 1 (low-order)	UDC1	UDC1L	R/W			○	○	Undefined
0FFFAAH	DMA up/down counter 1 (high-order)		UDC1H	R/W		○	○		Undefined
0FFFACH	DMA compare register 1 (low-order)	DCM1	DCM1L	R/W			○	○	Undefined
0FFFAEH	DMA compare register 1 (high-order)		DCM1H	R/W		○	○		Undefined
0FFFB0H	DMA memory address register 1 (low-order)	MAR1	MAR1L	R/W			○	○	Undefined
0FFFB2H	DMA memory address register 1 (high-order)		MAR1H	R/W		○	○		Undefined
0FFFB4H	DMA read/write pointer 1 (low-order)	DPTC1	DPTC1L	R/W			○	○	Undefined
0FFFB6H	DMA read/write pointer 1 (high-order)		DPTC1H	R/W		○	○		Undefined
0FFFBCH	DMA mode register 1	DMAM1		R/W	○	○			E0H
0FFFBDH	DMA control register 1	DMAC1		R/W	○	○			00H

Table 3-5 List of Special Function Registers (12/12)

Adress	Special Function Register Name	Symbol	R/W	Manipulatable Bit Units				After Reset
				1 bit	8-bit	18-bit	32-bit	
0 F F F E 0 H	Software timer/counter	STC	R			○		Undefined
0 F F F E 2 H	Software timer/counter compare regisyer	STMC	R/W			○		FFFFH
0 F F F E 8 H	Programmable wait control register 0	PWC0	R/W	○	○			EAH
0 F F F E 9 H	Programmable wait control register 1	PWC1	R/W	○	○			AAH
0 F F F E A H	Memory block control register	MBC	R/W	○	○			F0H
0 F F F E C H	Refresh mode register	RFM	R/W	○	○			77H
0 F F F E E H	Standby control register	STBC	R/W*1	○	○			Maintained*2
0 F F F E F H	Proccessor control register	PRC	R/W					EEH

* 1. The standby control register SBF bit can be set (1) by an instruction, however, it cannot be cleaned (0). (W can only be "1".)

2. After power ON/Reset: 00H

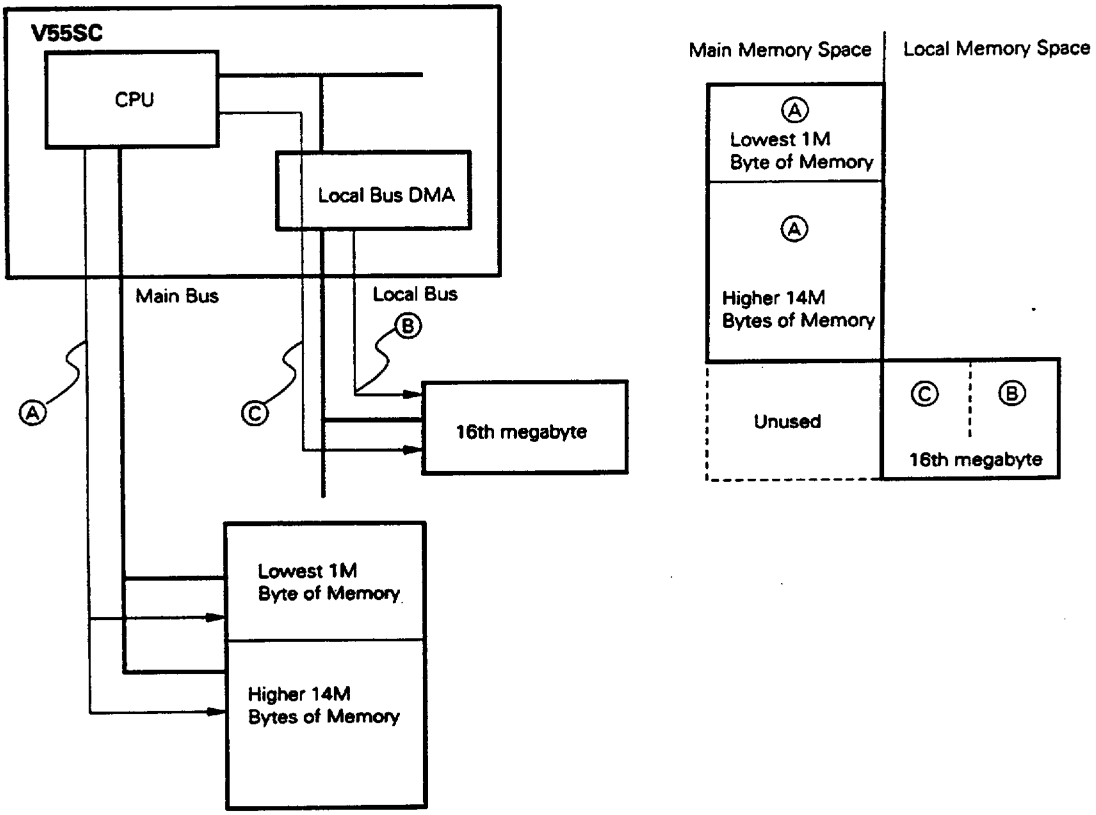
3.5.3 Local Memory Space

The local memory space is mapped onto the 16th megabyte (F00000H to FFFFFFH) of the expansion memory space and is fixed at 1M byte. Mapping of the local memory space to the expansion memory space is ON. The width of the data bus of the local bus can be switched between 8-bit and 16-bit by the local bus programmable wait control register (LPWC) LBW bit manipulation. When used as an 8-bit unit, data access from the CPU is only possible by the 8-bit memory manipulation instruction, the 16-bit memory manipulation instruction cannot be used.

The local memory space is mapped separate from the basic memory space and so only data access is possible, and it is possible to arrange large volume data such as receive data for MPSC.

Also, the bus control signal for the main bus for the main memory and the bus control signal for the local bus are controlled independently. Therefore, the conditions for suing the main bus by the CPU program processing are completely and independently controlled, and high-speed between MPSC and the local memory data transfer using the local bus DMA controller is possible.

Fig. 3-5 Local Memory Space



Remarks → is the access direction.

3.5.4 Vector Table Area

The 1K byte area 000000H to 0003FFH of the main memory space retains, in 256 vector portions (4 bytes to 1 vector), theinterrupt routine start address corresponding to aninterrupt request or break instruction.

At initialization, the V55 family exclusive on-chip peripheral and software interrupt vectors are reserved to be vectors 0 to 47. A vector address of the hardware interrupt except NMI can be changed using bits V0 and V1 of the interrupt mode control register (IMC).

Vector 0	(00000H)	:	Divide error
Vector 1	(00004H)	:	Single step
Vector 2	(00008H)	:	NMI instruction
Vector 3	(0000CH)	:	BRK 3 instruction
Vector 4	(00010H)	:	BRKV instruction
Vector 5	(00014H)	:	CHKIND instruction
Vector 6	(00018H)	:	Input/output instruction
Vector 7	(0001CH)	:	FPO instruction/exception trap

V1 = V0 = 0 :

Vector 8	(00020H)	:	INTWDT
Vector 9	(00024H)	:	INTP0
Vector 10	(00028H)	:	INTP1
Vector 11	(0002CH)	:	INTP2
Vector 12	(00030H)	:	INTP3
Vector 13	(00034H)	:	System reserved
Vector 14	(00038H)	:	INTCM00
Vector 15	(0003CH)	:	INTCM01
Vector 16	(00040H)	:	INTCM10
Vector 17	(00044H)	:	INTCM11
Vector 18	(00048H)	:	INTCM20
Vector 19	(0004CH)	:	INTCM30
Vector 20	(00050H)	:	INTD0 DMA#0
Vector 21	(00054H)	:	INTD1 DMA#1
Vector 22	(00058H)	:	INTD2 DMA#2
Vector 23	(0005CH)	:	INTD3 DMA#3
Vector 24	(00060H)	:	INTSP_A
Vector 25	(00064H)	:	INTSP_B
Vector 26	(00068H)	:	INTSR_A
Vector 27	(0006CH)	:	INTSR_B
Vector 28	(00070H)	:	INTST_A/INTD4 DMA#4
Vector 29	(00074H)	:	INTST_B/INTD5 DMA#5
Vector 30	(00078H)	:	INTES_A
Vector 31	(0007CH)	:	INTES_B
Vector 32	(00080H)	:	INTSIT
Vector 33	(00084H)	:	System reserved
Vector 34	(00088H)	:	System reserved
Vector 35	(0008CH)	:	INTSER0
Vector 36	(00090H)	:	INTSR0
Vector 37	(00094H)	:	INTST0
Vector 38	(00098H)	:	System reserved
Vector 39	(0009CH)	:	System reserved
Vector 40	(000A0H)	:	System reserved

	Vector 41	(000A4H)	:	System reserved
	Vector 42	(000A8H)	:	System reserved
	Vector 43	(000ACH)	:	System reserved
	Vector 44	(000B0H)	:	System reserved
	Vector 45	(000B4H)	:	System reserved
	Vector 46	(000B8H)	:	System reserved
	Vector 47	(000BCH)	:	System reserved

V1 = 0, V0 = 1 :

	Vector 72	(00120H)	:	INTWDT
	Vector 73	(00124H)	:	INTP0
	to	to		to
★	Vector 100	(00190H)	:	INTSR0
★	Vector 101	(00194H)	:	INTST0

V1 = 1, V0 = 0 :

	Vector 136	(00220H)	:	INTWDT
	Vector 137	(00224H)	:	INTP0
	to	to		to
★	Vector 164	(00290H)	:	INTSR0
★	Vector 165	(00294H)	:	INTST0

V1 = 1, V0 = 1 :

	Vector 200	(00320H)	:	INTWDT
	Vector 201	(00324H)	:	INTP0
	to	to		to
★	Vector 228	(00390H)	:	INTSR0
★	Vector 229	(00394H)	:	INTST0

3.6 REGISTER FILE SPACE

The register file space is shown in Figure 3-6.

The size of the register file space is 512 bytes. Macro service control register banks 0 and 1 are mapped in the register file space. A maximum of 16 register banks are assigned and can be used.

The register file space is separated from the main memory space as shown in Fig. 3-6 and the access method using software is different. To access the register file space, besides using the register manipulation instruction, a new "IRAM:" prefix instruction is added to the memory manipulation instruction.

When the IRAM: prefix instruction is added to the memory manipulation instruction, the CPU performs data access using the lower 9 bits of the memory address offset value as the register file address. At this time the segment register and physical address are not added and the external bus cycle is not activated.

Example

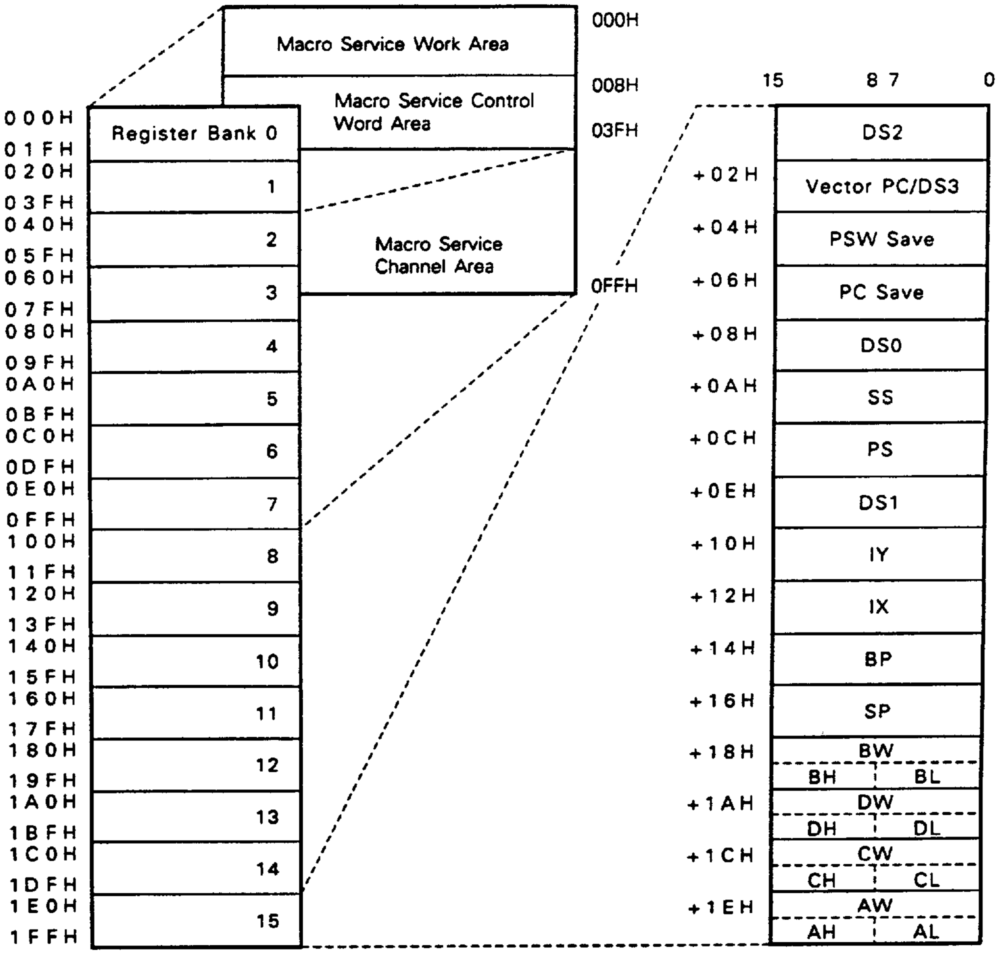
```
Label : MOV     IRAM  : [0024H], AW ....... ①
        MOV             [0056H], BW ....... ②
```

① Indicates the case when data is transferred to the register file space using a IRAM: prefix. The AW register value is stored in address 24H of theregister file.

② Indicates the case when data is transferred to the main memory.

Also, in the register file space, the macro service control word area (008H to 03FH), the macro service work area (000H to 007H), and the area used by the macro service channel (008H to 0FFH) overlap. If this work area does not use a specific macro service (BCDMA, SYNC, HDCMP) required by the work, it can be used as arbitrary data space.

Fig. 3-6 Register File Space



(Offset from the starting address of each register bank)

3.7 I/O SPACE

V55SC has a 64K byte I/O space.

The map of the I/O space is shown in Fig. 3-7.

The I/O space is accessed using the address/data bus and a control signal ($\overline{\text{IORD}}$, $\overline{\text{IOWR}}$, etc.).

0 are output from the unused upper 8 bits of the address bus.

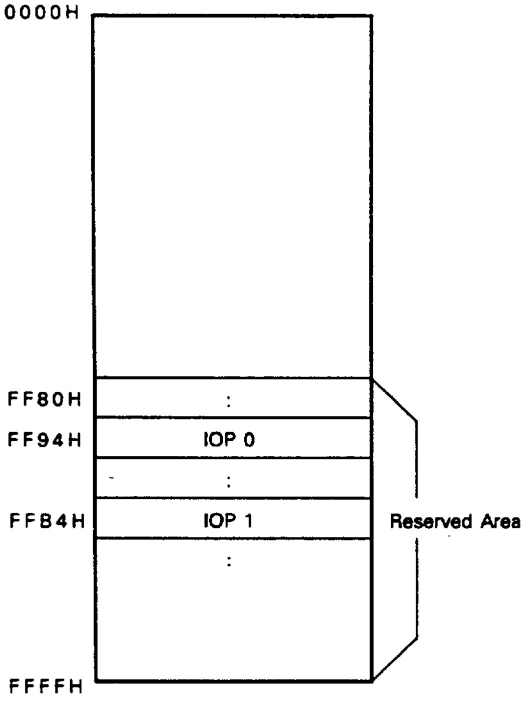
It is also possible to insert a wait cycle in the I/O cycle by software and the READY pin.

Also, FF80H to FFFFH of the I/O space is reserved area and is assigned to the two peripheral DMA input/output read/write pointers (IOP) which V55SC incorporates. The address for IOP0 is FF94Hand the address for IOP1 is FFB4H.

When the CPU executesan input/output instruction which uses the IOP address as an operand, the DMA controller takes the IOP contents asthe address value and reads data from or writes data toDMA controller transfer buffer and the IOP value isincremented (or decremented) automatically according tothe contents of the DMA control register.

It is alsopossible to reference the data written by the DMAcontroller using the input/output instruction.

Fig. 3-7 I/O Map (64K bytes)



Remarks IOPn corresponds to the DMA read/write pointer (DPTCn).

4. MAIN BUS CONTROL FUNCTION

For Pins for controlling the external main bus of the V55SC refer to 1.1.2 (1) **"Pin functions for main bus control"**.

For pins the serve a dual-function as port pins, the appropriate function must be selected using the port mode control register (PMCn).

4.1 BASIC BUS CYCLE

The V55SC main bus cycle operates basically in 3 clocks (when no wait cycle is inserted). Part of the address output signal and the data input/output signal are multiplexed, and address output and data input/output are performed in time multiplexing.

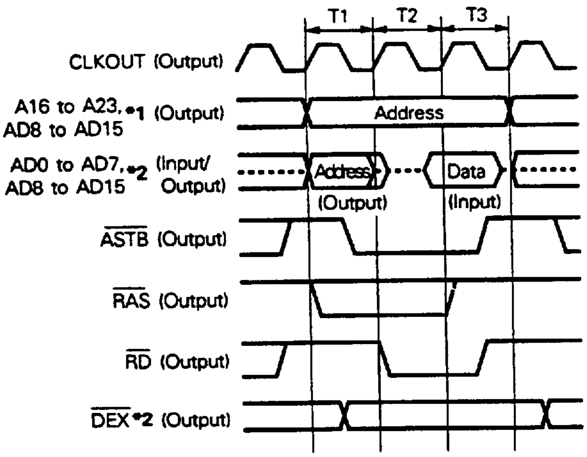
Address output is performed in the T1 cycle (first 1 clock) of the address cycle and data input/output is performed in the T2 and T3 cycles (remaining two clocks) of the data cycle.

The V55SC starts the following 7 bus cycles for the external main bus.

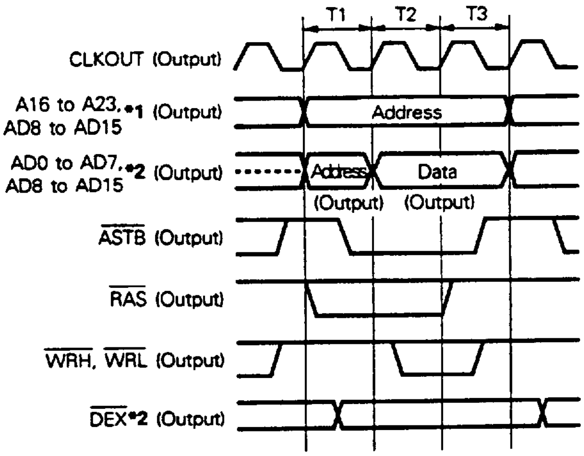
(1) Program code fetch cycle
(2) Memory read cycle
(3) Memory write cycle
(4) I/O read cycle
(5) I/O write cycle
(6) Refresh cycle
(7) DMA transfer cycle (external I/O to external memory)

An outline of the timing chart of each cycle is shown below.

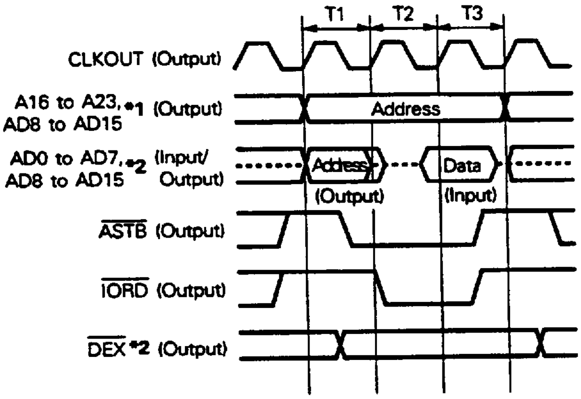
(1) Memory read cycle, program code fetch cycle



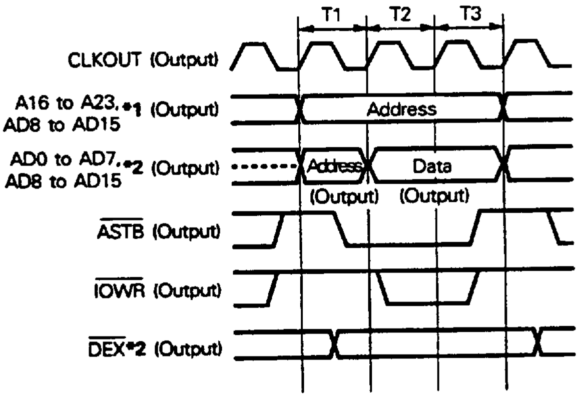
(2) Memory write cycle



(3) I/O read cycle



(4) I/O write cycle

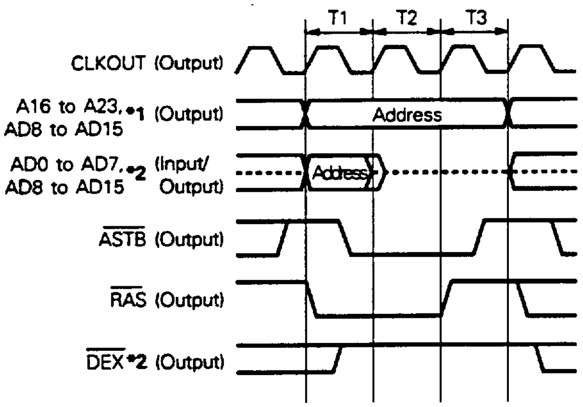


* 1. When external bus width is 8 bits.

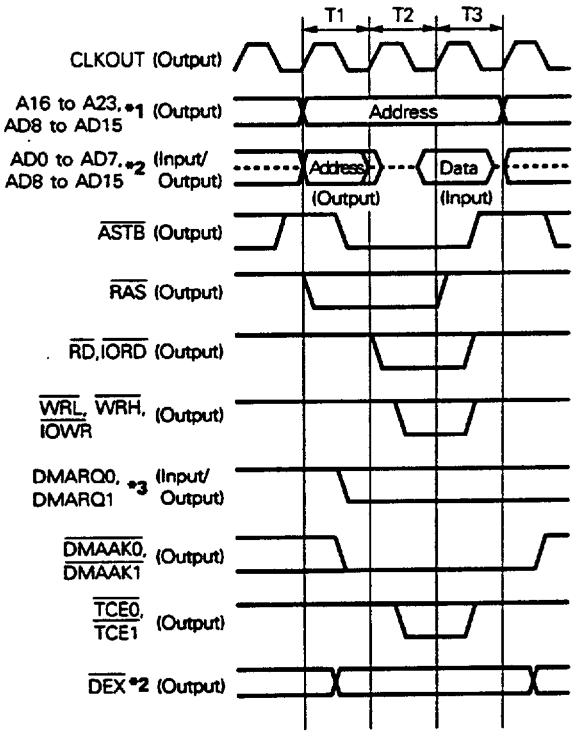
2. When external bus width is 16 bits.

Remarks The dotted line indicates high impedance.

(5) Refresh cycle



(6) DMA transfer cycle (external I/O to external memory)



* 1. When external bus width is 8 bits
  2. When external bus width is 16 bits
  3. In demand release mode

Remarks The dotted line indicates high impedance.

4.2 MAIN BUS WAIT

In the V55SC, the basic memory space (000000H to 0FFFFFH) is divided into variable memory sizes with a maximum of 5 blocks, and the portion of the basic memory space (100000H to FFFFFFH) that is not mapped by the expansion memory space is taken to be 1 block and a wait control is performed for each block.

The memory size of each block in the basic memory is specified by the memory block control register (MBC). The block size of the portion of the basic memory (100000H to FFFFFFH) which is not mapped by the expansion memory is 1 block and is fixed.

Fig. 4-1 shows the memory block configuration when the MBC register value is set to ADH.

Fig. 4-1 Memory Division Control

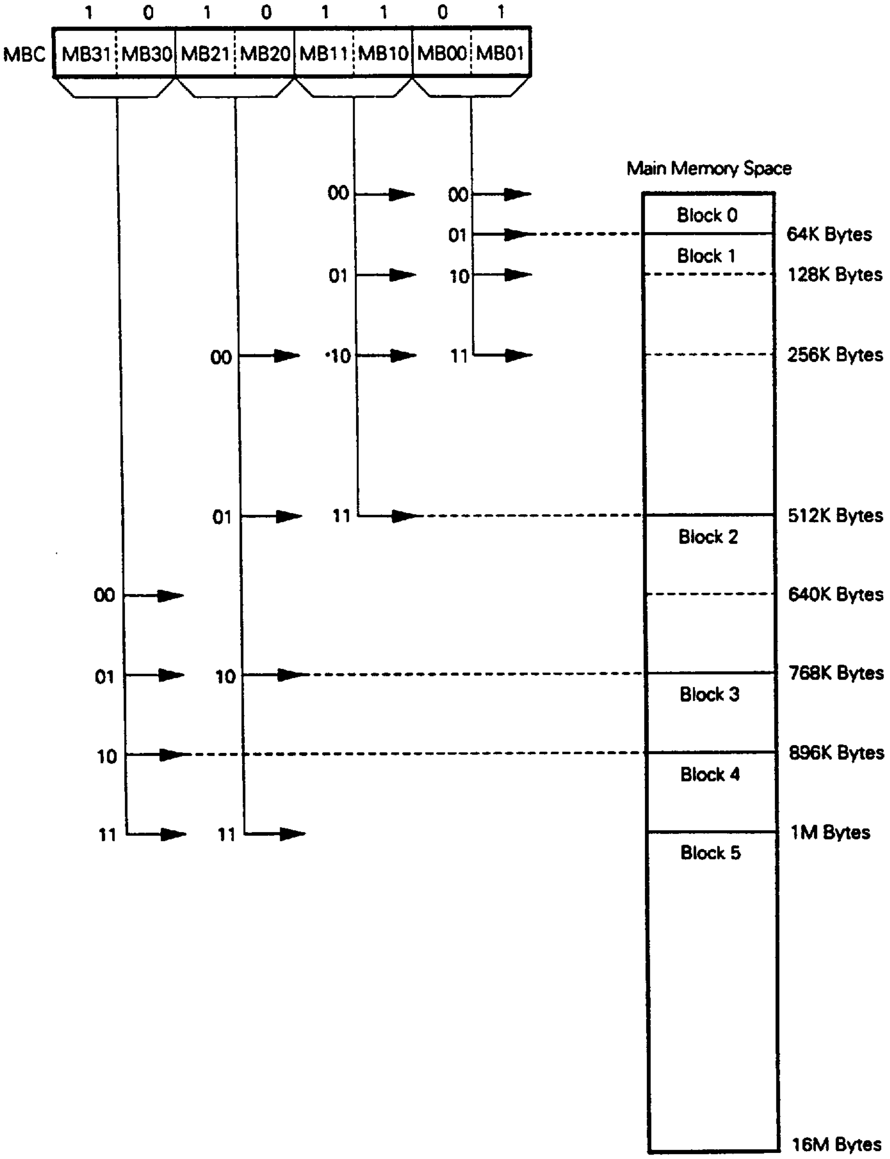


Fig. 4-2 Memory Wait Control

	7	6	5	4	3	2	1	0
	(BLOCK3)		(BLOCK2)		(BLOCK1)		(BLOCK0)	
PWC1	DW31	DW30	DW21	DW20	DW11	DW10	DW01	DW00

	7	6	5	4	3	2	1	0
	(BLOCK5)	(BLOCK2)	(I/O Space)		(BLOCK0)		(BLOCK4)	
PWC1	AW1	AW0	IOW1	IOW0	DW51	DW50	DW41	DW40

Data Wait (DW, IOW)

DWn1/IOW1	DWn0/IOW0	Wait State	READY Signal Influence
0	0	0	READY signal is ignored.
0	1	1	Additional control by READY signal is possible
1	0	2	
1	1	3	

Address Wait (AW)

AWn		Wait State
AW0	0	Not inserted (block 2)
	1	Inserted (Block 2)
AW1	0	Not inserted (Block 5)
	1	Inserted (Block 5)

5. LOCAL BUS CONTROL FUNCTION

The configuration of the local bus control pins is virtually the same as that of the main bus control pins, but the local bus control pins have no control pins equivalent to the $\overline{\text{DEX}}$ or $\overline{\text{BUSLOCK}}$ pin, When using the dual-function as the port pins (all pins otyher than the LHLD0 pin), it is necessary to select the corresponding function by the port mode control register (PMCn).

For the local bus control pins, see **1.1.2 (2) "Pin functions for local bus control"**.

Note After reset release, the control pins other than the LHLD0 pin function as input ports. Thus, when using the local bus, specify the local bus control gunction for each reset operation.

5.1 BASIC BUS CYCLE

The local bus cycle also operates basically in 3 clocks in the same way as the main bus (when no wait cycles are inserted). Part of the address output signal and the data input/output signal are multiplexed, and address output and data input/output are performed in time multiplexing. In addition, the operation timing of the other control sgnals is completely the same as that of the main bus.

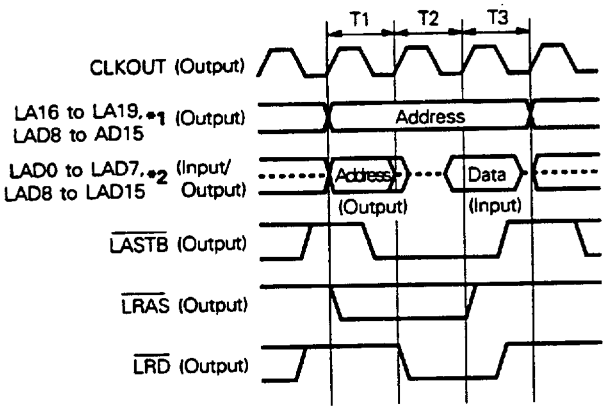
The V55SC starts the following 3 bus bybles for the local bus.

(1) Memory read cycle
(2) Memory write cycle
(3) Refresh cycle

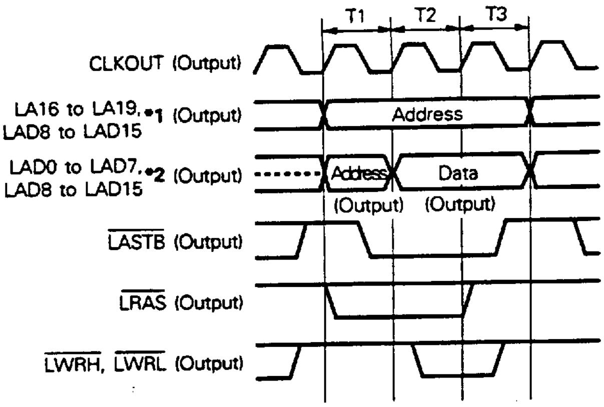
The memory read cycle and the DMA transfer cycle (MPSC ← external local memory), and the memory write cycle and the DMA transfer cycle (MPSC → external local memory) operate at completely the same timing in the external sense.

An outline of the timing chart of each cycle is shown below.

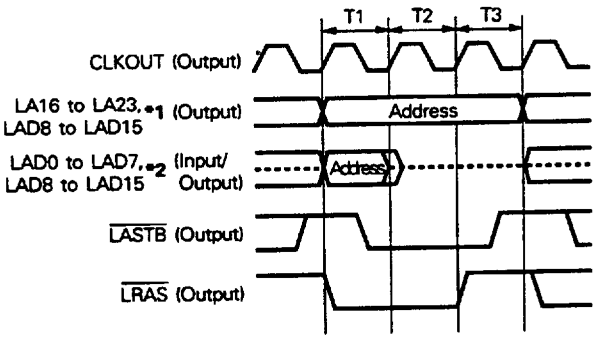
(1) Memory read cycle



(2) Memory write cycle



(3) Refresh cycle



* 1. When external bus width is 8 bits.
  2. When external bus width is 16 bits.

Remarks The dotted line indicates high impedance.

5.2 LOCAL BUS WAIT

The number of wait states inserted into the bus cycle can be controlled by the local bus programmable wait control register (LPWC). Also, it is possible to insert a wait state from the LRDY pin.

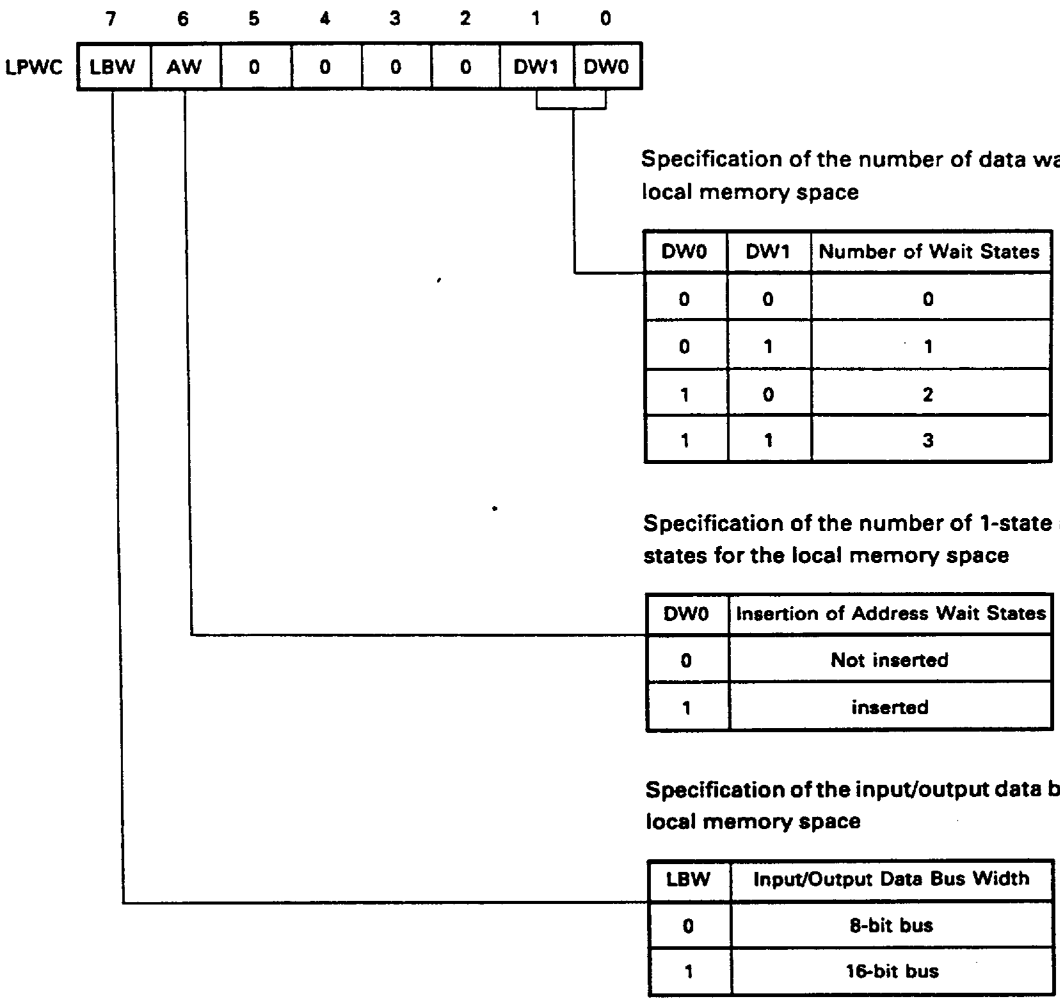


Fig. 5-1 Local Bus Control

	7	6	5	4	3	2	1	0
LPWC	LBW	AW	0	0	0	0	DW1	DW0

Specification of the number of data wait states for the local memory space

DW0	DW1	Number of Wait States
0	0	0
0	1	1
1	0	2
1	1	3

Specification of the number of 1-state address wait states for the local memory space

DW0	Insertion of Address Wait States
0	Not inserted
1	inserted

Specification of the input/output data bus width for the local memory space

LBW	Input/Output Data Bus Width
0	8-bit bus
1	16-bit bus

★ 6. INTERRUPT FUNCTION

The V55SC incorporates a strong interrupt controller (INTC) which specifies arbitrarily 4 levels of priority order for the total of 28, 23 internal and 5 external maskable hardware interrupt requests by the software and can control their processing. The interrupt controller controls multiple-interrupt servicing according to the programmable priority order. In addition, it is provided with three interrupt servicing modes; vectored interrupt function, macro service function, register bank switching function.

6.1 FEATURES

The interrupt function of the V55SC has the following features.

- Comprehensive service mode for interrupt request

• Vectored interrupt function	: Branch to interrupt service routine specified by vector table
• Register bank switching function	: High-speed interrupt response by automatic register bank switching
• Macro service function	: High-speed interrupt servicing by microprogram (firmware)

- 4-level programmable priority order control
- Multiple interrupt servicing control according to priority order
- Comprehensive macro service function (following 8 modes) in close contact with V55SC peripheral hardware

EVTCNT	: Event count processing
BLKTRS	: Data transfer between special function register and external memory buffer
BLKTRS-C	: Data transfer (with transfer data detection function) between special function register and external memory buffer
DTACMP	: Special function register status detection
DTADIF	: Time measurement by timer capture function
BCDMA	: Local bus DMA controller descript information save/update
SYNC	: MPSC (SYNC mode) character detection
HDCMP	: MPSC (HDLC mode) address detection (up to 8 kinds of address possible)

- 5 external interrupt request inputs (NMI, INTP0 to INTP3)
- Maskable interrupt requests can be masked separately

Table 6-1 shows a list of interrupt causes.

