DATA SHEET

NEC

MOS INTEGRATED CIRCUIT

μPD70108H, 70116H

V20HL™, V30HL™
16/8, 16-BIT MICROPROCESSOR

DESCRIPTION

The μPD70108H (V20HL) and μPD70116H (V30HL) are CMOS 16-bit microprocessors developed from the μPD70108 (V20™) and μPD70116 (V30™). It offers higher processing speed and lower power consumption than the μPD70108 and μPD70116.

The μPD70108H and μPD70116H are capable of operating at 16 MHz. In addition to the conventional standby function, fully static internal circuits are employed in the μPD70108H and μPD70116H. This allows the μPD70108H and μPD70116H to have the clock stop function. Therefore, power consumption is significantly reduced in the μPD70108H and μPD70116H. Additionally, the μPD70108H and μPD70116H can operate at 5 V. However, the μPD70108H and μPD70116H are designed to also operate at 3 V.

FEATURES

- High-speed, low-power consumption version of μPD70108 and μPD70116
- Single power supply (5 V or 3 V)
- 125-ns minimum instruction execution time at 16 MHz (5 V)
  250-ns minimum instruction execution time at 8 MHz (3 V)
- High-speed multiplication/division instructions:
  1.2 to 3.6 μs (at 16 MHz, 5 V)
  2.4 to 7.1 μs (at 8 MHz, 3 V)
- High-speed block transfer instructions:
  μPD70108H: 2M bytes/second (at 16 MHz, 5 V)
  1M bytes/second (at 8 MHz, 3 V)
  μPD70116H: 2M words/second (at 16 MHz, 5 V)
  1M words/second (at 8 MHz, 3 V)

C

Document No. IC-3552A
(O. D. No. IC-8154C)
Date Published March 1995 P
Printed in Japan

The mark ★ shows major revised points.

ORDERING INFORMATION

(1) μPD70108H

Part number	Package	Maximum operating frequency (MHz)
μPD70108HCZ-10	40-pin plastic DIP (600 mil)	10
μPD70108HCZ-12	40-pin plastic DIP (600 mil)	12.5
μPD70108HCZ-16	40-pin plastic DIP (600 mil)	16
μPD70108HGC-10-3B6	52-pin plastic QFP (□14 mm) (resin thickness: 2.70 mm)	10
μPD70108HGC-12-3B6	52-pin plastic QFP (□14 mm) (resin thickness: 2.70 mm)	12.5
μPD70108HGC-16-3B6	52-pin plastic QFP (□14 mm) (resin thickness: 2.70 mm)	16
μPD70108HG-10-22	52-pin plastic QFP (□14 mm) (resin thickness: 1.50 mm)	10
μPD70108HG-12-22	52-pin plastic QFP (□14 mm) (resin thickness: 1.50 mm)	12.5
μPD70108HG-16-22	52-pin plastic QFP (□14 mm) (resin thickness: 1.50 mm)	16
μPD70108HLM-10	44-pin plastic QFJ (□650 mil)	10
μPD70108HLM-12	44-pin plastic QFJ (□650 mil)	12.5
μPD70108HLM-16	44-pin plastic QFJ (□650 mil)	16

Remark Plastic QFJ is a new name for PLCC.

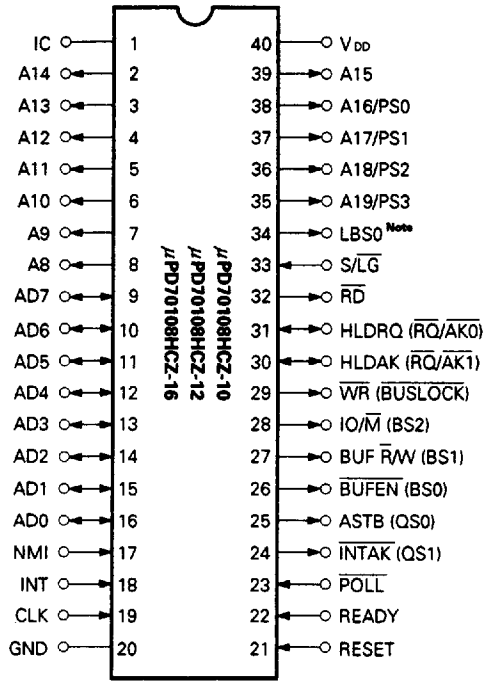
(2) μPD70116H

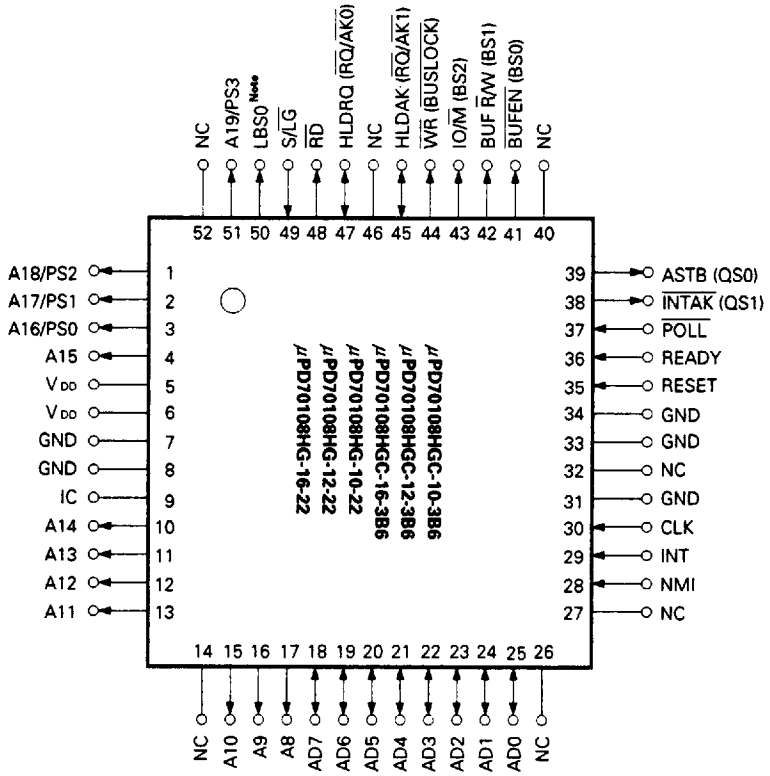
Part number	Package	Maximum operating frequency (MHz)
μPD70116HCZ-10	40-pin plastic DIP (600 mil)	10
μPD70116HCZ-12	40-pin plastic DIP (600 mil)	12.5
μPD70116HCZ-16	40-pin plastic DIP (600 mil)	16
μPD70116HGC-10-3B6	52-pin plastic QFP (□14 mm) (resin thickness: 2.70 mm)	10
μPD70116HGC-12-3B6	52-pin plastic QFP (□14 mm) (resin thickness: 2.70 mm)	12.5
μPD70116HGC-16-3B6	52-pin plastic QFP (□14 mm) (resin thickness: 2.70 mm)	16
μPD70116HG-10-22	52-pin plastic QFP (□14 mm) (resin thickness: 1.50 mm)	10
μPD70116HG-12-22	52-pin plastic QFP (□14 mm) (resin thickness: 1.50 mm)	12.5
μPD70116HG-16-22	52-pin plastic QFP (□14 mm) (resin thickness: 1.50 mm)	16
μPD70116HLM-10	44-pin plastic QFJ (□650 mil)	10
μPD70116HLM-12	44-pin plastic QFJ (□650 mil)	12.5
μPD70116HLM-16	44-pin plastic QFJ (□650 mil)	16

Remark Plastic QFJ is a new name for PLCC.

PIN CONFIGURATIONS (Top View)

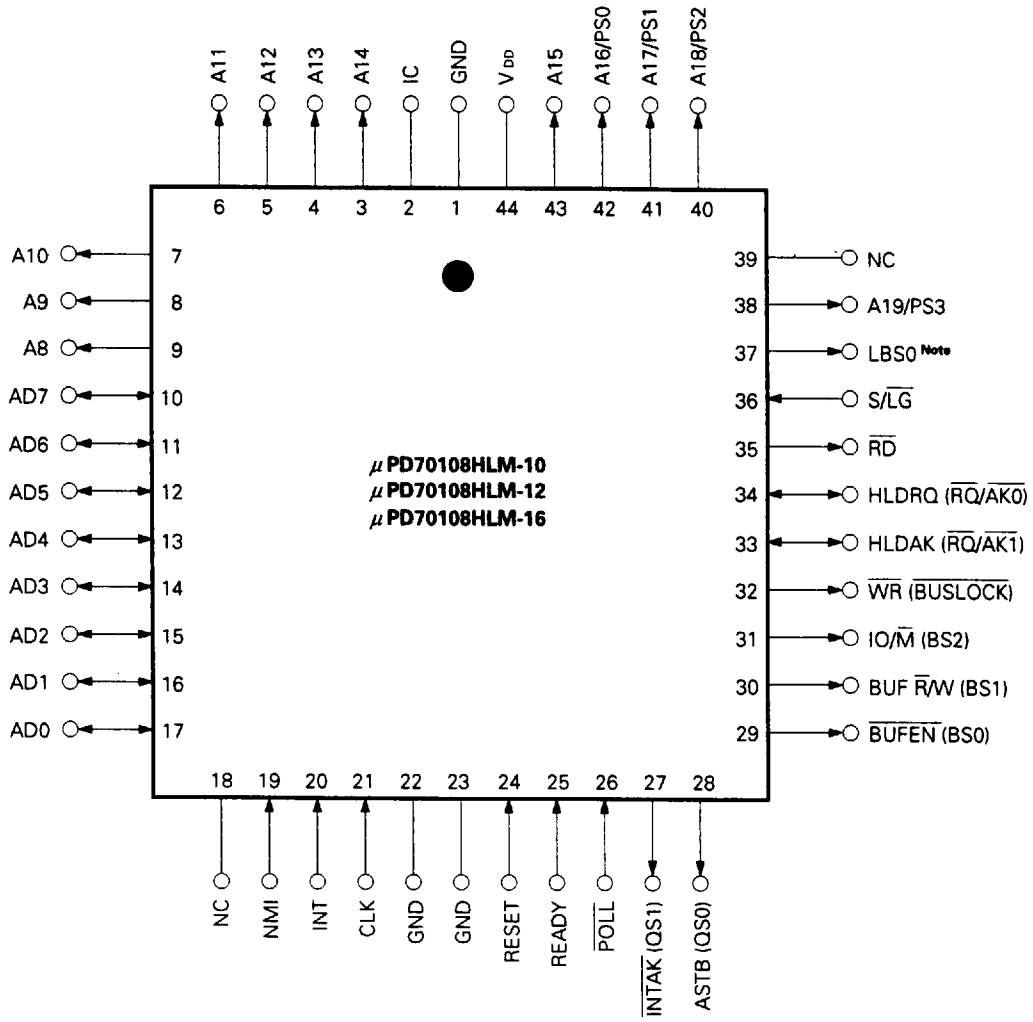
(1) μPD70108H





Note Outputs high level in large-scale mode.

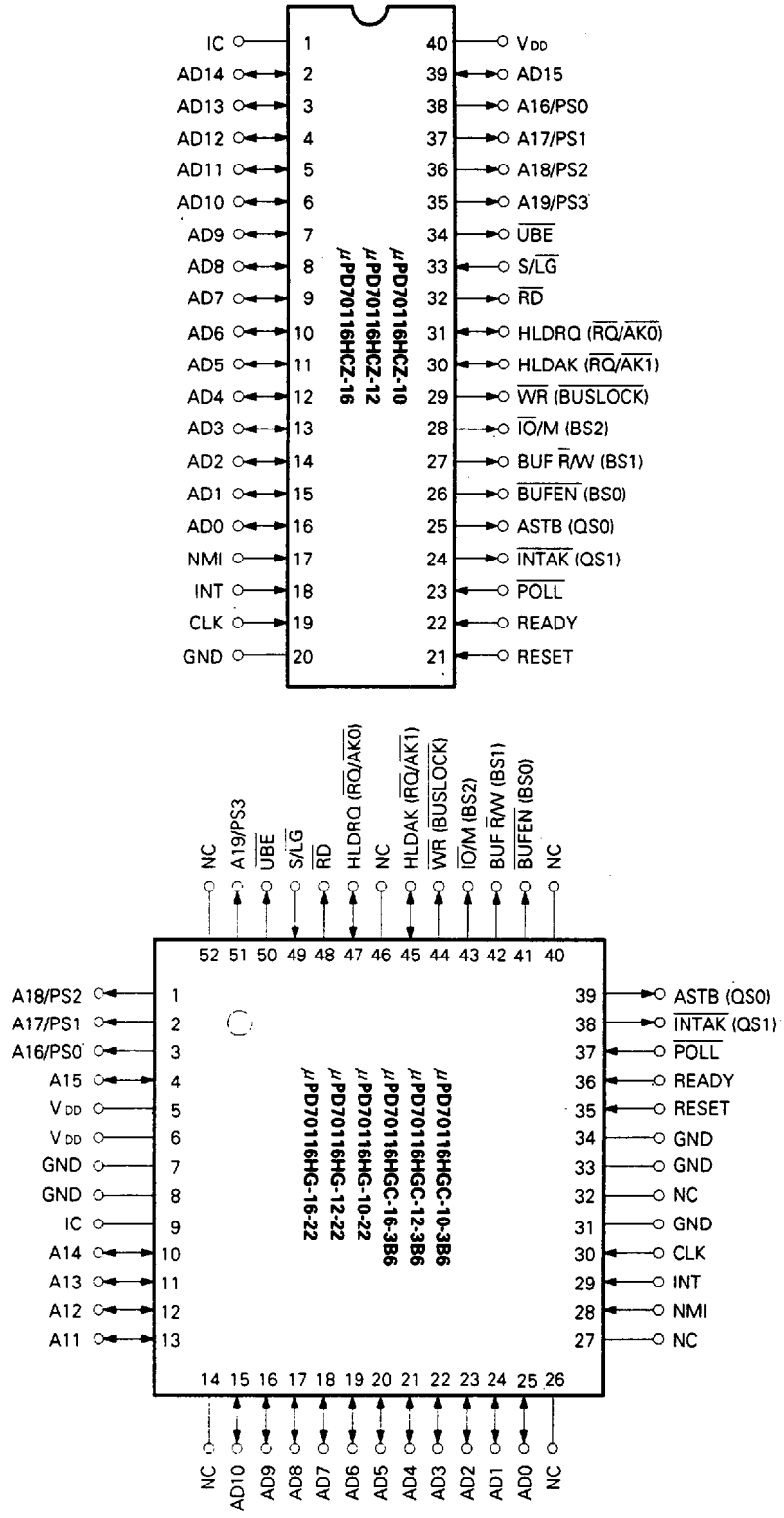
NC: No connection
IC : Internally connected to GND.
( ) : In large-scale mode