Table 6-1 List of Interrupt Causes (1/2)

Interrupt Classification	Default Priority	Interrupt Request Signal	Interrupt Cause			Default Vector Table Number	Vectored Address	Macro Service	Register Bank Switching	Macro Service Control Word Address
			Interrupt Request Control Register	Generating Source	Generating Unit					
Nonmaskable	1	NMI	—	NMI pin inputExternsl	External	2	00008H	No	No	—
	2	INTWDT		Watchdog timer overflow	WDT	8	00x20H	No	No	
Maskable	3	INTP0	IC9	INTP0 pin input	External	9	00x24H	Yes	Yes	012H
	4	INTP1	IC10	INTP1 pin input		10	00x28H	Yes	Yes	014H
	5	INTP2	IC11	INTP2 pin input		11	00x2CH	Yes	Yes	016H
	6	INTP3	IC12	INTP3 pin input		12	00x30H	Yes	Yes	018H
	7	INTCM00	IC14	CM00 coincidence detection	Timer	14	00x38H	Yes	Yes	01CH
	8	INTCM01	IC15	CM01 coincidence detection		15	00x40H	Yes	Yes	01EH
	9	INTCM10	IC16	CM10 coincidence detection		16	00x44H	Yes	Yes	020H
	10	INTCM11	IC17	CM11 coincidence detection		17	00x48H	Yes	Yes	022H
	11	INTCM20	IC18	CM20 coincidence detection		18	00x4CH	Yes	Yes	024H
	12	INTCM30	IC19	CM30 coincidence detection		19	00x50H	Yes	Yes	026H
	13	INTD0	IC20	DMA channel 0	DMA	20	00x54H	Yes	Yes	028H
	14	INTD1	IC21	DMA channel 1		21	00x58H	Yes	Yes	02AH
	15	INTD2	IC22	LDMA channel 2	LDMA	22	00x5CH	Yes	Yes	02CH
	16	INTD3	IC23	LDMA channel 3		23	00x60H	Yes	Yes	02EH
	17	INTSP_A	IC24	MPSC_A special receive condition	MPSC	24	00x64H	Yes	Yes	030H
	18	INTSP_B	IC25	MPSC_B special receive condition		25	00x68H	Yes	Yes	032H
	19	INTSR_A	IC26	MPSC_A receive		26	00x40H	Yes	Yes	034H
	20	INTSR_B	IC27	MPSC_B receive		27	00x6CH	Yes	Yes	036H
	21	INTSR_A /INTD4	IC28	MPSC_A send	MPSC	28	00x70H	Yes	Yes	038H
				LDMA channel 4	LDMA					

Table 6-1 List of Interrupt Causes (1/2)

Interrupt Classification	Default Priority	Interrupt Request Signal	Interrupt Cause			Default Vector Table Number	Vectored Address	Macro Service	Register Bank Switching	Macro Service Control Word Address
			Interrupt Request Control Register	Generating Source	Generating Unit					
Maskable	22	INTSR_B /INTD5	IC29	MPSC_B send	MPSC	29	00×74H	No	No	03AH
				LDMA channel 5	LDMA					
	23	INTES_A	IC30	MPSC_A external/status	MPSC	30	00×78H	Yes	Yes	03CH
	24	INTES_B	IC31	MPSC_B external/status		31	00×7CH	Yes	Yes	03EH
	25	INTSIT	IC32	STM coincidence detection	SIT	32	00×80H	No	Yes	—
	26	INTSER0	IC35	UART receive error	·UART	35	00×8CH	No	Yes	—
	27	INTSR0	IC36	UART receive		36	00×90H	Yes	Yes	008H
	28	INTST0	IC37	UART send		37	00×94H	Yes	Yes	00AH
Software	—	—	—	Divide error	—	0	00000H	No	No	—
				BRK flag (single-step)		1	00004H	No	No	
				BRK3 instruction		3	0000CH	No	No	
				BRKV instruction		4	00010H	No	No	
				CHKIND instruction		5	00014H	No	No	
				Input/output instruction (IBRK flag)		6	00018H	No	No	
				BRK imm8		×	00xxxH	No	No	
				BRKCS instruction		—	—	No	No	
Exception trap				FP0 instruction Exchange trap		7	0001CH	No	No	

Remarks Portions indicated with "x" are values that can change.

7. DMA FUNCTION (DMA CONTROLLER)

The V55SC has an on-chip 2-channel general-purpose DMA controller which controls execution of DMA transfer based on a DMA request using on-chip peripheral hardware (UART, timer), external DMARQ pins or a software trigger.

7.1 FEATURES ★

- 2 independent DMA channels
- 4 kinds of transfer modes
  - Single-transfer mode ... 1 DMA transfer cycle is executedfor 1 DMA request
  - Demand release mode ... DMA transfer cycles are consecutively executed while a DMA request is active
  - Single-step mode ... After generation of a DMA request, DMA transfer cycles and CPU bus cycles are alternately executed
  - Burst mode ... After generation of a DMA request, DMA transfer cycles are consecutively executed
- 4 kinds of operating modes
  - Normal transfer mode (I/O to memory transfer)... Control of DMA transfer between SFR (internal I/O) or I/O and memory
  - Intelligent DMA mode (ring buffer system) ... Control of DMA transfer to ring buffer
  - Next address specification mode ... Consecutive transfers are possible between different transfer buffers
  - Memory-memory transfer mode ... 2 bus cycles are started for 1 DMA transfer cycle and a memory to memory transfer is executed
- 3 clocks/1 bus cycle (no wait)
- Kinds of transfer
  - External I/O to memory ... 1 bus cycle/1 DMA transfer cycle
  - SFR (internal I/O) to memory ... 1 bus cycle/1 DMA transfer cycle
  - Memory to memory (memory includes SFR) ... 2 bus cycle/1 DMA transfer cycle
- Selectable byte transfer/word transfer
- Selectable transfer address increment/decrement/no change
- DMA transfer end signal output pins ($\overline{TCE0}$, $\overline{TCE1}$) ($\overline{TC00}$, $\overline{TC01}$)
- 24-bit long memory address register (MARn)
- 21-bit long terminal counter (TCn)
- DMA request signal pins (DMARQ0, DMARQ1: Dual-function as port pins P90 to P91)
  DMA acknowledge signal pins ($\overline{DMAAK0}$, $\overline{DMAAK1}$)

Table 7-1 Correspondence Table of Operating Modes and Transfer Modes

Operating mode		Transfer Target	Usable Transfer Mode			
			①*	②*	③*	④*
Normal transfer mode (I/O to memory transfer)		I/O ↔ memory	Usable	Usable	Usable	Usable
Intelligent DMA mode		I/O (SFR) → memory	Usable	Usable	Unusable	Unusable
Next address specification mode		I/O (SFR) ↔ memory	Usable	Usable	Unusable	Unusable
Memory to memory transfer mode	(Normal end)	Memory ↔ memory	Usable	Usable	Usable	Usable
	(Repeat)	Memory ↔ memory	Usable	Usable	Unusable	Unusable

* ① Single transfer mode
② Demande release mode
③ Single-step mode
④ Burst mode

8. LOCAL BUS DMA FUNCTION (LOCAL BUS DMA CONTROLLER)

Besides the DMA function (DMAC), the V55SC has an on-chip 4-channel local bus DMA controller.

The local bus DMA controller executes DMA transfer between the MPSC and the local memory connected to the local bus based on a DMA request from the MPSC.

8.1 FEATURES

- 4 independent DMA channels
- DMA transfer using the single transfer mode
- 2 types of operating modes
  - Block chain operating mode
  - Normal transfer mode
- 3 clocks/1 bus cycle
- Byte transfer
- Selectable transfer address increment/decrement
- 20-bit long current block memory address register (MARn)
- 20-bit long current block terminal counter (TCCn)

8.2 LOCAL BUS DMA CONTROLLER (LDMAC) CONFIGURATION

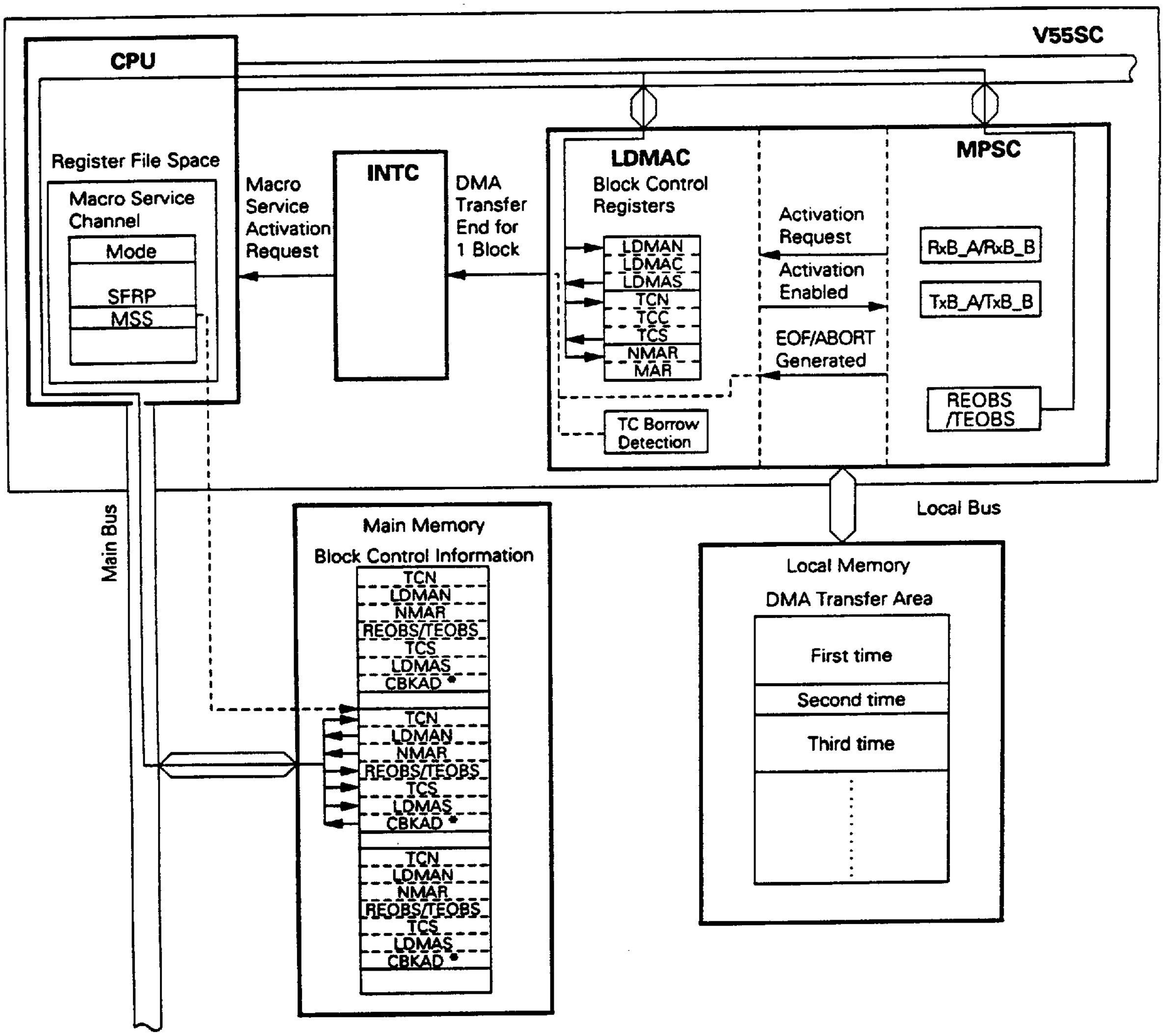
The local bus DMA controller has 4 independent DMA channels and controls DMA transfer between the external memory connected to the local bus (dedicated MPSC memory connection bus) and MPSC channel A or B.

MPSC channel A or B (HDLC, SYNC, ASYNC modes) is a dedicated DMA transfer channel.

The transfer memory for which address are directly specifiable is 1M bytes of the local memory space.

The maximum number of the DMA transferable bytes is 1M. The entire configuration of the local bus DMA controller is shown in Fig. 8-1.

Fig. 8-1 Entire Configuration of the Local Bus DMA Controller



- CBKAD: Current block segment address

9. UART FUNCTION

The V55SC, for the serial communication function, has 1 channel for the start-step format exclusive communication unit (UART), and has 2 channels for the communication unit (MPSC) which supports 3 protocol (refer to **10. MPSC Function**). The ASYNC communication protocol for UART and MPSC is the same, however, the method of setting commands and some functions are different.

Here, the UART will be described.

9.1 FEATURES

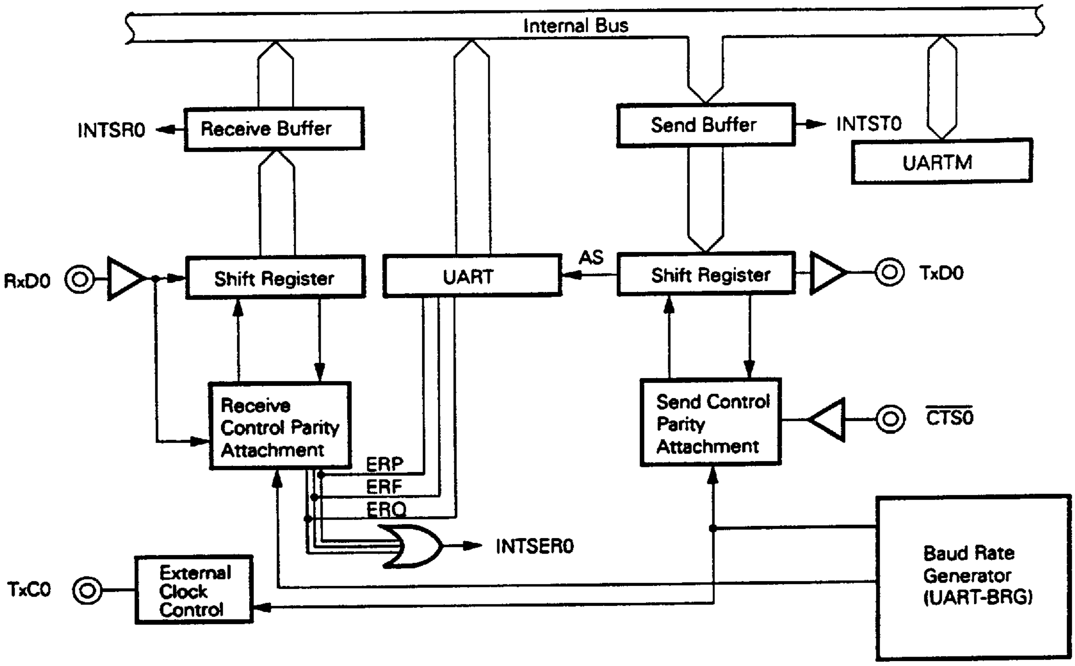
- Transfer speed 50 to 390 Kbps (system clock ϕ = 12.5 MHz)
- Full-duplex operation possible
- Exclusive (for sending and for receiving) on-chip baud rate generators
- Wake-up function
- 0 parity function
- Parity error detection
- Framing error detection
- Overrun error detection
- UART exclusive interrupt sources (3 kinds)
  - UART receive error interrupts (INTSER0)
  - UART receive complete interrupt (INTSR0)
  - UART send complete interrupt (INTST0)
- Macro service function
  - UART receive complete interrupt (INTSR0)
  - UART send complete interrupt (INTST0)

9.2 UART CONFIGURATION

The V55SC has a UART with an on-chip exclusive baud rate generator. A block diagram of the UART is shown in Fig. 9-1.

The UART is comprised of a serial data input (RxD0), serial data output (TxD0), serial clock output (TxC0), pins for the sendable state control input (CTS0) and a transfer control section, send/receive 8-bit shift register, UART mode register (UARTM), UART status register (UARTS), send buffer (TxB0), receive buffer (RxB0), and a send/receive baud rate generator. There are shift registers and buffers for sending and receiving and so sending and receiving are performed independently (full-duplex operation possible).

Fig. 9-1 UART Block Diagram



10. MPSC FUNCTION

The MPSC in the V55SC is a multi-function communication unit which can be used for a wide range of serial data communication.

The basic function of the MPSC is to convert, using a set format, parallel data to serial data and serial data to parallel data.

In normal serial data communication, regulations and procedure of how to format the data, or what configuration to use in sending and receiving data is called the data communication protocol, and are standardized. The V55SC supports the following three kinds of protocol.

(1) Bit oriented protocol (HDLC mode)
(2) Character oriented protocol (SYNC mode)
(3) Start-stop synchronization format (ASYNC mode)

In the MPSC, there are various circuits assembled to make it possible to use these efficiently.

10.1 FEATURES

- Multi protocol operation: HDLC mode, SYNC mode, ASYNC mode
- High-speed transfer: 2.5 Mbps
- Full-duplex operation
- 2 channels on-chip
- Transmitter: Double buffer
- Receiver: 4-way buffer
- Data format: NRZ, NRZI
- DMA transfer possible using the on-chip local bus DMA controller
  DMA request signals × 4 (send DMA × 2, receive DMA × 2)
- On-chip baud rate generators (send × 2, receive × 2)
- On-chip DPLL circuit

The features of each protocol are as follows.

(1) Bit oriented protocol (HDLC mode)

- Address field detection
- Abort send
- Zero insert/remove
- Flag insert/remove
- 16-bit FCS generation/check
  CRC format [Generating function: $X^{16} + X^{12} + X^{5} + 1$]
- Overrun error detection
- Short frame detection
- CRC error detection
- EOF flag/LDMAC error detection
- Abort detection
- Underline detection
- $\overline{\text{CTS}}$ pin state change detection
- Hunt detection
- $\overline{\text{DCD}}$ pin state change detection
- All Sent detection
- Mark idle detection

(2) Character oriented protocol (SYNC mode)

- Mono-sync/Bi-sync protocol supported
- SYNC character lengt: 1 or 2 characters
- SYNC character bit length: 6, 8, 16 bits
- Character bit length: 5, 6, 7, 8 bits
- Parity attachment/check
- 16-bit BCS generation/check
  CRC format [Generating function: $X^{16} + X^{12} + X^5 + 1$]
  [Generating function: $X^{16} + X^{15} + X^2 + 1$]
- Overrun error detection
- Parity error detection
- CRC error detection
- EOF flag/LDMAC error detection
- Underline detection
- $\overline{\text{CTS}}$ pin state change detection
- Hunt detection
- $\overline{\text{DCD}}$ pin state change detection
- Mark idle detection

(3) Start-stop synchronization method (ASYNC mode)

- Framing error detection
- Overrrun error detection
- Parity error detection
- EOF flag/LDMAC error detection
- Break send detection
- $\overline{\text{CTS}}$ pin state change detection
- $\overline{\text{DCD}}$ pin state change detection
- All Send detection
- Clock rate: Baud rate ×1, ×16, ×32, ×64
- Break send

10.2 SUMMARY

The V55SC is prepared with the UART and MPSC as the serial communication function. The MPSC has three types of operating modes, the HDLC mode, SYNC mode, and ASYNC mode.

The communication protocol of the ASYNC mode is the same for both the UART and the MPSC, however, the method of setting commands and some of the functions are different.

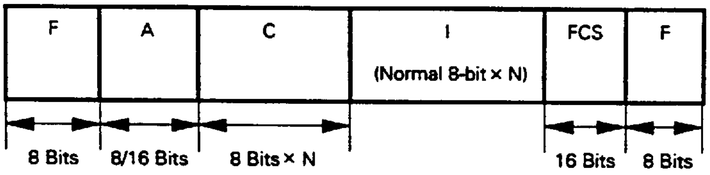
10.2.1 Bit Oriented Protocol (HDLC Mode)

The HDLC mode is an optimized subset which uses HDLC (High Level Data Link Control Procedure) by the microprocessor based on the function specification of the BOP mode of NEC's communication LST, μPD72001.

In the HDLC mode, it is possible to support the SDLC (Synchronous Data Link Control) protocol recommended by IBM. However, it does not support the SDLC-Loop mode.

In the HDLC protocol, sending data using an arbitrary bit length is allowed. Data is send with the format shown in **Fig. 10-1 Frame configuration.**

Fig. 10-1 Frame Configuration



Flag sequence (F) : Header ... Start flag (01111110)
: Tail ... Conclusion flag (01111110)
If the frame continues, the start flag and the conclusion flag may overlap.

Address field (A) : Address detection is performed.

Control field (C) : Handled as transparent single data.

Information field (I) : An arbitrary bit length is allowed however, it is generally 8 bits.

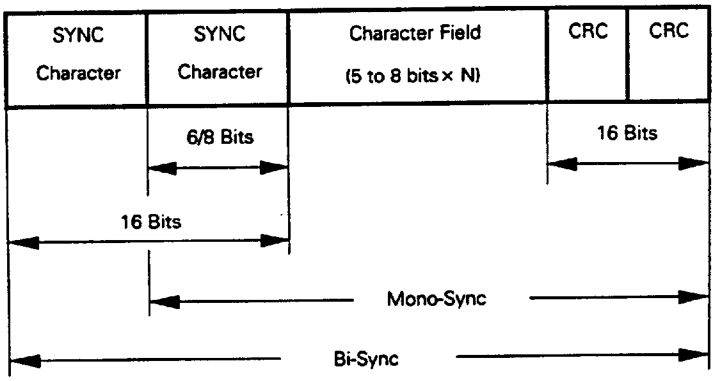
Frame check sequence (FCS) : CRC calculation result.

10.2.2 Character Oriented Protocol (SYNC Mode)

The SYNC mode is an optimized subset which uses SYNC with a single-chip microcomputer based on the function specifications of the COP mode of the μPD72001.

The SYNC mode is a protocol which is executed using a synchronized format and supports Mono-Sync format and Bi-Sync format as the character synchronization format.

Fig. 10-2 Data Format



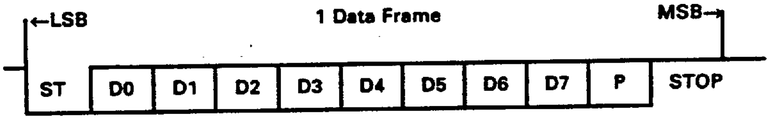
10.2.3 Start-stop Synchronized Format (ASYNC Mode)

The ASYNC mode is an optimized subset which used ASYNC with the microprocessor based on the function specifications of the ASYNC mode of the μPD72001.

In the ASYNC mode, on the send side, a start bit, stop bit, and if necessary a parity bit are attached to each data block to be sent. If there is no characters to be sent, the mark state is set ("H" continues).

The receiving side knows where the data block starts when it detects the start bit, and it receives the continuing serial data based on it. Receiving the serial data ends when thestop bit is detected. The data block between start bitsis recognized as one characters.

Fig. 10-3 Data Format



Start bit	(ST)	: 1 bit
Character bits	(D0 to D7)	: 1 to 8 bits (send)
Parity bit	(P)	: Even/Odd/Not added
Stop bit	(STOP)	: 1/1.5/2 bits

11. TIMER FUNCTION

The V55SC timer unit can be used as an interval timer, free running timer, and event counter.

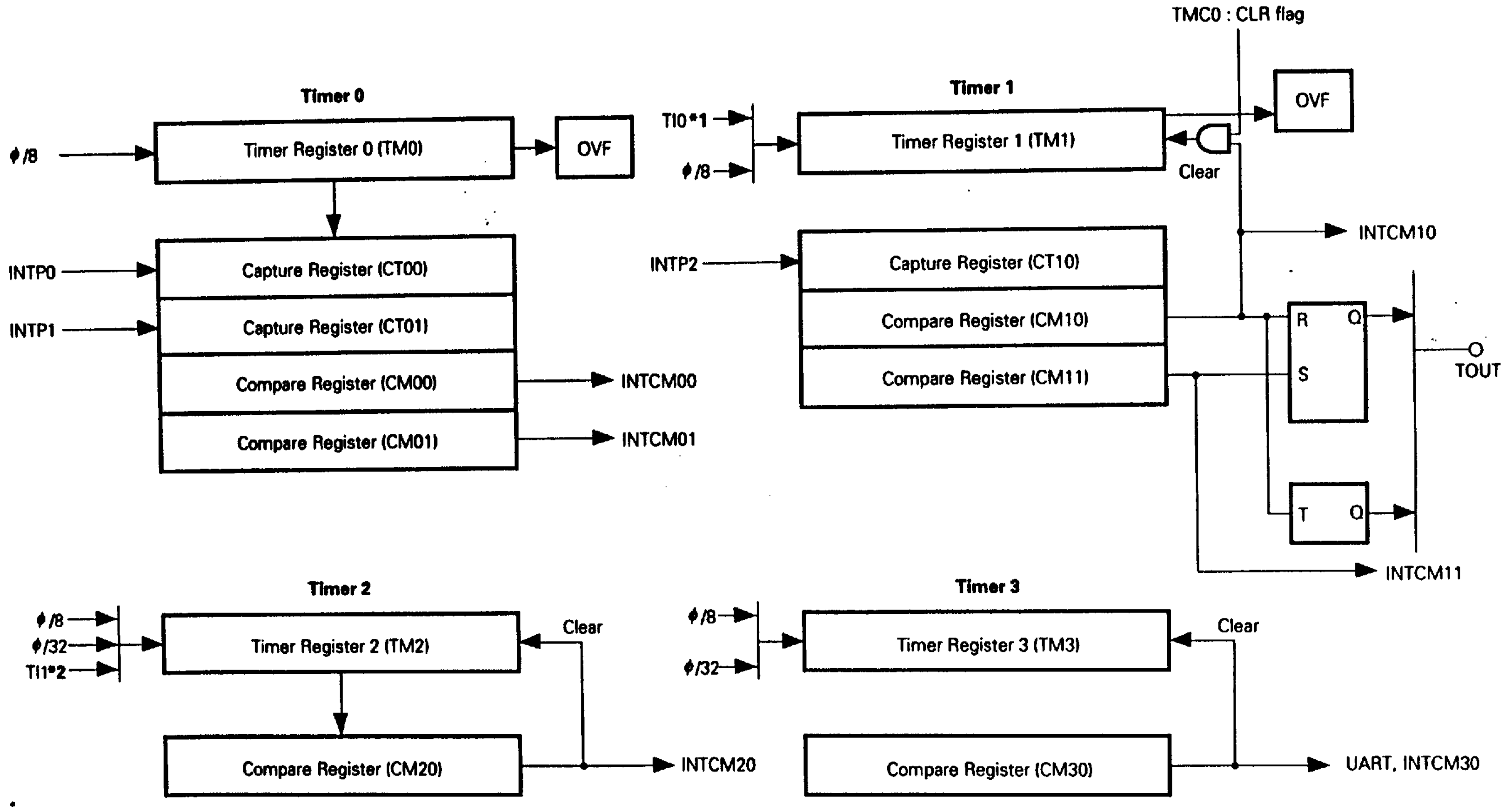
11.1 FEATURES

- 16-bit timer × 4
- 2 types of count clock sources
  - Selectable system clock scaled output ($\phi/8$, $\phi/32$: system clock ϕ)
  - Selectable external input pulse from pins T10 and T11
- External count output signal (TOUT output)
- 6 types of timer unit exclusive interrupt sources (INTCM00, INTCM01, INTCM10, INTCM11, INTCM20, INTCM30)

11.2 TIMER UNIT CONFIGURATION

The timer unit configuration is shown in Fig. 11-1.

Fig. 11-1 Timer Unit Configuration



Remarks ϕ: System Clock

* 1. TI0 is also used as INTP2.
2. TI1 is also used as INTP3.

12. WATCHDOG TIMER FUNCTION

The watchdog timer is a function which prevents program upset and dead-lock.

12.1 FEATURES

- 3 overflow times can be set (10.4, 41.9, 167.7 ms: System clock ϕ = 12.5 MHz)
- Output pin directly connectable with the $\overline{\text{RESET}}$ pin ($\overline{\text{WDTOUT}}$ pin)

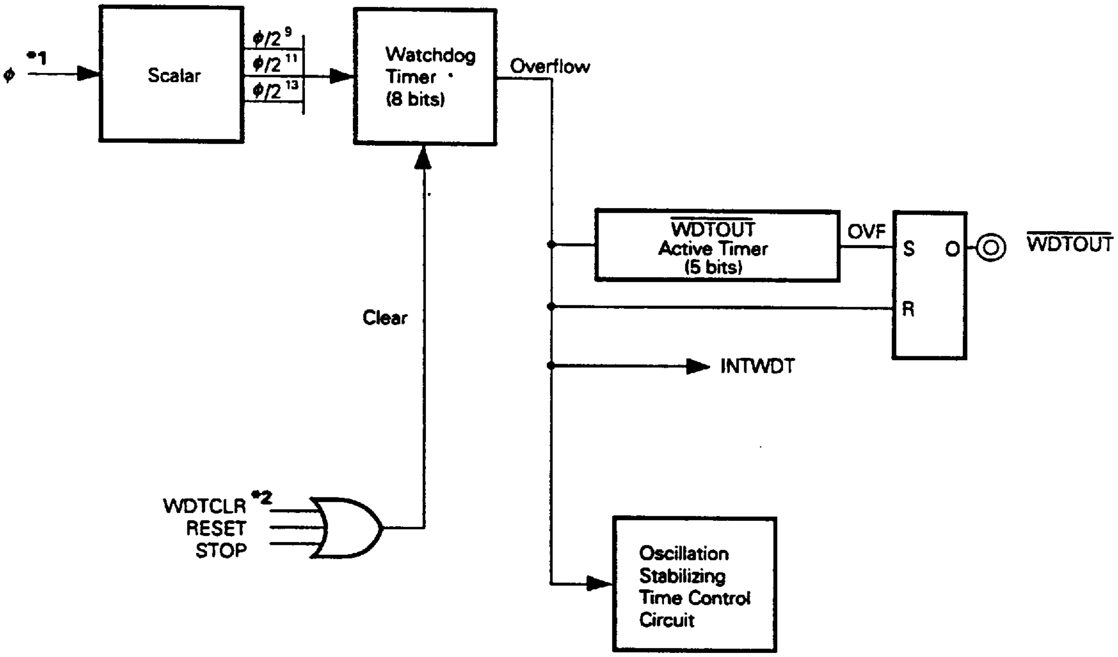
12.2 WATCHDOG TIMER CONFIGURATION AND OPERATION

When a watchdog timer interrupt is not generated, it checks that the program or system operates normally. To use the watchdog timer function, it is necessary to input an instruction clearing the watchdog timer (count start) for each program module.

If the instruction that clears the watchdog timer is not executed within the set time, the watchdog timer overflows and together with a watchdog timer interrupt (INTWDT) being generated, the output at pin WDTOUT is low, notifying thatthe program is abnormal.

The configuration of the watchdog timer is shown in Fig. 12-1.

Fig. 12-1 Watchdog Timer Configuration



ϕ *1
Scalar
$\phi/2^9$
$\phi/2^{11}$
$\phi/2^{13}$
Watchdog Timer (8 bits)
Overflow
$\overline{\text{WDTOUT}}$ Active Timer (5 bits)
OVF
S
O
R
$\overline{\text{WDTOUT}}$
INTWDT
Clear
WDTCLR *2
RESET
STOP
Oscillation Stabilizing Time Control Circuit

* 1. ϕ: System clock

2. WDTCLR: Clears the watchdog timer by instruction

13. STANDBY FUNCTION

In the V55SC there are three modes the control the operating clock and they serve as a low power consuming standby function. Transition can be made to any standby mode by a special instruction.

13.1 HALT MODE

This mode causes the CPU operating clock to stop.

By setting the HALT mode in the empty time of the CPU, the power consumption of the entire system is decreased. The HALT mode is set by executing the HALT mode instruction.

In the HALT mode, the CPU clock stops and execution of the program stops, however, before that all of the contents of the registers and on-chip RAM is saved.

If the HALT mode instruction is executed during DMA transfer by the DMA controller (DMAC), moving to the HALT mode is held until the end of the transfer bus cycle corresponding to the first DMA request.

13.2 STOP MODE

This mode stops oscillation.

This mode is valid when the entire application system is stopped, and it uses very little power. By executing the STOP mode instruction , the STOP mode is set. In the STOP mode, all of the clocks stop. Program execution is stopped, however, before that the contents of all of the registers and on-chip RAM are saved.

Note During local bus DMA transfer by the local bus DMA controller, or during the local bus refresh operation, operation stops when the STOP instruction is executed. Therefore, there is a possibility that the contents of the memory connected to the local bus may be corrupted. Therefore, beforeexecuting the STOP instruction, it is necessary to prohibit the operation of the local bus DMA controller and the refresh operation.

13.3 IDLE MODE

With the crystal oscillation circuit still oscillating, this modes stops all of the operating clocks except the local bus DMA controller and standby control circuit. The IDLE mode is set by executing the IDLE instruction. Even if the IDLE instruction is executed during a transmit data transfer to MPSC by the local bus DMA controller, the local bus DMA controller continues the operation and transmits the transmit data to MPSC.

In contrast to the STOP mode, it is not necessary to secure the oscillation stabilizing time of the oscillator and so, it is possible to quickly move to normal operation. Program execution is stopped however, before that, the contents of all registers and the on-chip RAM is saved.

14. CLOCK GENERATION CIRCUIT

The clock generation circuit is a circuit which supplies the various clocks of the CPU and peripheral hardware, and controls the CPU operating mode.

14.1 CLOCK GENERATION CIRCUIT CONFIGURATION AND OPERATION

The configuration of the clock generation circuit is shown in Fig. 14-1.

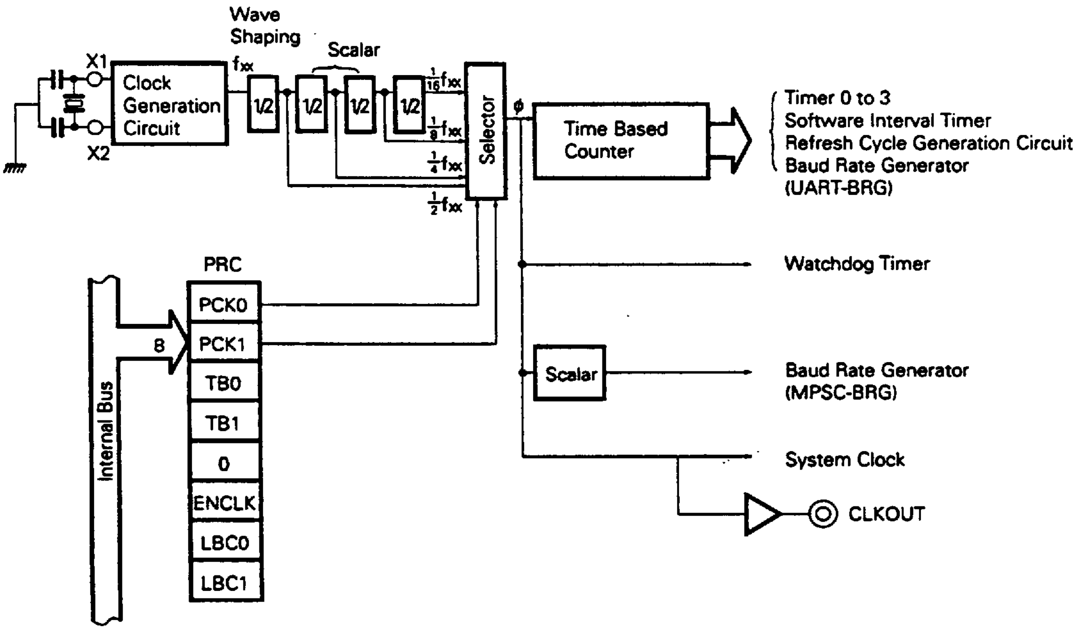
The reference clock of the clock generation circuit oscillates using a crystal oscillator or ceramic oscillator connected to pins X1 and X2. The output of the clock generation circuit is scaled wave shaping (1/2 scale) and the scale ratio isselected and it is used as the system clock ϕ.

The scale ration of the system clock o/ is specified by bits PCK1 and PCK0 of the processor control register (PRC), and the oscillation frequency (fxx) can be selected to be 1/2, 1/4, 1/8 or 1/16.

By reducing the speed of the system clock ϕ, the consumed current is reduced and operation is possible for a long time even if the voltage is lowered by the battery driven system.

Also, it is possible to input an external clock. In that case, input the clock signal at pin X1, and leave open pin X2.

Fig. 14-1 Clock Generation Circuit



fxx : Oscillation Frequency
φ : System clock
PRC: Processor Control Register

In the V55SC, the scaler (time based counter TBC) which scales the internal system clock ϕ is common for each timer unit.

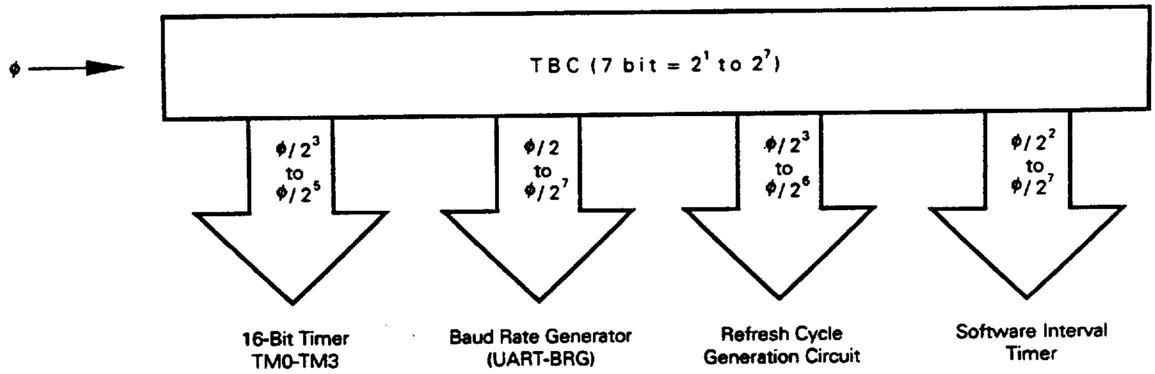
The TBC cannot be read from or written to by an instruction.

The tap output of the TBC (2^n scaled clock) is supplied as the count clock to the following units.

(1) Timer 0 to timer 3
(2) Baud rate generator (UART-BRG)
(3) Refresh cycle generation circuit
(4) Software interval timer

The TBC is cleared to 00H by only the reset input, after which it is always incremented. The operation of the TBC stops in the STOP mode and IDLE mode. The configuration of the TBC is shown in Fig. 14-2.

Fig. 14-2 Scaler (Time Based Counter TBC) Configuration



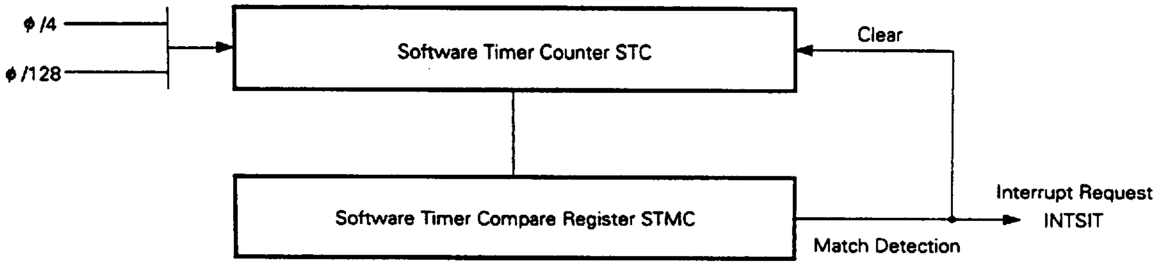
15. SOFTWARE INTERVAL TIMER FUNCTION

The V55SC has an on-chip 16-bit software interval timer which is used for the software timer function and the clock function.

15.1 SOFTWARE INTERVAL TIMER CONFIGURATION

The configuration of the software interval timer is shown in Fig. 15-1.

Fig. 15-1 Software Interval timer Configuration


ϕ/4
ϕ/128
Software Timer Counter STC
Clear
Software Timer Compare Register STMC
Interrupt Request
INTSIT
Match Detection

16. INSTRUCTION SET

The instruction set of the V55SC has upward compatibility with the instruction sets of V20 and V30 (native mode) and V25 and V35.

16.1 INSTRUCTION ADDED TO V20 AND V30 OR V25 AND V35

Instructions that have been added to those of V20 and V30 or V25 and V35, and instructions that expand the applicable range are as follows.

- **IRAM :** ....**Register File Space Access Override Prefix instruction (Added)**
  By adding this to the memory manipulation instruction, it separates the main memory space and access data as a 512 byte register file.
  The address of the register file is the lower 9 bits of the offset value of the memory specification operand.

Mnemonic	Operand
IRAM :	None

- **DS2 :** ....**Expansion Segment Override Prefix Instruction (Added)**
  When data is accessed in the 16M byte expansion memory space, this is attached to a memory operand which is capable of a segment override prefix, and specifies the segment register DS2.

Mnemonic	Operand
DS2 :	None

- **DS3 :** ....**Expansion Segment Override Prefix Instruction (Added)**
  When data is accessed in the 16M byte expansion memory space, this is attached to a memory operand which is capable of a segment override prefix, and specifies the segment register DS3.

Mnemonic	Operand
DS3 :	None

- **MOV DS2, reg16, mem32** ....**Instruction to Transfer from a 32-bit Memory to a 16-Bit Register and DS3 (Added)**
  Transfer the lower 16 bits of the 32-bit memory specified by the third operand to the 16-bit register specified by the second operand, and transfers the upper 16 bits to the expansion segment register DS2.

Mnemonic	Operand		
MOV	DS2	reg16	mem32

- **MOV DS3, reg16, mem32 ....Instruction to Transfer from a 32-bit Memory to a 16-Bit Register and DS3 (Added)**

Transfers the lower 16 bits of the 32-bit memory specified by the third operand to the 16-bit register specified by the second operand, and transfers the upper 16 bits to the expansion segment register DS3.

Mnemonic	Operand		
MOV	DS3	reg16	mem32

- **PUSH DS2**
  PUSH DS3
  PUSH VPC*

....Expansion Segment Register Stack Manipulation Instruction (Added)

(SP-1, SP-2) ← DS2/DS3/VPC
SP ← SP-2

The expansion segment register (DS2 or DS3/VPC) specified by the operand is saved in a stack.

Mnemonic	Operand
PUSH	DS2
PUSH	DS3
PUSH	VPC

*** VPC means vector PC. The VPC has the same address as DS3.**

- **POP DS2**
  POP DS3
  POP VPC

....Expansion Segment Register Stack Manipulation Instruction (Added)

DS2/DS3/VPC ← (SP + 1, SP)
SP ← SP + 2

The contents of the stack are transferred to the expansion segment register (DS2 or DS3/VPC) specified by the operand.

An interrupt cannot be placed between this instruction and the next instruction.

Mnemonic	Operand
POP	DS2
POP	DS3
POP	VPC

- **RSTWDT imm8, imm8** **....Watchdog Timer Manipulation Instruction (Added)**

Compares the instruction code of the 4th byte and the instruction code of the 3rd byte and manipulates the watchdog timer. This instruction used a special instruction code configuration (4 bytes) in order that the register is not written over during program upset, and it does not write unless the operands of the 3rd byte and 4th byte have a one's compliment relationship.

Mnemonic	Operand	
RSTWDT	imm8	imm8'

- **IDLE** **....Instruction to Move to the IDLE Mode Sets (Added)**

The IDLE mode of the standby mode.

Mnemonic	Operand
IDLE	None

- **BTCLRL sfrl, imm3, short-label....Conditioned Branch Instruction (Added)**

When the bit indicated by specified register imm3 in the lower 256 bytes (0FFE00H to 0FFEFFH) of the special function register space is set (1), that bit is reset (0), and the short-label value is added to the current PC value and loaded to the PC.

Branching is possible in the address from - 128 to +127 in the segment where this instruction is placed.

This instruction is paired with the BTCLR instruction, and when the BTCLR instruction targets the upper 240 bytes (0FFF00H to 0FFFEFH) of the special function register space, the BTCLRL instruction the lower 256 bytes of the same space. All other functions are the same as the BTCLR instruction.

Mnemonic	Operand		
BTCLRL	sfrl	imm3	short-label

- **MOV xsreg, reg16**
  MOV VPC, reg16 **....Instruction to Transfer Data from a Register to an Expansion Segment Register or Vector PC (Added)**

The expansion segment register (DS2, DS3/VPC) is added and so a register which can be specified for the operand is added.

The contents of a 16-bit register specified by the second operand is transferred to the expansion segment register (DS2 or DS3/VPC) specified by the first operand.

Mnemonic	Operand	
MOV	DS2	reg16
MOV	DS3	reg16
MOV	VPC	reg16

- **MOV xsreg, mem16**
  MOV VPC, mem16

....Instruction to Transfer Data from a Memory to an Expansion Segment Register or Vector PC (Added)

The expansion segment register (DS2, DS3/VPC) is added and so a register which can be specified for the operand is added.

The countents of a 16-bit memory specified by the second operand is transferred to the expansion segment register (DS2 or DS3/VPC) specified by the first operand.

Mnemonic	Operand	
MOV	DS2	mem16
MOV	DS3	mem16
MOV	VPC	mem16

- **MOV reg16, xsreg**
  MOV reg16, VP

....Instruction to Transfer Data from a Registe to an Expansion Segment Register or Vector PC (Added)

The expansion segment register (DS2, DS3/VPC) is added and so a register which can be specified for the operand is added.

The contents of a 16-bit register specified by the second operand is transferred to the expansion segment register (DS2 or DS3/VPC) specified by the first operand.

Mnemonic	Operand	
MOV	reg16	DS2
MOV	reg16	DS3
MOV	reg16	VPC

- **MOV mem16, xsreg**
  MOV mem16, VPC

....Instruction to Transfer Data from a Memory to an Expansion Segment Register or Vector PC (Added)

The expansion segment register (DS2, DS3/VPC) is added and so a register which can be specified for the operand is added.

The contents of a 16-bit memory specified by the second operand is transferred to the expansion segment register (DS2, DS3/VPC) specified by the first operand.

Mnemonic	Operand	
MOV	mem16	DS2
MOV	mem16	DS3
MOV	mem16	VPC

- **BSCH reg**
  BSCH mem

....Bit Manipulation Instruction (Added)

Searches for "1" starting from bit 0 up to bit 7 or bit 15 of a reg/mem, and returns the first searched bit number to CL. If there results of the search is that there are no "1s" (reg/mem = 0), the "ZF" (Zero Flag: Bit 6 of PSW) is set (1).

If the search result is that there was a "1", "ZF" is reset (0).

Mnemonic	Operand
BSCH	reg/mem

- **QHOUT imm16**

....Quene Manipulation Instruction to Release the Queue Header Block (Added)

This releases a block queued in the queue header and stores the segment address in the parameter table (register file). If the specified queue is empty, no manipulation is performed on the queue and "ZF" (Zero Flag: Bit 6 of PSW) is set (1). Otherwise, "ZF" is reset (0).

Mnemonic	Operand
QHOU	imm16

- **QOUT imm16**

....Queue Manipulation Instruction to Release an Arbitrary Queue Block (Added)

Releases blocks shown in the queue parameter table.

If the queue specified by the parameter table is empty, no manipulation is performed on the queue, and "ZF" (Zero Flag: Bit 6 of PSW) is set (1). Otherwise, "ZF" is reset (0).

Mnemonic	Operand
QOU	imm16

- **QTIN imm16**

....Queue Manipulation Instruction for Queuing a Block in a Queue (Added)

Queues-in a block indicated in the parameter table at the end of a queue.

Mnemonic	Operand
QTIN	imm16

- **MOVSPB reg16**

....SS, SP Transfer Instruction (Expanded)

Transfer the SS and SP values of the current (before switching) register bank to the SS and SP of the register bank switched to and indicated by the lower 4 bits of the contents of the 16-bit register entered in the operand. In comparison with V25, the number of switched register banks has increased.

(This expands the application range for the same instruction of the V25 and V35. This instruction is added to the instructions of the V20 and V30.)

Mnemonic	Operand
MOVSPB	reg16

The following 5 instruction have been added to the instruction of the V20 and V30 (native mode).

- BTCLR sfr, imm3, short-label ....Special Function Register Test Instruction (The upper 240 bytes of the special function register space (0FFF00H to 0FFFEFH) are targeted.)
- MOVSPA ....Instruction which transfers the SS and SP values of the register bank before switching, to the SS and SP of the current (after switching) register bank.
- RETRBI ....Register bank return instruction
- FINT ....Instruction which indicates to the interrupt controller the end of the interrupt process.
- STOP ....Instruction which moves to the STOP mode.

16.2 INSTRUCTION SET OPERATIONS

Table 16-1 Operand Type Legend

Identifier	Description
reg	8/16-bit general register (Destination register for an instruction which used two 8/16-bit general registers.)
reg'	Source register for an instruction which uses two 8/16-bit general registers.
reg8	8-bit general register (Destination register for an instruction which uses two 8-bit general registers.)
reg8'	Source register for an instruction which uses two 8-bit general registers.
reg16	16-bit general register (Destination register for an instruction which uses two 16-bit general registers.)
reg16'	Source register for an instruction which uses two 16-bit general registers.
mem	8/16-bit memory address
mem8	8-bit memory address
mem16	16-bit memory address
mem32	32-bit memory address
sfr	Special function register location: FFF00H to FFFEFH
sfrl	Special function register location: FFE00H to FFEFFH
dmem	16-bit direct memory address
imm	8/16-bit immediate data
imm3	3-bit immediate data
imm4	4-bit immediate data
imm8	8-bit immediate data
imm8'	8-bit immediate date (one's compliment of imm8)
imm16	16-bit immediate data
acc	Accumulator AW or AL
sreg	Segment register
xsreg	Expansion segment register
src-table	256-byte translation table name
src-block	Source block name addressed by register IX
dst-block	Destination block name addressed by register IY
src-string	Source string name addressed by register IX
dst-string	Destination string name addressed by register IY
near-proc	Procedure in the current program segment
far-proc	Procedure in a different program segment
near-label	Label of the current program segment
short-label	Label of the range of bytes -128 to +127 from the end of an instruction
far-label	Label of a different program segment
regptr16	16-bit general register having the offset of the call address of the current program segment
memptr16	16-bit memory address having the offset of the call address of the current program segment
memptr32	32-bit memory address having the offset and segment data of the call address of a different program segment
pop-value	Number of bytes to be deleted from the stack (0 to 64K,normally an even number)
repeat	Repeat prefix instruction
fp-op	Immediate value which identifies the instruction code of anexternal floating point arithmetic chip
IRAM :	Override prefix instruction for accessing the register file space
R	%Register set (AW, BW, CW, DW, SP, BP, IX, IY)
src-spec	DS2, DS1, DS0, SS, PS or the segment name/group name assumed in the DS2, DS1, DS0, SS, PS
Dst1-spec	DS3, DS1 or the segment name/group name assumed in DS3 and DS1
Dst2-spec	DS2, DS0 or the segment name/group name assumed in DS2 and DS0
Seg-spec	Arbitrary segment register (DS1, DS0, SS, PS) or the segment name/ group name assumed in the segment register
Xseg-spec	Arbitrary expansion segment register (DS3, DS2) or the segment name/group name assumed in an expansion segment register
[ ], ( )	Can be abbreviated
or,/	Or

Table 16-2 Instruction Word Format Legend

Identifier	Description
W	Word/byte specification bit (1: word, 0: byte). When s = 1, the sign expansion byte data is a 16-bit operand even if W=1.
reg, reg'	8/16-bit general register specification bits (000 to 111)
mod, mem	Memory address specification bits (mod: 00 to 10, mem: 000 to 111)
(disp-low)	Option 16-bit displacement of low-order bytes
(disp-high)	Option 16-bit displacement of high-order bytes
disp-low	6-bit displacement of low-order bytes for PC relative addition
disp-high	16-bit displacement of high-order bytes for PC relative addition
imm3	3-bit immediate data
imm4	4-bit immediate data
imm8, imm8'	8-bit immediate data
imm16-low	Low-order bytes of 16-bit immediate data
imm16-high	High-order bytes of 16-bit immediate data
addr-low	Low-order bytes of a 16-bit direct address
addr-high	High-order bytes of a 16-bit direct address
sreg	Segment register specification bits (00 to 11)
xsreg	Expansion segment register specification bits (10 to 11)
s	Sign expansion specification bit (1: sign expansion, 0: no sign expansion)
offset-low	Low-order bytes of 16-bit offset data loaded to the PC
offset-high	High-order bytes of 16-bit offset data loaded to the PC
seg-low	Low-order bytes of 16-bit segment data loaded to the PS
seg-high	High-order bytes of 16-bit segment data loaded to the PS
pop-value-low	Low-order bytes of 16-bit data which specifies the numberof bytes to delete from the stack
pop-value-high	High-order bytes of 16-bit data which specifies the numberof bytes to delete from the stack
disp8	8-bit displacement relatively added to the PC
X XXX YYY ZZZ	Operation code of an external floating pointarithmetic coprocessor

Table 16-3 Operation Description Legend

Identifier	Description
AW	Accumulator (16 bits)
AH	Accumulator (high-order bytes)
AL	Accumulator (low-order bytes)
BW	Register BW (16 bits)
CW	Register CW (16 bits)
CL	Register CW (low-order bytes)
DW	Register DW (16 bits)
SP	Stack pointer (16 bits)
PC	Program counter (16 bits)
PSW	Program status word (16 bits)
IX	Index register (source) (16 bits)
IY	Index register (destination) (16 bits)
PS	Program segment register (16 bits)
SS	Stack segment register (16 bits)
DS0	Data segment 0 register (16 bits)
DS1	Data segment 1 register (16 bits)
DS2	Expansion data segment 2 register (16 bits)
DS3	Expansion data segment 3 register (16 bits)
VPC	Vector PC
AC	Auxiliary carry flag
CY	Carry flag
P	Parity flag
S	Sign flag
Z	Zero flag
DIR	Direction flag
IE	Interrupt enable flag
V	Overflow flag
RB0 to RB3	Register bank flag
BRK	Break flag
$\overline{\text{IBRK}}$	I/O break flag
(...)	Memory contents shown in ( )
disp	Displacement (8/16-bit)
ext-disp8	8-bit displacement sign expanded to 16-bits
temp	Temporary register (8/16/32 bits)
tmpcy	Temporary carry flag (1 bit)
seg	Immediate segment data (16 bits)
offset	Immediate offset data (16 bits)
←	Transfer direction
+	Addition
−	Subtraction
×	Multiplication
÷	Division
%	Modulo
∧	Logical AND
∨	Logical OR
⊻	Exclusive OR
xxH	2-digit hexadecimal number
xxxxH	4-digit hexadecimal number
/	Dual-function, or

Table 16-4 Flag Operation Legend

Identifier	Description
(Blank)	No change
0	Cleared to 0
1	Set to 1
×	Set or cleared depending on result
U	undefined
R	Restore previously saved value

Table 16-5 Memory Addressing

mod mem	00	01	10
000	BW+IX	BW+IX+disp8	BW+IX+disp16
001	BW+IY	BW+IY+disp8	BW+IY+disp16
010	BP+IX	BP+IX+disp8	BW+IX+disp16
011	BP+IY	BP+IY+disp8	BP+IY+disp16
100	IX	IX+disp8	IX+disp16
101	IY	IY+disp8	IY+disp16
110	Direct address	BP+disp8	BP+disp16
111	BW	BW+disp8	BW+disp16

Note For other than primitive instruction memory addressing when BP is used, the default segment register becomes SS. Also, when BP is not used, the default segment register becomes DS0.
For primitive instruction memory addressing, the default segment register of the destination block becomes DS1. Also, for memory addressing the default segment register of the source block becomes DS0.

Table 16-6 8/16 Bit General Register Selection

reg. reg'	W = 0	W = 1
000	AL	AW
001	CL	CW
010	DL	DW
011	BL	BW
100	AH	SP
101	CH	BP
110	DH	IX
111	BH	IY

Table 16-7 Segment Register Selection

sreg	
00	DS1
01	PS
10	SS
11	DS0

Table 16-8 Expansion Segment Register Selection

xsreg	
10	DS3/VPC
11	DS2

Clock Number

In the case of a memory operand, the number of clocks differs depending on the addressing mode and so use the numerical values shown below to the portion with "EA" written in **Table 16-9 "Clock Number"**.

mod / mem	00		01		10	
		Clock		Clock		Clock
000	BW+IX	3	BW+IX+disp8	3	BW+IX+disp16	3
001	BW+IY	3	BW+IY+disp8	3	BW+IY+disp16	3
010	BP+IX	3	BP+IX+disp8	3	BP+IX+disp16	3
011	BP+IY	3	BP+IY+disp8	3	BP+IY+disp16	3
100	IX	2	IX+disp8	2	IX+disp16	2
101	IY	2	IY+disp8	2	IY+disp16	2
110	Direct address	2	BP+disp8	2	BP+disp16	2
111	BW	2	BW+disp8	2	BW+disp16	2

Also, "T" indicates the number of wait states. Use an arbitrary number of wait states for "0" (no wait).
The "bus width" indicates the main bus bus width.

Table 16-9 Clock Number (1/20)

Instruction Group	Mnemonic	Operand	Bus Width*	Byte Processing		Word Processing	
				On-chip RAM access	Other than On-chip RAM access	On-chip RAM access	Other than On-chip RAM access
Data Transfer Instructions	MOV	reg, reg'	—	2	2	2	2
		mem, reg	—	EA + 2	EA + 3	EA + 2	EA + 3
		reg, mem	8	EA + 2	EA + 5 + T	EA + 2	EA + 8 + 2T
			16				EA + 5 + T
		mem, imm	—	EA + 2	EA + 3	EA + 2	EA + 3
		reg, imm	—	2	2	2	3
		acc, dmem	8	4	7 + T	4	10 + 2T
			16				7 + T
		dmem, acc	—	4	5	4	5
		sreg, reg16	—	—	—	2	2
		xsreg, reg16 VPC, reg16	8	—	—	2	2
			16				
		sreg, mem16	8	—	—	EA + 2	EA + 8 + 2T
			16				EA + 5 + T
		xsreg,mem16 /VPC, mem16	8	—	—	EA + 2	EA + 8 + 2T
			16				EA + 5 + T
		reg16, sreg	—	—	—	2	2
		reg16, xsreg /reg16, VPC	8	—	—	2	2
			16				
		mem16, sreg	—	—	—	2	2
		mem16, xsreg /mem16, VPC	8	—	—	EA + 2	EA + 3
			16				
		DS0, reg16, mem32	8	—	—	EA + 5	EA + 17 + 4T
			16				EA + 11 + 2T
		DS2, reg16, mem32	8	—	—	EA + 5	EA + 17 + 4T
			16				EA + 11 + 2T
		DS1, reg16, mem32	8	—	—	EA + 5	EA + 17 + 4T
			16				EA + 11 + 2T
		DS3, reg16, mem32	8	—	—	EA + 5	EA + 17 + 4T
			16				EA + 11 + 2T

* 8 : 8-bit width
16 : 16-bit width
— : Common 8-bit bus width and 16-bit bus width

Table 16-9 Clock Number (2/20)

Instruction Group	Mnemonic	Operand	Bus Width*	Byte Processing		Word Processing	
				On-chip RAM access	Other than On-chip RAM access	On-chip RAM access	Other than On-chip RAM access
Data Transfer Instructions	MOV	AH, PSW	—	2	2	—	—
		PSW, AH	8	3	3	—	—
			16	2	2		
	LDEA	reg16, mem16	—	—	—	EA + 2	EA + 2
	TRANS /TRANSB	src-table	—	EA + 2	EA + 3	EA + 2	EA + 3
	XCH	reg, reg'	—	4	4	4	4
		mem, reg /reg, mem		EA + 4	EA + 7 + T	EA + 4	EA + 10 + 2T
							EA + 7 + T
		AW, reg16 /reg16, AW	—	—	—	4	4
	MOVSPA		—	—	—	8	8
	MOVSPB	reg16	—	—	—	4	4
Repeat Prefix	REPC		—	0 to 1	0 to 1	0 to 1	0 to 1
	REPNC		—	0 to 1	0 to 1	0 to 1	0 to 1
	REP /REPE /REPZ		—	0 to 1	0 to 1	0 to 1	0 to 1
	REPNE /REPNZ		—	0 to 1	0 to 1	0 to 1	0 to 1
Primitive Block Transfer Instructions	MOVBK	dst-block, src-block	8	18 + T (rep) 9 + (11 + T)n (rep CW = 0) 5	19 + T 9 + (12 + 2T)n 5	21 + 2T 9 + (11 + 2T)n 5	22 + 2T 9 + (18 + 4T)n 5
	MOVBKB /MOVBKW		16			18 + T 9 + (11 + T)n 5	19 + T 9 + (12 + 2T)n 5
	CMPBK	src-block dst-block	8	20 + T (rep) 9 + (13 + T)n (rep CW = 0) 5	22 + 2T 9 + (15 + 2T)n 5	23 + 2T 9 + (16 + 2T)n 5	28 + 4T 9 + (21 + 4T)n 5
	CMPBKB /CMPBKW		16			20 + T 9 + (13 + T)n 5	22 + 2T 9 + (15 + 2T)n 5

* 8 : 8-bit width
16 : 16-bit width
— : Common 8-bit bus width and 16-bit bus width

Remarks n: Number of times repeated

Table 16-9 Clock Number (3/20)

Instruction Group	Mnemonic	Operand	Bus Width*	Byte Processing		Word Processing	
				On-chip RAM access	Other than On-chip RAM access	On-chip RAM access	Other than On-chip RAM access
Primitive Block Transfer Instructions	CMPM	dst-block	8	15	17 + T	15	20 + T
	CMPMB /CMPMW			(rep) 10 + 7n	10 + (9 + T)n	10 + 7n	10 + (12 + 2T)n
				(rep CW = 0) 5	5	5	5
			16				17 + T
							10 + (9 + T)n
							5
	LDM	arc-block	8	10	13 + T	10	16 + T
	LDMB /LDMW			(rep) 9 + 3n	9 + (6 + T)n	9 + 3n	9 + (9 + 2T)n
				(rep CW = 0) 5	5	5	5
			16				13 + T
							9 + (6 + T)n
							5
	STM	dst-block	8	12	13	12	13
	STMB /STMW			(rep) 10 + 7n	10 + (9 + T)n	10 + 7n	9 + (9 + 2T)n
				(rep CW = 0) 5	5	5	5
			16				13
							9 + (6 + T)n
							5
Bit Field Manipulation Instructions	INS	reg8, reg8'	8	—	—	22 to 63	31 to 72
			16				23 to 64
		reg8, imm4	8	—	—	22 to 63	31 to 72
			16				23 to 64
	EXT	reg8, reg8'	8	—	—	19 to 41	19 + 2T to 48 + 4T
			16				19 to 42 + 2T
		reg8, imm4	8	—	—	19 to 41	19 + 2T to 48 + 4T
			16				19 to 42 + 2T

* 8 : 8-bit width
16 : 16-bit width
— : Common 8-bit bus width and 16-bit bus width

Remarks n: Number of times repeated

Table 16-9 Clock Number (4/20)

Instruction Group	Mnemonic	Operand	Bus Width*1	Byte Processing		Word Processing	
				On-chip RAM access	Other than On-chip RAM access	On-chip RAM access	Other than On-chip RAM access
Input/output Instructions	IN*2	acc8, imm8	8	—	7 + T	—	10 + 2T
			16				7 + T
		acc, DW	8	—	7 + T	—	10 + 2T
			16				7 + T
	OUT*2	imm8, acc	8	—	5	19 – 41	5
			16				
		DW, acc	8	—	5	19 – 41	5
			16				
Primitive Input/output Instructions	INM*2	dst-block, DW	8	17 + T	18 + T	20 + 2T	21 + 2T
				(rep) 9 + (10 + T)n	9 + (11 + 2T)n	9 + (13 + 2T)n	9 + (17 + 4T)n
				(rep CW = 0) 5	5	5	5
			16			17 + T	18 + T
						9 + (10 + T)n	9 + (11 + 2T)n
						5	5
	OUTM*2	DW, src-block	8	14 + T	17 + 2T	17 + 2T	23 + 4T
				(rep) 9 + (7 + T)n	9 + (10 + 2T)n	9 + (10 + 2T)n	9 + (16 + 4T)n
				(rep CW = 0) 5	5	5	5
			16			14 + T	17 + 2T
						9 + (7 + T)n	9 + (10 + 2T)n
						5	5

* 1. 8 : 8-bit width
16 : 16-bit width
— : Common 8-bit bus width and 16-bit bus width

2. Shows when $\overline{\text{IBRK}}$ = 1. The following occuurs when $\overline{\text{IBRK}}$ = 0.

Remarks n: Number of times repeated

Table 16-9 Clock Number (5/20)

Instruction Group	Mnemonic	Operand	Bus Width*	Byte Processing		Word Processing	
				On-chip RAM access	Other than On-chip RAM access	On-chip RAM access	Other than On-chip RAM access
Input/output Instructions	IN	acc8, imm8	8	—	60 + 10T	—	60 + 10 T
			16		40 + 5T		40 + 5T
		acc, DW	8	—	60 + 10 T	—	60 + 10 T
			16		40 + 5T		40 + 5T
	OUT	imm8, acc	8	—	60 + 10 T	—	60 + 10 T
			16		40 + 5T		40 + 5T
		DW, acc	8	—	60 + 10 T	—	60 + 10 T
			16		40 + 5T		40 + 5T
Primitive Input /output Instructions	INM	dst-block, DW	8	—	60 + 10 T	—	60 + 10 T
			16		40 + 5T		40 + 5T
	OUTM	DW, src-block	8	—	60 + 10 T	—	60 + 10 T
			16		40 + 5T		40 + 5T

* 8 : 8-bit width
16 : 16-bit width
— : Common 8-bit bus width and 16-bit bus width

Table 16-9 Clock Number (6/20)

Instruction Group	Mnemonic	Operand	Bus Width*	Byte Processing		Word Processing	
				On-chip RAM access	Other than On-chip RAM access	On-chip RAM access	Other than On-chip RAM access
Addition/substruction Instructions	ADD	reg, reg'	—	3	3	3	3
		mem, reg	8	EA + 4	EA + 7 + T	EA + 4	EA + 10 + 2T
			16				EA + 7 + T
		reg, mem	8	EA + 2	EA + 6 + T	EA + 2	EA + 9 + 2T
			16				EA + 6 + T
		reg, imm	—	2	2	2	2
		mem, imm	8	EA + 4	EA + 7 + T	EA + 4	EA + 10 + 2T
			16				EA + 7 + T
		acc, imm	—	2	2	2	2
	ADDC	reg, reg'	—	3	3	3	3
		mem, reg	8	EA + 4	EA + 7 + T	EA + 4	EA + 10 + 2T
			16				EA + 7 + T
		reg, mem	8	EA + 2	EA + 6 + T	EA + 2	EA + 9 + 2T
			16				EA + 6 + T
		reg, imm	—	2	2	2	2
		mem, imm	8	EA + 4	EA + 7 + T	EA + 4	EA + 10 + 2T
			16				EA + 7 + T
		acc, imm	—	2	2	2	2
	SUB	reg, reg'	—	3	3	3	3
		mem, reg	8	EA + 4	EA + 7 + T	EA + 4	EA + 10 + 2T
			16				EA + 7 + T
		reg, mem	8	EA + 2	EA + 6 + T	EA + 2	EA + 9 + 2T
			16				EA + 6 + T
		reg, imm	—	2	2	2	2
		mem, imm	8	EA + 4	EA + 7 + T	EA + 4	EA + 10 + 2T
			16				EA + 7 + T
		acc, imm	—	2	2	2	2

- 8 : 8-bit width

16 : 16-bit width

— : Common 8-bit bus width and 16-bit bus width

Table 16-9 Clock Number (7/20)

Instruction Group	Mnemonic	Operand	Bus Width*	Byte Processing		Word Processing	
				On-chip RAM access	Other than On-chip RAM access	On-chip RAM access	Other than On-chip RAM access
Addition/substruction Instructions	SUBC	reg, reg'	—	3	3	3	3
		mem, reg	8	EA + 4	EA + 7 + T	EA + 4	EA + 10 + 2T
			16				EA + 7 + T
		reg, mem	8	EA + 2	EA + 6 + T	EA + 2	EA + 9 + 2T
			16				EA + 6 + T
		reg, imm	—	2	2	2	2
		mem, imm	8	EA + 4	EA + 7 + T	EA + 4	EA + 10 + 2T
			16				EA + 7 + T
		acc, imm	—	2	2	2	2
BCD Calculation Instructions	ADD4S	dst-string, src-string	8	6 + (15 + T)n	6 + (19 + 3T)n	—	—
			16				
	SUB4S	dst-string, src-string	8	6 + (16 + T)n	6 + (20 + 3T)n	—	—
			16				
	CMP4S	dst-string, src-string	8	6 + (15 + T)n	6 + (18 + 2T)n	—	—
			16				
	ROL$	reg8	8	5	5	—	—
		mem8	16	EA + 5	EA + 8 + T	—	—
	ROR4	reg8	8	5	5	—	—
		mem8	16	EA + 5	EA + 8 + T	—	—
Increment/decrement Instructions	INC	reg8	—	2	2	—	—
		mem	8	EA + 3	EA + 7 + T	EA + 3	EA + 10 + 2T
			16				EA + 7 + T
		reg16	—	—	—	2	2
	DEC	reg8	—	2	2	—	—
		mem	8	EA + 3	EA + 7 + T	EA + 3	EA + 10 + 2T
			16				EA + 7 + T
		reg16	—	—	—	2	2

* 8 : 8-bit width
16 : 16-bit width
— : Common 8-bit bus width and 16-bit bus width

Remarks n: 1/2 the number of BCD digits

Table 16-9 Clock Number (8/20)

Instruction Group	Mnemonic	Operand	Bus Width*	Byte Processing		Word Processing	
				On-chip RAM access	Other than On-chip RAM access	On-chip RAM access	Other than On-chip RAM access
Multiplication Instructions	MULU	reg8	—	11	11	15	15
		mem8	8	EA + 12	EA + 14 + T	EA + 16	EA + 21 + 2T
			16				EA + 18 + T
		reg16	—	11	11	15	15
		mem16	8	EA + 12	EA + 14 + T	EA + 16	EA + 21 + 2T
			16				EA + 18 + T
	MUL	reg8	—	10	10	14	14
		mem8	8	EA + 11	EA + 13 + T	EA + 15	EA + 20 + 2T
			16				EA + 17 + T
		reg16	—	10	10	14	14
		mem16	8	EA + 11	EA + 13 + T	EA + 15	EA + 20 + 2T
			16				EA + 17 + T
		reg16, reg16', imm8/reg16, imm8	—	—	—	14	14
		reg16, mem16, imm8	8	—	—	EA + 15	EA + 20 + 2T
			16				EA + 17 + T
		reg16, reg16', imm16/reg16, imm16	—	—	—	14	14
		reg16, mem16, imm16	8	—	—	EA + 15	EA + 20 + 2T
			16				EA + 17 + T

* 8 : 8-bit width
16 : 16-bit width
— : Common 8-bit bus width and 16-bit bus width

Table 16-9 Clock Number (9/20)

Instruction Group	Mnemonic	Operand	Bus Width*	Byte Processing		Word Processing	
				On-chip RAM access	Other than On-chip RAM access	On-chip RAM access	Other than On-chip RAM access
Division Instructions	DIVU	reg8	8	15/62 + 10T	15/62 + 10T	23/57 + 10T	23/57 + 10T
			16	15/42 + 5T	15/42 + 5T	23/42 + 5T	23/42 + 5T
		mem8	8	EA + 16/63 + 10T	EA+18 +T/63+10T	EA + 24/58 + 10T	EA+30+2T/58+10T
			16	EA + 16/43 + 5T	EA+ 18+ T/63+5T	EA + 24/43 + 5T	EA+26+T/43+5T
		reg16	8	15/62 + 10T	15/62 + 10T	23/57 + 10T	23/57 + 10T
			16	15/42 + 5T	15/42 + 5T	23/42 + 5T	23/42 + 5T
		mem16	8	EA + 16/63 + 10T	EA+18+T/63+10T	EA + 24/58 + 10T	EA+30+2T/58+10T
			16	EA + 16/43 + 5T	EA+18+T/63+5T	EA + 24/43 + 5T	EA+26+T/43+5T
	DIV	reg8	8	17/64 + 10T	17/64 + 10T	25/59 + 10T	25/59 + 10T
			16	17/44 + 5T	17/44 + 5T	25/44 + 5T	25/44 + 5T
		mem8	8	EA + 18/65 + 10T	EA+20+T/65+10T	EA + 26/60 + 10T	EA+31+2T/60+10T
			16	EA + 18/45 + 5T	EA+20+T/45+5T	EA + 26/45 + 5T	EA+28+T/45+5T
		reg16	8	17/64 + 10T	17/64 + 10T	25/59 + 10T	25/59 + 10T
			16	17/44 + 5T	17/44 + 5T	25/44 + 5T	25/44 + 5T
		mem16	8	EA + 18/65 + 10T	EA+20+T/65+10T	EA + 26/60 + 10T	EA+31+2T/60+10T
			16	EA + 18/45 + 5T	EA+20+T/45+5T	EA + 26/45 + 5T	EA+28+T/45+5T
BCD Correction Instructions	ADJBA		8	6	9	—	—
			16	9			
	ADJ4A		—	3	3	—	—
	ADJBS		8	6	6	—	—
			16	9	9		
	ADJ4S		—	3	3	—	—
Data Transformation Instructions	CVTBD		—	18	18	—	—
	CVTDB		—	8	8	—	—
	CVTBW		—	3	3	—	—
	CVTWL		—	—	—	3	3

* 8 : 8-bit width
16 : 16-bit width
— : Common 8-bit bus width and 16-bit bus width

Remarks The right side of / occurs for adivide error.