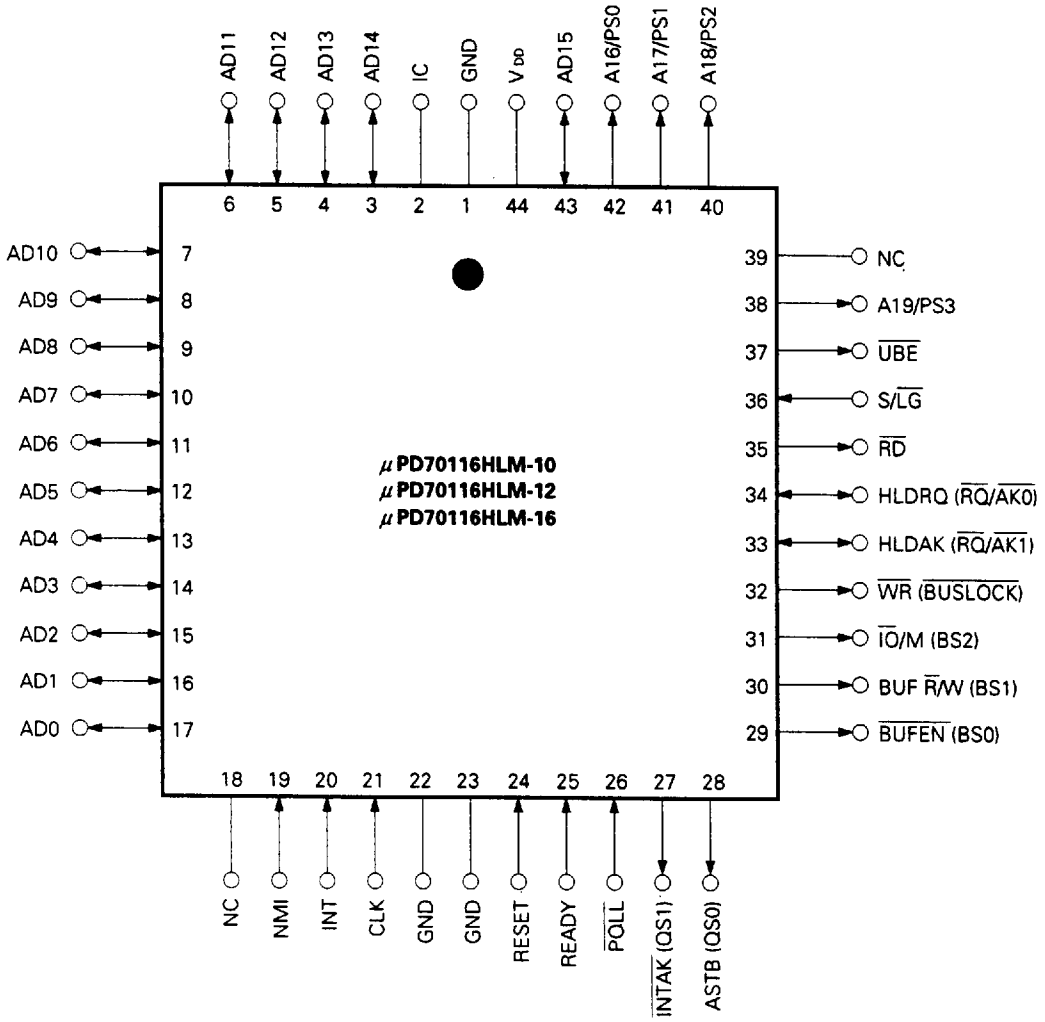
Note Outputs high level in large-scale mode.

NC: No connection
IC : Internally connected to GND.
( ) : In large-scale mode

(2) μPD70116H



NC: No connection
IC : Internally connected to GND.
( ) : In large-scale mode



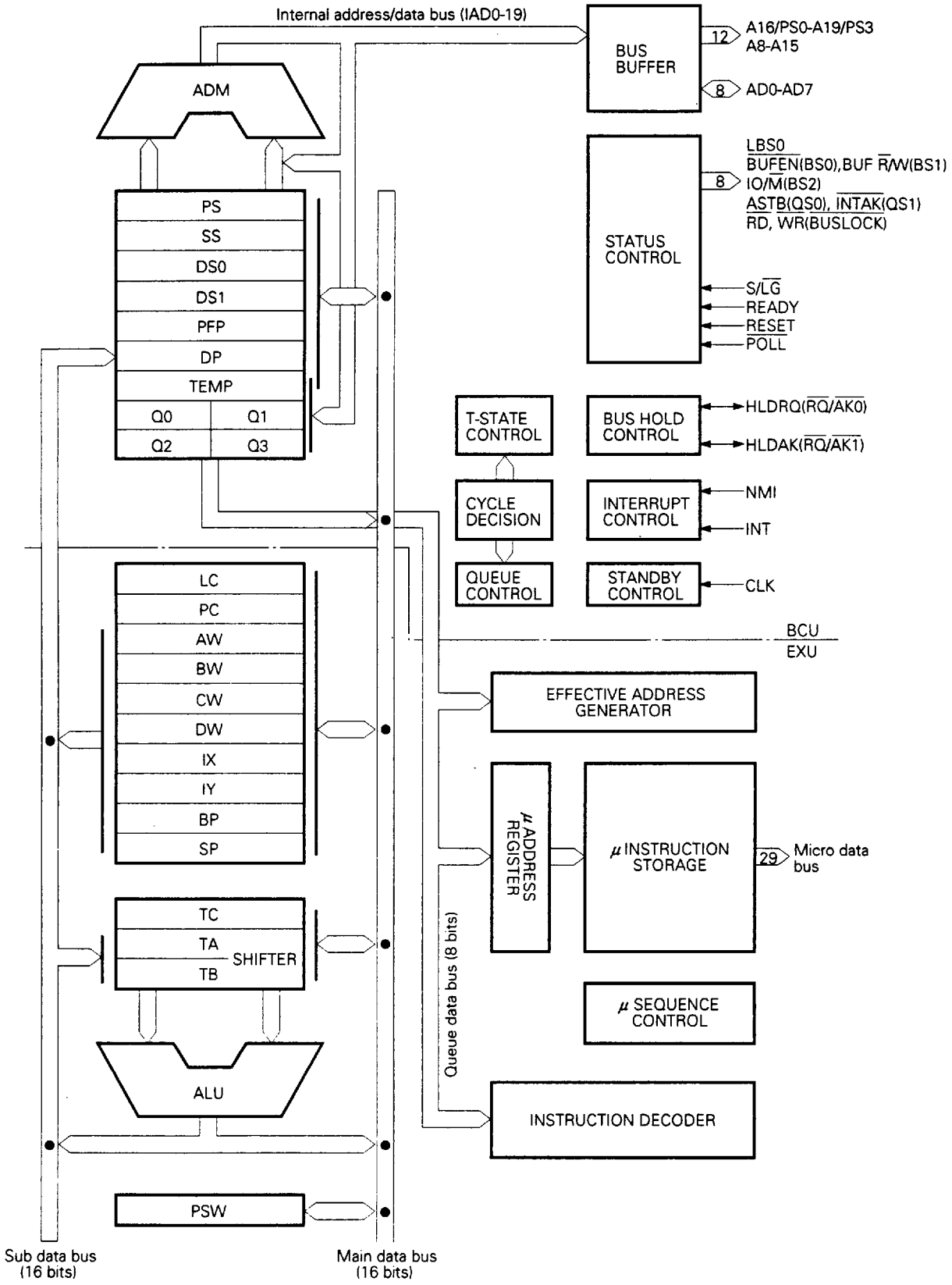
NC: No connection

IC : Internally connected to GND.

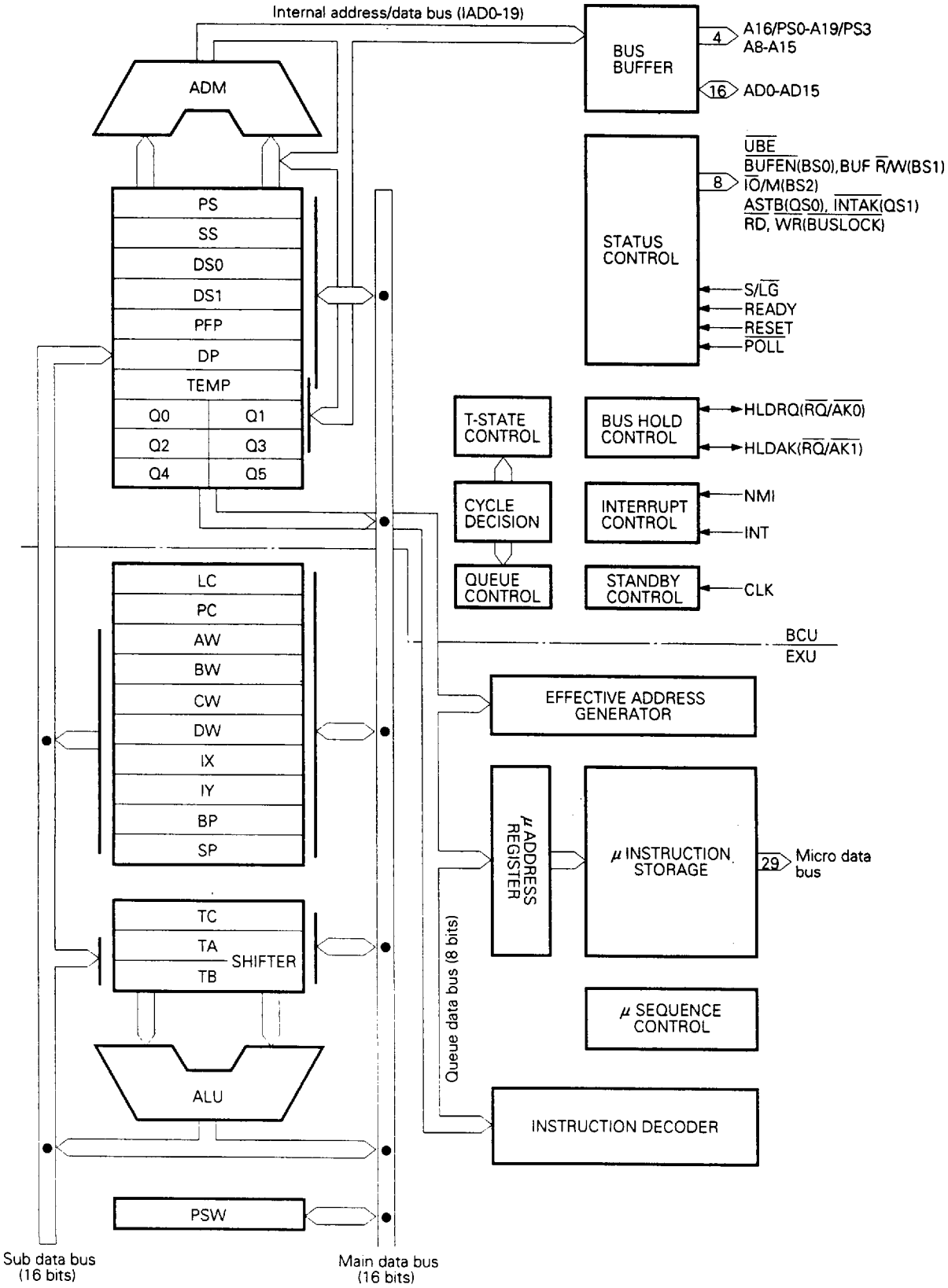
( ) : In large-scale mode

INTERNAL BLOCK DIAGRAM

(1) μPD70108H



(2) μPD70116H



Differences between μPD70108H and μPD70116H, and μPD70108 and μPD70116

Item \ Product Name		μPD70108H, 70116H	μPD70108, 70116
Operating Voltage		5 V, 3 V	5 V
Operating Frequency	At V_{DD}=5 V	MAX.: 10, 12.5, 16 MHz MIN. : DC	MAX.: 5, 8, 10 MHz MIN. : 2 MHz
	At V_{DD}=3 V	MAX.: 5, 6, 8 MHz MIN. : DC	Does not operate
Pull-up Resistor of $\overline{RQ}/\overline{AK}$ Pin (in Small- Scale Mode)		Not provided	Provided
On Reset	$\overline{BUFEN}$ (BS0)[Note1]	H	Hi-Z
	$\overline{BUFR/W}$ (BS1)[Note1]	H	Hi-Z
	IO/$\overline{M}$[Note2] (BS2)[Note1]	H	Hi-Z
	$\overline{IO}$/M[Note3] (BS2)[Note1]		
	$\overline{WR}$ ($\overline{BUSLOCK}$)[Note1]	H	Hi-Z
	$\overline{RD}$	H	Hi-Z
52-Pin Plastic QTF (Resin Thick 1.50 mm)		Provided	Not provided

Note 1. ( ): In large-scale mode.
2. μPD70108, 70108H
3. μPD70116, 70116H

CONTENTS

1. PIN FUNCTIONS

1.1 LIST OF PIN FUNCTIONS

(1) Small-scale mode (S/$\overline{LG}$ = H)