Table 16-9 Clock Number (10/20)

Instruction Group	Mnemonic	Operand	Bus Width*	Byte Processing		Word Processing	
				On-chip RAM access	Other than On-chip RAM access	On-chip RAM access	Other than On-chip RAM access
Comparison Instructions	CMP	reg, reg'	—	3	3	3	3
		mem, reg	8	EA + 4	EA + 6 + T	EA + 4	EA + 9 + 2T
			16				EA + 6 + T
		reg, mem	8	EA + 2	EA + 6 + T	EA + 2	EA + 9 + 2T
			16				EA + 6 + T
		reg, imm	—	2	2	2	2
		mem, imm	8	EA + 4	EA + 6 + T	EA + 4	EA + 9 + 2T
			16				EA + 6 + T
		acc, imm	—	2	2	2	2
Complimentary Operation Instructions	NOT	reg	—	2	2	2	2
		mem	8	EA + 3	EA + 7 + T	EA + 3	EA + 10 + 2T
			16				EA + 7 + T
	NEG	reg	—	2	2	2	2
		mem	8	EA + 3	EA + 7 + T	EA + 3	EA + 10 + 2T
			16				EA + 7 + T
Logical Operation Instructions	TEST	reg, reg'	—	3	3	3	3
		mem, reg /reg, mem	8	EA + 4	EA + 6 + T	EA + 4	EA + 9 + 2T
			16				EA + 6 + T
		reg, imm	—	2	2	2	2
		mem, imm	8	EA + 4	EA + 6 + T	EA + 4	EA + 9 + 2T
			16				EA + 6 + T
		acc, imm	—	2	2	2	2
	AND	reg, reg'	—	3	3	3	3
		mem, reg	8	EA + 4	EA + 7 + T	EA + 4	EA + 10 + 2T
			16				EA + 7 + T
		reg, mem	8	EA + 4	EA + 6 + T	EA + 4	EA + 9 + 2T
			16				EA + 6 + T
		reg, imm	—	2	2	2	2
		mem, imm	8	EA + 4	EA + 7 + T	EA + 4	EA + 10 + 2T
			16				EA + 7 + T
		acc, imm	—	2	2	2	2

* 8 : 8-bit width
16 : 16-bit width
— : Common 8-bit bus width and 16-bit bus width

Table 16-9 Clock Number (11/20)

Instruction Group	Mnemonic	Operand	Bus Width*	Byte Processing		Word Processing	
				On-chip RAM access	Other than On-chip RAM access	On-chip RAM access	Other than On-chip RAM access
Logical Operation Instructions	OR	reg, reg'	—	3	3	3	3
		mem, reg	8	EA + 4	EA + 7 + T	EA + 4	EA + 10 + 2T
			16				EA + 7 + T
		reg, mem	8	EA + 2	EA + 6 + T	EA + 2	EA + 9 + 2T
			16				EA + 6 + T
		reg, imm	—	2	2	2	2
		mem, imm	8	EA + 4	EA + 7 + T	EA + 4	EA + 10 + 2T
			16				EA + 7 + T
		acc, imm	—	2	2	2	2
	XOR	reg, reg'	—	3	3	3	3
		mem, reg	8	EA + 4	EA + 7 + T	EA + 4	EA + 10 + 2T
			16				EA + 7 + T
		reg, mem	8	EA + 2	EA + 6 + T	EA + 2	EA + 9 + 2T
			16				EA + 6 + T
		reg, imm	—	2	2	2	2
		mem, imm	8	EA + 4	EA + 7 + T	EA + 4	EA + 10 + 2T
			16				EA + 7 + T
		acc, imm	—	2	2	2	2
Bit Manipulation Instructions	TEST1	reg8, CL	—	3	3	3	3
		mem8, CL	8	EA + 4	EA + 6 + T	EA + 4	EA + 9 + 2T
			16				EA + 6 + T
		reg16, CL	—	3	3	3	3
		mem16, CL	8	EA + 4	EA + 6 + T	EA + 4	EA + 9 + 2T
			16				EA + 6 + T
		reg8, imm3	—	2	2	2	2
		mem8, imm3	8	EA + 4	EA + 6 + T	EA + 4	EA + 9 + 2T
			16				EA + 6 + T
		reg16, imm3	—	3	3	3	3
		mem16, imm3	8	EA + 4	EA + 6 + T	EA + 4	EA + 9 + 2T
			16				EA + 6 + T

* 8 : 8-bit width
16 : 16-bit width
— : Common 8-bit bus width and 16-bit bus width

Table 16-9 Clock Number (12/20)

Instruction Group	Mnemonic	Operand	Bus Width*	Byte Processing		Word Processing	
				On-chip RAM access	Other than On-chip RAM access	On-chip RAM access	Other than On-chip RAM access
Bit Manipulation Instructions	NOT1	reg8, CL	—	3	3	3	3
		mem8, CL	8	EA + 4	EA + 7 + T	EA + 4	EA + 10 + 2T
			16				EA + 7 + T
		reg16, CL	—	3	3	3	3
		mem16, CL	8	EA + 4	EA + 7 + T	EA + 4	EA + 10 + 2T
			16				EA + 7 + T
		reg8, imm3	—	2	2	2	2
		mem8, imm3	8	EA + 4	EA + 7 + T	EA + 4	EA + 10 + 2T
			16				EA + 7 + T
		reg16, imm4	—	3	3	3	3
		mem16, imm4	8	EA + 4	EA + 7 + T	EA + 4	EA + 10 + 2T
			16				EA + 7 + T
		CY	—	2	2	2	2
	CLR1	reg8, CL	—	3	3	3	3
		mem8, CL	8	EA + 4	EA + 7 + T	EA + 4	EA + 10 + 2T
			16				EA + 7 + T
		reg16, CL	—	3	3	3	3
		mem16, CL	8	EA + 4	EA + 7 + T	EA + 4	EA + 10 + 2T
			16				EA + 7 + T
		reg8, imm3	—	2	2	2	2
		mem8, imm3	8	EA + 4	EA + 7 + T	EA + 4	EA + 10 + 2T
			16				EA + 7 + T
		reg16, imm4	—	3	3	3	3
		mem16, imm4	8	EA + 4	EA + 7 + T	EA + 4	EA + 10 + 2T
			16				EA + 7 + T
		CY	—	2	2	2	2
		DIR	—	2	2	2	2

* 8 : 8-bit width
16 : 16-bit width
— : Common 8-bit bus width and 16-bit bus width

Table 16-9 Clock Number (13/20)

Instruction Group	Mnemonic	Operand	Bus Width*	Byte Processing		Word Processing	
				On-chip RAM access	Other than On-chip RAM access	On-chip RAM access	Other than On-chip RAM access
Bit Manipulation Instructions	SET1	reg8, CL	—	3	3	3	3
		mem8, CL	8	EA + 4	EA + 7 + T	EA + 4	EA + 10 + 2T
			16				EA + 7 + T
		reg16, CL	—	3	3	3	3
		mem16, CL	8	EA + 4	EA + 7 + T	EA + 4	EA + 10 + 2T
			16				EA + 7 + T
		reg8, imm3	—	2	2	2	2
		mem8, imm3	8	EA + 4	EA + 7 + T	EA + 4	EA + 10 + 2T
			16				EA + 7 + T
		reg16, imm4	—	3	3	3	3
		mem16, imm4	8	EA + 4	EA + 7 + T	EA + 4	EA + 10 + 2T
			16				EA + 7 + T
		CY	—	2	2	2	2
		DIR	—	2	2	2	2
	BSCH	mem	8	EA + 8 + 3n + T	EA + 8 + 3n + T	EA + 11 + 3n + 2T	EA + 11 + 3n + 2T
			16	EA + 8 + 3n + T	EA + 8 + 3n + T	EA + 8 + 3n + T	EA + 8 + 3n + T
		reg	—	4 + 3n	4 + 3n	4 + 3n	4 + 3n
Shift Instructions	SHL	reg, 1	—	3	3	3	3
		mem, 1	8	EA + 3	EA + 7 + T	EA + 3	EA + 10 + 2T
			16				EA + 7 + T
		reg, CL	—	5 + n	5 + n	5 + n	5 + n
		mem, CL	8	EA + 5 + n	EA + 8 + T + n	EA + 6 + n	EA + 11 + 2T + n
			16				EA + 8 + T + n
		reg, imm8	—	5 + n	5 + n	5 + n	5 + n
		mem, imm8	8	EA + 6 + n	EA + 8 + T + n	EA + 6 + n	EA + 11 + 2T + n
			16				EA + 8 + T + n

* 8 : 8-bit width
16 : 16-bit width
— : Common 8-bit bus width and 16-bit bus width

Remarks Number of shifts (n in a bit manipulation instruction indicates the bit number searched.)

Table 16-9 Clock Number (14/20)

Instruction Group	Mnemonic	Operand	Bus Width*	Byte Processing		Word Processing	
				On-chip RAM access	Other than On-chip RAM access	On-chip RAM access	Other than On-chip RAM access
SHift Instructions	SHR	reg, 1	—	3	3	3	3
		mem, 1	8	EA + 3	EA + 7 + T	EA + 3	EA + 10 + 2T
			16				EA + 7 + T
		reg, CL	—	5 + n	5 + n	5 + n	5 + n
		mem, CL	8	EA + 5 + n	EA + 8 + T + n	EA + 6 + n	EA + 11 + 2T + n
			16				EA + 8 + T + n
		reg, imm8	—	5 + n	5 + n	5 + n	5 + n
		mem, imm8	8	EA + 6 + n	EA + 8 + T + n	EA + 6 + n	EA + 11 + 2T + n
			16				EA + 8 + T + n
	SHRA	reg, 1	—	3	3	3	3
		mem, 1	8	EA + 3	EA + 7 + T	EA + 3	EA + 10 + 2T
			16				EA + 7 + T
		reg, CL	—	5 + n	5 + n	5 + n	5 + n
		mem, CL	8	EA + 5 + n	EA + 8 + T + n	EA + 6 + n	EA + 11 + 2T + n
			16				EA + 8 + T + n
		reg, imm8	—	5 + n	5 + n	5 + n	5 + n
		mem, imm8	8	EA + 6 + n	EA + 8 + T + n	EA + 6 + n	EA + 11 + 2T + n
			16				EA + 8 + T + n
Rotate Instructions	ROL	reg, 1	—	3	3	3	3
		mem, 1	8	EA + 3	EA + 7 + T	EA + 3	EA + 10 + 2T
			16				EA + 7 + T
		reg, CL	—	5 + n	5 + n	5 + n	5 + n
		mem, CL	8	EA + 5 + n	EA + 8 + T + n	EA + 6 + n	EA + 11 + 2T + n
			16				EA + 8 + T + n
		reg, imm8	—	5 + n	5 + n	5 + n	5 + n
		mem, imm8	8	EA + 6 + n	EA + 8 + T + n	EA + 6 + n	EA + 11 + 2T + n
			16				EA + 8 + T + n

* 8 : 8-bit width
16 : 16-bit width
— : Common 8-bit bus width and 16-bit bus width

Remarks Number of shifts

Table 16-9 Clock Number (15/20)

Instruction Group	Mnemonic	Operand	Bus Width*	Byte Processing		Word Processing	
				On-chip RAM access	Other than On-chip RAM access	On-chip RAM access	Other than On-chip RAM access
Rotate Instructions	ROR	reg, 1	—	3	3	3	3
		mem, 1	8	EA + 3	EA + 7 + T	EA + 3	EA + 10 + 2T
			16				EA + 7 + T
		reg, CL	—	5 + n	5 + n	5 + n	5 + n
		mem, CL	8	EA + 5 + n	EA + 8 + T + n	EA + 6 + n	EA + 11 + 2T + n
			16				EA + 8 + T + n
		reg, imm8	—	5 + n	5 + n	5 + n	5 + n
		mem, imm8	8	EA + 6 + n	EA + 8 + T + n	EA + 6 + n	EA + 11 + 2T + n
			16				EA + 8 + T + n
	ROLC	reg, 1	—	3	3	3	3
		mem, 1	8	EA + 3	EA + 7 + T	EA + 3	EA + 10 + 2T
			16				EA + 7 + T
		reg, CL	—	5 + n	5 + n	5 + n	5 + n
		mem, CL	8	EA + 5 + n	EA + 8 + T + n	EA + 6 + n	EA + 11 + 2T + n
			16				EA + 8 + T + n
		reg, imm8	—	5 + n	5 + n	5 + n	5 + n
		mem, imm8	8	EA + 6 + n	EA + 8 + T + n	EA + 6 + n	EA + 11 + 2T + n
			16				EA + 8 + T + n
	RORC	reg, 1	—	3	3	3	3
		mem, 1	8	EA + 3	EA + 7 + T	EA + 3	EA + 10 + 2T
			16				EA + 7 + T
		reg, CL	—	5 + n	5 + n	5 + n	5 + n
		mem, CL	8	EA + 5 + n	EA + 8 + T + n	EA + 6 + n	EA + 11 + 2T + n
			16				EA + 8 + T + n
		reg, imm8	—	5 + n	5 + n	5 + n	5 + n
		mem, imm8	8	EA + 6 + n	EA + 8 + T + n	EA + 6 + n	EA + 11 + 2T + n
			16				EA + 8 + T + n

* 8 : 8-bit width
16 : 16-bit width
— : Common 8-bit bus width and 16-bit bus width

Remarks Number of shifts

Table 16-9 Clock Number (16/20)

Instruction Group	Mnemonic	Operand	Bus Width*1	Byte Processing		Word Processing	
				On-chip RAM access	Other than On-chip RAM access	On-chip RAM access	Other than On-chip RAM access
Subroutine Control Instructions	CALL	near-proc	8	—	—	—	19 + 2T
			16				16 + T
		regptr16	8	—	—	—	18 + 2T
			16				15 + T
		memptr16	8	—	—	EA + 19 + 2T	EA + 24 + 4T
			16			EA + 16 + T	EA + 18 + 2T
		far-proc	8	—	—	—	29 + 4T
			16				23 + 2T
		memptr32	8	—	—	EA + 32 + 4T	EA + 44 + 8T
			16			EA + 26 + 2T	EA + 32 + 4T
	RET		8	—	—	—	18 + 2T
			16				15 + T
		pop-value	8	—	—	—	19 + 2T
			16				16 + T
		*2	8	—	—	—	26 + 4T
			16				20 + 2T
		pop-value*2	8	—	—	—	27 + 4T
			16				21 + 2T
Stack Manipulation Instructions	PUSH	mem16	8	—	—	EA + 7	EA + 13 + 2T
			16				EA + 10 + T
		reg16	—	—	—	—	7
		sreg	—	—	—	—	7
		xsreg/VPC	—	—	—	—	7
		PSW	—	—	—	—	6
		R	8	—	—	—	57 + 14T
			16				36 + 7T
		imm8	—	—	—	—	6
		imm16	—	—	—	—	6

* 1. 8 : 8-bit width
16 : 16-bit width
— : Common 8-bit bus width and 16-bit bus width

2. Indicates outside the segment.

Remarks n: Number of shifts

Table 16-9 Clock Number (17/20)

Instruction Group	Mnemonic	Operand	Bus Width*1	Byte Processing		Word Processing	
				On-chip RAM access	Other than On-chip RAM access	On-chip RAM access	Other than On-chip RAM access
Stack Manipulation Instructions	PUSH	mem16	8	—	—	EA + 13 + 2T	EA + 14 + 2T
			16			EA + 10 + T	EA + 11 + T
		reg16	8	—	—	—	10 + 2T
			16				7 + T
		sreg	8	—	—	—	10 + 2T
			16				7 + T
		xsreg/VPC	8	—	—	—	10 + 2T
			16				7 + T
		PSW	8	—	—	—	11 + 2T
			16				8 + T
		R	8	—	—	—	76 + 16T
			16				52 + 8T
	PREPARE*2	imm16, imm8	—	—	—	—	9
	DISPOSE		8	—	—	—	10 + 2T
			16				7 + T
Branch Instructions	BR	near-label	—	—	—	—	9
		short-label	—	—	—	—	9
		regptr16	—	—	—	—	8
		memptr16	8	—	—	EA + 9	EA + 14 + 2T
			16				EA + 11 + T
		far-label	—	—	—	—	9
		memptr32	8	—	—	EA + 12	EA + 24 + 4T
			16				EA + 18 + 2T

* 1. 8 : 8-bit width

16 : 16-bit width

— : Common 8-bit bus width and 16-bit bus width

2. Indicates when imm8 = 0. The following occurs when imm8 ≧ 1.

	PREPARE	imm16, imm8	8	—	—	—	15+2T+(16+4T)n
			16				14 + (12 + T)n

Table 16-9 Clock Number (18/20)

Instruction Group	Mnemonic	Operand	Bus Width	Byte Processing		Word Processing	
				On-chip RAM access	Other than On-chip RAM access	On-chip RAM access	Other than On-chip RAM access
Conditional Branch Instructions	BV	short-label	—	—	—	9/3	9/3
	BNV	short-label	—	—	—	9/3	9/3
	BC/BL	short-label	—	—	—	9/3	9/3
	BNC/BNL	short-label	—	—	—	9/3	9/3
	BE/BZ	short-label	—	—	—	9/3	9/3
	BNE/BNZ	short-label	—	—	—	9/3	9/3
	BNH	short-label	—	—	—	9/3	9/3
	BH	short-label	—	—	—	9/3	9/3
	BN	short-label	—	—	—	9/3	9/3
	BP	short-label	—	—	—	9/3	9/3
	BPE	short-label	—	—	—	9/3	9/3
	BPO	short-label	—	—	—	9/3	9/3
	BLT	short-label	—	—	—	9/3	9/3
	BGE	short-label	—	—	—	9/3	9/3
	BLE	short-label	—	—	—	9/3	9/3
	BGT	short-label	—	—	—	9/3	9/3
	DBNZNE	short-label	—	—	—	10/5	10/5
	DBNZE	short-label	—	—	—	10/5	10/5
	DBNZ	short-label	—	—	—	10/5	10/5
	BCWZ	short-label	—	—	—	10/5	10/5
	BTCLR	sfr, imm3 short-label	8	—	21/14	—	—
			16				
	BTCLRL	sfr, imm3 short-label	8	—	20/13	—	—
			16				

* 8 : 8-bit width
16: 16-bit width
—: Common 8-bit bus width and 16-bit bus width

Table 16-9 Clock Number (19/20)

Instruction Group	Mnemonic	Operand	Bus Width*1	Byte Processing		Word Processing	
				On-chip RAM access	Other than On-chip RAM access	On-chip RAM access	Other than On-chip RAM access
Interrupt Instructions	BRK*2	3	8	—	—	—	50 + 10T
			16				36 + 4T + t
		imm8 (≠3)	8	—	—	—	52 + 10T
			16				38 + 4T + t
	BRKV*2		8	—	—	—	51 + 10 T
			16				37 + 4T + t
	RETI		8	—	—	—	28 + 4T
			16				22 + 2T
	RETRBI		—	—	—	—	9
	FINT		—	3	3	3	3
	CHKIND*3		8	—	—	EA + 11	EA + 21 + 4T
			16				EA + 15 + 2T
*4	BRKCS	reg16	—	—	—	12	12
	TSKSW	reg16	—	—	—	13	13
CPU Control Instructions	HALT		—	—	—	—	—
	STOP		—	—	—	—	—
	IDLE		—	—	—	—	—
	POLL		—	—	—	—	—
	DI		—	3	3	3	3
	EI		—	3	3	3	3
	BUSLOCK		—	0 to 1	0 to 1	0 to 1	0 to 1

* 1. 8 : 8-bit width
16 : 16-bit width
— : Common 8-bit bus width and 16-bit bus width

2. For BRK = 1, 50 + 10T are added for an 8-bit bus width, and 34 + 4T are added fpr a 16-bit bus width.
3. When (mem32) > reg16 or when (mem32 + 2) < reg16, 50 + 10T are added for an 8-bit bus width, and 34 + 4T + t are added for a 16-bit bus width.
4. Register bank switching instruction

Remarks When T ≧ 2, t = T − 1

Table 16-9 Clock Number (20/20)

Instruction Group	Mnemonic	Operand	Bus Width*1	Byte Processing		Word Processing	
				On-chip RAM access	Other than On-chip RAM access	On-chip RAM access	Other than On-chip RAM access
CPU Control Instructions	FPO1	fp-op	8	—	—	—	50 + 10T
			16				36 + 4T + t
		fp=op, mem	8	—	—	—	EA + 50 + 10 T
			16				EA + 36 + 4T + t
	FPO2	fp-op	8	—	—	—	50 + 10T
			16				36 + 4T + t
		fp=op, mem	8	—	—	—	EA + 50 + 10 T
			16				EA + 36 + 4T + t
	NOP		—	4	4	4	4
*2	RSTWDT	imm8, imm8'	8	—	9/54 + 10T*3	—	—
			16		9/40 + 4T + t*3		
*4			—	0 to 1	0 to 1	0 to 1	0 to 1
*5	QHOUT	imm16	—	—	—	—	—
	QHOUT	imm16	—	—	—	—	—
	QTIN	imm16	—	—	—	—	—

* 1. 8 : 8-bit width
16 : 16-bit width
— : Common 8-bit bus width and 16-bit bus width

2. Watchdog timer manipulation instructions
3. From /, word proccessing is performed for a data error.
When T ≧ 2, t = T − 1
4. Segment override prefix (DS :, DS1 :, PS :, SS :)
Expansion segment override prefix (DS2 :, DS3 :)
Override prefix (IRAM :) for accessing the register file space.
5. Queue manipulation instructions

16.3 LIST OF THE INSTRUCTION SET

Instruction Group	Mnemonic	Operand	Operation Code		Byte Number	Operation	Flag					
			7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0			AC	CY	V	P	S	Z
Data Transfer Instruction	MOV	reg, reg'	1 0 0 0 1 0 1 W	1 1 reg reg'	2	reg←reg'						
		mem, reg	1 0 0 0 1 0 0 W	mod reg mem	2 to 4	(mem)←reg						
		reg, mem	1 0 0 0 1 0 1 W	mod reg mem	2 to 4	reg←(mem)						
		mem, imm	1 1 0 0 0 1 1 W	mod 0 0 0 mem	3 to 6	(mem)←imm						
		reg, imm	1 0 1 1 W reg		2 to 3	reg←imm						
		acc, dmem	1 0 1 0 0 0 0 W		3	When W = 0, AL←(dmem) When W = 1, AH←(dmem + 1), AL←(dmem)						
		dmem, acc	1 0 1 0 0 0 1 W		3	When W = 0, (dmem)← AL When W = 1, (dmem + 1)←AH, (dmem)←AL						
		sreg, reg16	1 0 0 0 1 1 1 0	1 1 0 sreg reg	2	sreg←reg16 sreg : SS, DS0, DS1						
		xsreg, reg16*	1 0 0 0 1 1 1 0	1 1 1 xsreg reg	2	xsreg←reg16 xsreg : DS2, DS3						
		sreg, mem16	1 0 0 0 1 1 1 0	mod 0 sreg mem	2 to 4	sreg←(mem16) sreg : SS, DS0, DS1						
		xsreg, mem16*	1 0 0 0 1 1 1 0	mod 1 xsreg mem	2 to 4	xsreg←(mem16)						
		reg16,sreg	1 0 0 0 1 1 0 0	1 1 0 sreg reg	2	reg16←sreg						
		reg16, xsreg*	1 0 0 0 1 1 0 0	1 1 1 xsreg reg	2	reg16←xsreg						
		mem16, sreg	1 0 0 0 1 1 0 0	mod 0 sreg mem	2 to 4	(mem16)←sreg						
		mem16, xsreg*	1 0 0 0 1 1 0 0	mod 1 xsreg mem	2 to 4	(mem16)←xsreg						
		DS0, reg16, mem32	1 1 0 0 0 1 0 1	mod reg mem	2 to 4	reg16←(mem32) DS0←(mem32+2)						
		DS1, reg16, mem32	1 1 0 0 0 1 0 0	mod reg mem	2 to 4	reg16←(mem32) DS1←(mem32+2)						
		DS2, reg16* mem32	0 0 0 0 1 1 1 1 mod reg mem	0 0 1 1 1 1 1 0	3 to 5	reg16←(mem32) DS2←(mem32+2)						
		DS3, reg16* mem32	0 0 0 0 1 1 1 1 mod reg mem	0 0 1 1 0 1 1 0	3 to 5	reg16←(mem32) DS3←(mem32+2)						

* Instruction added to the instructions of V25 and V35.

Instruction Group	Mnemonic	Operand	Operation Code		Byte Number	Operation	Flag					
			7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0			AC	CY	V	P	S	Z
Repeat Prefix	MOV	AH, PSW	1 0 0 1 1 1 1 1		1	AH←S, Z, F1, AC, F0, P, IBRK, CY						
		PSW, AH	1 0 0 1 1 1 1 0		1	S, Z, F1, AC, F0, P, IBRK, CY←AH	×	×		×	×	×
	LDEA	reg16, mem16	1 0 0 0 1 1 0 1	mod reg mem	2 to 4	reg16←mem16						
	TRANS TRANSB*1	src-table	1 1 0 1 0 1 1 1		1	AL←(BW + AL)						
	XCH	reg, reg'	1 0 0 0 0 1 1 W	1 1 reg reg	2	reg↔reg'						
		mem, reg reg, mem	1 0 0 0 0 1 1 W	mod reg mem	2 to 4	(mem)↔reg						
		AW, reg16 reg16, AW	1 0 0 1 0 reg		1	AW↔reg16						
	MOVSPA*2		0 0 0 0 1 1 1 1	0 0 1 0 0 1 0 1	2	New register bank SS, SP + Old register bank SS, SP						
	MOVSPB*2	reg16	0 0 0 0 1 1 1 1	1 0 0 1 0 1 0 1	3							
			1 1 1 1 1 reg									
Data Transfer Instruction	REPC		0 1 1 0 0 1 0 1		1	While CW ≠ 0, the primitive block transfer instruction of the continuing byte is executed, and CW is decremented (−1). If there isa hold interrupt it is processed. When CY ≠ 1, it leaves the loop.						
	REPNC		0 1 1 0 0 1 0 0		1	Same as above. When CY ≠ 0, it leaves the loop.						
	REP REPE REPZ		1 1 1 1 0 0 1 1		1	While CW ≠ 0, the primitive block transfer instruction of the continuing byte is executed, and CW is decremented (−1). If there isa hold interrupt it is processed. If the primitive clock transfer instruction is CMPBK or CMPM, it leaves the loop when Z ≠ 1.						
	REPNE REPNZ		1 1 1 1 0 0 1 0		1	Same as above. When Z ≠ 0, it leaves the loop.						

* 1. For the TRANS instruction, the operand can be abbreviated. For the TRANSB instruction, there is no operand.
2. Instruction added to the instructions of V20 and V30.

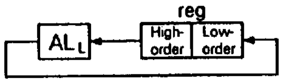
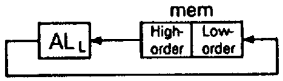
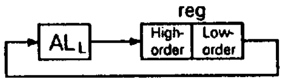
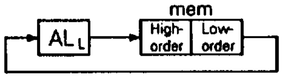
Instruction Group	Mnemonic	Operand	Operation Code		Byte Number	Operation	Flag					
			7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0			AC	CY	V	P	S	Z
Primitive Block Transfer Instructions	MOVBK MOVBKB* MOVBKW*	dst-block src-block	1 0 1 0 0 1 0 W		1	When W = 0, (IY)←(IX) DIR = 0: IX←IX + 1, IY←IY + 1 DIR = 1: IX←IX − 1, IY←IY − 1						
						When W = 1, (IY + 1, IY)(IX + 1, IX) DIR = 0: IX←IX + 2, IY←IY + 2 DIR = 1: IX←IX − 2, IY←IY − 2						
	CMPBK CMPBKB* CMPBKW*	src-block dst-block	1 0 1 0 0 1 1 W		1	When W = 0, (IY) − (IX) DIR = 0: IX←IX + 1, IY←IY + 1 DIR = 1: IX←IX − 1, IY←IY − 1	×	×	×	×	×	×
						When W = 1, (IY + 1, IY) − (IX + 1, IX) DIR = 0: IX←IX + 2, IY←IY + 2 DIR = 1: IX←IX − 2, IY←IY − 2						
	CMPM CMPMB* CMPMW*	dst-block	1 0 1 0 1 1 1 W		1	When W = 0, AL − (IY) DIR = 0, IY←IY + 1 ; DIR = 1, IY←IY − 1	×	×	×	×	×	×
						When W = 1, AW − (IY + 1, IY) DIR = 0, IY←IY + 2 ; DIR = 1, IY←IY − 2						
	LDM LDMB* LDMW*	src-block	1 0 1 0 1 1 0 W		1	When W = 0, AL←(IX) DIR = 0, IX←IX + 1 ; DIR = 1, IX←IX − 1						
						When W = 1, AW + (IX + 1, IX) DIR = 0, IX + 2 ; DIR = 1, IX←IX − 2						
	STM STMB* STMW*	dst-block	1 0 1 0 1 0 1 W		1	When W = 0, (IY)←AL DIR = 0, IY←IY + 1 ; DIR = 1, IY←IY − 1						
						When W = 1, AW − (IY + 1, IY)←AW DIR = 0, IY←IY + 2 ; DIR = 1, IY←IY − 2						

* Instruction added to the instructions of V20 and V30.

Instruction Group	Mnemonic	Operand	Operation Code		Byte Number	Operation	Flag					
			7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0			AC	CY	V	P	S	Z
Bit Field Manipulation Instructions	INS	reg8, reg8'	0 0 0 0 1 1 1 1 1 1 reg' reg	0 0 1 1 0 0 0 1	3	16-bit field←AW						
		reg8, imm4	0 0 0 0 1 1 1 1 1 1 0 0 0 reg	0 0 1 1 1 0 0 1	4	16-bit field←AW						
	EXT	reg8, reg8'	0 0 0 0 1 1 1 1 1 1 reg' reg	0 0 1 1 0 0 1 1	3	AW←16-bit field						
		reg8, imm4	0 0 0 0 1 1 1 1 1 1 0 0 0 reg	0 0 1 1 1 0 1 1	4	AW←16-bit field						
Input/output Instructions	IN*	acc, dmem	1 1 1 0 0 1 0 W		2	When W = 0, AL←(imm8) When W = 1, AH←(imm8 + 1), AL←(imm8)						
		acc, DW	1 1 1 0 1 1 0 W		1	When W = 0, AL←(DW) When W = 1, AH←(DW + 1), AL←(DW)						
	OUT*	imm8, acc	1 1 1 0 0 1 1 W		2	When W = 0, AL←(imm8) When W = 1, AH←(imm8+ 1), AL←(imm8)						
		DW, acc	1 1 1 0 1 1 1 W		1	When W = 0, (DW)←AL When W = 1, (DW+ 1)←AH, (DW)←AL						
Primitive Input /output Instructions	INM*	dst-block DW	0 1 1 0 1 1 0 W		1	When W = 0, (IY)←(DW) DIR = 0: IY←IY + 1 ; DIR = 1: IY←IY − 1 When W = 0, (IY + 1. IY)←(DW + 1, DW) DIR = 0: IY←IY + 2 ; DIR = 1: IY←IY − 2						
	OUTM*	DW, src-block	0 1 1 0 1 1 1 W		1	When W = 0, (DW)←(IX) DIR = 0: IX←IX + 1 ; DIR = 1: IX←IX − 1 When W = 0, (DW + 1, DW)←(IX + 1. IX) DIR = 0: IX←IX + 2 ; DIR = 1: IX←IX − 2						

* When $\overline{\text{IBRK}}$ = 0, a software interrupt is automatically generated and the instruction is not executed.

Instruction Group	Mnemonic	Operand	Operation Code		Byte Number	Operation	Flag					
			7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0			AC	CY	V	P	S	Z
Addition/subtraction Instructions	ADD	reg, reg'	0 0 0 0 0 0 1 W	1 1 reg reg'	2	reg←reg + reg'	×	×	×	×	×	×
		mem, reg	0 0 0 0 0 0 0 W	mod reg mem	2 to 4	(mem)←(mem) + reg	×	×	×	×	×	×
		reg, mem	0 0 0 0 0 0 1 W	mod reg mem	2 to 4	reg←reg + (mem)	×	×	×	×	×	×
		reg, imm	1 0 0 0 0 0 s W	1 1 0 0 0 reg	3 to 4	(mem)←reg + imm	×	×	×	×	×	×
		mem, imm	1 0 0 0 0 1 s W	mod 0 0 0 mem	3 to 6	reg←(mem) + imm	×	×	×	×	×	×
		acc, imm	0 0 0 0 0 1 0 W		2 to 3	When W = 0, AL←AL + imm When W = 1, AW←AW + imm	×	×	×	×	×	×
	ADDC	reg, reg'	0 0 0 1 0 0 1 W	1 1 reg reg'	2	reg←reg + reg' + CY	×	×	×	×	×	×
		mem, reg	0 0 0 1 0 0 0 W	mod reg mem	2 to 4	(mem)←(mem) + reg + CY	×	×	×	×	×	×
		reg, mem	0 0 0 1 0 0 1 W	mod reg mem	2 to 4	reg←reg + (mem) + CY	×	×	×	×	×	×
		reg, imm	1 0 0 0 0 0 s W	1 1 0 1 0 reg	3 to 4	(mem)←reg + imm + CY	×	×	×	×	×	×
		mem, imm	1 0 0 0 0 1 s W	mod 0 1 0 mem	3 to 6	reg←(mem) + imm + CY	×	×	×	×	×	×
		acc, imm	0 0 0 1 0 1 0 W		2 to 3	When W = 0, AL←AL + imm + CY When W = 1, AW←AW + imm + CY	×	×	×	×	×	×
	SUB	reg, reg'	0 0 1 0 1 0 1 W	1 1 reg reg'	2	reg←reg – reg'	×	×	×	×	×	×
		mem, reg	0 0 1 0 1 0 0 W	mod reg mem	2 to 4	(mem)←(mem) – reg	×	×	×	×	×	×
		reg, mem	0 0 1 0 1 0 1 W	mod reg mem	2 to 4	reg←reg – (mem)	×	×	×	×	×	×
		reg, imm	1 0 0 0 0 0 s W	1 1 1 0 1 reg	3 to 4	(mem)←reg – imm	×	×	×	×	×	×
		mem, imm	1 0 0 0 0 0 s W	mod 1 0 1 mem	3 to 6	reg←(mem) – imm	×	×	×	×	×	×
		acc, imm	0 0 1 0 1 1 0 W		2 to 3	When W = 0, AL←AL – imm When W = 1, AW←AW – imm	×	×	×	×	×	×
	SUBC	reg, reg'	0 0 0 1 1 0 1 W	1 1 reg reg'	2	reg←reg – reg' – CY	×	×	×	×	×	×
		mem, reg	0 0 0 1 1 0 0 W	mod reg mem	2 to 4	(mem)←(mem) – reg' – CY	×	×	×	×	×	×
		reg, mem	0 0 0 1 1 0 1 W	mod reg mem	2 to 4	reg←reg – (mem)' – CY	×	×	×	×	×	×
		reg, imm	1 0 0 0 0 0 s W	1 1 0 1 1 reg	3 to 4	(mem)←reg – imm' – CY	×	×	×	×	×	×
		mem, imm	1 0 0 0 0 0 s W	mod 0 1 1 mem	3 to 6	reg←(mem) – imm' – CY	×	×	×	×	×	×
		acc, imm	0 0 0 1 1 1 0 W		2 to 3	When W = 0, AL←AL – imm' – CY When W = 1, AW←AW – imm' – CY	×	×	×	×	×	×

Instruction Group	Mnemonic	Operand	Operation Code		Byte Number	Operation	Flag					
			7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0			AC	CY	V	P	S	Z
Addition/subtraction Instructions	ADD4S*1	(dst-string, src-string)	0 0 0 0 1 1 1 1	0 0 1 0 0 0 0 0	2	dst BCD string←dst BCD string + srv BCD string*2	U	×	U	U	U	×
	SUB4S*1	(dst-string, src-string)	0 0 0 0 1 1 1 1	0 0 1 0 0 0 1 0	2	dst BCD string←dst BCD string – srv BCD string*2	U	×	U	U	U	×
	CMP4S*1	(dst-string, src-string)	0 0 0 0 1 1 1 1	0 0 1 0 0 1 1 0	2	dst BCD string – srv BCD string*2	U	×	U	U	U	×
	ROL4	reg8	0 0 0 0 1 1 1 1 1 1 0 0 0 reg	0 0 1 0 1 0 0 0	3							
		mem8	0 0 0 0 1 1 1 1 mod 0 0 0 mem	0 0 1 0 1 0 0 0	3 to 5							
	ROR4	reg8	0 0 0 0 1 1 1 1 1 1 0 0 0 reg	0 0 1 0 1 0 1 0	3							
		mem8	0 0 0 0 1 1 1 1 mod 0 0 0 mem	0 0 1 0 1 0 1 0	3 to 5							
Increment/decrement Instructions	INC	reg8	1 1 1 1 1 1 1 0	1 1 0 0 0 reg	2	reg8←reg8 + 1	×		×	×	×	×
		mem	1 1 1 1 1 1 1 W	mod 0 0 0 mem	2 to 4	(mem)←(mem) + 1	×		×	×	×	×
		reg16	0 1 0 0 0 reg		1	reg16←reg16 + 1	×		×	×	×	×
	DEC	reg8	1 1 1 1 1 1 1 0	1 1 0 0 1 reg	2	reg8←reg8 – 1	×		×	×	×	×
		mem	1 1 1 1 1 1 1 W	mod 0 0 1 mem	2 to 4	(mem)←(mem) – 1	×		×	×	×	×
		reg16	0 1 0 0 1 reg		1	reg16←reg16 – 1	×		×	×	×	×

* 1. The operand can be abbreviated.