Pin Name	Input/Output	Function
A16/PS0-A19/PS3	Tri-state output	Address bus/processor status
A8-A15 (μPD70108H)	Tri-state output	Address bus
AD0-AD7 (μPD70108H)	Tri-state output	Address/data bus
AD0-AD15 (μPD70116H)	Tri-state input/output	Address/data bus
ASTB	Output	Address strobe
$\overline{INTAK}$	Output	Interrupt acknowledge display
$\overline{BUFEN}$	Tri-state output	External data buffer enable
BUF$\overline{R}$/W	Tri-state output	External data buffer read/write selection
IO/$\overline{M}$ (μPD70108H)	Tri-state output	I/O, memory selection
$\overline{IO}$/M (μPD70116H)		
LSB0 (μPD70108H)	Tri-state output	Bus status
$\overline{UBE}$ (μPD70116H)	Tri-state output	Data bus upper byte enable
$\overline{RD}$	Tri-state output	Read strobe
$\overline{WR}$	Tri-state output	Write strobe
HLDRQ	Input	Bus hold request
HLDAK	Output	Bus hold enable
S/$\overline{LG}$	Input	Small-scale/large-scale mode selection (=H)
NMI	Input	Non-maskable interrupt request
INT	Input	Maskable interupt request
CLK	Input	System clock
READY	Input	External ready display
$\overline{POLL}$	Input	External polling display
RESET	Input	System reset

(2) Large-scale mode (S/$\overline{LG}$ = L)

Pin Name	Input/Output	Function
A16/PS0-A19/PS3	Tri-state output	Address bus/processor status
A8-A15 (μPD70108H)	Tri-state output	Address bus
AD0-AD7 (μPD70108H)	Tri-state input/output	Address/data bus
AD0-AD15 (μPD70116H)	Tri-state input/output	Address/data bus
QS0, QS1	Output	Queue status
BS0-BS2	Tri-state output	Bus status
LSB0 (μPD70108H) (pin for small-scale mode)	Output	Outputs high level
$\overline{UBE}$ (μPD70116H)	Tri-state output	Data bus upper byte enable
$\overline{RD}$	Tri-state output	Read strobe
$\overline{BUSLOCK}$	Tri-state output	Write strobe
$\overline{RQ}$/AK0, $\overline{RQ}$/AK1	Tri-state input/output	Bus hold request/enable
S/$\overline{LG}$	Input	Small-scale/large-scale mode selection (=L)
NMI	Input	Non-maskable interrupt request
INT	Input	Maskable interupt request
CLK	Input	System clock
READY	Input	External ready display
$\overline{POLL}$	Input	External polling display
RESET	Input	System reset

1.2 PIN STATUS UNDER SPECIFIC CONDITIONS

(1) Small-scale mode

Pin Name	Conditions		
	Hold	Halt	Reset
A8-A15[Note1] (μPD70108H)	Hi-Z	H or L	Hi-Z
AD0-AD7[Note1] (μPD70108H)	Hi-Z	H or L	Hi-Z
AD0-AD15[Note1] (μPD70116H)	Hi-Z	H or L	Hi-Z
A16/PS0-A19/PS3[Note1]	Hi-Z	H or L	Hi-Z
ASTB	L	**Note2**	L
$\overline{\text{INTAK}}$	H	H	H
$\overline{\text{BUFEN}}$[Note1]	Hi-Z	H	**Note3**
BUFR/$\overline{\text{W}}$[Note1]	Hi-Z	H or L	**Note3**
IO/$\overline{\text{M}}$[Note1] (μPD70108H)	Hi-Z	H or L	**Note3**
$\overline{\text{IO}}$/M[Note1] (μPD70116H)	Hi-Z	H or L	**Note3**
LSB0[Note1] (μPD70108H)	Hi-Z	H	Hi-Z
$\overline{\text{UBE}}$[Note1] (μPD70116H)	Hi-Z	H	Hi-Z
$\overline{\text{RD}}$[Note1]	Hi-Z	H	**Note3**
$\overline{\text{WR}}$[Note1]	Hi-Z	H	**Note3**
HLDAK	H	L	L

Note 1. An internal latch is provided, so that the status before this pin goes into a high-impedance (Hi-Z) state is retained, until the pin is driven by an external source.

2. High level only once (for about half a clock cycle) during the halt acknowledge cycle, otherwise low level.

3. Outputs the high level as long as the reset signal is active, and goes into a Hi-Z state after the reset signal has become inactive and until the first bus cycle is started. Therefore, if the pin is not driven by an external source, the high level is retained by a latch.

(2) Large-scale mode

Pin Name	Conditions		
	Hold	Halt	Reset
A8-A15[Note1] (μPD70108H)	Hi-Z	H or L	Hi-Z
AD0-AD7[Note1] (μPD70108H)	Hi-Z	H or L	Hi-Z
AD0-AD15[Note1] (μPD70116H)	Hi-Z	H or L	Hi-Z
A16/PS0-A19/PS3[Note1]	Hi-Z	H or L	Hi-Z
QS0	**Note 2**	L	L
QS1	L	L	L
BS0[Note1]	Hi-Z	H	**Note 3**
BS1[Note1]	Hi-Z	H	**Note 3**
BS2[Note1]	Hi-Z	H	**Note 3**
$\overline{UBE}$[Note1] (μPD70116H)	Hi-Z	H	Hi-Z
$\overline{RD}$[Note1]	Hi-Z	H	**Note 3**
$\overline{BUSLOCK}$[Note1]	Hi-Z	H or L	**Note 3**
$\overline{RQ}$/AK0[Note4]	H	H	H
$\overline{RQ}$/$\overline{AK1}$[Note4]	H	H	H
LSB0[Note2] (μPD70108H) (pin for small-scale mode)	Hi-Z	H	Hi-Z

Note 1. An internal latch is provided, so that the status before this pin goes into a high-impedance (Hi-Z) state is retained, until the pin is driven by an external source.

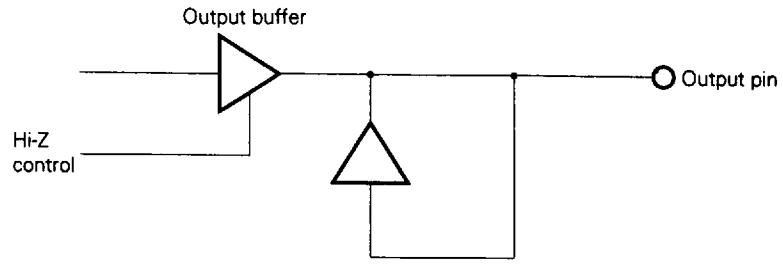
2. High level only once (for about half a clock cycle) during the halt acknowledge cycle, otherwise low level.

3. Outputs the high level as long as the reset signal is active, and goes into a Hi-Z state after the reset signal has become inactive and until the first bus cycle is started. Therefore, if the pin is not driven by an external source, the high level is retained by a latch.

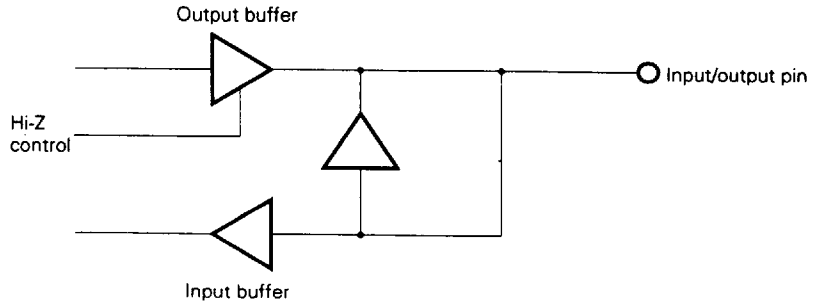
4. The $\overline{RQ}$/$\overline{AK}$ pin is connected with a pull-up resistor in the large-scale mode.

Remark The circuit configuration of the latch is as shown below. To invert the status of the pin with a latch, a drive current higher than the latch inverting current (I_{ILH}, I_{ILL}) is required.

Output pin



Input/output pin



1.3 FUNCTIONAL DESCRIPTION

Some pins have meaning only for small-scale systems or large-scale systems, while others are used for both small- and large-scale systems.

(1) A8-A15 (Address Bus) ... Small/Large Scale (μPD70108H)

A8-A15 output the middle 8 bits of the 20-bit address information.

A8-A15 are 3-state output pins and become high impedance during hold acknowledge cycles.

(2) AD0-AD7 (Address/Data Bus) ... Small/Large Scale (μPD70108H)

AD0-AD7 serve as the time-multiplexed address/data bus. The lower 8-bit output of the 20-bit address information and data input/output are multiplexed.

16-bit data is input/output in two operations, the lower byte is the first byte, and the upper byte is the second.

These pins are 3-state input/output pins and become high impedance during hold acknowledge cycles and interrupt acknowledge cycles.

(3) AD0 - AD15 (Address/Data Bus)... Small/Large Scale (μPD70116H)

This bus, which is for both addresses and data, performs the low 16-bit output of the 20-bit address information. The byte/word data input/output are performed as well in the time-sharing method.

μPD70116H accesses memory or I/O operands, in terms of two separate areas, i.e., the byte data bank which is accessed with an even-numbered address (AD0=0) and the byte data bank which is accessed with an odd-number address (AD0=1). The least significant bit (AD0) does not have a meaning of its own as the address information word data; however, it is used to distinguish the byte bank of the odd-number address from that of the even-number address.

In addition to AD0, there are also the $\overline{UBE}$ signals available for the purpose of accessing byte/word data. The $\overline{UBE}$ signals are used in a combination as shown in the table below.

Operand		$\overline{UBE}$	AD0	Bus Cycle Count
Even-number Address Word		0	0	1
Odd-number Address Word	First bus cycle	0	1	2
	Second bus cycle	1	0	
Even-number Address Byte		1	0	1
Odd-number Address Byte		0	1	1

The word operand of an odd-number address is accessed twice, i.e., in terms of the odd-number byte bank and the even-number byte bank. In the course of the access, "AD0=1" which indicates the odd-number address bank is output in the first bus cycle; and in the second bus cycle, "AD0=0" is automatically output in order to indicate the continuing even-number address bank. Each of the pins for 3-stated output. During hold acknowledge or interrupt acknowledge, they become high-impedance. In standby mode, they are fixed to either a high or low level.

(4) NMI (Non-Maskable Interrupt) ... Small/Large Scale

NMI is a non-maskable interrupt request signal input, which cannot be masked by software.

This signal is rising edge active, and is sampled in any clock cycle. However, interrupt processing is initiated after completing the current instruction execution.

The interrupt start address for this interrupt is determined by interrupt vector 2.

The NMI signal must be held at high level, at least for five clock periods, after the rising edge.

The priority order for the NMI interrupt is as indicated below. The hold request is accepted during NMI

acknowledge cycles.

INT < NMI < HLDRQ (small scale) or $\overline{RQ}$ (large scale)

This interrupt is also used to release the standby mode.

(5) INT (Maskable Interrupt) .... Small/Large Scale

INT is a maskable interrupt request signal input, which can be masked by software.

This input is high level active and is sampled in the last clock period of an instruction. It is accepted, when interrupt is enabled (the interrupt enable flag (IE) is set). The external device checks whether or not the interrupt is accepted by the $\overline{INTAK}$ signal output from the CPU.

The INT signal must be held at high level at least until the first $\overline{INTAK}$ signal is output.

The priority order is as shown below. If NMI is simultaneously generated, the priority is given to the NMI, and INT will not be accepted. The hold request can be accepted during INT acknowledge cycles.

INT < NMI < HLDRQ (small scale) or $\overline{RQ}$ (large scale)

This interrupt is also used to release the standby mode.

(6) CLK (Clock) ... Small/Large Scale

CLK is the external clock input. The power supply current can be significantly reduced by stopping this CLK input (DC level). However, if the CLK input is halted while the RESET input is active (high level), the source current specifications are not satisfied.

(7) RESET (Reset) ... Small/Large Scale

RESET is the high level active CPU reset input. This input has priority over any other operation.

After reset, the CPU starts program execution from address FFFF0H.

The RESET input is used for normal CPU start. In addition, the RESET input is also used for releasing the standby mode.

Do not stop the clock input when making the RESET input active.

(8) READY (Ready) ... Small/Large Scale

When the memory or I/O cannot complete read/write operation within the basic CPU access time, this signal can be set to inactive (low level) to request the CPU to generate wait state (TW), in order to extend the read/write cycle.

The CPU will not generate wait state, if the READY signal is active (high level) during T3 or TW states.

The setup/hold time for this signal must be satisfied; otherwise, no correct operation can be guaranteed. Therefore, external synchronization is necessary.

(9) $\overline{POLL}$ (Poll) ... Small/Large Scale

The $\overline{POLL}$ input is sampled by the POLL instruction. If this signal is low when checked, the processing moves to the next instruction execution. If it is high, the $\overline{POLL}$ input is sampled every 5 clock periods, until it becomes low.

This function is used to synchronize the CPU program with the external device operation.

(10) $\overline{INTAK}$ (Interrupt Acknowledge) ... Small Scale

$\overline{INTAK}$ signal is output, when an INT signal is accepted. The external device inputs the interrupt vector to the CPU through the data bus (AD7-AD0) in synchronization with this signal.

This output is fixed to high level in the standby mode.

(11) ASTB (Address Strobe) ... Small Scale

ASTB is the address strobe signal, used to latch the address information into the external latch.

In standby mode, fix this pin to low level after setting it to a high level (for about 1/2 of a clock cycle).

(12) $\overline{\text{BUFEN}}$ (Buffer Enable) ... Small Scale

$\overline{\text{BUFEN}}$ signal is used as the external bi-directional buffer output enable signal. This signal is output, when transferring data between the memory and I/O or when inputting interrupt vector.

The $\overline{\text{BUFEN}}$ output is fixed to high during the standby mode.

This pin is a 3-state output, and becomes high impedance during hold acknowledge cycles.

(13) BUF $\overline{\text{R}}$/W (Buffer Read/Write) ... Small Scale

BUF $\overline{\text{R}}$/W signal is output to determine the data transfer direction for the external bi-directional buffer. High indicates CPU-to-external device transfer. Low indicates data transfer from the external device to the CPU.

The BUF $\overline{\text{R}}$/W output is fixed to high or low during the standby mode.

This pin is a 3-state output, and becomes high impedance during hold acknowledge cycles.

(14) IO/$\overline{\text{M}}$ (IO/Memory) ... Small Scale (μPD70108H)

IO/$\overline{\text{M}}$ signal distinguishes between I/O access and memory access. Low indicates I/O access, and high indicates memory access.

The IO/$\overline{\text{M}}$ output is fixed to high or low during the standby mode.

This pin is a 3-state output. It becomes high impedance during hold acknowledge cycles.

(15) $\overline{\text{IO}}$/M (IO/Memory) ... Small Scale (μPD70116H)

The signal for distinguishing the I/O access from the memory access is output. I/O is displayed at low; memory is displayed at high.

In standby mode, this output is fixed either to a high or low level.

This pin is for tri-state output. During hold acknowledge, it becomes high-impedance.

(16) $\overline{\text{WR}}$ (Write Strobe) ... Small Scale

$\overline{\text{WR}}$ signal is output, when writing to I/O or memory. Whether write operation is performed to the I/O or memory is determined by the IO/$\overline{\text{M}}$ signal (μPD70108H) or $\overline{\text{IO}}$/M signal (μPD70116H).

The $\overline{\text{WR}}$ output is fixed to high, during the standby mode.

This pin is a 3-state output, and becomes high impedance during hold acknowledge cycles.

(17) HLDAK (Hold Acknowledge) ... Small Scale

HLDAK is the hold acknowledge signal, which indicates that the CPU has accepted the hold request (HLDRQ).

The address bus, address/data bus, and 3-state output control bus become high impedance when this pin is active.

(18) HLDRQ (Hold Request) ... Small Scale

HLDRQ signal is used by an external device to request the CPU for the release of the address bus, address/data bus, and the control bus.

The setup time for this signal must be satisfied; otherwise, no correct operation can be guaranteed. Therefore, external synchronization is necessary.

(19) $\overline{RD}$ (Read Strobe) ... Small/Large Scale

$\overline{RD}$ signal is output when reading from I/O or memory. Whether read operation is performed with the I/O or memory is determined by the IO/$\overline{M}$ signal (μPD70108H) or $\overline{IO}$/M signal (μPD70116H).

$\overline{RD}$ signal is provided basically for the small scale mode. However, this signal is output in the same timing in the large scale mode.

The $\overline{RD}$ output is fixed to high, during the standby mode.

This pin is a 3-state output, and becomes high impedance during hold acknowledge cycles.

(20) S/$\overline{LG}$ (Small/Large) ... Small/Large Scale

S/$\overline{LG}$ signal is used to determine the CPU operation mode. S/$\overline{LG}$ signal must be fixed to high or low. When set to high, the CPU operates in the small scale mode. When set to low, the CPU operates in the large scale mode.

Functions of pins listed below change, depending on the mode selected. These individual pins are assigned with different names as follows:

Pin No.			Pin Name	
DIP	QFP	QFJ	In small-scale mode (S/$\overline{LG}$=High level)	In large-scale mode (S/$\overline{LG}$=Low level)
24	38	27	$\overline{INTAK}$	QS1
25	39	28	ASTB	QS0
26	41	29	$\overline{BUFEN}$	BS0
27	42	30	BUF$\overline{R}$/W	BS1
28	43	31	IO/$\overline{M}$ (μPD70108H)	BS2
			$\overline{IO}$/M (μPD70116H)	
29	44	32	$\overline{WR}$	$\overline{BUSLOCK}$
30	45	33	HLDAK	$\overline{RQ}$/AK1
31	47	34	HLDRQ	$\overline{RQ}$/AK0
34	50	37	LBS0 (μPD70108H)	(Outputs high level)

(21) LBS0 (Latched Bus Status 0) ... Small Scale (μPD70108H)

This signal is used in conjunction with the IO/$\overline{M}$ and BUF$\overline{R}$/W signal to externally indicate the type of the current bus cycle.

IO/$\overline{M}$	BUF$\overline{R}$/W	LBS0	Type of Bus Cycle
0	0	0	Program fetch
0	0	1	Memory read
0	1	0	Memory write
0	1	1	Passive state
1	0	0	Interrupt acknowledge
1	0	1	I/O read
1	1	0	I/O write
1	1	1	Halt

Remark Outputs high level in large-scale mode.

(22) $\overline{UBE}$ (Upper Byte Enable) ... Small/Large Scale (μPD70116H)

This pin outputs the signal which indicates that the upper 8 bits (AD8 to AD15) of AD0 to AD15 are used in bus cycles T2 to T4.

The $\overline{UBE}$ signal becomes active (low level) during bus cycles T1 to T4. The bus cycles in which this signal becomes active include the following:

- Bus cycle by the byte access to an odd-number address
- Bus cycle by the first byte access to an odd-number address for the purpose of word data
- Bus cycle by the access to an even-number address for the purpose of word data

Each cycle can be identified based on a combination of the address information and the $\overline{UBE}$ signal which are output during bus cycle T1 by the AD0 pin.

Operand		$\overline{UBE}$	AD0	Bus Cycle Count
Even-number Address Word		0	0	1
Odd-number Address Word	First bus cycle	0	1	2
	Second bus cycle	1	0	
Even-number Address Byte		1	0	1
Odd-number Address Byte		0	1	1

The $\overline{UBE}$ signal continues to be at a low level during interrupt acknowledge (word access of an even-number address is required due to vector reading).

This pin is for tri-stated output. During hold acknowledge. It becomes high-impedance. In standby mode, it is fixed to a high level.

(23) A16/PS0-A19/PS3 (Address Bus/Processor Status) ... Small/Large Scale

A16/PS0-A19/PS3 serve as bits 16 to 19 of the address bus. They also serve as the processor status signal. These contents are output in multiplexed mode.

The upper 4 bits in the 20-bit memory address are output. During I/O access, all bits output 0.

The processor status signal is output for both memory and I/O accesses. PS3 is always set to 0 in the native mode, and 1 in the emulation mode. PS2 outputs the contents of the interrupt enable flag (IE). PS1 and PS0 indicate which segment is currently used.

A17/PS1	A16/PS0	Segment
0	0	Data segment 1
0	1	Stack segment
1	0	Program segment
1	1	Data segment 0

These outputs are fixed to high during the standby mode.

The A16/PS0-A19/PS3 pins are 3-state outputs, and become high impedance during hold acknowledge cycles.

(24) QS0, QS1 (Queue Status) ... Large Scale

QS0 and QS1 inform the external device (floating-point operation coprocessor) of the internal instruction queue status for the CPU.

QS1	QS0	Instruction Queue Status
0	0	No operation (no change in queue status)
0	1	First byte of instruction
1	0	Empty
1	1	After first byte of instruction

"Instruction queue status" indicates the status, when the EXU accesses the instruction queue. The contents output to QS0 and QS1 are effective only in the first clock period, immediately after the instruction queue is accessed.

This status signal is provided for the floating-point operation coprocessor to monitor the CPU's program execution status, so that the coprocessor can perform processing in synchronization with the CPU when the control is transferred (by FPO: Floating-Point Operation instruction).

These outputs are fixed to low in the standby mode.

(25) BS0-BS2 (Bus Status) ... Large Scale

BS0-BS2 are encoded to indicate the current bus cycle category to the external bus controller.

The external bus controller decodes these signals and generates a control signal for accessing the memory or I/O.

BS2	BS1	BS0	Type of Bus Cycle
0	0	0	Interrupt acknowledge
0	0	1	I/O read
0	1	0	I/O write
0	1	1	HALT
1	0	0	Program fetch
1	0	1	Memory read
1	1	0	Memory write
1	1	1	Passive state

These outputs are fixed to high in the standby mode.

These pins are 3-state outputs. They become high impedance during hold acknowledge cycles.

These signals are high during the period from the rising edge of the clock immediately after RESET becomes active to the next rising edge of the clock. After the one clock cycle, these signals become high impedance.

(26) $\overline{\text{BUSLOCK}}$ (Bus Lock) ... Large Scale

$\overline{\text{BUSLOCK}}$ is used to request other master CPUs in a multi-processor system not to use the system bus, when executing one instruction which follows the $\overline{\text{BUSLOCK}}$ prefixed instruction.

This output is fixed to high during the standby mode. (However, if the $\overline{\text{BUSLOCK}}$ instruction is placed before the HALT instruction, this signal is fixed to low during the standby mode.)

This pin is a 3-state output, and becomes high impedance during hold acknowledge cycles.

(27) $\overline{\text{RQ}}/\overline{\text{AK1}}$, $\overline{\text{RQ}}/\overline{\text{AK0}}$ (Hold Request/Acknowledge) ... Large Scale

$\overline{\text{RQ}}/\overline{\text{AK1}}$ and $\overline{\text{RQ}}/\overline{\text{AK0}}$ serve as the bus hold request input pins ($\overline{\text{RQ}}$) and the bus hold acknowledge signal output pins ($\overline{\text{AK}}$). The priority is as indicated below:

$\overline{\text{RQ}}/\overline{\text{AK1}} < \overline{\text{RQ}}/\overline{\text{AK0}}$

These pins are 3-state inputs/outputs, and are provided with internal pull-up resistors. These pins are set to inactive state (high level) when open (float).

When used for bus hold request input ($\overline{\text{RQ}}$), the setup/hold time for this signal must be satisfied; otherwise, no correct operation can be guaranteed. Therefore, external synchronization is necessary.

(28) V_{DD} (Power Supply) ... Small/Large Scale

Positive voltage power supply pins.

(29) GND (Ground) ... Small/Large Scale

GND.

(30) IC (Internally Connected)

IC must be set to GND potential level.

2. REGISTER CONFIGURATION

2.1 PFP (PREFETCH POINTER)

The prefetch pointer is a 16-bit binary counter, which retains the program memory address offset information for the instruction to be prefetched to the instruction queue by the BCU.

The PFP is incremented each time an instruction byte is prefetched from the program memory by the BCU. When branch, call, return, or break instruction is executed, a new location is loaded into the PFP. In this case, the PFP contents become the same as those for the PC (Program Counter).

The PFP is always used, together with PS (Program Segment) register.

2.2 Q0-Q3, Q0-Q5 (PREFETCH QUEUES)

The μPD70108H contains a 4-byte instruction queue (FIFO), which can store up to 4 bytes of instruction code prefetched by the BCU. The μPD70116H is equipped with a 6-byte instruction queue (FIFO) and thus capable of storing up to 6 bytes of instruction codes prefetched by BCU.

The instructions stored in the queue are executed by the EXU.

The queue contents are cleared, when a branch, call, return, or break instruction is executed or external interrupt is processed, and an instruction at a new location will be prefetched.

Normally, the μPD70108H performs prefetching if there is an unused space of at least 1 byte available in the queue; and the μPD70116H does so if at least 1 word (2 bytes) is available.

When successively executing a number of instructions, if the average instruction execution time is faster than the number of clock periods necessary to prefetch each instruction code, when an instruction execution is completed, the next instruction code to be executed by the EXU is already provided in the queue. Therefore, the time necessary to fetch an instruction from the external memory can be excluded from the instruction execution time. As a result, the processing speed can be improved, compared to a CPU which fetches and executes instructions one at a time.

The queue effectiveness decreases as the number of instructions, which clear the queue, increases or the number of instructions, having shorter instruction execution time, increases.

2.3 DP (DATA POINTER)

The data pointer is a 16-bit register, which specifies the variable read/write address.

The effective address, generated by the EAG, and the contents of the register, including the memory address offset, are transferred to the data pointer.

2.4 TEMP (TEMPORARY COMMUNICATION REGISTER)

TEMP is a 16-bit temporary register used for communication between the external data bus and the EXU.

The TEMP can be accessed in byte units. Therefore, the upper byte and the lower byte can be independently read/written.

Basically, the EXU completes write operation when data is transferred to the TEMP; it completes read operation when confirming that data is transferred from an external data bus to TEMP.

2.5 SEGMENT REGISTERS (PS, SS, DS0, DS1)

In the μPD70108H and μPD70116H, memory addresses are divided into logical segments, which are 64K bytes each. The start address for each segment register is specified by the corresponding segment register. The offset from the start address is specified by a different register or by the effective address.

The μPD70108H contains the following four kinds of segment registers.

Segment Register	Default Offset
PS (Program Segment)	PFP
SS (Stack Segment)	SP, Effective address
DS0 (Data Segment 0)	IX, Effective address
DS1 (Data Segment 1)	IY

The PS and the PFP (Prefetch Pointer), and the DS1 and the IY registers are always paired.

The SS is normally paired with the SP. However, when the BP register is selected as the base register, effective address is used as the offset.

The DS0 is used together with the IX register for block transfer processing. However, for other general processing, the effective address is used as the offset.

In addressing using the BP register as the base register and the SS register as the segment register, any one of three other registers can be selected as the segment selection, using the the segment override prefix instruction (PS:, DS0:, DS1:). However, eight or more prefix instructions cannot be attached to other than prefix instructions.

2.6 ADM (ADDRESS MODIFIER)

ADM (Address Modifier) is used for generating a physical address (addition of segment register and PFP or DP), and increments the PFP (Prefetch Pointer).

2.7 GENERAL-PURPOSE REGISTERS (AW, BW, CW, DW)

The μPD70108H contains four 16-bit registers. Each of these 16-bit registers can be used as a 16-bit register. In addition, each of these 16-bit registers can be divided into upper and lower 8 bits, so that each can be accessed as an 8-bit register (AH, AL, BH, BL, CH, CL, DH, DL).

Therefore, these registers can be used as 8-bit or 16-bit registers for various instructions, such as transfer, arithmetic operation, logical operation instruction, etc.

These registers are used as default registers for certain instruction processings as follows:

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

2.8 POINTERS (SP, BP) AND INDEX REGISTER (IX, IY)

SP and BP, and IX and IY are used as the base pointers or index registers, when accessing the memory in the based addressing mode, indexed addressing mode, based indexed addressing mode, etc.

In the same way as for general-purpose registers, these pointers and index registers are used for transfer, arithmetic, or logical operation instructions. However, in this case, these pointers and index registers cannot be used as 8-bit registers.

These registers are used as default registers for certain instruction processing, as follows:

SP : Stack manipulation
IX : Block transfer (source side), BCD string operation
IY : Block transfer (destination side), BCD string operation

2.9 TA/TB (TEMPORARY REGISTER/SHIFTER A/B)

TA/TB is a 16-bit temporary register/shifter used for multiplication/division, and shift/rotate (including BCD rotate) instructions.

When executing a multiplication/division instruction, TA and TB are paired to form a 32-bit temporary register/shifter. When executing a shift/rotate instruction, only the TB serves as the 16-bit temporary register/shifter.

The upper and lower bytes individually for TA and TB can be independently read/written through the internal bus.

TA/TB becomes the ALU input.