2. The number of BCD digites, given by CL register, can be set between 1 and 254.

Instruction Group	Mnemonic	Operand	Operation Code		Byte Number	Operation	Flag					
			7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0			AC	CY	V	P	S	Z
Multiplication Instructions	MULU	reg8	1 1 1 1 0 1 1 0	1 1 1 0 0 reg	2	AW←AL × reg8 AH = 0: CY←0, V←0 AH ≠ 0: CY←1, V←1	U	×	×	U	U	U
		mem8	1 1 1 1 0 1 1 0	mod 1 0 0 mod	2 to 4	AW←AL × (mem8) AH = 0: CY←0, V←0 AH ≠ 0: CY←1, V←1	U	×	×	U	U	U
		reg16	1 1 1 1 0 1 1 1	1 1 1 0 0 reg	2	DW, AW←AW × reg16 DW = 0: CY←0, V←0 DW ≠ 0: CY←1, V←1	U	×	×	U	U	U
		mem16	1 1 1 1 0 1 1 1	mod 1 0 0 mod	2 to 4	DW, AW←AW × (mem16) DW = 0: CY←0, V←0 DW ≠ 0: CY←1, V←1	U	×	×	U	U	U
	MUL	reg8	1 1 1 1 0 1 1 0	1 1 1 0 1 reg	2	AW←AL × reg8 AH = AL sign expansion: CY←0, V←0 AH ≠ AL sign expansion: CY←1, V←1	U	×	×	U	U	U
		mem8	1 1 1 1 0 1 1 0	mod 1 0 1 mem	2 to 4	AW←AL × (mem8) AH = AL sign expansion: CY←0, V←0 AH ≠ AL sign expansion: CY←1, V←1	U	×	×	U	U	U
		reg16	1 1 1 1 0 1 1 1	1 1 1 0 1 reg	2	DW, AW←AW × reg16 DW = AW sign expansion: CY←0, V←0 DW ≠ AW sign expansion: CY←1, V←1	U	×	×	U	U	U
		mem16	1 1 1 1 0 1 1 1	mod 1 0 1 mod	2 to 4	DW, AW←AW × (mem16) DW = AW sign expansion: CY←0, V←0 DW ≠ AW sign expansion: CY←1, V←1	U	×	×	U	U	U
		reg16, (reg16',)* imm8	0 1 1 0 1 0 1 1	1 1 reg reg'	2	reg16←reg16' × imm8 Product ≦ 16 bits: CY←0, V←0 Product > 16 bits: CY←1, V←1	U	×	×	U	U	U
		reg16, mem16, imm8	0 1 1 0 1 0 1 1	mod reg mem	3 to 5	AW←AL × reg8 Product ≦ 16 bits: CY←0, V←0 Product > 16 bits: CY←1, V←1	U	×	×	U	U	U
		reg16, (reg16',)* imm16	0 1 1 0 1 0 0 1	1 1 reg reg'	2	AW←AL × reg8 Product ≦ 16 bits: CY←0, V←0 Product > 16 bits: CY←1, V←1	U	×	×	U	U	U
		reg16, mem16, imm16	0 1 1 0 1 0 0 1	mod reg mem	4 to 6	AW←AL × reg8 Product ≦ 16 bits: CY←0, V←0 Product > 16 bits: CY←1, V←1	U	×	×	U	U	U

* The Second operand can be omitted. When it is omitted, the same register as that specified by the first operand is specified.

Instruction Group	Mnemonic	Operand	Operation Code		Byte Number	Operation	Flag					
			7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0			AC	CY	V	P	S	Z
Non-coded Division Instructions	DIVU	reg8	1 1 1 1 0 1 1 0	1 1 1 1 0 reg	2	temp←AW When temp + reg8 ≦ FFH AH←temp%reg8, AL←temp + reg8 When temp + reg8 > FFH (SP − 1, SP − 2)←PSW, (SP − 3, SP − 4)←PS (SP − 5, SP − 6)←PC, SP←SP − 6 IE←0, BRK←0, PS←(3, 2), PC←(1, 0)	U	U	U	U	U	U
		mem8	1 1 1 1 0 1 1 0	mod 1 1 0 mem	2 to 4	temp←AW When temp + (mem8) ≦ FFH AH←temp%(mem8), AL←temp + (mem8) When temp + (mem8) > FFH (SP − 1, SP − 2)←PSW, (SP − 3, SP − 4)←PS (SP − 5, SP − 6)←PC, SP←SP − 6 IE←0, BRK←0, PS←(3, 2), PC←(1, 0)	U	U	U	U	U	U
		reg16	1 1 1 1 0 1 1 1	1 1 1 1 0 reg	2	temp←DW, AW When temp + reg16 ≦ FFFFH DW←temp%reg16, AW←temp + reg16 When temp + reg16 > FFFFH (SP − 1, SP − 2)←PSW, (SP − 3, SP − 4)←PS (SP − 5, SP − 6)←PC, SP←SP − 6 IE←0, BRK←0, PS←(3, 2), PC←(1, 0)	U	U	U	U	U	U
		mem16	1 1 1 1 0 1 1 0	mod 1 1 0 mem	2 to 4	temp←DW, AW When temp + (mem16) ≦ FFFFH DW←temp%(mem16), AW←temp + (mem16) When temp + (mem16) > FFFFH (SP − 1, SP − 2)←PSW, (SP − 3, SP − 4)←PS (SP − 5, SP − 6)←PC, SP←SP − 6 IE←0, BRK←0, PS←(3, 2), PC←(1, 0)	U	U	U	U	U	U

Instruction Group	Mnemonic	Operand	Operation Code		Byte Number	Operation	Flag					
			7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0			AC	CY	V	P	S	Z
Coded Division Instructions	DIV	reg8	1 1 1 1 0 1 1 0	1 1 1 1 1 reg	2	temp←AW When temp + reg8 > 0 and temp + reg8 ≦ 7FH; or when temp + reg8 < 0 and temp + reg8 > 0 – 7FH – 1 AH←temp%reg8, AL←temp + reg8 When temp + reg8 > 0 and temp + reg8 > 7FH; or when temp + reg8 < 0 and temp + reg8 ≦ 0 – 7FH – 1 (SP – 1, SP – 2)←PSW, (SP – 3, SP – 4)←PS (SP – 5, SP – 6)←PC, SP←SP – 6 IE←0, BRK←0, PS←(3, 2), PC←(1, 0)	U	U	U	U	U	U
		mem8	1 1 1 1 0 1 1 0	mod 1 1 1 mem	2 to 4	temp←AW When temp + (mem8) > 0 and temp + (mem8) ≦ 7FH; or when temp + (mem8) and temp + (mem8) > 0 – 7FH – 1 AH←temp%(mem8), AL←temp + (mem8) When temp + (mem8) > 0 and temp + (mem8) > 7FH; or when temp + (mem8) < 0 and temp + (mem8) ≦ 0 – 7FH – 1 (SP – 1, SP – 2)←PSW, (SP – 3, SP – 4)←PS (SP – 5, SP – 6)←PC, SP←SP – 6 IE←0, BRK←0, PS←(3, 2), PC←(1, 0)	U	U	U	U	U	U
		reg16	1 1 1 1 0 1 1 1	1 1 1 1 1 reg	2	temp←DW, AW When temp + reg16 > 0 and temp + reg16 ≦ 7FFFH; or when temp + reg16 < 0 and temp + reg16 > 0 – 7FFFH – 1 DW←temp%reg16, AW←temp + reg16 When temp + reg16 > 0 and temp + reg16 > 7FFFH; or when temp + reg16 < 0 and temp + reg16 ≦ 0 – 7FFFH – 1 (SP – 1, SP – 2)←PSW, (SP – 3, SP – 4)←PS (SP – 5, SP – 6)←PC, SP←SP – 6 IE←0, BRK←0, PS←(3, 2), PC←(1, 0)	U	U	U	U	U	U
		mem16	1 1 1 1 0 1 1 1	mod 1 1 1 mem	2 to 4	temp←DW, AW When temp + (mem16) > 0 and temp + (mem16) ≦ 7FFFH; or when temp + (mem16) and temp + (mem16) > 0 – 7FFFH – 1 AH←temp%(mem16), AL←temp + (mem16) When temp + (mem16) > 0 and temp + (mem16) > 7FFFH; or when temp + (mem16) < 0 and temp + (mem16) ≦ 0 – 7FFFH – 1 (SP – 1, SP – 2)←PSW, (SP – 3, SP – 4)←PS (SP – 5, SP – 6)←PC, SP←SP – 6 IE←0, BRK←0, PS←(3, 2), PC←(1, 0)	U	U	U	U	U	U

Instruction Group	Mnemonic	Operand	Operation Code		Byte Number	Operation	Flag					
			7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0			AC	CY	V	P	S	Z
BCD Correction Instruction	ADJBA		0 0 1 1 0 1 1 1		1	When AL ∧ 0FH > 9 or AC = 1, AL←AL + 6 AH←AH + 1, AC←1, CY←AC, AL←AL ∧ 0FH	×	×	U	U	U	U
	ADJ4A		0 0 1 0 0 1 1 1		1	When AL ∧ 0FH > 9 or AC = 1 AL←AL + 6, AC←1 When AL > 9FH or CY = 1 AL←AL + 60H, CY←1	×	×	U	×	×	×
	ADJBS		0 0 1 1 1 1 1 1		1	When AL ∧ 0FH > 9 or AC = 1 AL←AL - 6, AH←AH - 1, AC←1 CY←AC, AL←AL ∧ 0FH	×	×	U	U	U	U
	ADJ4S		0 0 1 0 1 1 1 1		1	When AL ∧ 0FH > 9 or AC = 1 AL←AL - 6, AC←1 When AL > 9FH or CY = 1 AL←AL - 60H, CY←1	×	×	U	×	×	×
*1	CVTBD		1 1 0 1 0 1 0 0	0 0 0 0 1 0 1 0	2	AH←AH + 0AH, AL←AL%0AH	U	U	U	×	×	×
	CVTDB		1 1 0 1 0 1 0 1	0 0 0 0 1 0 1 0	2	AL←AH × 0AH + AL, AH←0	U	U	U	×	×	×
	CVTBW		1 0 0 1 1 0 0 0		1	When AL < 80H, AH←0, for all other times AH←FFH						
	CVTWL		1 0 0 1 1 0 0 1		1	When AW < 8000H, DW←0, for all other times DW←FFFFH						
Comparison Instructions	CMP	reg, reg'	0 0 1 1 1 0 1 W	1 1 reg reg	2	reg - reg'	×	×	×	×	×	×
		mem, reg	0 0 1 1 1 0 0 W	mod reg reg	2 to 4	(mem)- reg	×	×	×	×	×	×
		reg, mem	0 0 1 1 1 0 1 W	mod reg mem	2 to 4	reg - (mem)	×	×	×	×	×	×
		reg, imm	1 0 0 0 0 0 S W	1 1 1 1 1 reg	3 to 4	reg - imm	×	×	×	×	×	×
		mem, imm	1 0 0 0 0 0 S W	mod 1 1 1 mem	3 to 6	(mem) - imm	×	×	×	×	×	×
		acc, imm	0 0 1 1 1 1 0 W		2 to 3	When W = 0, AL - imm When W = 1, AW - imm	×	×	×	×	×	×
*2	NOT	reg	1 1 1 1 0 1 1 W	1 1 0 1 0 reg	2	reg←$\overline{\text{reg}}$						
		mem	1 1 1 1 0 1 1 W	mod 0 1 0 mem	2 to 4	(mem) - ($\overline{\text{mem}}$)						
	NEG	reg	1 1 1 1 0 1 1 W	1 1 0 1 1 reg	2	reg←$\overline{\text{reg}}$ + 1	×	×	×	×	×	×
		mem	1 1 1 1 0 1 1 W	mod 0 1 1 mem	2 to 4	(mem)←($\overline{\text{mem}}$) + 1	×	×	×	×	×	×

* 1. Data transformation instructions
2. Complimentary operation instructions

Instruction Group	Mnemonic	Operand	Operation Code		Byte Number	Operation	Flag					
			7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0			AC	CY	V	P	S	Z
Logical Operation Instructions	TEST	reg, reg'	1 0 0 0 0 1 0 W	1 1 reg' reg	2	reg ∧ reg'	U	0	0	×	×	×
		mem, reg reg, mem	1 0 0 0 0 1 0 W	mod reg mem	2 to 4	(mem) ∧ reg	U	0	0	×	×	×
		reg, imm	1 1 1 1 0 1 1 W	1 1 0 0 0 reg	3 to 4	reg ∧ imm	U	0	0	×	×	×
		mem, imm	1 1 1 1 0 1 1 W	mod 0 0 0 mem	3 to 6	(mem) ∧ imm	U	0	0	×	×	×
		acc, imm	1 0 1 0 1 0 0 W		2 to 3	When W = 0, AL ∧ imm When W = 1, AW ∧ imm	U	0	0	×	×	×
	AND	reg, reg'	0 0 1 0 0 0 1 W	1 1 reg reg'	2	reg←reg ∧ reg'	U	0	0	×	×	×
		mem, reg	0 0 1 0 0 0 0 W	mod reg mem	2 to 4	(mem)←(mem) ∧ reg	U	0	0	×	×	×
		reg, mem	0 0 1 0 0 0 1 W	mod reg mem	2 to 4	reg←reg ∧ (mem)	U	0	0	×	×	×
		reg, imm	1 0 0 0 0 0 0 W	1 1 1 0 0 reg	3 to 4	reg←reg ∧ imm	U	0	0	×	×	×
		mem, imm	1 0 0 0 0 0 0 W	mod 1 0 0 mem	3 to 6	(mem)←(mem) ∧ imm	U	0	0	×	×	×
		acc, imm	0 0 1 0 0 1 0 W		2 to 3	When W = 0, AL←AL ∧ imm8 When W = 1, AW←AW ∧ imm16	U	0	0	×	×	×
	OR	reg, reg'	0 0 0 0 1 0 1 W	1 1 reg reg'	2	reg←reg ∨ reg'	U	0	0	×	×	×
		mem, reg	0 0 0 0 1 0 0 W	mod reg mem	2 to 4	(mem)←(mem) ∨ reg	U	0	0	×	×	×
		reg, mem	0 0 0 0 1 0 1 W	mod reg mem	2 to 4	reg←reg ∨ (mem)	U	0	0	×	×	×
		reg, imm	1 0 0 0 0 0 0 W	1 1 0 0 1 reg	3 to 4	reg←reg ∨ imm	U	0	0	×	×	×
		mem, imm	1 0 0 0 0 0 0 W	mod 0 0 1 mem	3 to 6	(mem)←(mem) ∨ imm	U	0	0	×	×	×
		acc,imm	0 0 0 0 1 1 0 W		2-3	When W = 0, AL←AL ∨ imm8 When W = 1, AW←AW ∨ imm16	U	0	0	×	×	×
	XOR	reg, reg'	0 0 1 1 0 0 1 W	1 1 reg reg'	2	reg←reg ⊻ reg'	U	0	0	×	×	×
		mem, reg	0 0 1 1 0 0 0 W	mod reg mem	2 to 4	(mem)←(mem) ⊻ reg'	U	0	0	×	×	×
		reg, mem	0 0 1 1 0 0 1 W	mod reg mem	2 to 4	reg←reg ⊻ (mem)	U	0	0	×	×	×
		reg, imm	1 0 0 0 0 0 0 W	1 1 1 1 0 reg	3 to 4	reg←reg ⊻ imm	U	0	0	×	×	×
		mem, imm	1 0 0 0 0 0 0 W	mod 1 1 0 mem	3 to 6	(mem)←(mem) ⊻ imm	U	0	0	×	×	×
		acc, imm	0 0 1 1 0 1 0 W		2 to 3	When W = 0, AL←AL ⊻ imm8 When W = 1, AW←AW ⊻ imm16	U	0	0	×	×	×

Instruction Group	Mnemonic	Operand	Operation Code		Byte Number	Operation	Flag					
			7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0			AC	CY	V	P	S	Z
Bit Manipulation Instructions	TEST1	reg8, CL	0 0 0 1 0 0 0 0	1 1 0 0 0 reg	3	reg8 bit NO.CL = 0 : Z←1 reg8 bit NO.CL = 1 : Z←0	U	0	0	U	U	×
		mem8, CL	0 0 0 0	mod 0 0 0 mem	3 to 5	(mem)8 bit NO.CL = 0 : Z←1 (mem)8 bit NO.CL = 1 : Z←0	U	0	0	U	U	×
		reg16, CL	0 0 0 1	1 1 0 0 0 reg	3	reg16 bit NO.CL = 0 : Z←1 reg16 bit NO.CL = 1 : Z←0	U	0	0	U	U	×
		mem16, CL	0 0 0 1	mod 0 0 0 mem	3 to 5	(mem16) bit NO.CL = 0 : Z←1 (mem16) bit NO.CL = 1 : Z←0	U	0	0	U	U	×
		reg8, imm3	1 0 0 0	1 1 0 0 0 reg	4	reg8 bit NO.imm3 = 0 : Z←1 reg8 bit NO.imm3 = 1 : Z←0	U	0	0	U	U	×
		mem8, imm3	1 0 0 0	mod 0 0 0 mem	4 to 6	(mem8) bit NO.imm3 = 0 : Z←1 (mem8) bit NO.imm3 = 1 : Z←0	U	0	0	U	U	×
		reg16, imm4	1 0 0 1	1 1 0 0 0 reg	4	reg16 bit NO.imm4 = 0 : Z←1 reg16 bit NO.imm4 = 1 : Z←0	U	0	0	U	U	×
		mem16, imm4	1 0 0 0	mod 0 0 0 mem	4 to 6	(mem16) bit NO.imm4 = 0 : Z←1 (mem16) bit NO.imm4 = 1 : Z←0	U	0	0	U	U	×
	NOT1	reg8, CL	0 1 1 0	1 1 0 0 0 reg	3	reg8 bit NO.CL←$\overline{\text{reg8 bit NO.CL}}$						
		mem8, CL	0 1 1 0	mod 0 0 0 mem	3 to 5	(mem8) bit NO.CL←$\overline{\text{(mem8) bit NO.CL}}$						
		reg16, CL	0 1 1 1	1 1 0 0 0 reg	3	reg16 bit NO.CL←$\overline{\text{reg16 bit NO.CL}}$						
		mem16, CL	0 1 1 1	mod 0 0 0 mem	3 to 5	(mem16) bit NO.CL←$\overline{\text{(mem16) bit NO.CL}}$						
		reg8, imm3	1 1 1 0	1 1 0 0 0 reg	4	reg8 bit NO.imm3←$\overline{\text{reg8 bit NO.imm3}}$						
		mem8, imm3	1 1 1 0	mod 0 0 0 mem	4 to 6	(mem8) bit NO.imm3←$\overline{\text{(mem8) bit NO.imm4}}$						
		reg16, imm4	1 1 1 1	1 1 0 0 0 reg	4	reg16 bit NO.imm4←$\overline{\text{reg16 bit NO.imm3}}$						
		mem16, imm4	1 1 1 1	mod 0 0 0 mem	4 to 6	(mem16) bit NO.imm4←$\overline{\text{(mem16) bit NO.imm4}}$						
			2nd byte *	3rd byte *								

* 1st byte = 0FH

	NOT1	CY	1 1 1 1 0 1 0 1		1	CY←$\overline{\text{CY}}$		×				

Instruction Group	Mnemonic	Operand	Operation Code		Byte Number	Operation	Flag					
			7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0			AC	CY	V	P	S	Z
Bit Manipulation Instructions	CLR1	reg8, CL	0 0 0 1 0 0 1 0	1 1 0 0 0 reg	3	reg8 bit NO.CL←0						
		mem8, CL	0 0 1 0	mod 0 0 0 mem	3 to 5	(mem8) bit NO.CL←0						
		reg16, CL	0 0 1 1	1 1 0 0 0 reg	3	reg16 bit NO.CL←0						
		mem16, CL	0 0 1 1	mod 0 0 0 mem	3 to 5	(mem16) bit NO.CL←0						
		reg8,imm3	1 0 1 0	1 1 0 0 0 reg	4	reg8 bit NO.imm3←0						
		mem8, imm3	1 0 1 0	mod 0 0 0 mem	4 to 6	(mem8) bit NO.imm3←0						
		reg16, imm4	1 0 1 1	1 1 0 0 0 reg	4	reg16 bi NO.imm4←0						
		mem16, imm4	1 0 1 1	mod 0 0 0 mem	4 to 6	(mem16) bit NO.imm4←0						
	SET1	reg8, CL	0 1 0 0	1 1 0 0 0 reg	3	reg8 bit NO.CL←1						
		mem8, CL	0 1 0 0	mod 0 0 0 mem	3 to 5	(mem8) bit NO.CL←1						
		reg16, CL	0 1 0 1	1 1 0 0 0 reg	3	reg16 bit NO.CL←1						
		mem16, CL	0 1 0 1	mod 0 0 0 mem	3 to 5	(mem16) bit NO.CL←1						
		reg8,imm3	1 1 0 0	1 1 0 0 0 reg	4	reg8 bit NO.imm3←1						
		mem8, imm3	1 1 0 0	mod 0 0 0 mem	4 to 6	(mem8) bit NO.imm3←1						
		reg16, imm4	1 1 0 1	1 1 0 0 0 reg	4	reg16 bi NO.imm4←1						
		mem16, imm4	1 1 0 1	mod 0 0 0 mem	4 to 6	(mem16) bit NO.imm4←1						
			2nd byte *	3rd byte *	* 1st byte = 0FH							

	CLR1	CY	1 1 1 1 1 0 0 0		1	CY←0		0				
		DIR	1 1 1 1 1 1 0 0		1	DIR←0						
	SET1	CY	1 1 1 1 1 0 0 1		1	CY←1		1				
		DIR	1 1 1 1 1 1 0 1		1	DIR←0						

Instruction Group	Mnemonic	Operand	Operation Code		Byte Number	Operation	Flag					
			7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0			AC	CY	V	P	S	Z
Bit Manipulation Instructions	BSCH*	reg8	0 0 0 0 1 1 1 1 1 1 0 0 0 reg	0 0 1 1 1 1 0 0	3	"1" is searched for in order started from bit 0 of reg8, The bit NO. searched first→CL If there is no "1"→ Z = 1	U	U	U	U	U	×
		mem8	0 0 0 0 1 1 1 1 mod 0 0 0 reg	0 0 1 1 1 1 0 0	3 to 5	"1" is searched for in order started from bit 0 of (mem8), The bit NO. searched first→CL If there is no "1"→ Z = 1	U	U	U	U	U	×
		reg16	0 0 0 0 1 1 1 1 1 1 0 0 0 reg	0 0 1 1 1 1 0 1	3	"1" is searched for in order started from bit 0 of reg16, The bit NO. searched first→CL If there is no "1"→ Z = 1	U	U	U	U	U	×
		mem16	0 0 0 0 1 1 1 1 mod 0 0 0 reg	0 0 1 1 1 1 0 1	3 to 5	"1" is searched for in order started from bit 0 of (mem16), The bit NO. searched first→CL If there is no "1"→ Z = 1	U	U	U	U	U	×
Que Manipulation Instructions	QHOUT*	imm16	0 0 0 0 1 1 1 1	0 1 1 1 0 0 0 0	4	The block queued header is removed, and the segment is stored in P2.	U	U	U	U	U	×
	QOUT*	imm16	0 0 0 0 1 1 1 1	0 1 1 1 0 0 0 1	4	The queue block indicated by P2 is removed.	U	U	U	U	U	×
	QTIN*	imm16	0 0 0 0 1 1 1 1	0 1 1 1 0 0 1 0	4	The block indicated by P2 is queued in at the very end.						

* Instruction added to the instructions of V25 and V35.

Remarks P2: Parameter table (in the register file)

Instruction Group	Mnemonic	Operand	Operation Code		Byte Number	Operation	Flag					
			7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0			AC	CY	V	P	S	Z
Shift Instructions	SHL	reg, 1	1 1 0 1 0 0 0 W	1 1 1 0 0 reg	2	CY←reg MSB, reg←reg × 2 When reg MSB ≠ CY, V←1 When reg MSB = CY, V←0	U	×	×	×	×	×
		mem, 1	1 1 0 1 0 0 0 W	mod1 0 0 mem	2 to 4	CY←(mem) MSB, (mem)←(mem) × 2 When (mem) MSB ≠ CY, V←1 When (mem) MSB = CY, V←0	U	×	×	×	×	×
		reg, CL	1 1 0 1 0 0 1 W	1 1 1 0 0 reg	2	While temp←CL and temp ≠ 0, the next operation repeats. CY←reg MSB, reg←reg × 2 temp←temp – 1	U	×	U	×	×	×
		mem, CL	1 1 0 1 0 0 1 W	mod1 0 0 mem	2 to 4	While temp←CL and temp ≠ 0, the next operation repeats. CY←(mem) MSB, (mem)←(mem) × 2 temp←temp – 1	U	×	U	×	×	×
		reg, imm8	1 1 0 0 0 0 0 W	1 1 1 0 0 reg	3	While temp←imm8 and temp ≠ 0, the next operation repeats. CY←reg MSB, reg←reg × 2 temp←temp – 1	U	×	U	×	×	×
		mem, imm8	1 1 0 0 0 0 0 W	mod1 0 0 mem	3 to 5	While temp←imm8 and temp ≠ 0, the next operation repeats. CY←(mem) MSB, (mem)←(mem) × 2 temp←temp – 1	U	×	U	×	×	×
	SHR	reg, 1	1 1 0 1 0 0 0 W	1 1 1 0 1 reg	2	CY←reg LSB, reg←reg + 2 reg MSB ≠ next bit of reg MSB: V←1 reg MSB = next bit of reg MSB: V←0	U	×	×	×	×	×
		mem, 1	1 1 0 1 0 0 0 W	mod1 0 1 mem	2 to 4	CY←(mem) LSB, (mem)←(mem) + 2 (mem) MSB ≠ next bit of (mem) MSB: V←1 (mem) MSB = next bit of (mem) MSB: V←0	U	×	×	×	×	×
		reg, CL	1 1 0 1 0 0 1 W	1 1 1 0 1 reg	2	While temp←CL and temp ≠ 0, the next operation repeats. CY←reg LSB, reg←reg + 2 temp←temp – 1	U	×	U	×	×	×
		mem, CL	1 1 0 1 0 0 1 W	mod1 0 1 mem	2 to 4	While temp←CL and temp ≠ 0, the next operation repeats. CY←(mem) LSB, (mem)←(mem) + 2 temp←temp – 1	U	×	U	×	×	×
		reg, imm8	1 1 0 0 0 0 0 W	1 1 1 0 1 reg	3	While temp←imm8 and temp ≠ 0, the next operation repeats. CY←reg LSB, reg←reg + 2 temp←temp – 1	U	×	U	×	×	×
		mem, imm8	1 1 0 0 0 0 0 W	mod1 0 1 mem	3 to 5	While temp←imm8 and temp ≠ 0, the next operation repeats. CY←(mem) LSB, (mem)←(mem) + 2 temp←temp – 1	U	×	U	×	×	×

Instruction Group	Mnemonic	Operand	Operation Code		Byte Number	Operation	Flag					
			7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0			AC	CY	V	P	S	Z
Shift Instructions	SHRA	reg, 1	1 1 0 1 0 0 0 W	1 1 1 1 1 reg	2	CY←reg LSB, reg←reg ÷ 2 The operand MSB does not change.	U	×	0	×	×	×
		mem, 1	1 1 0 1 0 0 0 W	mod 1 1 1 mem	2 to 4	CY←(mem) LSB, (mem)←(mem) ÷ 2 The operand MSB does not change.	U	×	0	×	×	×
		reg, CL	1 1 0 1 0 0 1 W	1 1 1 1 1 reg	2	While temp←CL and temp ≠ 0, the next operation repeats. CY←reg LSB, reg←reg ÷ 2 temp←temp – 1 and the operand MSB does not change.	U	×	U	×	×	×
		mem, CL	1 1 0 1 0 0 1 W	mod 1 1 1 mem	2 to 4	While temp←CL and temp ≠ 0, the next operation repeats. CY←(mem) LSB, (mem)←(mem) ÷ 2 temp←temp – 1 and the operand MSB does not change.	U	×	U	×	×	×
		reg, imm8	1 1 0 0 0 0 0 W	1 1 1 1 1 reg	3	While temp←imm8 and temp ≠ 0, the next operation repeats. CY←reg LSB, reg←reg ÷ 2 temp←temp – 1 and the operand MSB does not change.	U	×	U	×	×	×
		mem, imm8	1 1 0 0 0 0 0 W	mod 1 1 1 mem	3 to 5	While temp←imm8 and temp ≠ 0, the next operation repeats. CY←(mem) LSB, (mem)←(mem) ÷ 2 temp←temp – 1 and the operand MSB does not change.	U	×	U	×	×	×
Rotate Instructions	ROL	reg, 1	1 1 0 1 0 0 0 W	1 1 0 0 0 reg	2	CY←reg MSB, reg←reg × 2 + CY reg MSB ≠ CY: V←1 reg MSB = CY: V←0		×	×			
		mem, 1	1 1 0 1 0 0 0 W.	mod 0 0 0 mem	2 to 4	CY←(mem) MSB, (mem)←(mem) × 2 + CY (mem) MSB ≠ CY: V←1 (mem) MSB = CY: V←0		×	×			
		reg, CL	1 1 0 1 0 0 1 W	1 1 0 0 0 reg	2	While temp←CL and temp ≠ 0, the next operation repeats. CY←reg MSB, reg←reg × 2 + CY temp←temp – 1		×	U			
		mem, CL	1 1 0 1 0 0 1 W	mod 0 0 0 mem	2 to 4	While temp←CL and temp ≠ 0, the next operation repeats. CY←(mem) MSB, (mem)←(mem) × 2 + CY temp←temp – 1		×	U			
		reg, imm8	1 1 0 0 0 0 0 W	1 1 0 0 0 reg	3	While temp←imm8 and temp ≠ 0, the next operation repeats. CY←reg MSB, reg←reg × 2 + CY temp←temp – 1		×	U			
		mem, imm8	1 1 0 0 0 0 0 W	mod 0 0 0 mem	3 to 5	While temp←imm8 and temp ≠ 0, the next operation repeats. CY←(mem) MSB, (mem)←(mem) × 2 + CY temp←temp – 1		×	U			

Instruction Group	Mnemonic	Operand	Operation Code		Byte Number	Operation	Flag					
			7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0			AC	CY	V	P	S	Z
Rotate Instructions	ROR	reg, 1	1 1 0 1 0 0 0 W	1 1 0 0 1 reg	2	CY←reg LSB, reg←reg + 2 reg MSB←CY reg MSB ≠ next bit of reg MSB: V←1 reg MSB = next bit of reg MSB: V←0		×	×			
		mem, 1	1 1 0 1 0 0 0 W	mod 0 0 1 mem	2 to 4	CY←(mem) LSB, (mem)←(mem) + 2 (mem) MSB←CY (mem) MSB ≠ next bit of (mem) MSB: V←1 (mem) MSB = next bit of (mem) MSB: V←0		×	×			
		reg, CL	1 1 0 1 0 0 1 W	1 1 0 0 1 reg	2	While temp←CL and CL ≠ 0, the next operation repeats. CY←reg LSB, reg←reg + 2 reg MSB←CY temp←temp − 1		×	U			
		mem, CL	1 1 0 1 0 0 1 W	mod 0 0 1 mem	2	While temp←CL and CL ≠ 0, the next operation repeats. CY←(mem) LSB, (mem)←(mem) + 2 (mem) MSB←CY temp←temp − 1		×	U			
		reg, imm8	1 1 0 0 0 0 0 W	1 1 0 0 1 reg	3	While temp←imm8 and CL ≠ 0, the next operation repeats. CY←reg LSB, reg←reg + 2 reg MSB←CY temp←temp − 1		×	U			
		mem, imm8	1 1 0 0 0 0 0 W	mod 0 0 1 mem	3 to 5	While temp←imm8 and CL ≠ 0, the next operation repeats. CY←(mem) LSB, (mem)←(mem) + 2 (mem) MSB←CY temp←temp − 1		×	U			
	ROLC	reg, 1	1 1 0 1 0 0 0 W	1 1 0 1 0 reg	2	tmpcy←CY, CY←reg MSB reg←reg × 2 + tmpcy reg MSB ≠ CY: V←1 reg MSB = CY: V←0		×	×			
		mem, 1	1 1 0 1 0 0 0 W	mod 0 1 0 mem	2 to 4	tmpcy←CY, CY←(mem) MSB (mem)←(mem) × 2 + tmpcy (mem) MSB ≠ CY: V←1 (mem) MSB = CY: V←0		×	×			
		reg, CL	1 1 0 1 0 0 1 W	1 1 0 1 0 reg	2	While temp←CL and CL ≠ 0, the next operation repeats. tmpcy←CY, CY←reg MSB reg←reg × 2 + tmpcy temp←temp − 1		×	U			
		mem, CL	1 1 0 1 0 0 1 W	mod 0 1 0 mem	2 to 4	While temp←CL and CL ≠ 0, the next operation repeats. tmpcy←CY, CY←(mem) MSB (mem)←(mem) × 2 + tmpcy temp←temp − 1		×	U			

Instruction Group	Mnemonic	Operand	Operation Code		Byte Number	Operation	Flag					
			7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0			AC	CY	V	P	S	Z
Rotate Instructions	ROLC	reg, imm8	1 1 0 0 0 0 0 W	1 1 0 1 0 reg	3	While temp←imm8 and CL ≠ 0, the next operation repeats. tmpcy←CY, CY←reg MSB reg←reg × 2 + tmpcy temp←temp – 1		×	U			
		mem, imm8	1 1 0 0 0 0 0 W	mod 0 1 0 mem	3 to 5	While temp←imm8 and CL ≠ 0, the next operation repeats. tmpcy←CY, CY←(mem) MSB (mem)←(mem) × 2 + tmpcy temp←temp – 1		×	U			
	RORC	reg, 1	1 1 0 1 0 0 0 W	1 1 0 1 1 reg	2	tmpcy←CY, CY←reg LSB reg←reg ÷ 2 reg MSB←tmpcy reg MSB ≠ next bit of reg MSB: V←1 reg MSB = next bit of reg MSB: V←0		×	×			
		mem, 1	1 1 0 1 0 0 0 W	mod 0 1 1 mem	2 to 4	tmpcy←CY, CY←(mem) LSB (mem)←(mem) ÷ 2 (mem) MSB←tmpcy (mem) MSB ≠ next bit of (mem) MSB: V←1 (mem) MSB = next bit of (mem) MSB: V←0		×	×			
		reg, CL	1 1 0 1 0 0 1 W	1 1 0 1 1 reg	2	While temp←CL and CL ≠ 0, the next operation repeats. tmpcy←CY, CY←reg LSB reg←reg ÷ 2 reg MSB←tmpcy temp←temp – 1		×	U			
		mem, CL	1 1 0 1 0 0 1 W	mod 0 1 1 mem	2 to 4	While temp←CL and CL ≠ 0, the next operation repeats. tmpcy←CY, CY←(mem) LSB (mem)←(mem) ÷ 2 (mem) MSB←tmpcy temp←temp – 1		×	U			
		reg, imm8	1 1 0 0 0 0 0 W	1 1 0 1 1 reg	3	While temp←imm8 and CL ≠ 0, the next operation repeats. tmpcy←CY, CY←reg LSB reg←reg ÷ 2 reg MSB←tmpcy temp←temp – 1		×	U			
		mem, imm8	1 1 0 0 0 0 0 W	mod 0 1 1 mem	3 to 5	While temp←imm8 and CL ≠ 0, the next operation repeats. tmpcy←CY, CY←(mem) LSB reg←reg ÷ 2 (mem) MSB←tmpcy temp←temp – 1		×	U			

Instruction Group	Mnemonic	Operand	Operation Code		Byte Number	Operation	Flag					
			7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0			AC	CY	V	P	S	Z
Subroutine Control Instructions	CALL	near-proc	1 1 1 0 1 0 0 0		3	(SP – 1, SP – 2)←PC, SP←SP – 2 PC←PC + disp						
		regptr16	1 1 1 1 1 1 1 1	1 1 0 1 0 reg	2	(SP – 1, SP – 2)←PC, SP←regptr16 SP←SP – 2						
		memptr16	1 1 1 1 1 1 1 1	mod 0 1 0 mem	2 to 4	(SP – 1, SP – 2)←PC, SP←SP – 2 PC←(memptr16)						
		far-proc	1 0 0 1 1 0 1 0		5	(SP – 1, SP – 2)←PC, (SP – 3, SP – 4←)PC SP←SP – 4 PC←seg, PC←offset						
		memptr32	1 1 1 1 1 1 1 1	mod 0 1 0 mem	2 to 4	(SP – 1, SP – 2)←PS, (SP – 3, SP – 4←)PC SP←SP – 4 PC←(memptr32 + 2), PC←(memptr32)						
	RET		1 1 0 0 0 0 1 1		1	PC←(SP + 1, SP) SP←SP + 2						
		pop-value	1 1 0 0 0 0 1 0		3	PC←(SP + 1, SP) SP←SP + 2, SP←SP + pop-value						
			1 1 0 0 1 0 1 1		1	PC←(SP + 1, SP) PS←(SP + 3, SP + 2) SP←SP + 4						
		pop-value	1 1 0 0 1 0 1 0		3	PC←(SP + 1, SP) PS←(SP + 3, SP + 2) SP←SP + 4, SP←SP + pop-value						
Stack Manipulation Instruction	PUSH	mem16	1 1 1 1 1 1 1 1	mod 1 1 0 mem	2 to 4	(SP – 1, SP – 2)←(mem16) SP←SP – 2						
		reg16	0 1 0 1 0 reg		1	(SP – 1, SP – 2)←reg16 SP←SP – 2						
		sreg	0 1 0 sreg 1 1 1		1	(SP – 1, SP – 2)←sreg SP←SP – 2						
		PSW	1 0 0 1 1 1 0 0		1	(SP – 1, SP – 2)←PSW SP←SP – 2						
		R	0 1 1 0 0 0 0 0		1	Push registers on the stack						
		imm	0 1 1 0 1 0 S 0		2 to 3	(SP – 1, SP – 2)←imm Sign expansion when SP←SP – 2 and S = 1.						
		DS2*	0 0 0 0 1 1 1 1	0 0 1 1 1 1 1 0	2	(SP – 1, SP – 2)←DS2 SP←SP – 2						
		DS3/VPC*	0 0 0 0 1 1 1 1	0 0 1 1 0 1 1 0	2	(SP – 1, SP – 2)←DS3/VPC SP←SP – 2						

* Instruction added to the instructions of the V25 and V35.

Instruction Group	Mnemonic	Operand	Operation Code		Byte Number	Operation	Flag					
			7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0			AC	CY	V	P	S	Z
Stack Manipulation Instruction	POP	mem16	1 0 0 0 1 1 1 1	mod 0 0 0 mem	2 to 4	(mem16)←(SP + 1, SP) SP←SP + 2						
		reg16	0 1 0 1 1 reg		1	reg16←(SP + 1, SP) SP←SP + 2						
		sreg	0 0 0 sreg 1 1 1		1	sreg←(SP + 1, SP) sreg : SS, DS0, DS1 SP←SP + 2						
		PSW	1 0 0 1 1 1 0 1		1	PSW←(SP + 1, SP) SP←SP + 2	R	R	R	R	R	R
		R	0 1 1 0 0 0 0 1		1	Push registers on the stack						
		imm	0 1 1 0 1 0 S 0		2 to 3	(SP - 1, SP - 2)←imm Sign expansion when SP←SP - 2 and S = 1.						
		DS2*	0 0 0 0 1 1 1 1	0 0 1 1 1 1 1 1	2	DS2←(SP + 1, SP) SP←SP + 2						
		DS3/VPC*	0 0 0 0 1 1 1 1	0 0 1 1 0 1 1 1	2	DS3←(SP + 1, SP) SP←SP + 2						
	PREPARE	imm16, imm8	1 1 0 0 1 0 0 0		4	Prepare New Stack Frame						
	DISPOSE		1 1 0 0 1 0 0 1		1	Dispose of Stack Frame						
Branch Instruction	BR	near-label	1 1 1 0 1 0 0 1		3	PC←PC + disp						
		short-label	1 1 1 0 1 0 1 1		2	PC←PC + ext-disp8						
		regptr16	1 1 1 1 1 1 1 1	1 1 0 0 0 reg	2	PC)←regptr16						
		memptr16	1 0 0 0 1 1 1 1	mod 0 0 0 mem	2 to 4	PC←(memptr16)						
		far-label	1 1 1 0 1 0 1 0		5	PS←seg PC←offset						
		memptr32	1 1 1 1 1 1 1 1	mod 1 0 1 mem	2 to 4	PS←(memptr32 + 2) PC←(memptr32)						

* Instruction added to the instructions of the V25 and V35.

Instruction Group	Mnemonic	Operand	Operation Code		Byte Number	Operation	Flag					
			7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0			AC	CY	V	P	S	Z
Conditional Branch Instructions	BV	short-label	0 1 1 1 0 0 0 0		2	if V = 1 PC←PC + ext-disp8						
	BNV	short-label	0 0 0 1		2	if V = 0 PC←PC + ext-disp8						
	BC BL	short-label	0 0 1 0		2	if CY = 1 PC←PC + ext-disp8						
	BNC BNL	short-label	0 0 1 1		2	if CY = 0 PC←PC + ext-disp8						
	BE BZ	short-label	0 1 0 0		2	if Z = 1 PC←PC + ext-disp8						
	BNE BNZ	short-label	0 1 0 1		2	if Z = 0 PC←PC + ext-disp8						
	BNH	short-label	0 1 1 0		2	if CY ∨ Z = 1 PC←PC + ext-disp8						
	BH	short-label	0 1 1 1		2	if CY ∨ Z = 0 PC←PC + ext-disp8						
	BN	short-label	1 0 0 0		2	if S = 1 PC←PC + ext-disp8						
	BP	short-label	1 0 0 1		2	if S = 0 PC←PC + ext-disp8						
	BPE	short-label	1 0 1 0		2	if P = 1 PC←PC + ext-disp8						
	BPO	short-label	1 0 1 1		2	if P = 0 PC←PC + ext-disp8						
	BLT	short-label	1 1 0 0		2	if S ∀ V = 1 PC←PC + ext-disp8						
	BGE	short-label	1 1 0 1		2	if S ∀ V = 0 PC←PC + ext-disp8						
	BLE	short-label	1 1 1 0		2	if (S ∀ V) ∨ Z = 1 PC←PC + ext-disp8						
	BGT	short-label	1 1 1 1		2	if (S ∀ V) ∨ Z = 0 PC←PC + ext-disp8						
	DBNZNE	short-label	1 1 1 0 0 0 0 0		2	CW = CW – 1 if Z = 0 and CW ≠ 0 PC←PC + ext-disp8						
	DBNZE	short-label	0 0 0 1		2	CW = CW – 1 if Z = 1 and CW ≠ 0 PC←PC + ext-disp8						
	DBNZ	short-label	0 0 1 0		2	CW = CW – 1 if CW ≠ 0 PC←PC + ext-disp8						
	BCWZ	short-label	0 0 1 1		2	if CW = 0 PC←PC + ext-disp8						
	BTCLR*1	sfr, imm3 short-label	0 0 0 0 1 1 1 1	1 0 0 1 1 1 0 0	5	When (sfr) bit No. imm3 = 1 PC←PC + ext-disp8, (sfr) bit No.imm3←0						
	BTCLRL*2	sfrl, imm3 short-label	0 0 0 0 1 1 1 1	1 0 0 1 1 1 0 1	5	When (sfrl) bit No. imm3 = 1 PC←PC + ext-disp8, (sfrl) bit No.imm3←0						

* 1. Instruction added to the instructions of V20 and V30.
2. Instruction added to the instructions of V25 and V35.

Instruction Group	Mnemonic	Operand	Operation Code		Byte Number	Operation	Flag					
			7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0			AC	CY	V	P	S	Z
Interrupt Instructions	BRK	3	1 1 0 0 1 1 0 0		1	(SP – 1, SP – 2)←PSW, (SP – 3, SP – 4)←PS (SP – 5, SP – 6)←PC, SP←SP – 6 IE←0, BRK←0 PS←(15, 14), PC←(13, 12)						
		imm8 (≠ 3)	1 1 0 0 1 1 0 1		2	(SP – 1, SP – 2)←PSW, (SP – 3, SP – 4)←PS (SP – 5, SP – 6)←PC, SP←SP – 6 IE←0, BRK←0 PS←(n x 4 + 3, n x 4 + 2), PC←(n x + 1, n x 4) n = imm8						
	BRKV		1 1 0 0 1 1 0 0		1	When V = 1 (SP – 1, SP – 2)←PSW, (SP – 3, SP – 4)←PS (SP – 5, SP – 6)←PC, SP←SP – 6 IE←0, BRK←0 PS←(19, 18), PC←(17, 16)						
	RETI		1 1 0 0 1 1 1 1		1	PC←(SP + 1, SP), PS←(SP + 3, SP + 2) PSW←(SP + 5, SP + 4), SP←SP + 6	R	R	R	R	R	R
	RETRBI*		0 0 0 0 1 1 1 1	1 0 0 1 0 0 0 1	2	PC←Save PC, PSW←Save PSW	R	R	R	R	R	R
	FINT*		0 0 0 0 1 1 1 1	1 0 0 1 0 0 1 0	2	Notifies the interrupt controller in the CPU that the interrupt process routine has ended.						
	CHKIND		1 1 0 0 1 1 0 0		1	When (mem32) > reg16 or (mem32 + 2) < reg16 (SP – 1, SP – 2)←PSW, (SP – 3, SP – 4)←PS (SP – 5, SP – 6)←PC, SP←SP – 6 IE←0, BRK←0 PS←(23, 22), PC←(21, 20)						
Register Bank Switching Instructions	BRKCS*	reg16	0 0 0 0 1 1 1 1 1 1 0 0 0 reg	0 0 1 0 1 1 0 1	3	RB2 to 0←lower 4 bits of reg16, IE←0, BRK←0 Save PSW←PSW, Save PC←PC, PC←Vector PC						
	TSKSW*	reg16	0 0 0 0 1 1 1 1 1 1 1 1 1 reg	1 0 0 1 0 1 0 0	3	RB2 to 0←lower 4 bits of reg16 Old register bank Save PSW, Save PC←PSW, PC, PSW, PC←new register bank Save PSW, Save PC	×	×	×	×	×	×

* Instruction added to the instructions of V20 and V30.

★

Instruction Group	Mnemonic	Operand	Operation Code		Byte Number	Operation	Flag					
			7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0			AC	CY	V	P	S	Z
CPU Control Instructions	HALT		1 1 1 1 0 1 0 0		1	CPU Halt						
	STOP*1		0 0 0 0 1 1 1 1	1 0 0 1 1 1 1 0	1	CPU Stop						
	IDLE*2		0 0 0 0 1 1 1 1	1 0 0 1 1 1 1 1	2	IDLE mode						
	POLL		1 0 0 1 1 0 1 1		1	Poll and wait						
	DI		1 1 1 1 1 0 1 0		1	IE←0						
	EI		1 1 1 1 1 0 1 1		1	IE←1						
	BUSLOCK		1 1 1 1 0 0 0 0		1	Bus Lock Prefix						
	FPO1	fp-op	1 1 0 1 1 X X X	1 1 Y Y Y Z Z Z	2	No Operation						
		fp-op, mem	1 1 0 1 1 X X X	mod Y Y Y mem	2 to 4	data bus←(mem)						
	FPO2	fp-op	0 1 1 0 0 1 1 X	1 1 Y Y Y Z Z Z	2	No Operation						
		fp-op, mem	0 1 1 0 0 1 1 X	mod Y y Y mem	2 to 4	bata bus←(mem)						
	NOP		1 0 0 1 0 0 0 0		1	No Operation						
*3	RSTWDT*2	imm8, $\overline{\text{imm8}}$	0 0 0 0 1 1 1 1	1 0 0 1 0 1 1 0	4	WDM←imm8 WDM is an AFR space register $\overline{\text{imm8}}$ is the one's compliment of imm8.						
			imm8	$\overline{\text{imm8}}$								
	*4		0 0 1 sreg 1 1 0		1	Segment override prefix						
	DS2: *2		0 1 1 0 0 0 1 1		1	Expansion segment override prefix						
	DS3: *2		1 1 0 1 0 1 1 0		1	Expansion segment override prefix						
*5	IRAM: *2		1 1 1 1 0 0 0 1		1	IRAM: Prefix						

* 1. Instruction added to the instructions of V20 and V30.
2. Instruction added to the instructions of V25 and V35.
3. Watchdog timer manipulation instructions
4. 4 types, DS0:, DS1:, PS:, and SS:
5. Override prefix instruction for accessing the register file space.

★ 17. ELECTRICAL SPECIFICATIONS (PRELIMINARY)

Absolute Maximum Rating (Ta = 25 °C)

PARAMETER	SYMBOL	TESTCONDITIONS	RATING	UNIT
Power supply voltage	V_{DD}		– 0.5 to + 0.7	V
Input voltage	V_I		– 0.5 to V_{DD} + 0.5	V
Output voltage	V_O		– 0.5 to V_{DD} + 0.5	V
Output current low	I_{OL}	One pin	4.0	mA
		Total of all output pins	100	mA
Output current high	I_{OH}	One pin	– 1.0	mA
		Total of all output pins	– 20	mA
Operating temperature	T_{opt}		– 40 to + 85	°C
Storage temperature	T_{stg}		– 65 to + 150	°C

DC Characteristics (Ta = –40 to +85 °C, V_{DD} = +5.0 V ± 10 %)

PARAMETER	SYMBOL	TEST CONDITIONS	MIN.	TYP.	MAX.	UNIT
Input voltage low	V_{IL1}	*1	0		0.8	V
	V_{IL2}	*2	0		0.2V_{DD}	V
Input voltage high	V_{IH1}	*1	2.2		V_{DD}	V
	V_{IH2}	*2	0.8V_{DD}		V_{DD}	V
Output voltage low	V_{OL}	I_{OL} = 2.0 mA			0.45	V
Output voltage high	V_{OH}	I_{OH} = – 0.4 mA	V_{DD} – 1.0			V
Input leakage current	I_{LI}	0 V ≦ V_I ≦ V_{DD}			±10	μA
Output leakage current	I_{LO}	0 V ≦ V_O ≦ V_{DD}			±10	μA
Power supply current *3	I_{DD1}	Operation mode		7.0fx + 30	7.0fx + 50	mA
	I_{DD2}	HALT mode		4.7fx	4.7fx + 20	mA
	I_{DD3}	STOP mode		30	280	μA
	I_{DD4}	IDLE mode		2.3fx	2.3fx + 15	mA

* 1. Other than *2

2. $\overline{\text{RESET}}$, NMI, INTP0 to INTP3, X1

3. The unit of constants 7.0, 4.7, 2.3 is mA/MHz.

Capacitance (Ta = 25 °C, V_{DD} = 0 V)

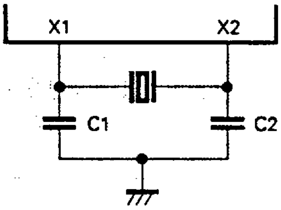
PARAMETER	SYMBOL	TEST CONDITIONS	MIN.	TYP.	MAX.	UNIT
Input capacitance	C_I	fc = 1MHz Unmeasured pins are 0 V			10	pF
Output capacitance	C_O				20	pF
I/O capacitance	C_{IO}				20	pF

Operating Conditions

INTERNAL OPERATING CLOCK FREQUENCY	OPERATING TEMPERATURE (T_{opt})	POWER SUPPLY VOLTAGE (V_{DD})
0.25 MHz ≦ fx ≦ 12.5 MHz	− 40 to + 85 °C	+ 5.0 V ±10 %

Recommended Oscillator

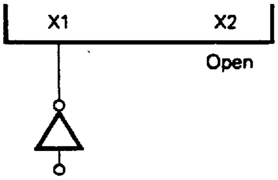
(a) Ceramic resonator connection (Ta = − 40 to + 85 °C, V_{DD} = 5 V ±10 %)



MANUFACTURER	OSCILLATION FREQUENCY fxx [MHz]	PRODUCT NAME	RECOMMENDED CONSTANT	
			C1 [pF]	C2 [pF]
Murata Mfg.	25	CSA25.00MXZ040	5	5

Remarks 1. The oscillator should be located as close as possible to the X1 and X2 pins.
2. No other signal lines should cross the shaded srea.
3. For matching between the μPD70423 and the resonator, evaluation should be carried out sufficiently.

(b) External clock input



AC Characteristics (Ta = – 40 to + 85 °C, V_{DD} = + 5.0 V ±10 %)

PARAMETER	SYMBOL		TEST CONDITIONS	MIN.	MAX.	UNIT
X1 input cycle time	1	t_{CYX}		40	250	ns
X1 input high–level width	2	t_{WXH}		15		ns
X1 input low–level width	3	t_{WXL}		15		ns
X1 input rise time	4	t_{XR}			10	ns
X1 input fall time	5	t_{XF}			10	ns
CLKOUT output cycle time	6	t_{CYK}		80	4000	ns
CLKOUT output high–level width	7	t_{WKH}		0.5T – 7		ns
CLKOUT output low–level width	8	t_{WKL}		0.5T – 7		ns
CLKOUT output rise time	9	t_{KR}			7	ns
CLKOUT output fall time	10	t_{KF}			7	ns
Input rise time	11	t_{IR1}	*1		10	ns
	12	t_{IR2}	*2		10	ns
Input fall time	13	t_{IF1}	*1		20	ns
	14	t_{IF2}	*2		20	ns
Output rise time	15	t_{OR}			10	ns
Output fall time	16	t_{OF}			10	ns
CLKOUT delay time from X1↑	111	t_{DXK}	External clock input		20	ns
Address delay time from CLKOUT↑	17	t_{DKA}		5	60	ns
Address hold time (from CLKOUT↑)	18	t_{HKA1}		5		ns
	19	t_{HKA2}		5		ns
Address float delay time from CLKOUT↑	20	t_{FKA}		t_{HKA1}	40	ns
Address setup time (to $\overline{\text{ASTB}}$↓)	21	t_{SAST}		(n + 0.5)T – 35		ns
Address hold time (from $\overline{\text{ASTB}}$↓)	22	t_{HSTA}		0.5T – 15		ns
$\overline{\text{ASTB}}$↓ delay time from CLKOUT↓	23	t_{DKSTL}		0	30	ns
$\overline{\text{ASTB}}$↑ delay time from CLKOUT↓	24	t_{DKSTH}		0	25	ns
$\overline{\text{ASTB}}$ high–level width	25	t_{WSTH}		(n + 1)T – 15		ns
$\overline{\text{RD}}$↑ delay time from CLKOUT	26	t_{DKRL}		0	30	ns
$\overline{\text{RD}}$↓ delay time from CLKOUT	27	t_{DKRH}		0	25	ns
$\overline{\text{RD}}$ low–level width	28	t_{WRL}		(N + 1.5)T – 15		ns
$\overline{\text{RD}}$↓ delay time from address float	29	t_{FARL}		0		ns
Address delay time from $\overline{\text{RD}}$↑	30	t_{DRA}		0.5T		ns

n : Number of address wait states
N : Number of data wait states
T : t_{CYK}

* 1. Other than *2
2. $\overline{\text{RESET}}$, NMI, INTP0 to INTP3, X1

Remarks The numbers in the symbol column correspond to the numbers in the timing chart.

PARAMETER	SYMBOL		TEST CONDITIONS	MIN.	MAX.	UNIT
$\overline{ASTB}$↑ delay time from $\overline{RD}$↑, $\overline{IORD}$↑	112	t_{DRSTH}		0		ns
$\overline{RD}$↑, $\overline{IORD}$↑ delay time from $\overline{WRL}$↑, $\overline{WRH}$↑, $\overline{IOWR}$↑	113	t_{DWRH}		0		ns
$\overline{DEX}$ delay time from CLKOUT↓	31	t_{DKDX}		0	30	ns
$\overline{DEX}$ hold time (from CLKOUT↓)	32	t_{HKDX}		0		ns
Data input setup time (to CLKOUT↓)	33	t_{SDK}		15		ns
Data input hold time (to CLKOUT↓)	34	t_{HKDR}		0		ns
$\overline{WR}$↓ delay time from CLKOUT↓	35	t_{DKWL}		0	30	ns
$\overline{WR}$↑ delay time from CLKOUT↓	36	t_{DKWH}		0	25	ns
$\overline{WR}$ low-level width	37	t_{WWL}		(N + 1)T − 15		ns
Data outout delay time from CLKOUT↑	38	t_{DKD}		3	60	ns
Data output hold time (to CLKOUT↓)	39	t_{HKDW}		0		ns
$\overline{ASTB}$↑ delay time from $\overline{WR}$↑	40	t_{DWSTH}		0		ns
$\overline{RAS}$↓ delay time from CLKOUT↑	41	t_{DKRAL}		nT	nT + 30	ns
$\overline{RAS}$↑ delay time from CLKOUT↑	42	t_{DKRAH}		0	25	ns
$\overline{RAS}$ high-level width	43	t_{WRAH}		(n + 1)T − 15		ns
$\overline{RAS}$↑ delay time from $\overline{WRH}$↓, $\overline{WRL}$↓	114	t_{DWRAH}		(N + 0.5)T − 15		ns
Address setup time (to $\overline{RAS}$↓)	115	t_{SARAL}		nT − 30		ns
READY setup time (to CLKOUT↓)	44	t_{SRYHK}		t_{WKH} − 10		ns
READY hold time (to CLKOUT↓)	45	t_{HKRYL}		15		ns
READY setup time (to CLKOUT↓)	46	t_{SRYLK}		t_{WKH} − 10		ns
READY hold time (to CLKOUT↓)	47	t_{HKRYH}		15		ns
$\overline{RESET}$ low-level width	48	t_{WRSL1}	STOP release/power ON reset	30		ns
	49	t_{WRSL2}	System reset	5		μs
NMI high-level width	50	t_{WNIH}		5		μs
NMI loe-level width	51	t_{WNIL}		5		μs
INTPm setup time (to CLKOUT↓)	52	t_{SIQK}	m = 0 to 3	30		ns
INTPm high-level width	53	t_{WIQH}	m = 0 to 3	10T		ns
INTPm low-level width	54	t_{WIQL}	m = 0 to 3	10T		ns
$\overline{POLL}$ setup time (to CLKOUT↓)	55	t_{SPLK}		30		ns
HLDRQ setup time (to CLKOUT↓)	56	t_{SHQK}		30		ns
$\overline{HLDAK}$↓ delay time from CLKOUT↑	57	t_{DKHA}		0	30	ns
Bus float delay time from $\overline{HLDAK}$↓	58	t_{FCHA}		0		ns
Bus output delay time from $\overline{HLDAK}$↑	59	t_{DHAC}		T − 40		ns
$\overline{HLDAK}$ delay time from HLDRQ↓	60	t_{DHQHA}			2.5T + 80	ns

n : Number of address wait states
N : Number of data wait states
T.: t_{CYK}

Remarks The numbers in the symbol column correspond to the numbers in the timing chart.

PARAMETER	SYMBOL		TEST CONDITIONS	MIN.	MAX.	UNIT
Bus output delay time from HLDRQ↓	61	t_{DHOC}		0.5T + 50		ns
HLDRQ low-level width	62	t_{WHQL}		2T		ns
$\overline{\text{HLDAK}}$ low-level width	63	t_{WHAL}		3T − 15		ns
$\overline{\text{CTSm}}$ high-level width	64	t_{WCTH}	m = 0, A, B	2T		ns
$\overline{\text{CTSm}}$ low-level width	65	t_{WCTL}	m = 0, A, B	2T		ns
$\overline{\text{DCDm}}$ high-level width	66	t_{WDCH}	m = A, B	2T		ns
$\overline{\text{DCDm}}$ low-level width	67	t_{WDCL}	m = A, B	2T		ns
Send/receive data cycle	68	t_{CYD1}	UART	32T		ns
	69	t_{CYD2}	MPSC (ASYNC/HDLC/SYNC mode)	5T		ns
TxC0 output clock cycle	70	t_{CYC1}	UART	32T		ns
TxC0 output clock high-level width	71	t_{WCH1}		16T − 10		ns
TxC0 output clock low-level width	72	t_{WCL1}		16T − 10		ns
TxCm, RxCm input clock cycle	73	t_{CYC2}	MPSC(ASYNC/HDLC/SYNC mode; m = A, B)	5T		ns
TxCm, RxCm input clock high-level width	74	t_{WCH2}		2.5T − 10		ns
TxCm, RxCm input clock low-level width	75	t_{WCL2}		2.5T − 10		ns
TxDm delay time from TxCm↓	76	t_{DTCTD1}	UART (TxC output; m = 0)	−10	90	ns
	77	t_{DTCTD2}	MPSC (x1 mode; m = A, B)		90	ns
	78	t_{DTCTD3}	MPSC (x16, 32, 64 mode; m = A, B)		270	ns
	79	t_{DTCTD4}	TxC output (m = A, B)	0	90	ns
RxDm setup time (to RxCm↑)	80	t_{SRDRC}	m = A, B	10		ns
RxDm hold time (to RxCm↑)	81	t_{HRCRD}	m = A, B	110		ns
TxDm delay time from $\overline{\text{CTSm}}$	82	t_{DCTTD1}	UART (m = 0)		$2t_{CYC1}$	ns
	83	t_{DCTTD2}	MPSC (ASYNC/SYNC mode; m = A, B)		$3t_{CYC2}$	ns
	84	t_{DCTTD3}	MPSC (HDLC mode; m = A, B)	$4t_{CYC2}$	$7t_{CYC3}$	ns
$\overline{\text{DCDm}}$ setup time (to RxCm↑)	85	t_{SDCRC}	m = A, B	t_{CYC2}		ns
$\overline{\text{DCDm}}$ hold time (to RxCm↑)	86	t_{HRCDC1}	MPSC (ASYNC mode; m = A, B)	7T		ns
	87	t_{HRCDC2}	MPSC (SYNC mode; m = A, B)	$20t_{CYC2}$ + 8T		ns
	88	t_{HRCDC3}	MPSC (HDLC mode; m = A, B)	$3t_{CYC2}$ + 8T		ns
RxCm hold time (to start bit, CRC MSB, end flag MSB)	89	t_{HRDRC1}	MPSC (ASYNC mode; m = A, B)	1		bit
	90	t_{HRDRC2}	MPSC (SYNC mode; m = A, B)	$22t_{CYC2}$		ns
	91	t_{HRDRC3}	MPSC (HDLC mode; m = A, B)	$5t_{CYC2}$		ns
RxCm setup time (to start bit, SYNC character)	92	t_{SRCRD1}	MPSC (ASYNC mode; m = A, B)	1		bit
	93	t_{SRCRD2}	MPSC (SYNC/HDLC mode; m = A, B)	t_{CYC2}		ns
TxDm delay time from RxDm	94	t_{DRDTD}	Echo-back mode (m = 0, 1)		90	ns
DMARQm setup time (to CLKOUT↓)	95	t_{SDQK}	Other than demand release mode; m = 0, 1	30		ns

T : t_{CYK}

Remarks The numbers in the symbol column correspond to the numbers in the timing chart.

PARAMETER	SYMBOL		TEST CONDITIONS	MIN.	MAX.	UNIT
DMARQm high-level width	96	t_{DDOH}	Other than demand release mode; m = 0, 1	2T		ns
DMARQm low-level width	97	t_{WDQL}	m = 0, 1	2T		ns
DMARQm↓ setup time (to CLKOUT↑)	98	t_{SKDQL}	Demand release mode; m = 0, 1		5	ns
DMARQm hold time (to CLKOUT↓)	99	t_{HKDQL}	Demand release mode; m = 0, 1	15		ns
$\overline{\text{DMAAKm}}$↓ delay time from CLKOUT↑	100	t_{DKDA}	m = 0, 1	0	30	ns
$\overline{\text{DMAAKm}}$ low-level width	101	t_{WDAL}	m = 0, 1	(3+ n+ N)T-15		ns
$\overline{\text{TCEm}}$↓ delay time from CLKOUT↑	102	t_{DKTE}	m = 0, 1	0	30	ns
$\overline{\text{TCEm}}$ loe-level width	103	t_{WTCL}	m = 0, 1	T - 15		ns
TOUT high-level width	104	t_{WTOH}		8T - 10		ns
TOUT low-level width	105	t_{WTOL}		8T - 10		ns
$\overline{\text{WDTOUT}}$ low-level width	106	t_{WWTL}		32T - 10		ns
$\overline{\text{BUSLOCK}}$ delay time from CLKOUT↑	107	t_{DKBL}		0	30	ns
Port output delay time (to CLKOUT↓)	116	t_{DKP}		10	55	ns
Port input setup time (to CLKOUT↓)	117	t_{SPK}		30		ns
Port input hold time (to CLKOUT↓)	118	t_{HKP}		20		ns
LHLD1 setup time (to CLKOUT↓)	119	t_{SLHOK}	With local bus master	30		ns
LHLD0 delay time from CLKOUT↓	120	t_{DKLHA}		5	50	ns
LHLD0↓ delay time from bus float	121	t_{FCLHA}		0.5T - 20		ns
Bus output delay time from LHLD0↑	122	t_{DLHAC}		0.5T - 20		ns
LHLD0↑ delay time from LHLD1↓	123	$t_{DLHOLHA}$			1.5T + 60	ns
Bus output delay time from LHLD1↓	124	t_{DLHOC}		1.5T		ns
LHLD1 low-level width	125	t_{WLHOL1}		2T		ns
LHLD0 low-level width	126	t_{WLHAL1}		3T - 15		ns
LHLD0 delay time (to CLKOUT↓)	127	t_{DLHOK}	With local bus slave	5	50	ns
LHLD1↓ setup time (to CLKOUT↓)	128	t_{SKLHA}		30		ns
Bus output delay time from LHLD1↓	129	t_{FLHAC}		1.5T - 30		ns
LHLD0↓ delay time from bus float	130	t_{FCLHO}			0.5T + 30	ns
LHLD1↑ setuo time (to LHLD0↓)	131	$t_{SLHOLHA}$			4T - 65	ns
LHLD0 low-level width	132	t_{WLHOL2}		2T - 15		ns
LHLD1 low-level width	133	t_{WLHAL2}		3T		ns

n : Number of address wait states
N : Number of data wait states
T : t_{CYK}

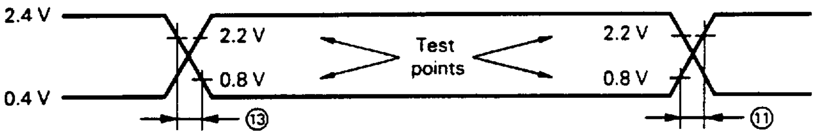
Remarks The numbers in the symbol column correspond to the numbers in the timing chart.

Data Memory STOP Mode Low-Power Supply Voltage Data Hold Characteristic (Ta = –40 to +85 °C)

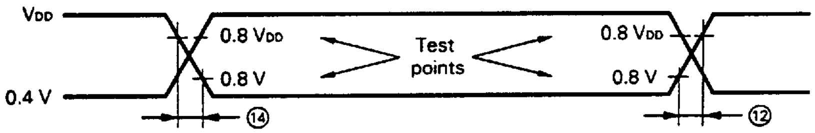
PARAMETER	SYMBOL		TEST CONDITIONS	MIN.	MAX.	UNIT
Data hold power supply voltage	108	V_{DDDR}		2.5	5.5	V
Power supply voltage rise time	109	t_{RVD}		200		μs
Power supply voltage fall time	110	t_{FVD}		200		μs

Remarks The numbers in the symbol column correspond to the numbers in the timing chart.

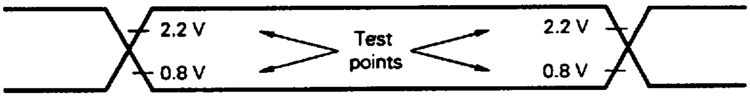
AC Test Wave Form (Except $\overline{\text{RESET}}$, NMI, INTP0 to INTP3, X1)



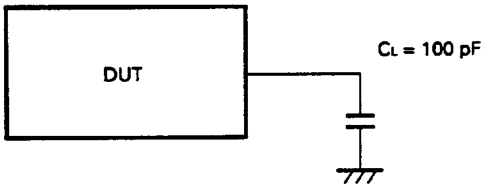
AC Test Input Wave Form ($\overline{\text{RESET}}$, NMI, INTP0 to INTP3, X1)



AC Test Output Test Points



Load Conditions

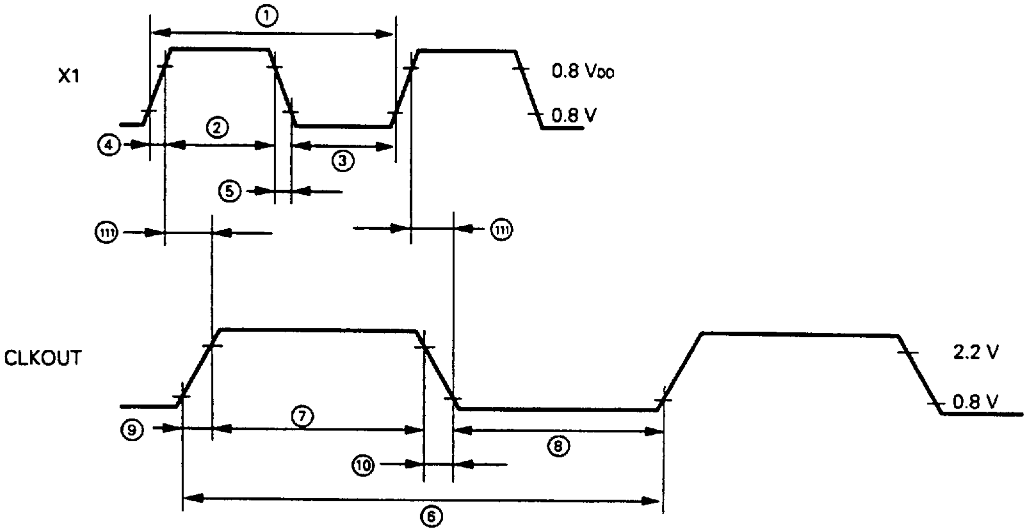


Note **When the load capacity exceeds 100 pF due to the circuit configuration, a buffer should be inserted to keep the load capacity below 100 pF.**

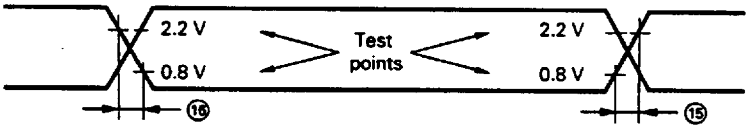
Remarks DUT : measured device

Clock Input/Output Timing

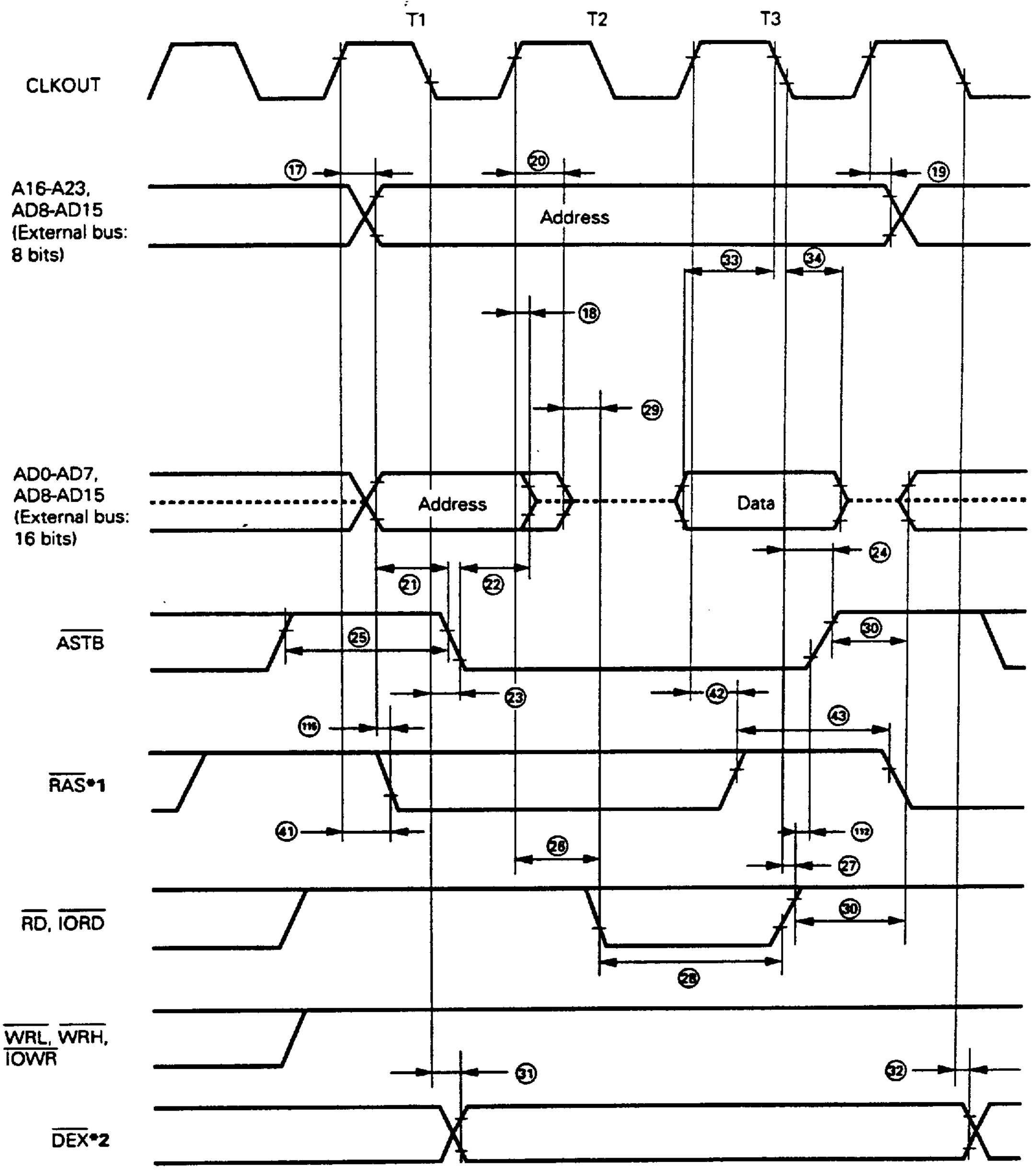
★



Output Wave Form (Except CLKOUT)



★ **Read Timing (Main Bus)**

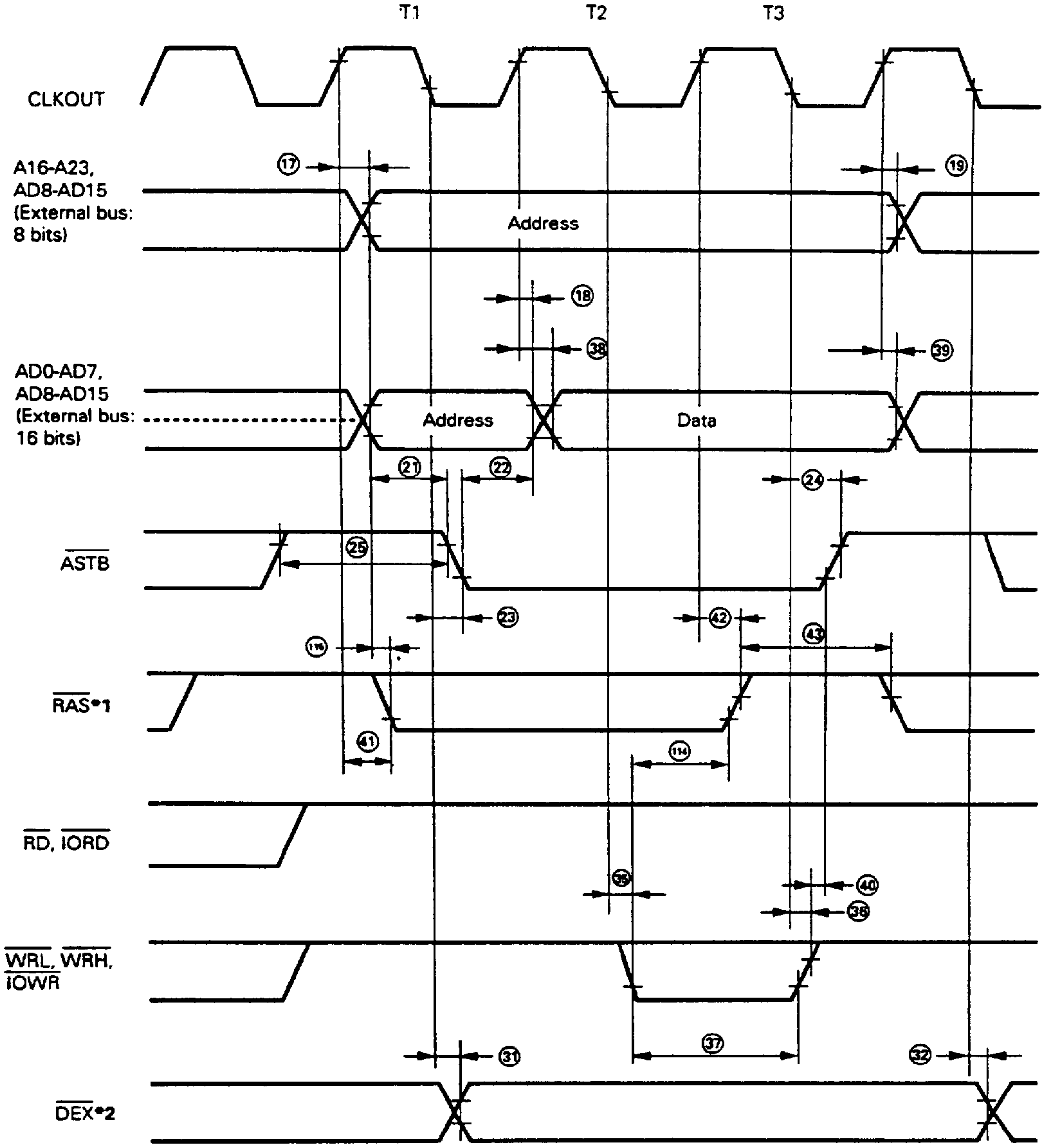


* 1. Becomes active only when accessing memory block 2 and 5 (set using the MBC register).