2.10 TC (TEMPORARY REGISTER C)

TC is a 16-bit temporary register, used for multiplication, division, and other internal processings.

The TC becomes the ALU input.

2.11 ALU (ARITHMETIC & LOGIC UNIT)

ALU (Arithmetic Logic Unit) consists of the full adder and the logic operation circuit, and performs arithmetic operations (addition, subtraction, increment, decrement, and complement operations) and logical operations (test, AND, OR, XOR, and test, set, clear, and invert in bit units).

2.12 PSW (PROGRAM STATUS WORD)

The program status word consists of six status flags and four control flags.

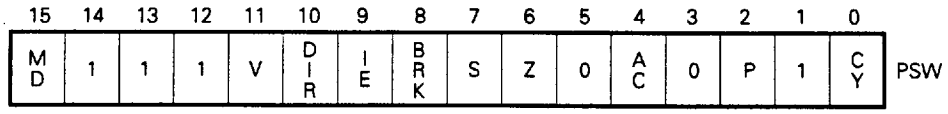
Status flags

- V (Overflow)
- S (Sign)
- Z (Zero)
- AC (Auxiliary Carry)
- P (Parity)
- CY (Carry)

Control flags

- MD (Mode)
- DIR (Direction)
- IE (Interrupt Enable)
- BRK (Break)

These flags are manipulated in the stack in the following word image:



Status flags are automatically set or reset, according to instruction execution.results (data value).

The CY flag can be directly set, reset, or inverted by an instruction.

The control flags are set or reset by an instruction, in order to control the CPU operation.

The MD flag can be modified only from the BRKEM instruction execution to the RETEM instruction. In other locations, the MD flag cannot be restored by executing the RETI or POP PSW instruction.

2.13 LC (LOOP COUNTER)

LC (Loop Counter) is a 16-bit register, which counts the number of loops for primitive block transfer, input/output instructions (MOVBK, OUTM, etc.) controlled by repeat prefix instructions (REP, REPC, etc.), or the number of shifts for multiple bit shift/rotate instructions.

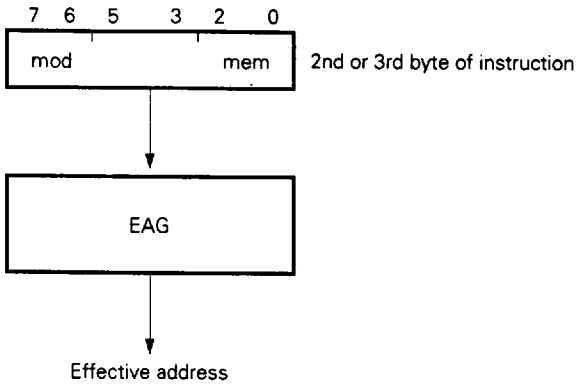
2.14 PC (PROGRAM COUNTER)

PC (Program Counter) is a 16-bit binary counter, which retains the offset information for the program memory address in the instruction to be executed next by the EXU.

The PC is incremented each time the decoder fetches an instruction byte from the instruction queue. When a branch, call, return, or break instruction is executed, a new address location is loaded into the PC, in this case, the PC contents become the same as those for the PFP (Prefetch Pointer).

2.15 EAG (EFFECTIVE ADDRESS GENERATOR)

EAG (Effective Address Generator) logic computes an effective address necessary for accessing the memory at high speed. An address computation is completed in two clock period in any addressing mode.



The byte (2nd or 3rd byte) in which the instruction operand is specified is clocked in. If memory accessing is necessary, the EAG generates a control signal necessary for manipulating the ALU and related register, computes the effective address, and transfers it to the data pointer (DP).

In addition, the EAG requests a bus cycle (memor read) to the BCU as necessary.

2.16 INSTRUCTION DECODER

The instruction decoder classifies the first byte in the instruction code into a group by function, and retains it during macro execution.

2.17 MICROADDRESS REGISTER

The microaddress register specifies the microinstruction ROM address to be executed next.

The first byte for the instruction stored in the queue is clocked into this register as the start address to indicate the specified microinstruction sequence start address, when starting the microinstruction execution.

2.18 MICROINSTRUCTION ROM

The microinstruction ROM contains 1024 words of 29-bit wide microinstructions.

2.19 MICROINSTRUCTION SEQUENCE CIRCUIT

This circuit manages microaddress register control, microinstruction ROM output control, and EXU and BCU synchronization.

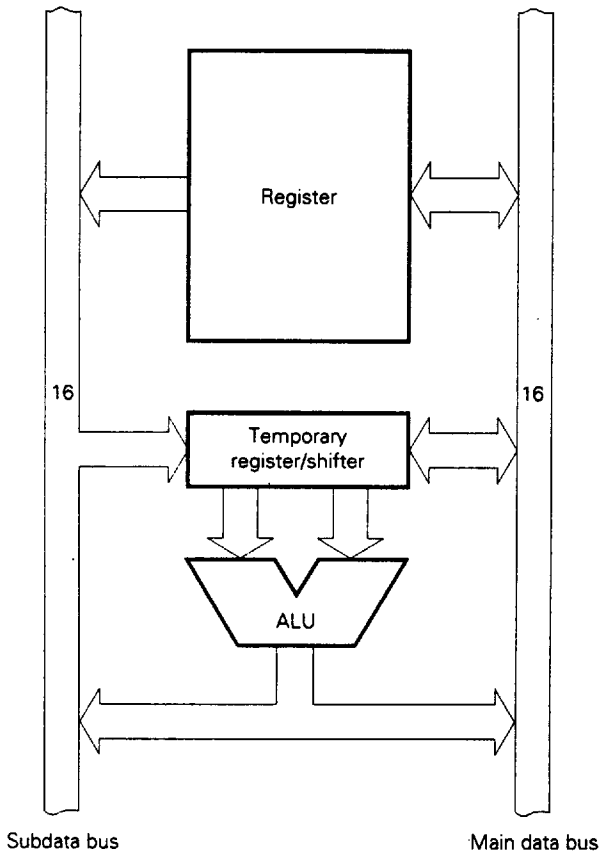
3. INCREASING INSTRUCTION EXECUTION SPEED

The μPD70108H and μPD70116H are provided with the following hardware functions in order to reduce the instruction execution time:

- EXU internal dual data bus
- Effective address generator
- 16/32-bit temporary register/shifter (TA, TB)
- 16-bit loop counter (LC)
- PC (Program Counter) and PFP (Prefetch Pointer)

3.1 DUAL DATA BUS METHOD

In order to reduce the number of processing steps necessary for instruction execution, a dual data bus concept, with the main data bus (16 bits) and the sub data bus (16 bits), is employed. With this concept, the processing time is reduced approximately 30%, compared to the processing time for a single bus system in implementing addition, subtraction, logic operation, and compare instructions.



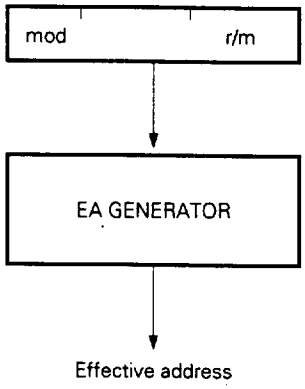
Example: ADD AW, BW; AW ← AW+BW

		Single bus	Dual bus
Step	1	ALU ← AW	ALU ← AW, BW
	2	ALU ← BW	AW ← ALU
	3	AW ← ALU	

3.2 EFFECTIVE ADDRESS GENERATOR

The effective address generator computes at a high speed an effective address necessary for accessing the memory.

In the microprogram method, it took 5 to 12 clock periods to compute an effective address. However, with this sole use hardware, an effective address can be computed in two clock period for any addressing mode.



3.3 16/32-BIT TEMPORARY REGISTER/SHIFTER (TA, TB)

The temporary register/shifter (TA, TB) are provided for multiplication, division, shift, and rotate instructions. With this circuit, especially, multiplication and division instruction execution speed is increased to approximately 4 time faster than a method using the microprogram.

TA+TB: 32-bit temporary register/shifter
- For multiplication/division instructions

TB: 16-bit temporary register/shifter
- For shift/rotate instructions

3.4 LOOP COUNTER (LC)

The loop counter (LC) counts the number of loops for primitive block transfer, and input/output instructions controlled by the repeat prefix instruction, or counts the number of shifts for the multiple-bit shift/rotate instructions.

For example, register multiple-bit rotation will be performed as shown below, and the processing speed is increased approximately 2 time faster than that of microprogram method.

RORC AW, CL; CL = 5

Microprogram method	LC method
8 + 4 x 5 = 28 clocks	7 + 5 = 12 clocks

3.5 PC AND PFP

With the prefetch pointer (PFP), which addresses the program memory when prefetching an instruction, and the program counter (PC), which addresses the program memory for the current instruction execution, provided by hardware, the instruction execution time is reduced by several clock periods for branch, call, return, and break instructions, compared to that for the PFP only.

4. UNIQUE μPD70108H INSTRUCTIONS

4.1 VARIABLE BIT FIELD MANIPULATION INSTRUCTIONS

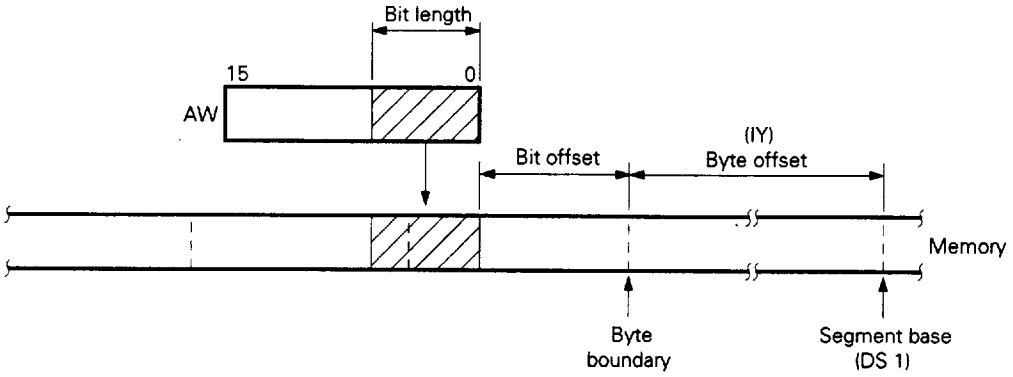
The INS (Insert Bit Field) and EXT (Extract Bit Field) instructions are provided as variable bit field manipulation instructions. These instructions are very effective for computer graphics and high level language. For example, these instructions are effective for Pascal packed array and record type data structure.

(1) INS reg8, reg8'/INS reg8, imm4

Of the 16 bits of data contained in the AW register, the data for the lower bits, specified by the second operand, is transferred to the memory area determined by the byte offset addressed by the DS1 segment register and IY indexed register plus the bit offset specified by the value (0-15) for the first operand.

After the transfer, the IY register and the register, specified by the first operand, are automatically updated to indicate the next bit field.

Only 0-15 (0 specifies 1-bit length, 15 specifies 16-bit length) are effective as the value for the second operand.



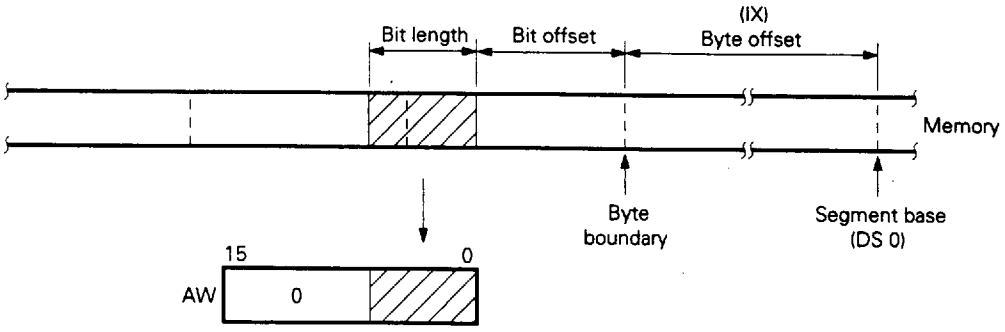
Bit field data can cross memory byte boundaries.

(2) EXT reg8, reg8'/EXT reg8, imm4

Data for the bit field, whose length is specified by the second operand, is loaded from the memory area determined by the byte offset addressed by the DS0 segment register and IX indexed register, plus the bit offset specified by the value (0-15) for the first operand to the AW register.

After the transfer, the IX register and the register specified by the first operand are automatically updated to indicate the next bit field.

Only 0–15 (0 specifies 1 bit length, 15 specifies 16 bit length) are effective as the value for the second operand.



Bit-field data can cross memory byte boundaries.

4.2 PACKED BCD OPERATION INSTRUCTIONS

The ADD4S, SUB4S, and CMP4S instructions process packed BCD as strings. The ROR4 and ROL4 instructions process packed BCD as byte or word format operands.

(1) ADD4S

This instruction adds the packed BCD string, addressed by the IX index register, to the packed BCD string, addressed by the IY index register, and stores the result in the string, addressed by the IY register. The length of the string (number of BCD digits) is specified by the CL register, and the operation result will affect the zero (Z) and carry (CY) flags.

BCD string (IY,CL) ← BCD string (IY,CL) + BCD string (IX,CL)

(2) SUB4S

This instruction subtracts the packed BCD string, addressed by the IX index register, from the packed BCD string, addressed by the IY index register, and stores the result in the string, addressed by the IY register. The length of the string (number of BCD digits) is specified by the CL register, and the operation result will affect the zero (Z) and carry (CY) flags.

BCD string (IY,CL) ← BCD string (IY,CL) – BCD string (IX,CL)

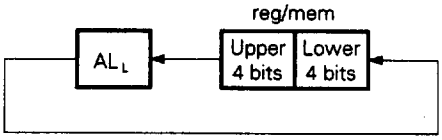
(3) CMP4S

This instruction performs the same operation as SUB4S, except that the result is not stored and only the zero flag (Z) and carry flag (CY) are affected.

BCD string (IY,CL) – BCD string (IX,CL)

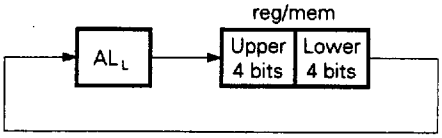
(4) ROL4

This instruction treats the byte data for the register or memory operand specified by the instruction as BCD data and uses the lower 4 bits of the AL register (AL_L) to rotate that data one BCD digit to the left.