2. Valid only when the external bus width is 16 bits.

Remarks The dotted lines show high impedance.

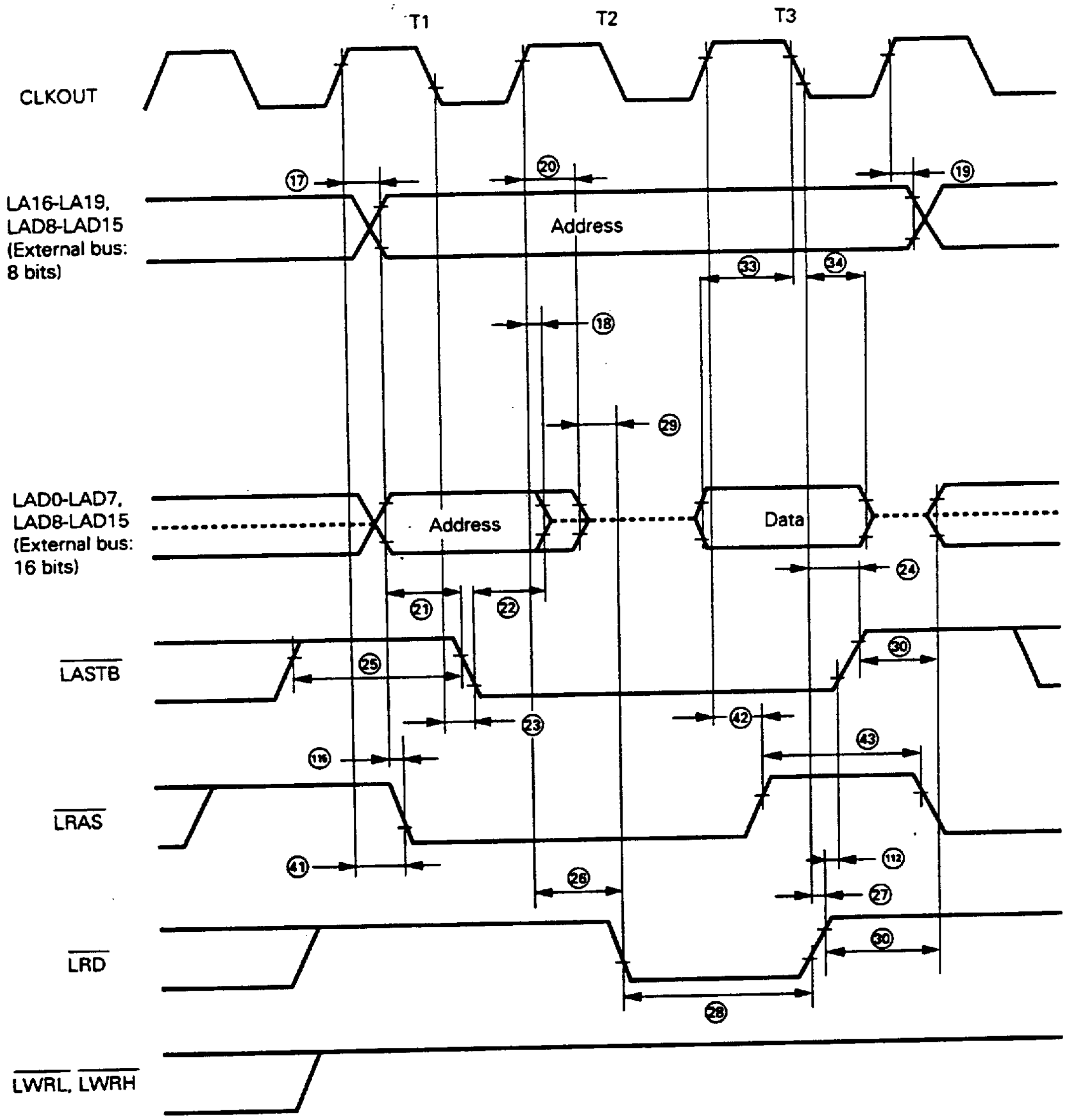
Write Timing (Main Bus) ★



* 1. Becomes active only when accessing memory block 2 and 5 (set using the MBC register).
  2. Valid only when the external bus width is 16 bits.

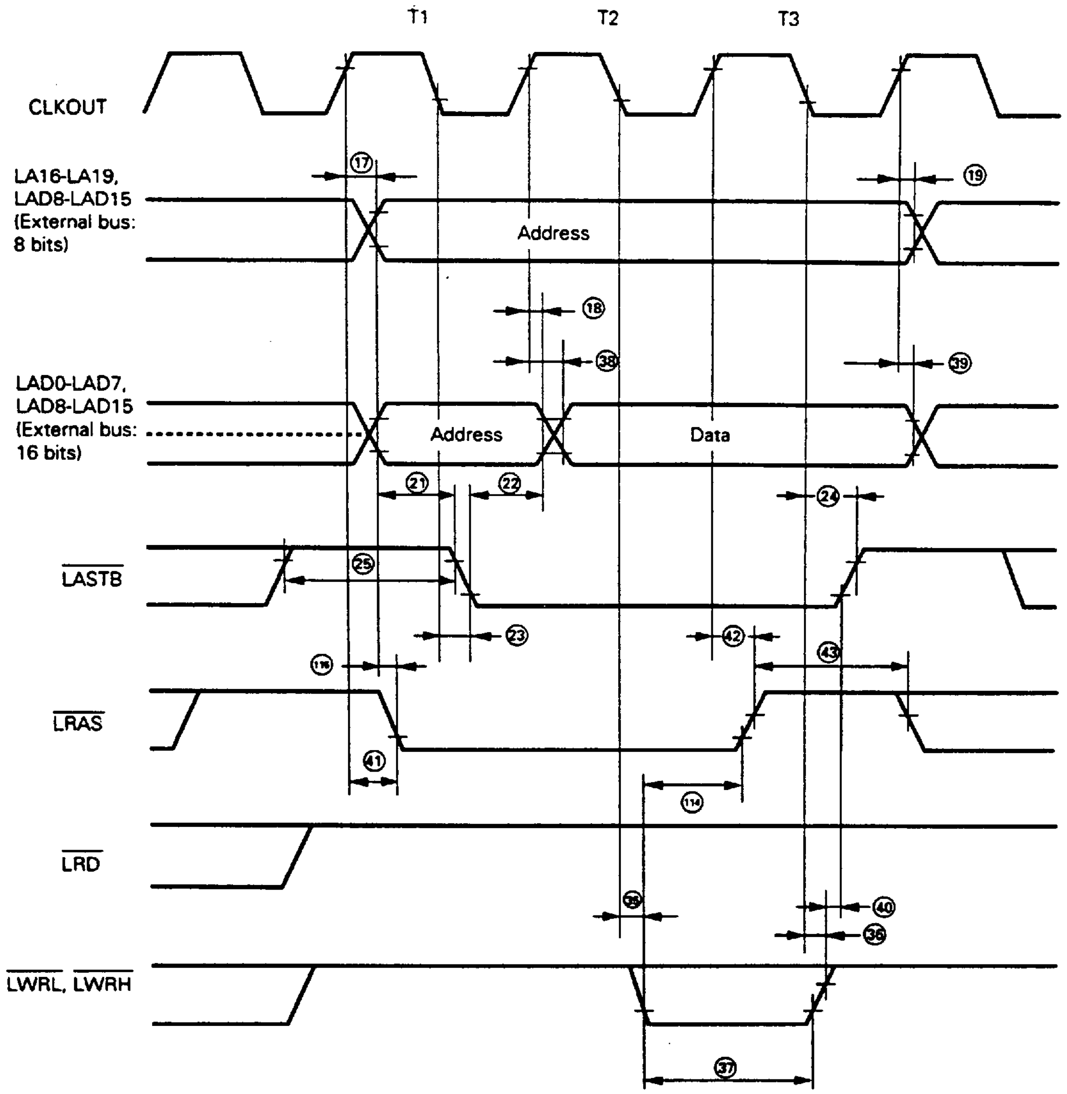
Remarks The dotted lines show high impedance.

★ **Read Timing (Local Bus)**



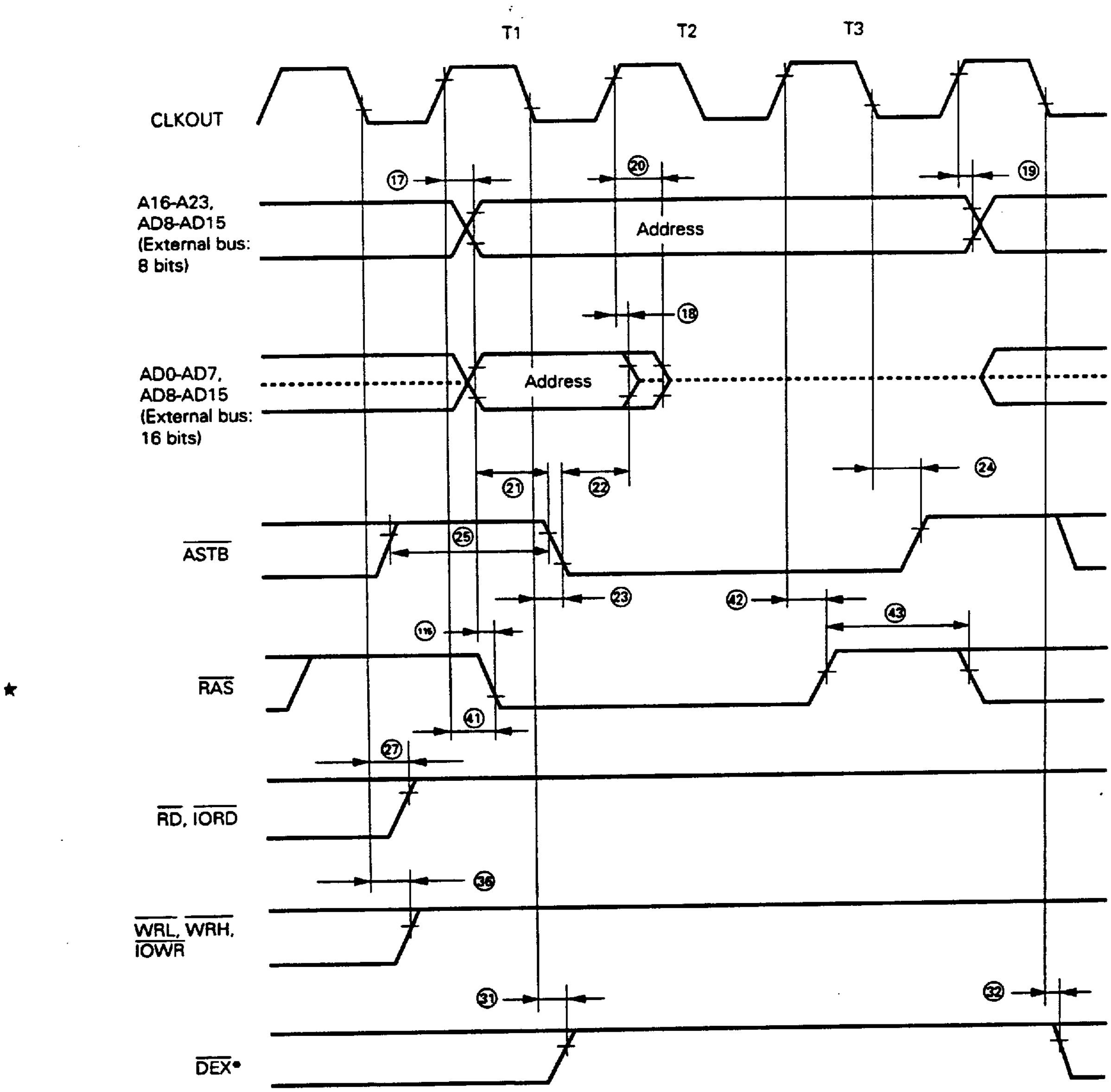
Remarks The dotted lines show high impedance.

Write Timing (Local Bus) ★



Remarks The dotted lines show high impedance.

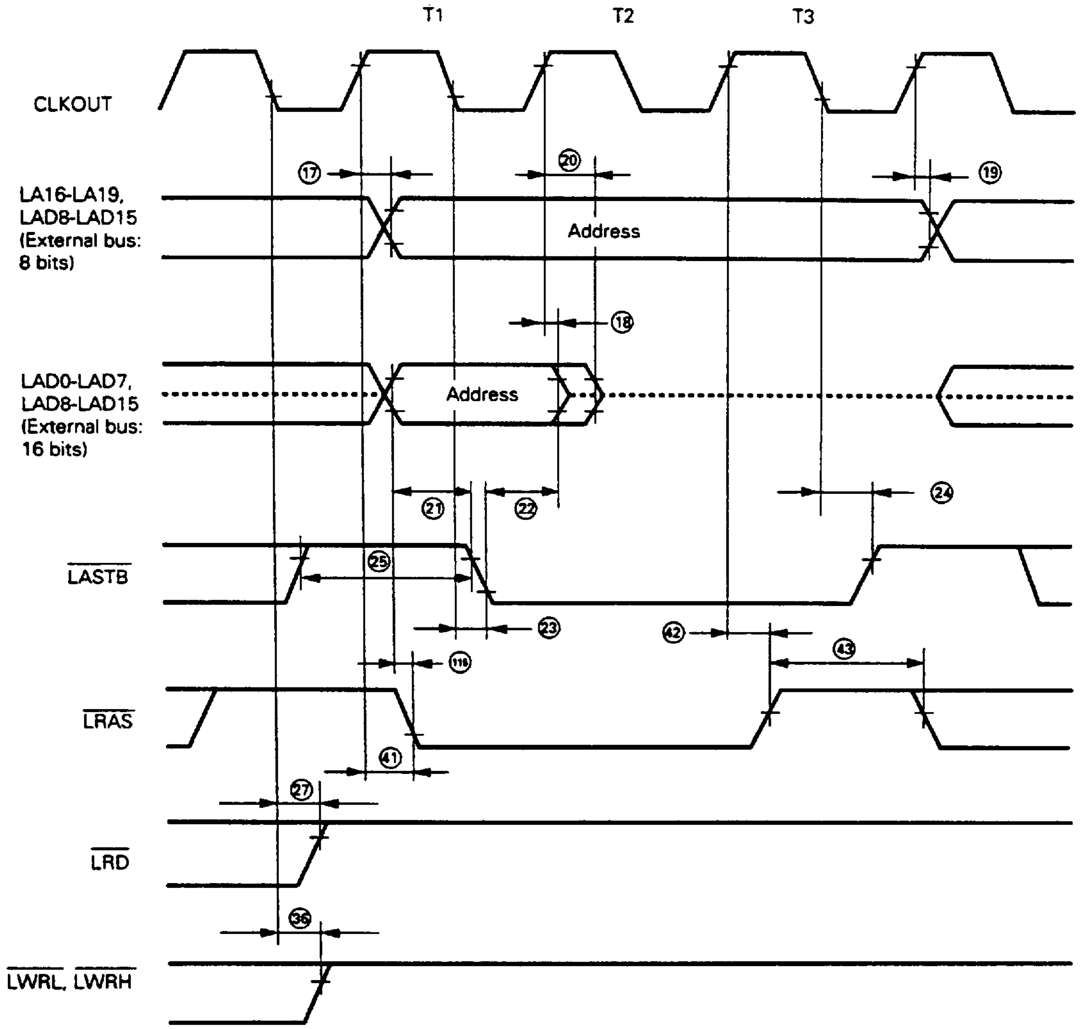
Refresh Timing (Main Bus)



* Valid only when the external bus width is 16 bits.

Remarks The dotted lines show high impedance.

Refresh Timing (Local Bus)

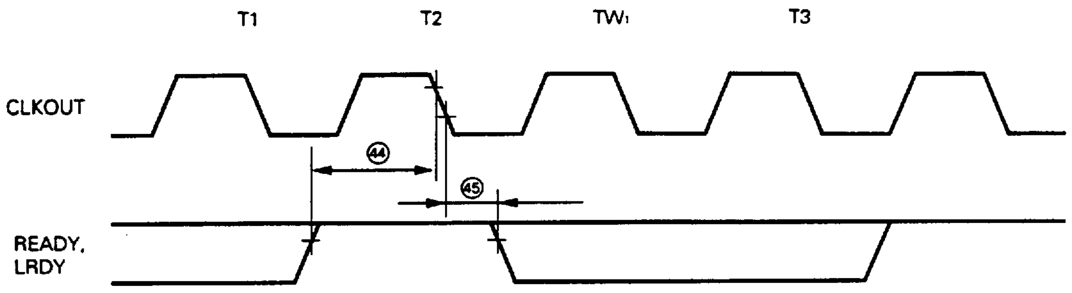


★

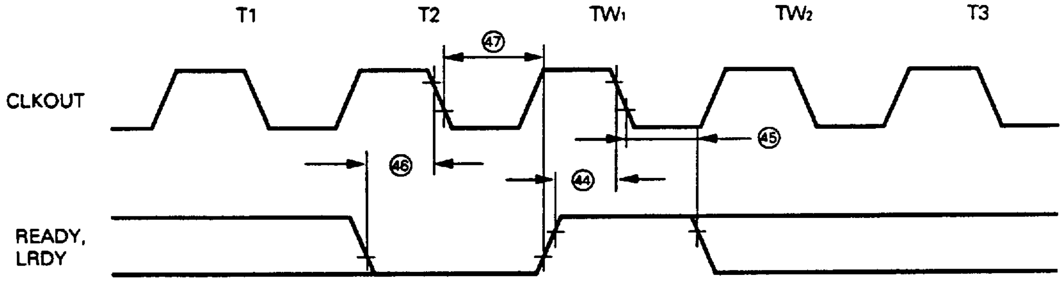
Remarks The dotted lines show high impedance.

READY, LRDY Input Timing

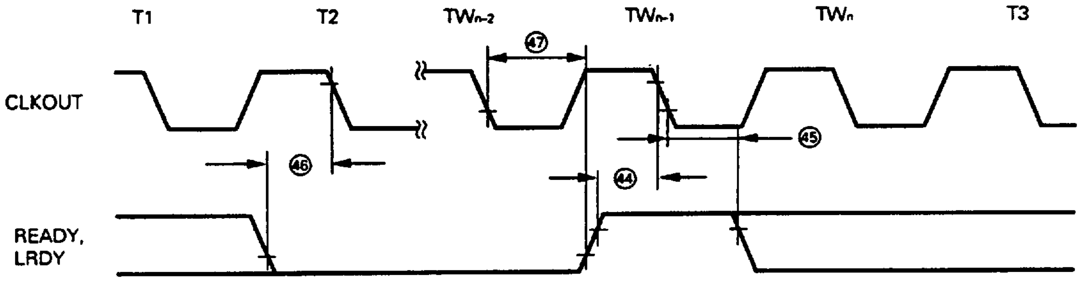
(1) 1 data wait inserted



(2) 2 data waits inserted

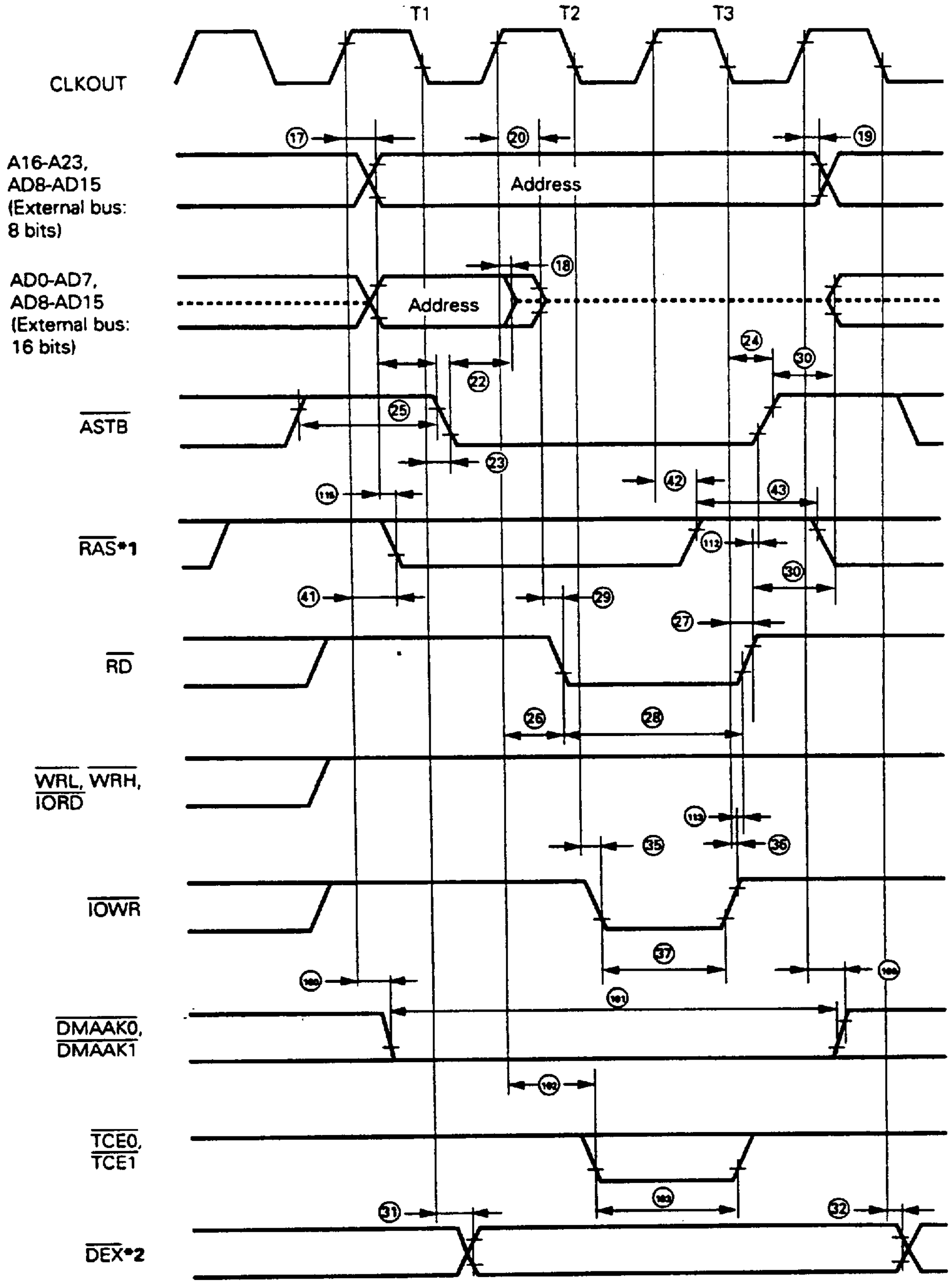


(3) n data waits inserted (n ≥ 3)



Remarks The READY input is valid except when the PWCn register (n = 0,1) field is '00' (binary).
The LRDY input is valid except when bits DW0 and DW1 of the LPWC register are '0', '0'.

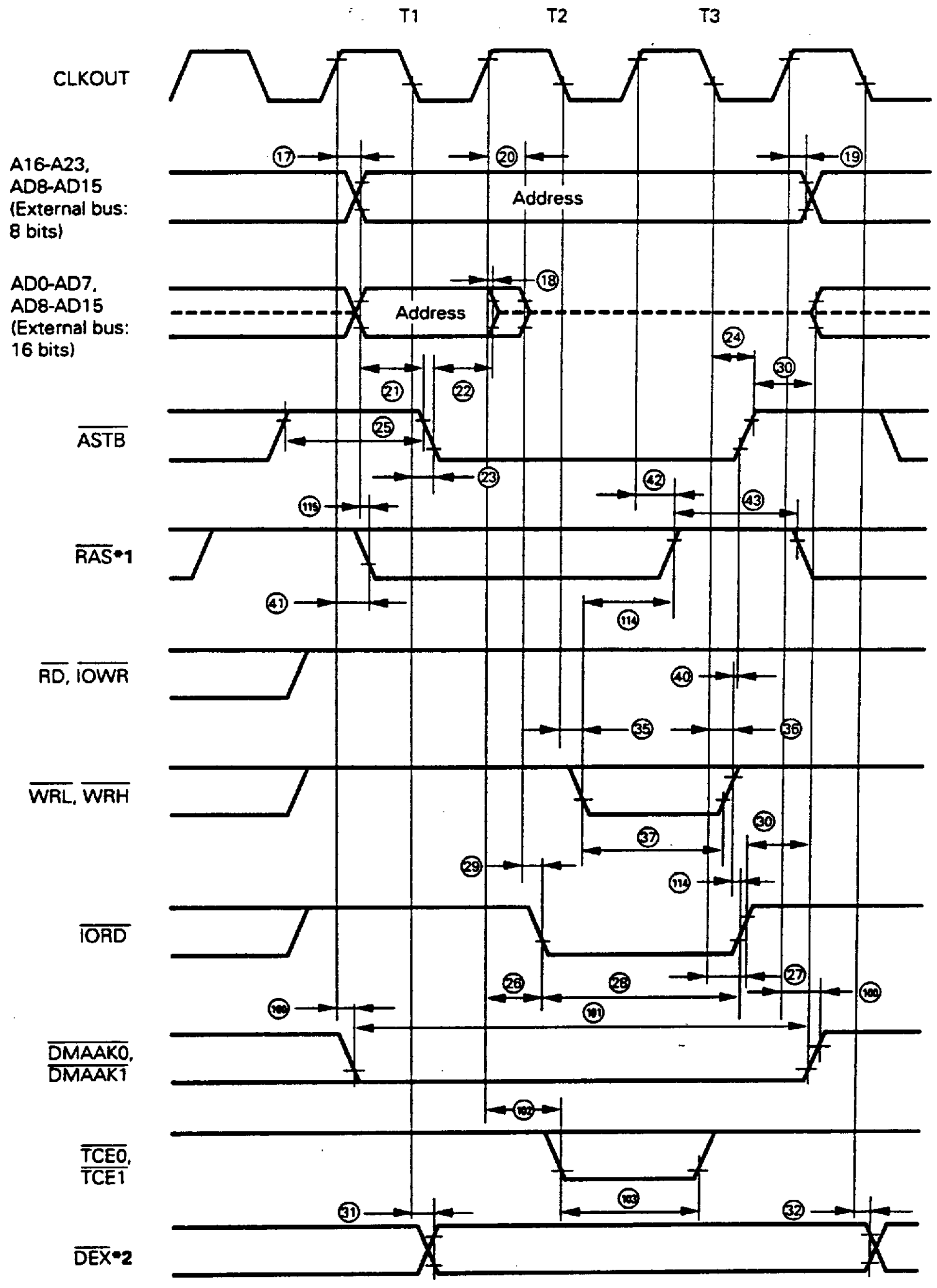
DMA Timing (External Memory → External I/O) ★



* 1. Becomes active only during DMA transfer to memory blocks 2 and 5 (set using the MBC register).
  2. Valid only when the external bus width is 16 bits.

Remarks The dotted lines show high impedance.

★ DMA Timing (External I/O → External Memory)

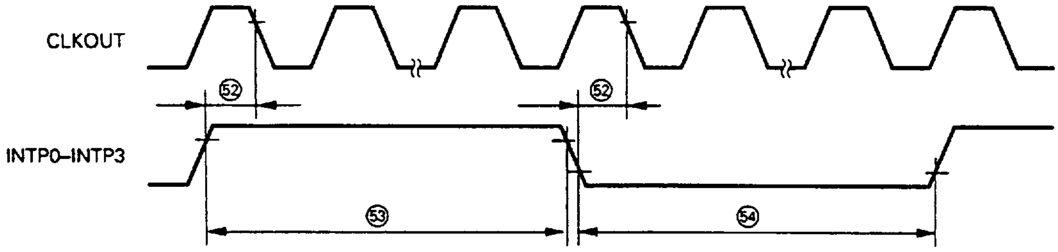


* 1. Becomes active only during DMA transfer to memory blocks 2 and 5 (set using the MBC register).

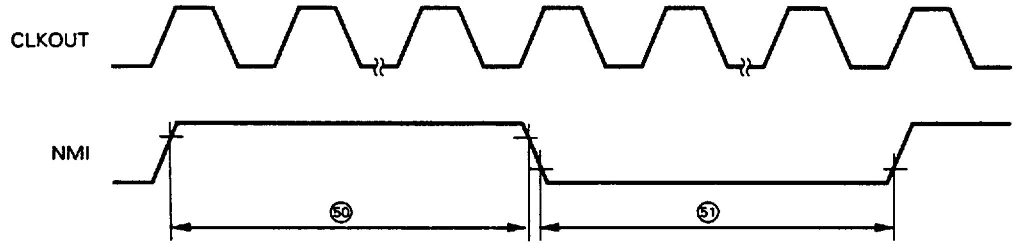
2. Valid only when the external bus width is 16 bits.

Remarks The dotted lines show high impedance.