(5) ROR4

This instruction treats the byte data for the register or memory operand specified by the instruction as BCD data and uses the lower 4 bits of the AL register (AL_L) to rotate that data one BCD digit to the right.



4.3 STACK MANIPULATION INSTRUCTIONS

(1) PREPARE imm16,imm8

This instruction is used to generate a "stack frame" necessary for a block structure high level language (such as Pascal and Ada). The stack frame contains the pointers to point frames of variables that can be referenced from the procedure, and local variable area.

The following explanation uses the following program example written in a Pascal style language.

```
program EXAMPLE;
    procedure P;
     var a,b,c;
     procedure Q;
       var d,e;
       procedure R;
         var f,g;
         begin
           d:=a+f+g;
         end;
       begin
         R;
         b:=d+e;
       end;
     begin
       a:=b+c;
       Q;
     end;
    (*main program*)
     begin
       P;
     end.
```

Remark All variables are word variables.

In this program, procedure blocks are nested in three levels. Procedure P defines variables a, b, and c, procedure Q defines variables d and e, and procedure R defines variables f and g. Therefore, a, b, and c are referenced from procedure Q, and d and e in addition to a, b, and c, are referenced from procedure R as global variables.

The PREPARE instruction copies the frame pointer, in order to assure local variable area and enable referencing to global variables. The first operand specifies the size (bytes) of the area assured for the local variables. The second operand indicates the depth of the procedure block (this depth is referred to as lexical level).

The base address for the frame generated by the PREPARE instruction is set into the base pointer BP.

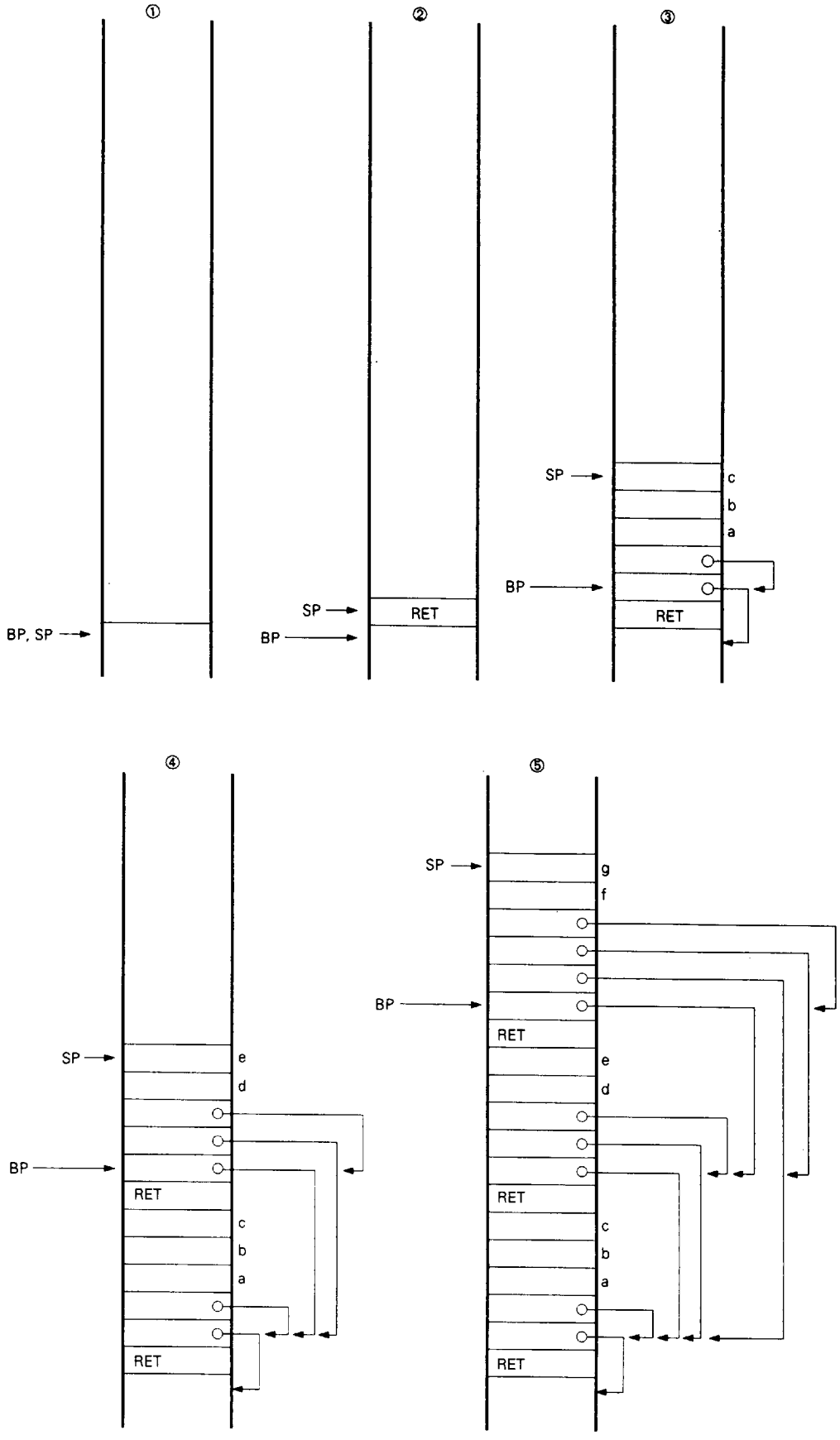
When the above EXAMPLE program is compiled, the assembler program, shown on the next page will be created (the DISPOSE instruction used in the assembler program returns the stack pointer SP and the base pointer BP to the condition which existed before the PREPARE instruction was executed. Refer to (2)).

```
:ASSEMBLER PROGRAM

START :  MOV      SP, SPTOP
         MOV      BP, SP        ; ①
         CALL     P             ; ②
         BR       SYSTEM
P:       PREPARE  6, 1          ; ③
         MOV      AW, [BP] [B+BLEVEL*2]
         ADD      AW, [BP] [C+CLEVEL*2]
         MOV      [BP] [A+ALEVEL*2], AW
         CALL     Q
         DISPOSE
         RET
Q:       PREPARE  4, 2          ; ④
         CALL     R
         MOV      AW, [BP] [D+DLEVEL*2]
         ADD      AW, [BP] [E+ELEVEL*2]
         MOV      IY, [BP] [BLEVEL*2]
         MOV      SS:[IY] [B+BLEVEL*2], AW
         DISPOSE
         RET
R:       PREPARE  4, 3          ; ⑤
         MOV      AW, [BP] [F+FLEVEL*2]
         ADD      AW, [BP] [G+GLEVEL*2]
         MOV      IY, [BP] [ALEVEL*2]
         ADD      AW,.SS:[IY] [A+ALEVEL*2], AW
         MOV      IY, [BP] [DLEVEL*2]
         MOV      SS:[IY] [D+DLEVEL*2], AW
         DISPOSE
         RET

;  A = -2   ALEVEL = -1
;  B = -4   BLEVEL = -1
;  C = -6   CLEVEL = -1
;  D = -2   DELVEL = -2
;  E = -4   ELEVEL = -2
;  F = -2   FLEVEL = -3
;  G = -4   GLEVEL = -3
```

The following shows the process in which stack frames are generated according to the program execution progress. The numbers correspond to the numbers written in the comment fields.


①
②
③
④
⑤
BP, SP
SP
BP
RET
c
b
a
e
d
g
f

The PREPARE instruction first stores the BP into the stack. This is to restore the BP for the procedure which made the call, when the procedure is completed. The frame pointers (stored BP) in the range that can be referenced from the called procedure are then loaded into the stack. The range that can be referenced means a value which is the lexical level of the procedure minus 1.

If the lexical level is 1 or greater, the own frame pointer is also loaded into the stack. This is to copy the frame pointer for the procedure, which made the call, when copying the frame pointer in the procedure called from this procedure.

The value for the new frame pointer is then set into the BP, and the local variable area to be used by the procedure is assured in the stack. That is, the SP is decremented by the number of local variables.

```
    display = 2nd operand
    dynamics = 1st operand

SP = SP-2;
(SP) = BP;
temp = SP;
if display > 0 then begin
      repeat display-1 times
        begin
          SP = SP-2;
          BP = BP-2;
          (SP) = (BP);
        end
      SP = SP-2;
      (SP) = temp;
      end
BP = temp;
SP = SP-dynamics
```

Data Access Method

(a) Accessing local variable

A local variable is allocated in the procedure's own frame. Therefore, the effective address for the local variable EA.L can be computed as follows:

EA.L = SS:(BP+offset)

Where, offset value is the result of addition of the frame size (base value for the frame that can be referenced) loaded in the flame, with the offset value from the base value of the local variable area to the variable.

(b) Accessing global variable

A global variable accesses the base pointer to be referenced from the old base pointers loaded in the stack frame, then add the offset value to the variable to be referenced to that value. This value is the address at which the global variable is located. Therefore, the the effective address for the global variable EA.G can be computed as follows:

EA.G = SS:((SS:(BP+offset1))+offset2)

Where, offset1 is the offset value from the base value (BP value) for the current frame to the address, in which the base address for frame containing the global variable is stored.

Offset2 is the offset value from the base value for the frame containing the variable to be referenced to that variable.

(2) DISPOSE

The DISPOSE instruction releases one frame from stack frames generated by the PREPARE instruction. The point value, indicating the previous frame, is loaded into the BP, and the point value, indicating the bottom position of the frame, is loaded into the SP.

SP = BP;

BP = (SP);

SP = SP+2

4.4 CHECK ARRAY BOUNDARY INSTRUCTION

This instruction is used to verify that index values, pointing to the elements of an array data structure, are within the defined range. If the index value is not between these defined ranges when CHKIND is executed, a BRK5 will occur.

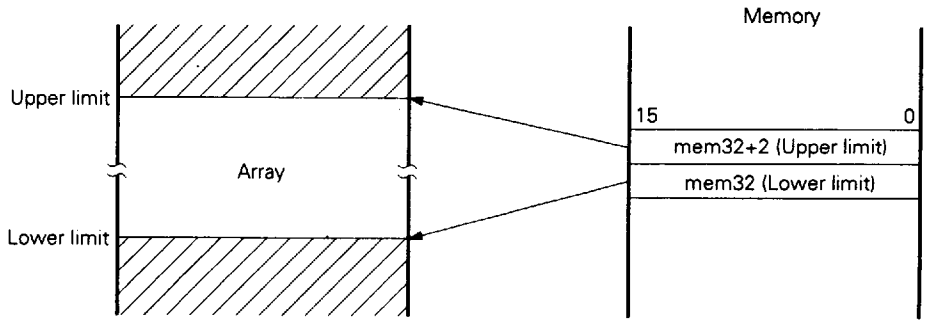
When using the CHKIND instruction, the defined value must be set into 2 words (the 1st word defines the lower limit, the second word specifies the upper limit) in the memory in advance. The index value should be a register (an arbitrary 16-bit register) used by the array manipulation program.

CHKIND, reg 16, mem 32

When (mem32) > reg16 or (mem 32+2) < reg16

TA ← (015H, 014H)
TC ← (017H, 016H)
SP ← SP–2, (SP+1, SP) ← PSW
IE ← 0, BRK ← 0
SP ← SP–2, (SP+1, SP) ← PS
PS ← TC
SP ← SP–2, (SP+1, SP), ← PC
PC ← TA

} = BRK 5



4.5 MODE MANIPULATION INSTRUCTIONS

The μPD70108H and 70116H has two operation modes: native mode (normal operation mode) and emulation mode (emulates μPD8080AF instruction set). Bit 15 is provided as the mode flag to select these modes. Setting MD to 1 specifies the native mode, and 0 specifies the emulation mode.

The MD is directly or indirectly set/reset by the mode manipulation instruction.

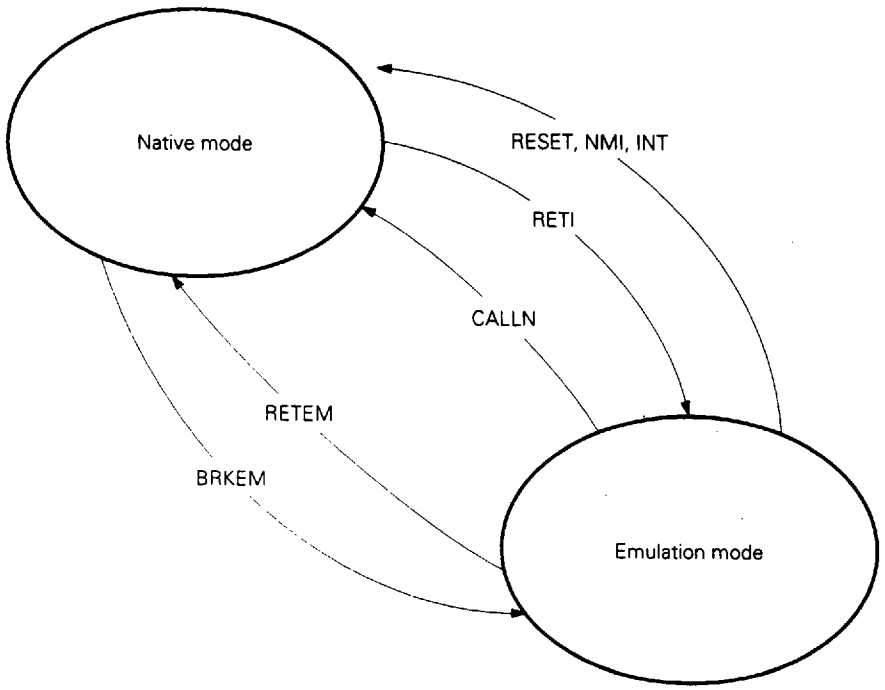
The following instructions change the mode from the native mode to the emulation mode:

BRKEM (Break for Emulation)
RETI (Return from Interrupt)

The following instructions change the mode from the emulation mode to the native mode:

RETEM (Return from Emulation)
CALLN (Call Native routine)

In addition, the mode returns from the emulation mode to the native mode, when a RESET is input or when an external interrupt is input (NMI, INT).