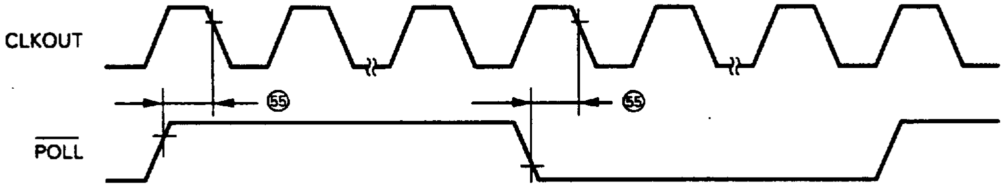
INTPm Input Timing (m = 0 to 3)



NMI Input Timing

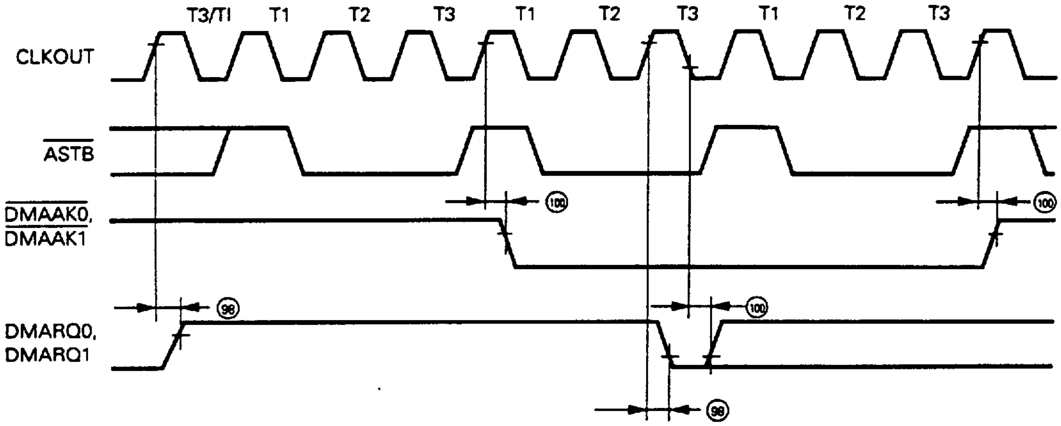


$\overline{\text{POLL}}$ Input Timing

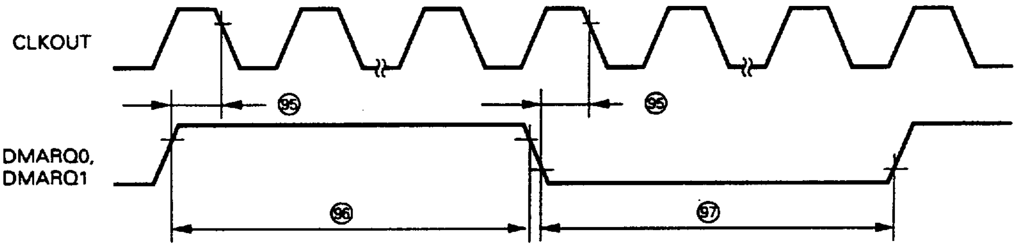


DMARQn Input Timing (n = 0, 1)

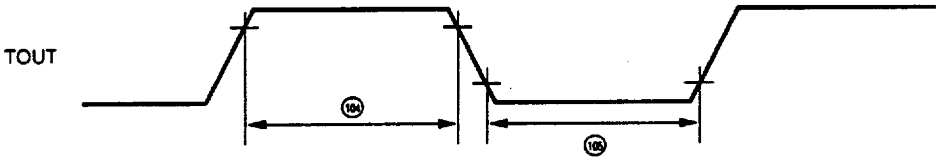
★ (1) In demand release mode



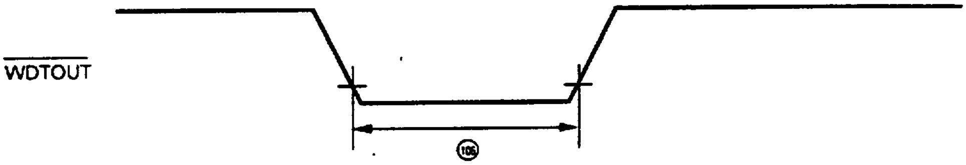
(2) In non-demand-release mode



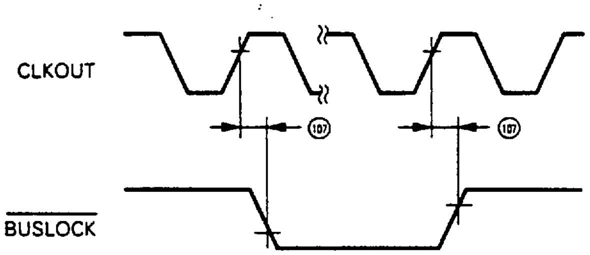
Timer output timing



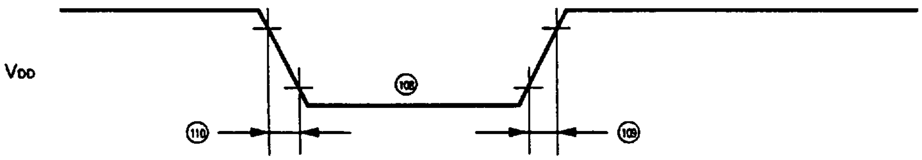
WDTOUT output timing



$\overline{\text{BUSLOCK}}$ Output Timing ★



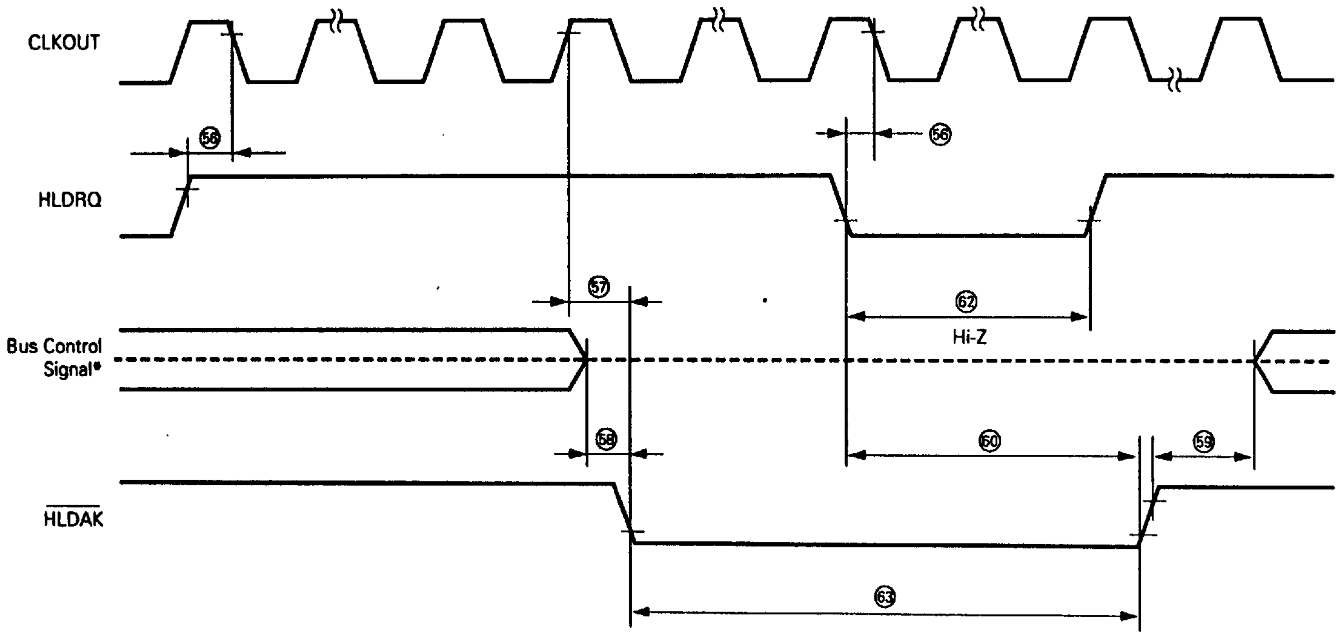
Data Hold Timing (STOP Mode)



★

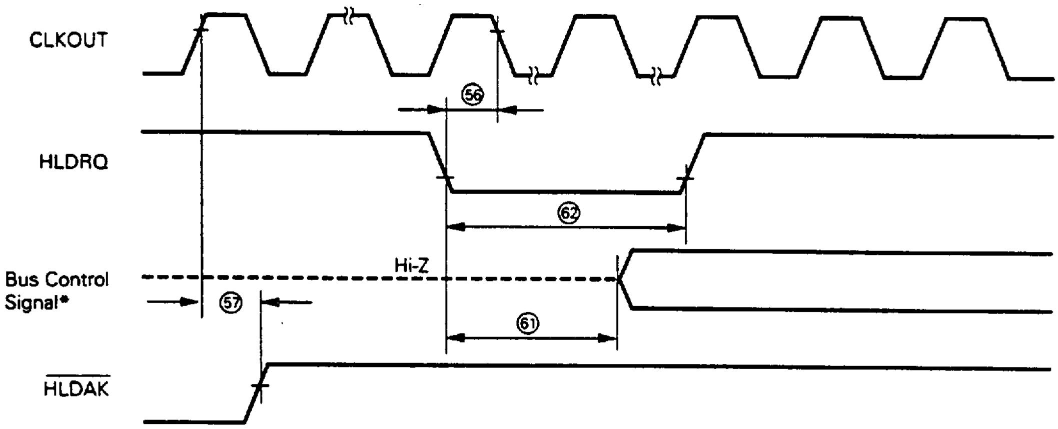
Main Bus Hold Request/Acknowledge Timing

(1) Normal mode



* $\overline{\text{ASTB}}$, $\overline{\text{RD}}$, $\overline{\text{WRH}}$, $\overline{\text{WRL}}$, $\overline{\text{DEX}}$, $\overline{\text{RAS}}$, $\overline{\text{BUSLOCK}}$, $\overline{\text{IORD}}$, $\overline{\text{IOWR}}$, AD0 to AD15, A16 to A23

(2) At hold mode release for inserting a refresh cycle

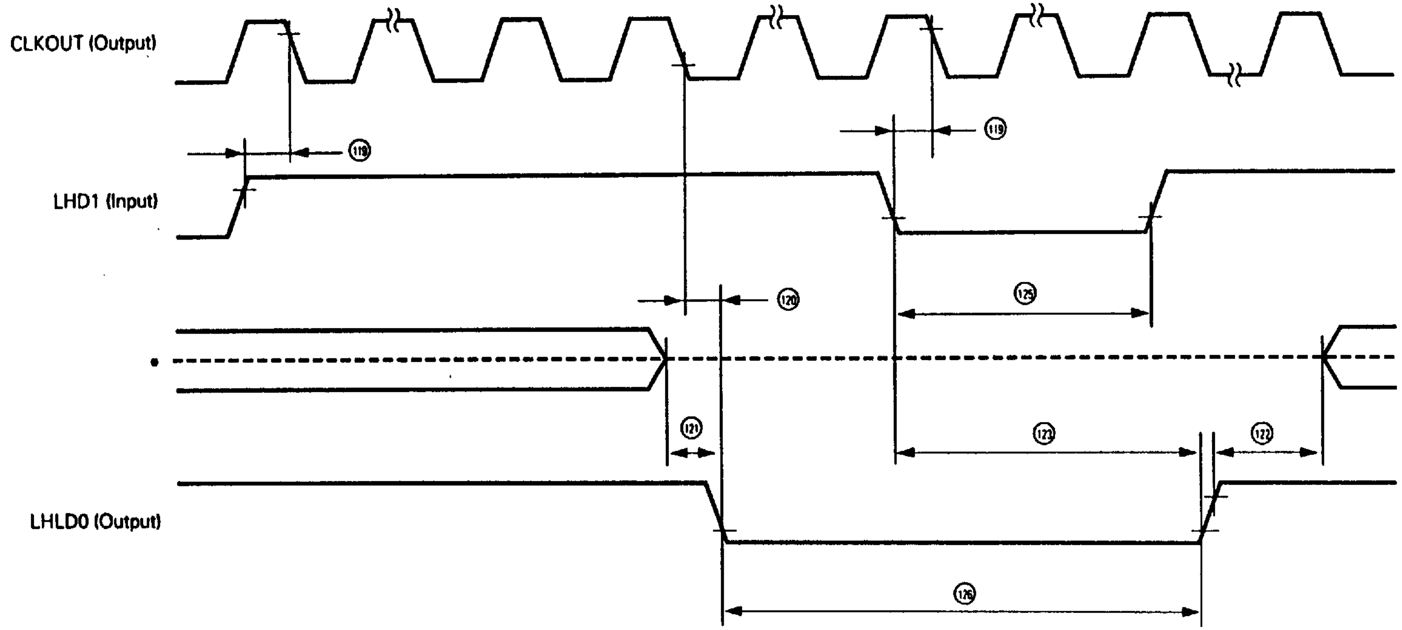


* $\overline{\text{ASTB}}$, $\overline{\text{RD}}$, $\overline{\text{WRH}}$, $\overline{\text{WRL}}$, $\overline{\text{DEX}}$, $\overline{\text{RAS}}$, $\overline{\text{BUSLOCK}}$, $\overline{\text{IORD}}$, $\overline{\text{IOWR}}$, AD0 to AD15, A16 to A23

★

Local Bus Hold Request/Acknowledge Timing

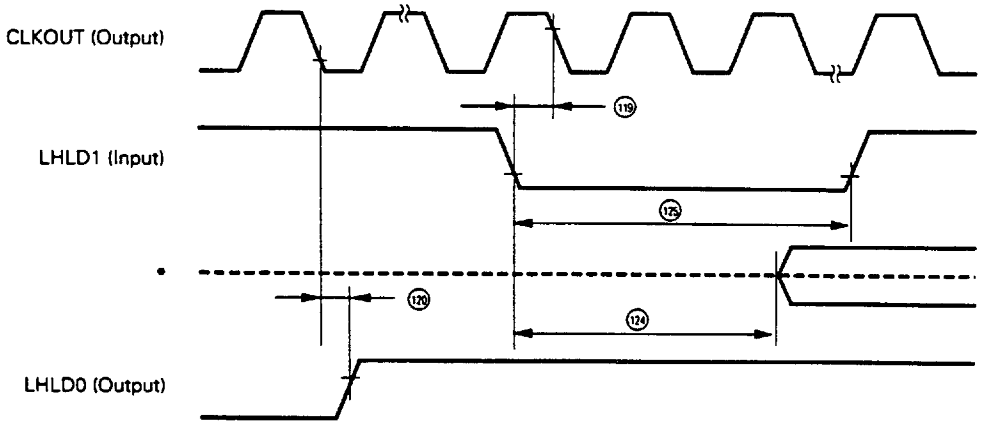
(1) In local bus master normal operation



Remarks The dotted line indicates high impedance.

- * $\overline{\text{LASTB}}$, $\overline{\text{LRD}}$, $\overline{\text{LWRH}}$, $\overline{\text{LWRL}}$, $\overline{\text{LRAS}}$, LAD0 to LAD15, LA16 to LA19

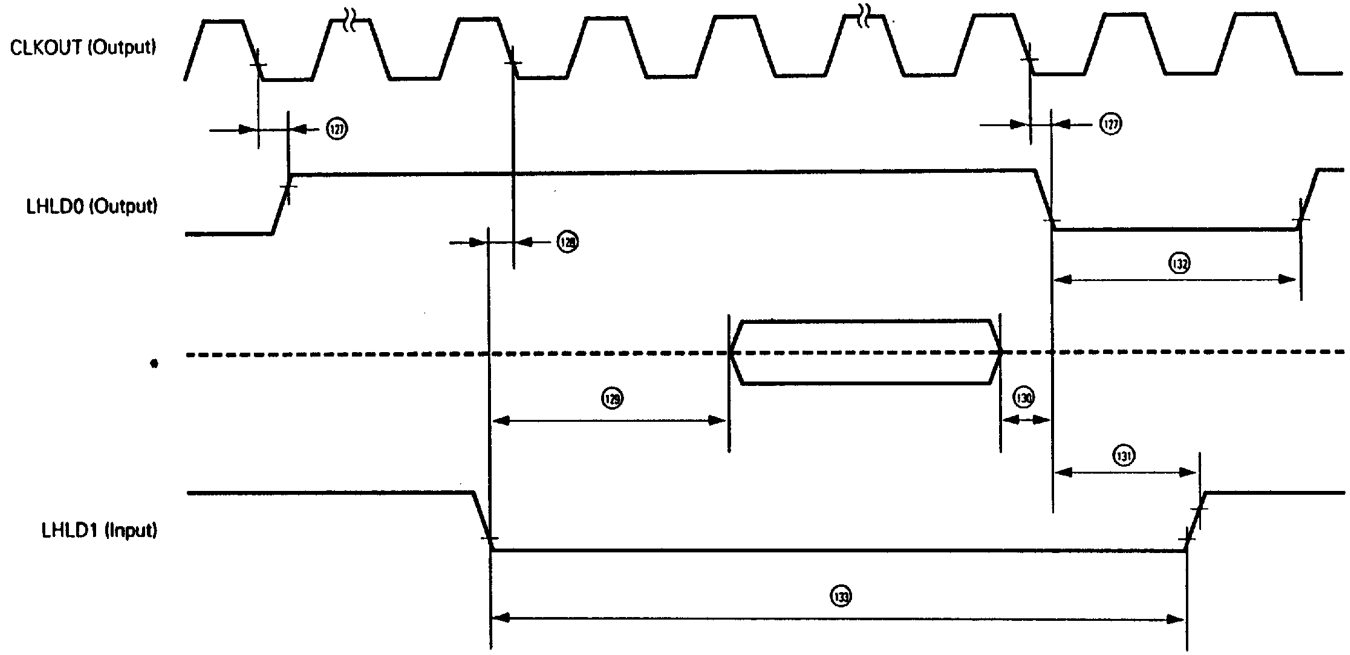
(2) **When bus hold is released for refresh cycle insertion in local bus master**



Remarks The dotted line indicates high impedance.

* $\overline{\text{LASTB}}$, $\overline{\text{LRD}}$, $\overline{\text{LWRH}}$, $\overline{\text{LWRL}}$, $\overline{\text{LRAS}}$, LAD0 to LAD15, LA16 to LA19

(3) In local bus slave normal operation



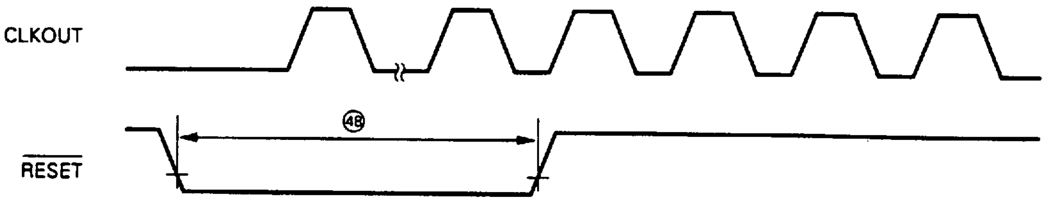
Remarks The dotted line indicates high impedance.

- * $\overline{\text{LASTB}}$, $\overline{\text{LRD}}$, $\overline{\text{LWRH}}$, $\overline{\text{LWRL}}$, $\overline{\text{LRAS}}$, LAD0 to AD15, LA16 to LA19

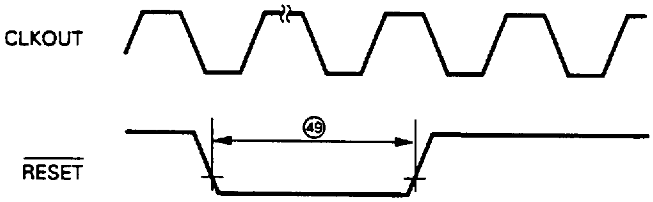
$\overline{\text{RESET}}$ Input Timing

★

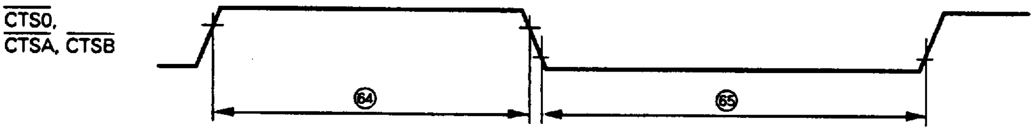
(1) After STOP mode release/power-on reset



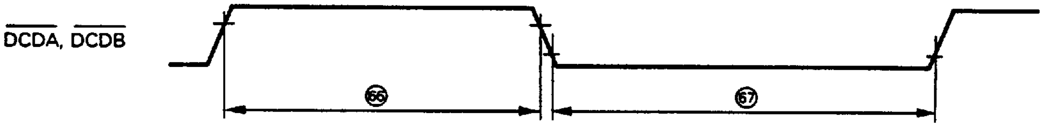
(2) After system reset



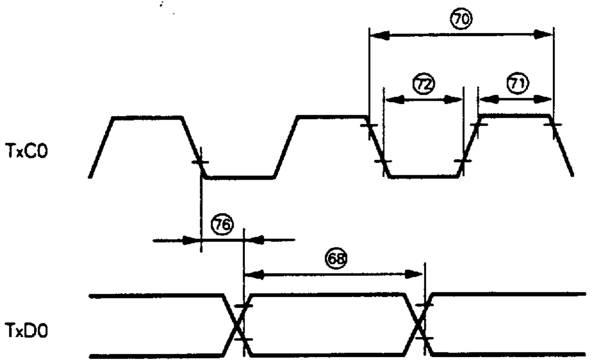
$\overline{\text{CTSm}}$ input timing (m = 0, A, B)



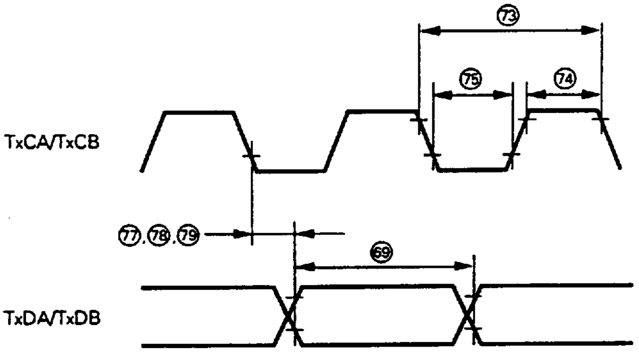
$\overline{\text{DCDm}}$ input timing (m = A, B)



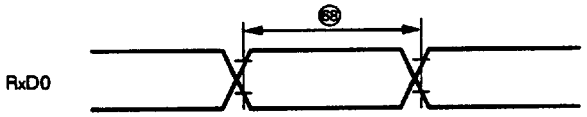
Send Timing (1)



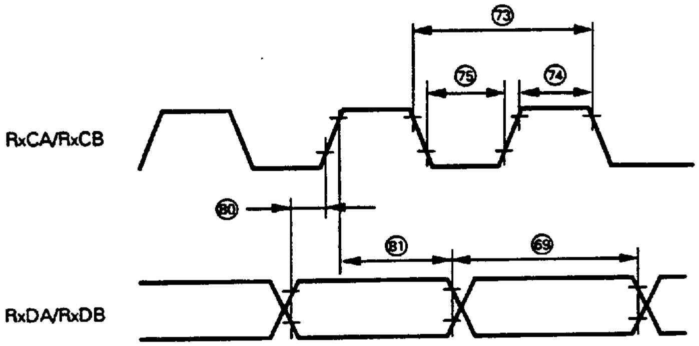
Send Timing (2)



Receive Timing (1)

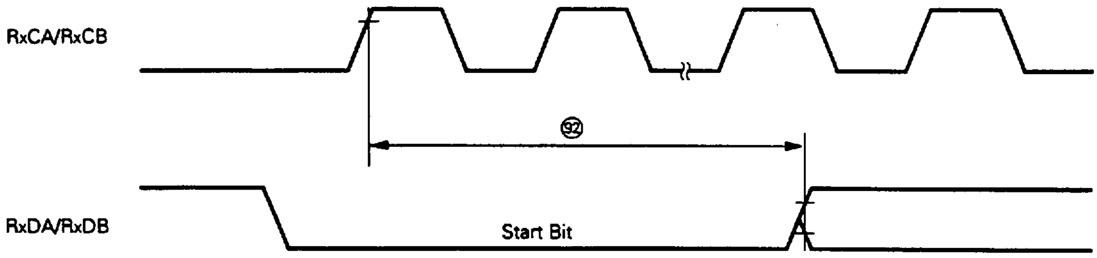


Receive Timing (2)

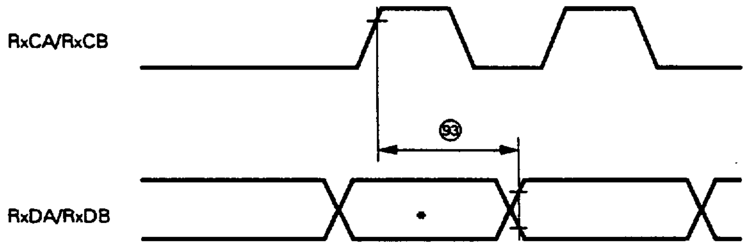


Receive Clock Setup Timing

(1) ASYNC mode

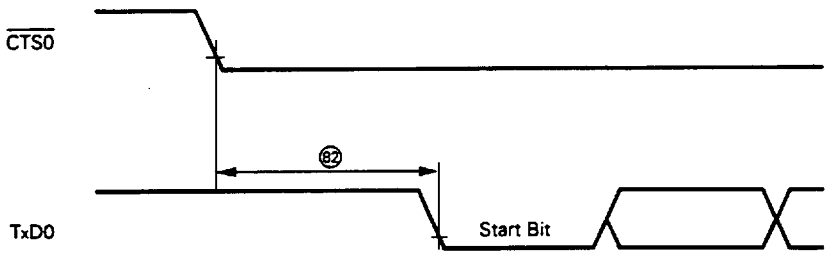


(2) SYNC/HDLC mode

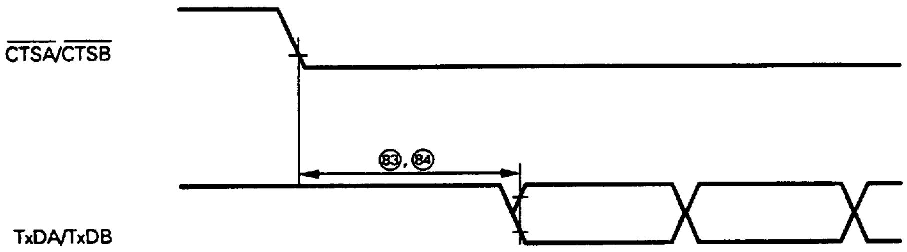


* LSB bit of the synchronization pattern (SYNC character, flag)

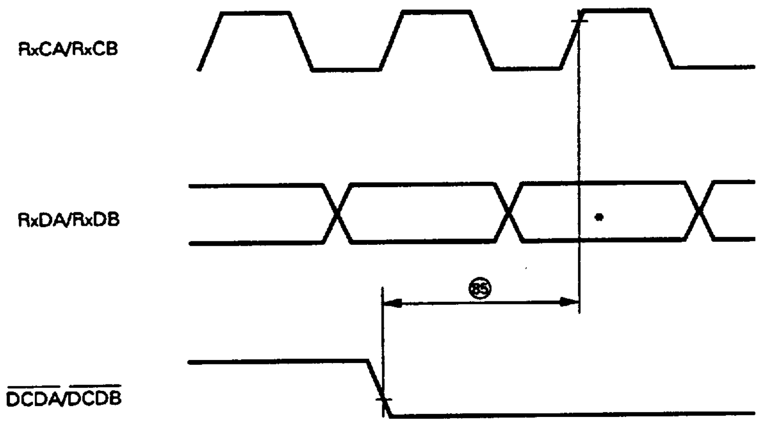
Send Enable Timing (1)



Send Enable Timing (2)



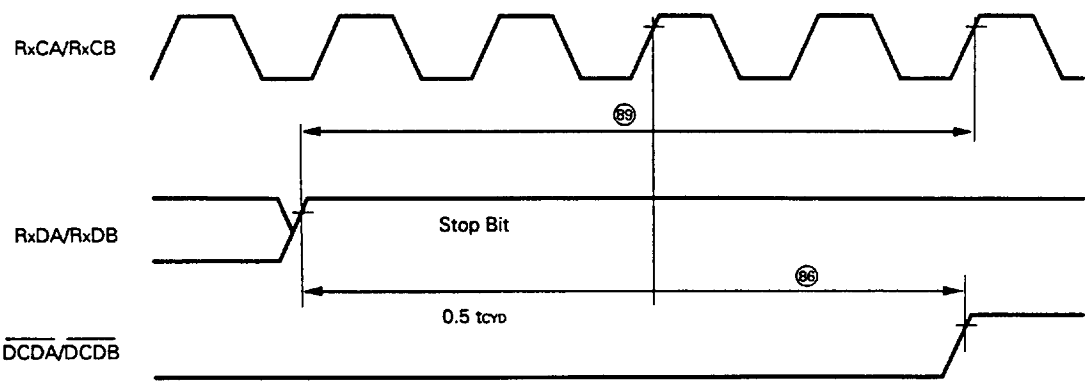
Receive Enable Timing



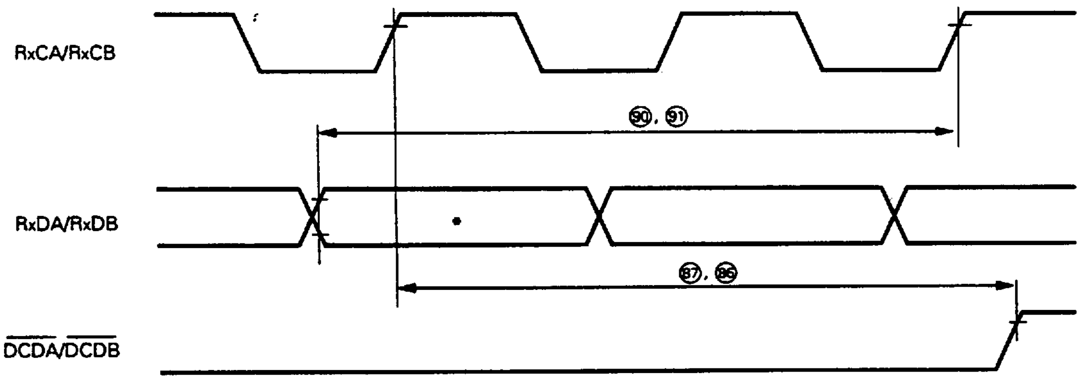
* The LSB bit of the first receive data (SYNC character, start flag)

DCD Timing and Receive Clock Hold Timing

(1) ASYNC mode

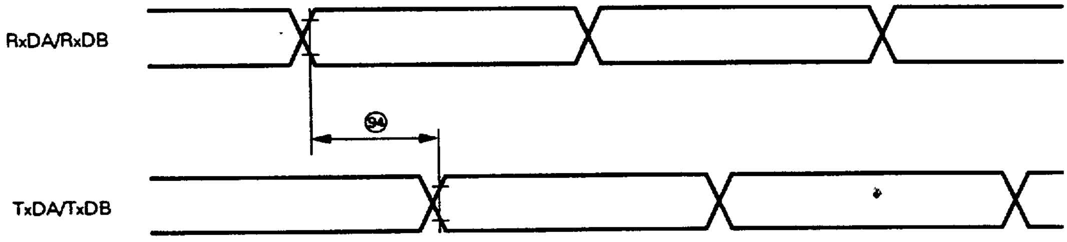


(2) SYNC/HDLC mode

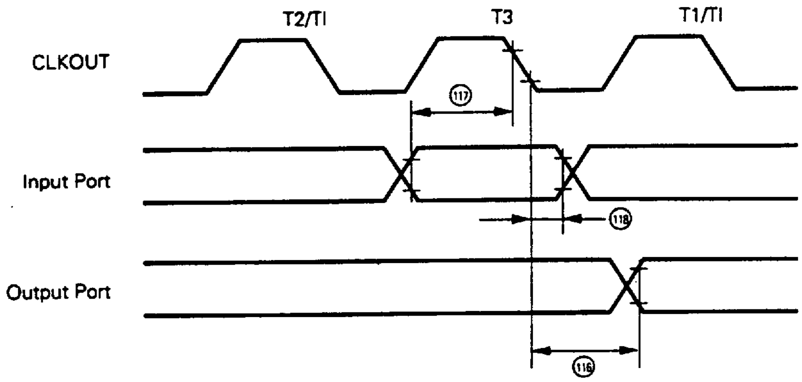


- * MSB bit of the FCS in the SYNC mode or the MSB bit of the end flag in the HDLC mode.

(3) Echo-Back mode



★ Port Input/Output Timing



18. AC CHARACTERISTICS (TARGET VALUE)

★

These characteristics are only target values; the volume production productsmay not always satisfy these specifications.

Only the target values that are different from the actual sample characteristics are shown here.

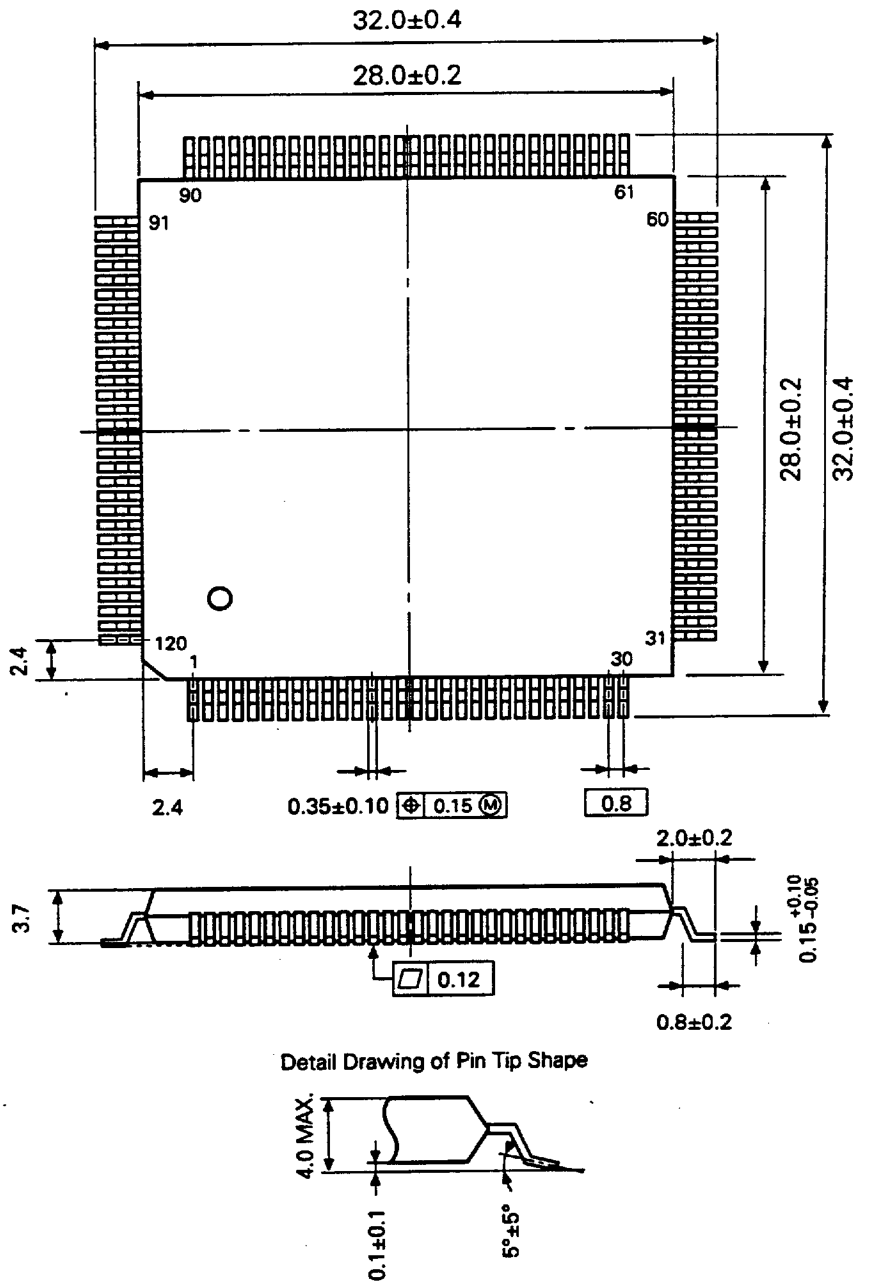
PARAMETER	SYMBOL		TEST CONDITIONS	MIN.	MAX.	UNIT
Address delay time from CLKOUT↑	17	t_{DKA}		5	30	V
$\overline{ASTB}$↓ delay time from CLKOUT↓	23	t_{DKSTL}		0	25	μs
$\overline{RAS}$↓ delay time from CLKOUT↑	41	t_{DKRAL}		nT	nT + 25	μs
$\overline{RD}$↓ delay time from CLKOUT↑	26	t_{DKRL}		0	25	ns
$\overline{WR}$↓ delay time from CLKOUT↓	35	t_{DKWL}		0	25	ns
Data output delay time from CLKOUT↑	38	t_{DKD}		3	30	ns
Address setup time (to $\overline{RAS}$↓)	115	t_{SARAL}		nT – 15		ns
Address setup time (to $\overline{ASTB}$↓)	21	t_{SAST}		(n + 0.5)T – 25		ns
$\overline{RAS}$↑ delay time from $\overline{WRH}$↓, $\overline{WRL}$↓	114	t_{DWRAH}		(N + 0.5)T – 10		ns
LHLD0 delay time from CLKOUT↓	120	t_{DKLHA}	with local bus master	5	35	ns
LHLD0 low-level width	126	t_{WLHAL1}		3T		ns
LHLD0 delay time (to CLKOUT↓)	127	t_{DLHOK}	with local bus slave	5	35	ns
LHLD0 low-level width	132	t_{WLHOL2}		2T		ns

n : Number of address wait states
N : Number of data wait states
T : t_{CYK}

Remarks Figures in the symbol column correspond to those in the timing chart in 17. **"Electrical Specifications (Preliminary)".**

19. PACKAGE INFORMATION

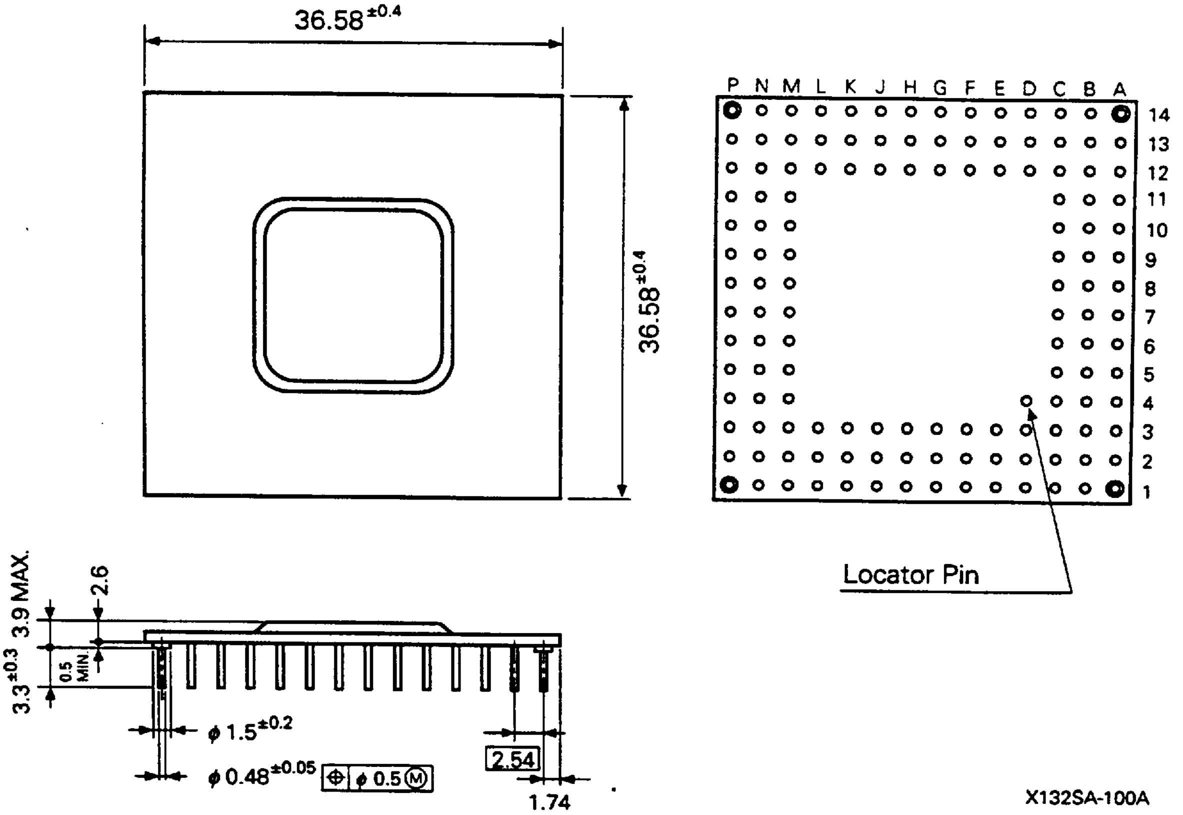
120-Pin Plastic QFP (□28) (Unit: mm)



Detail Drawing of Pin Tip Shape

P120GD-80-5BB-2

132-Pin Plastic PGA (Unit: mm)



★ 20. RECOMMENDED SOLDERING CONDITIONS

This product should be soldered and mounted under the conditions recommended in the table below.

For details of recommended soldering conditions, refer to the information document **"Semiconductor Device Mount Manual"** **(IEI-1207)**.

For soldering methods and conditions other than those recommended below, contact our salesman.

Table 20-1 Surface-Mounted Type Soldering Conditions

μPD70423GD-5BB : 120 pin Plastic QFP (□28)

Soldering Method	Solderring Conditions	Recommended Condition Symbol
Infrared reflow	Package peak temperature: 235°C, Duration: 30 sec. max. (at 210°C or above), Number of times: Once, Time limit: 7 days* (thereafter 36 hours 125°C prebanking required)	IR35-367-1
Pin part heating	Pin part temperature: 300°C or below, Duration: 3 sec. max. (per device side)	

* For the storage period after dry-pack decompression storage conditions are max. 25 °C, 65 % RH.

Note **Use of more than one soldering method should be avoided (except in the case of pin part heating).**

Table 20-2 Insertion Type Soldering Conditions

μPD70423SA : 132 pin Plastic PGA

Soldering Method	Solderring Conditions
Waving soldering (lead part only)	Soldering bath temperature: 260°C or below, Duration: 10 sec. max.
Pin part heating	Pin part temperature: 260°C or below, Duration: 10 sec. max.

Note **Ensure that the application of wave soldering is limited to the lead part and no solder touches the main unit directly.**