(1) BRKEM imm8

BRKEM is the basic instruction to initiate the emulation mode. This instruction stores the PSW, PS, and PC, resets the MD flag to 0, and loads the interrupt vector specified by the operand to the PS and PC. The interrupt enable flag (IE) and the break flag (BRK) are not affected by this instruction.

When an instruction code for interrupt service routine (for emulation) jumped in this manner is fetched, the CPU interprets and executes the instruction as an instruction of the μPD8080AF.

The CPU treats the emulation mode as an interrupt service.

In the emulation mode, the μPD8080AF registers and flags are substituted for by the following registers and flags:

μPD8080AF	μPD70108H, 70116H
A	AL
B	CH
C	CL
D	DH
E	DL
H	BH
L	BL
SP	BP
PC	PC

μPD8080AF	μPD70108H, 70116H
C	CY
Z	Z
S	S
P	P
AC	AC

For stack manipulation, SP serves as the stack pointer in the native mode. However, in the emulation mode, BP serves as the stack pointer. By employing an independent stack pointer, the stack area is assured independently for both modes, in order to prevent destroying the stack for some other mode by erroneous stack manipulation.

The SP, IX, IY, AH and four segment registers (PS, SS, DS0, DS1) used in the native mode are not affected by the emulation mode.

In the emulation mode, the segment base for the instruction is determined by the PS register (automatically determined by the interrupt vector), and the segment base of data is determined by the DS0 register (the programmer determins before entering the emulation mode).

(2) RETEM (no operand)

When the RETEM instruction is executed in the emulation mode (the instruction is interpreted as the μPD8080AF instruction), the CPU restores the PS, PC, and PSW in the same way as when returning from an interrupt service, then returns to the native mode. In this case, the contents for the native mode (that is 1), stored in the stack by the BRKEM instruction, is restored. This sets the CPU to the native mode.

(3) CALLN imm8

When this instruction is executed in the emulation mode (the instruction is interpreted as the μPD8080AF instruction), the CPU stores the PS, PC, and PSW into the stack (MD=0 is stored here), sets the mode flag (MD) to 1, and loads the interrupt vector specified by the operand to the PS and PC.

The interrupt enable flag (IE) and the break flag (BRK) are not affected by this instruction.

In this manner, the interrupt routine in the native mode can be called from the emulation mode.

Use RETI instruction to return to the emulation mode from this interrupt routine.

(4) RETI (no operand)

The RETI instruction is generally used to return from an interrupt routine initiated by the BRK instruction or external interrupt in the native mode. However, if this instruction is executed at the end of the interrupt routine initiated by the CALLN instruction, PS, PC, and PSW are restored in the same way as normal. However, when the PSW is restored, the value (=0) for the mode flag (MD) for the emulation mode is restored to the MD. For this reason, the CPU is set to the emulation mode. Afterwards, instructions will be interpreted and executed as μPD8080AD instructions.

In the same manner, the RETI instruction is used to return from an interrupt routine for the native mode, initiated by NMI, or INT interrupt request, generated in the emulation mode.

4.6 FLOATING-POINT OPERATION COPROCESSOR CONTROL INSTRUCTIONS

FPO1 fp-op/FPO1 fp-op, mem
FPO2 fp-op/FPO2 fp-op, mem

These instructions are used to control the external floating-point operation coprocessor. When the CPU fetches this instruction, the CPU outputs instructions for coprocessor to perform operations. The CPU only performs supportive processing (effective address computation, physical address generation, and memory read cycle initiation) necessary for the coprocessor to perform operations.

When the coprocessor monitors this instruction, the coprocessor interprets the instruction as its own instruction and executes the instruction. In this case, the coprocessor uses the address information for the memory read cycle initiated by the CPU only, or both the address and read data, depending on the type of instruction involved.

In terms of function, FPO1 and FPO2 are identical, except the type of code is different.

In general, when writing in an assembler language, mnemonics, corresponding to each instruction for the coprocessor, are used rather than FPO1 and FPO2 mnemonics.

When the FPO1 or FPO2 instruction is fetched, the CPU initiates memory read cycle, if the instruction is requesting memory accessing. However, the data read out by this operation is to be used by the coprocessor, so that the CPU will not clock in this data.

When the coprocessor needs memory write cycle, the CPU initiates memory write cycle for the coprocessor. The data, read out as a result, will be ignored by the floating-point operation coprocessor. The floating-point operation coprocessor only latches the memory address information, and use it to execute memory write cycle.

5. INTERRUPT OPERATION

The interrupts supported by the μPD70108H, 70116H can be divided into two types; interrupts generated by external interrupt requests and traps generated by software processing. They are:

(1) External interrupts

(a) NMI input (nonmaskable)

(b) INT input (maskable)

(2) Software traps

(a) By instruction execution result
- Divide error during DIV or DIVU instruction
- Array bound error during CHKIND

(b) Conditional break instruction
- When V = 1 when BRKV instruction is executed

(c) Unconditional break instruction
- 1-byte break instruction BRK 3
- 2-byte break instruction BRK imm8

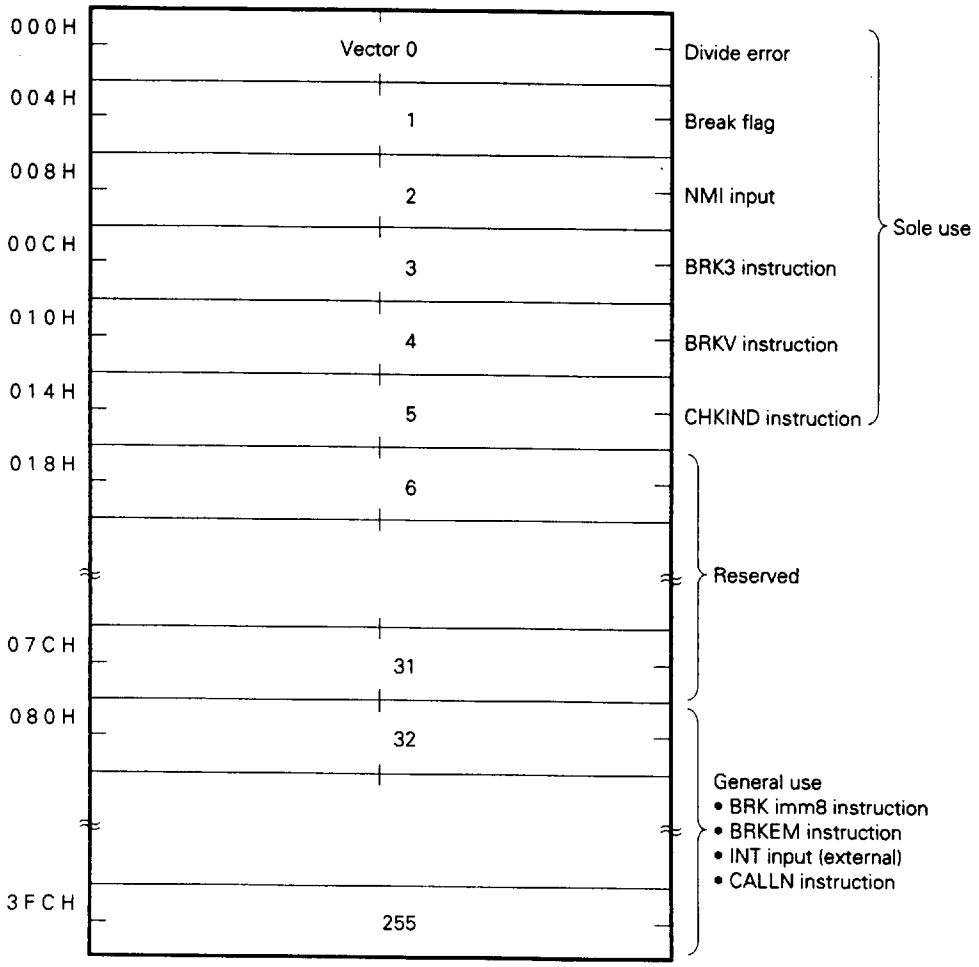
(d) Flag processing (single step)
- Sets the BRK flag by stack manipulation

(e) Emulation related instructions
- BRKEM imm8
- CALLN imm8

For any interrupt, one location of the provided interrupt vector table is automatically selected, or is selected each time by specification, to determine the interrupt routine start address.

Figure 5-1 shows the interrupt vector table. This table is allocated to the 1K-byte area for memory addresses 000H to 3FFH, and can contain interrupt routine start addresses for 256 vectors (4 bytes for each vector)

Fig. 5-1 Interrupt Vector Table



Vectors 0-5 are specified for sole uses. Vectors 6-31 are reserved and cannot be used for general purposes.

Vectors 32-255 can be used for general purposes, and the 2-byte break instruction, BRKEM instruction, INT input, and CALLN instruction (in the emulation mode) can be used.

Each interrupt vector consists of 4 bytes, and the lower 2 bytes are loaded into the PC as offset, and the upper 2 bytes are loaded into the PS as base.

Example: Vector 0

000H	001H
002H	003H

PS ← (003H, 002H)
PC ← (001H, 000H)

The programmer must initialize the contents of each vector in the beginning of a program according to this format.

The following is the basic step used to jump to an interrupt service routine.

TA ← Lower bytes in vector (offset)
TC ← Upper bytes in vector (base)
SP ← SP-2, (SP+1,SP) ← PSW
IE ← 0, BRK ← 0, MD ← 0
SP ← SP-2,(SP+1,SP) ← PS
PS ← TC
SP ← SP-2,(SP+1,SP) ← PC
PC ← TA

6. STANDBY FUNCTION

The μPD70108H and 70116H offer two standby modes to reduce power consumption. The standby mode is entered after the HALT instruction in the native mode or the HLT instruction in the emulation mode.

In the standby mode, the internal clock is supplied only to those circuits related to functions required to exit this mode and bus hold control functions. As a result, power consumption is reduced by about ten times the normal operation.

Standby mode is canceled by the RESET input or external interrupt input (NMI, INT).

The BUS HOLD function is still valid even during standby mode. If the request for BUS HOLD is withdrawn, the μPD70108H or 70116H is placed back in standby mode.

7. I/O ADDRESS RESERVE

The upper 256 bytes (FF00H-FFFFH) in I/O address space are reserved for future use. Therefore, use of these addresses is not recommended.

8. INSTRUCTION SET

Table 8-1 Operand Types

Symbol	Meaning
reg	8/16-bit general-purpose register (Destination register, when instruction uses two 8/16-bit registers)
reg'	Source register, when instruction uses two 8/16-bit registers
reg 8	8-bit general-purpose register (Destination register, when instruction uses two 8-bit registers)
reg 8'	Source register, when instruction uses two 8-bit registers
reg 16	16-bit general-purpose register (Destination register, when instruction uses two 16-bit registers)
reg 16'	Source register, when instruction uses two 16-bit registers
dmem	8/16-bit memory location
mem	8/16-bit memory location
mem 8	8-bit memory location
mem 16	16-bit memory location
mem 32	32-bit memory location
imm	Constant (0-FFFFH)
imm 3	Constant (0-7)
imm 4	Constant (0-FH)
imm 8	Constant (0-FFH)
imm 16	Constant (0-FFFFH)
acc	Register AW or AL
sreg	Segment register
src-table	256-byte conversion table
src-block	Block name addressed by register IX
dst-block	Block name addressed by register IY
near-proc	Procedure within the current segment
far-proc	Procedure within a different program segment
near-label	Label within the current segment
short-label	Label between -128 and +127 bytes from the end of the current instruction
far-label	Label within a different program segment
memptr 16	Word containing the destination offset address in the current segment
memptr 32	Double word containing the destination offset address and segment base address in another segment
regptr 16	16-bit general-purpose register containing the destination offset address in another segment
pop-value	Number of bytes to discard from the stack (0-64K, normally an even number)
fp-op	Immediate value which determins the operation code of an external floating-point coprocessor
R	Register set

Table 8-2 Operation Codes

Symbol	Meaning
W	Byte/word specification bit (0: byte, 1: word). However, when S = 1, byte date with extended sign is used as an 16-bit operand even if W = 1.
reg	Register field (000-111)
reg'	Register field (000-111) (Source register, when two registers are used)
mem	Memory field (000-111)
mod	Mode field (00-10)
S	Sign extension specification bit (0: sign not extended, 1: sign extended)
X,XXX,YYY,ZZZ	Data to identify the instruction code of the external floating-point coprocessor

Table 8-3 Operand Types

Symbol	Meaning
AW	Accumulator (16 bits)
AH	Accumulator (high byte)
AL	Accumulator (low byte)
BW	Register BW (16 bits)
CW	Register CW (16 bits)
CL	Register CW (low byte)
DW	Register DW (16 bits)
BP	Base pointer (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)
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
BRK	Break flag
MD	Mode flag
(...)	Memory contents indicated by parentheses
disp	Displacement (8/16 bits)
ext-disp8	8-bit displacement sign expanded to 16-bit
temp	Temporary register (8/16/32 bits)
TA	Temporary register A (16 bits)
TB	Temporary register B (16 bits)
TC	Temporary register C (16 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
∧	AND
∨	OR
∀	XOR
xxH	Two-digit hexadecimal value
xxxxH	Four-digit hexadecimal value

Table 8-4 Flag Operation

Symbol	Meaning
(blank)	No change
0	Cleared to 0
1	Set to 1
×	Set or cleared, according to result.
U	Undefined
R	Value saved earlier is restored

Table 8-5 Memory Addressing Mode

mem \ mod	00	01	10
000	BW+IX	BW+IX+disp 8	BW+IX+disp 16
001	BW+IY	BW+IY+disp 8	BW+IY+disp 16
010	BP+IX	BP+IX+disp 8	BP+IX+disp 16
011	BP+IY	BP+IY+disp 8	BP+IY+disp 16
100	IX	IX+disp 8	IX+disp 16
101	IY	IY+disp 8	IY+disp 16
110	Direct address	BP+disp 8	BP+disp 16
111	BW	BW+disp 8	BW+disp 16

Table 8-6 8- and 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 8-7 Segment Register Selection

sreg	
00	DS1
01	PS
10	SS
11	DS0

On the following pages, the instruction set is shown in table form.

The number of clocks shown in the table is the time required for the execution unit to execute an instruction and is under the following conditions:

- Does not include prefetch time and wait time for bus
- Memory access is assumed to have 0 wait cycle. Therefore, the number of clocks for 1 bus cycle is 4.
- I/O access is assumed to have 0 wait cycle.
- The primitive block transfer instruction and primitive I/O instruction include the repeat prefix.

The number of clocks of an instruction that can execute byte and word processing (that has W bit) is indicated as follows:

(1) μPD70108H

Left to / : Value for byte processing (W = 0)
Right to / : Value for word processing (W = 1)

For details of the clock count of the instructions related to block transfer of μPD70108H, refer to Table 8-8.

Table 8-8 Clock Count of Block-Transfer-Related Instructions (μPD70108H)

Instruction	Clock Count	
	Byte operation (W=0)	Word operation (W=1)
MOVBK	11 + 8 × rep (11)	11 + 16 × rep (19)
CMPBK	7 + 14 × rep (13)	7 + 22 × rep (21)
CMPM	7 + 10 × rep (7)	7 + 14 × rep (11)
LDM	7 + 9 × rep (7)	7 + 13 × rep (11)
STM	7 + 4 × rep (7)	7 + 8 × rep (11)
INM	9 + 8 × rep (10)	9 + 16 × rep (18)
OUTM	9 + 8 × rep (10)	9 + 16 × rep (18)

Remark The values in the brackets are applicable if the operation was performed only a single time.

(2) μPD70116H

Left to /: Value for byte operation (W=0), or for word operation (W=1) of an even-number address
Right to /: Value for word operation (W=1) of an odd-number

For details of the clock count of instructions related to block transfer of μPD70116H, refer to Table 8-9.

Table 8-9 Clock Count of Block-Transfer-Related Instructions (μPD70116H) (1/2)

Instruction	Clock Count			
	Byte operation (W=0)	Word operation (W=1)		
		Odd- & odd-number address	Odd-& even-number address	Even- & even-number address
MOVBK	11 + 8 × rep (11)	11 + 16 × rep (19)	11 + 12 × rep (15)	11 + 8 × rep (11)
CMPBK	7 + 14 × rep (13)	7 + 22 × rep (21)	7 + 18 × rep (17)	7 + 14 × rep (13)
INM	9 + 8 × rep (10)	9 + 16 × rep (18)	9 + 12 × rep (14)	9 + 8 × rep (10)
OUTM	9 + 8 × rep (10)	9 + 16 × rep (18)	9 + 12 × rep (14)	9 + 8 × rep (10)

Remark The values in the brackets are applicable if the operation was performed only a signle time.

Table 8-9 Clock Count of Block-Transfer-Related Instructions (μPD70116H) (2/2)

Instruction	Clock Count		
	Byte operation (W=0)	Word operation (W=1)	
		Odd-number address	Even-number address
CMPM	7 + 10 × rep (7)	7 + 14 × rep (11)	7 + 10 × rep (7)
LDM	7 + 9 × rep (7)	7 + 13 × rep (11)	7 + 9 × rep (7)
STM	7 + 4 × rep (7)	7 + 8 × rep (11)	7 + 4 × rep (7)

Remark The values in the brackets are applicable if the operation was performed only a signle time.

Instruction Group	Mnemonic	Operand	Operation Code																Number of Bytes	Number of Clocks		Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0		μPD70108H	μPD70116H		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	2	2	reg←reg'						
		mem, reg	1	0	0	0	1	0	0	W	mod		reg			mem			2-4	9/13	9/13	(mem)←reg						
		reg, mem	1	0	0	0	1	0	1	W	mod		reg			mem			2-4	11/15	11/15	reg←(mem)						
		mem, imm	1	1	0	0	0	1	1	W	mod	0	0	0		mem			3-6	11/15	11/15	(mem)←imm						
		reg, imm	1	0	1	1	W		reg										2-3	4	4	reg←imm						
		acc, dmem	1	0	1	0	0	0	0	W									3	10/14	10/14	When W = 0, AL←(dmem) When W = 1, AH←(dmem+1), AL←(dmem)						
		dmem,acc	1	0	1	0	0	0	1	W									3	9/13	9/13	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	2	2	sreg←reg16 sreg: SS, DS0, DS1						
		sreg,mem16	1	0	0	0	1	1	1	0	mod	0	sreg			mem			2-4	11/15	11/15	sreg←(mem16) sreg: SS, DS0, DS1						
		reg16, sreg	1	0	0	0	1	1	0	0	1	1	0	sreg		reg			2	2	2	reg16←sreg						
		mem16,sreg	1	0	0	0	1	1	0	0	mod	0	sreg			mem			2-4	10/14	10/14	(mem16)←sreg						
		DS0, reg16, mem32	1	1	0	0	0	1	0	1	mod		reg			mem			2-4	26	18/26	reg16←(mem32), DS0←(mem32+2)						
		DS1, reg16, mem32	1	1	0	0	0	1	0	0	mod		reg			mem			2-4	26	18/26	reg16←(mem32), DS1←(mem32+2)						
		AH, PSW	1	0	0	1	1	1	1	1									1	2	2	AH←S, Z, x, AC, x, P, x, CY						
		PSW, AH	1	0	0	1	1	1	1	0									1	3	3	S, Z, x, AC, x, P, x, CY←AH	x	x		x	x	x
	LDEA	reg16, mem16	1	0	0	0	1	1	0	1	mod		reg			mem			2-4	4	4	reg16←mem16						

Instruction Group	Mnemonic	Operand	Operation Code																Number of Bytes	Number of Clocks		Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0		μPD70108H	μPD70116H		AC	CY	V	P	S	Z
Data transfer instruction	TRANS	src-table	1	1	0	1	0	1	1	1									1	9	9	AL←(BW+AL)						
	XCH	reg, reg'	1	0	0	0	0	1	1	W	1	1	reg			reg'			2	3	3	reg↔reg'						
		mem, reg reg, mem	1	0	0	0	0	1	1	W	mod		reg			mem			2-4	16/24	16/24	(mem)↔reg						
		AW, reg16 reg16, AW	1	0	0	1	0	reg											1	3	3	AW↔reg16						
Repeat prefix	REPC		0	1	1	0	0	1	0	1									1	2	2	While CW ≠ 0, executes the primitive block transfer of the successive bytes, and decrements (−1) CW. If any interrupt has been on hold, the interrupt is processed. Exits from loop, when CY ≠ 1.						
	REPNC		0	1	1	0	0	1	0	0									1	2	2	Same as above. Exits from loop, when CY ≠ 0.						
	REP REPE REPZ		1	1	1	1	0	0	1	1									1	2	2	While CW ≠ 0, executes the primitive block transfer of the successive bytes, and decrements (−1) CW. If any interrupt has been on hold, the interrupt is processed. Exits from loop, when the primitive block transfer instruction is CMPBK or CMPM and Z ≠ 1.						
	REPNE REPNZ		1	1	1	1	0	0	1	0									1	2	2	Same as above. Exits from loop, when Z ≠ 0.						

Instruction Group	Mnemonic	Operand	Operation Code																Number of Bytes	Number of Clocks		Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0		μPD70108H	μPD70116H		AC	CY	V	P	S	Z
Primitive block transfer instruction	MOVBK	dst-block, src-block	1	0	1	0	0	1	0	W									1	Refer to Table 8-8.	Refer to Table 8-9.	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	src-block, dst-block	1	0	1	0	0	1	1	W									1	Refer to Table 8-8.	Refer to Table 8-9.	When W = 0, (IX) − (IY) DIR = 0: IX←IX+1, IY←IY+1 DIR = 1: IX←IX−1, IY←IY−1 When W = 1, (IX+1,IX) − (IY+1,IY) DIR = 0: IX←IX+2, IY←IY+2 DIR = 1: IX←IX−2, IY←IY−2	×	×	×	×	×	×
	CMPM	dst-block	1	0	1	0	1	1	1	W									1	Refer to Table 8-8.	Refer to Table 8-9.	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	src-block	1	0	1	0	1	1	0	W									1	Refer to Table 8-8.	Refer to Table 8-9.	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	dst-block	1	0	1	0	1	0	1	W									1	Refer to Table 8-8.	Refer to Table 8-9.	When W = 0, (IY)←AL DIR = 0: IY←IY+1; DIR = 1: IY←IY−1 When W = 1, (IY+1, IY)←AW DIR = 0: IY←IY+2; DIR = 1: IY←IY−2						

Instruction Group	Mnemonic	Operand	Operation Code																Number of Bytes	Number of Clocks		Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0		μPD70108H	μPD70116H		AC	CY	V	P	S	Z
Bit field operation instruction	INS	reg8,reg8'	0	0	0	0	1	1	1	1	0	0	1	1	0	0	0	1	3	35-133	31-117/ 35-133	16 bit field←AW						
			1	1	reg'			reg																				
		reg8,imm4	0	0	0	0	1	1	1	1	0	0	1	1	1	0	0	1	4	35-133	31-117/ 35-133	16-bit field←AW						
			1	1	0	0	0	reg																				
	EXT	reg8,reg8'	0	0	0	0	1	1	1	1	0	0	1	1	0	0	1	1	3	34-59	26-55/ 34-59	AW←16 bit field						
			1	1	reg'			reg																				
		reg8,imm4	0	0	0	0	1	1	1	1	0	0	1	1	1	0	1	1	4	34-59	26-55/ 34-59	AW←16 bit field						
			1	1	0	0	0	reg																				
I/O instruction	IN	acc,imm8	1	1	1	0	0	1	0	W									2	9/13	9/13	When W = 0, AL←(imm8)						
																						When W = 1, AH←(imm8+1), AL←(imm8)						
		acc,DW	1	1	1	0	1	1	0	W									1	8/12	8/12	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	8/12	8/12	When W = 0, (imm8)←AL						
																						When W = 1, (imm8+1)←AH,(imm8)←AL						
		DW,acc	1	1	1	0	1	1	1	W									1	8/12	8/12	When W = 0, (DW)←AL						
																						When W = 1, (DW+1)←AH,(DW)←AL						
Primitive I/O instruction	INM	dst-block, DW	0	1	1	0	1	1	0	W									1	Refer to Table 8-8.	Refer to Table 8-9.	When W = 0, (IY)←(DW)						
																						DIR = 0: IY←IY+1; DIR = 1: IY←IY-1						
																						When W = 1, (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	Refer to Table 8-8.	Refer to Table 8-9.	When W = 0, (DW)←(IX)						
																						DIR = 0: IX←IX+1; DIR = 1: IX←IX-1						
																						When W = 1, (DW+1,DW)←(IX+1,IX)						
																						DIR = 0: IX←IX+2; DIR = 1: IX←IX-2						

Instruction Group	Mnemonic	Operand	Operation Code																Number of Bytes	Number of Clocks		Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0		μPD70108H	μPD70116H		AC	CY	V	P	S	Z
Addition/Subtraction instruction	ADD	reg,reg'	0	0	0	0	0	0	1	W	1	1	reg			reg'			2	2	2	reg←reg+reg'	×	×	×	×	×	×
		mem,reg	0	0	0	0	0	0	0	W	mod		reg			mem			2-4	16/24	16/24	(mem)←(mem)+reg	×	×	×	×	×	×
		reg,mem	0	0	0	0	0	0	1	W	mod		reg			mem			2-4	11/15	11/15	reg←reg+(mem)	×	×	×	×	×	×
		reg,imm	1	0	0	0	0	0	s	W	1	1	0	0	0	reg			3-4	4	4	reg←reg+imm	×	×	×	×	×	×
		mem,imm	1	0	0	0	0	0	s	W	mod		0	0	0	mem			3-6	18/26	18/26	(mem)←(mem)+imm	×	×	×	×	×	×
		acc,imm	0	0	0	0	0	1	0	W									2-3	4	4	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	2	2	reg←reg+reg'+CY	×	×	×	×	×	×
		mem,reg	0	0	0	1	0	0	0	W	mod		reg			mem			2-4	16/24	16/24	(mem)←(mem)+reg+CY	×	×	×	×	×	×
		reg,mem	0	0	0	1	0	0	1	W	mod		reg			mem			2-4	11/15	11/15	reg←reg+(mem)+CY	×	×	×	×	×	×
		reg,imm	1	0	0	0	0	0	s	W	1	1	0	1	0	reg			3-4	4	4	reg←reg+imm+CY	×	×	×	×	×	×
		mem,imm	1	0	0	0	0	0	s	W	mod		0	1	0	mem			3-6	18/26	18/26	(mem)←(mem)+imm+CY	×	×	×	×	×	×
		acc,imm	0	0	0	1	0	1	0	W									2-3	4	4	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	2	2	reg←reg−reg'	×	×	×	×	×	×
		mem,reg	0	0	1	0	1	0	0	W	mod		reg			mem			2-4	16/24	16/24	(mem)←(mem)−reg	×	×	×	×	×	×
		reg,mem	0	0	1	0	1	0	1	W	mod		reg			mem			2-4	11/15	11/15	reg←reg−(mem)	×	×	×	×	×	×
		reg,imm	1	0	0	0	0	0	s	W	1	1	1	0	1	reg			3-4	4	4	reg←reg−imm	×	×	×	×	×	×
		mem,imm	1	0	0	0	0	0	s	W	mod		1	0	1	mem			3-6	18/26	18/26	(mem)←(mem)−imm	×	×	×	×	×	×
		acc,imm	0	0	1	0	1	1	0	W									2-3	4	4	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	2	2	reg←reg−reg'−CY	×	×	×	×	×	×
		mem,reg	0	0	0	1	1	0	0	W	mod		reg			mem			2-4	16/24	16/24	(mem)←(mem)−reg−CY	×	×	×	×	×	×
		reg,mem	0	0	0	1	1	0	1	W	mod		reg			mem			2-4	11/15	11/15	reg←reg−(mem)−CY	×	×	×	×	×	×
		reg,imm	1	0	0	0	0	0	s	W	1	1	0	1	1	reg			3-4	4	4	reg←reg−imm−CY	×	×	×	×	×	×
		mem,imm	1	0	0	0	0	0	s	W	mod		0	1	1	mem			3-6	18/26	18/26	(mem)←(mem)−imm−CY	×	×	×	×	×	×
		acc,imm	0	0	0	1	1	1	0	W									2-3	4	4	When W = 0, AL←AL−imm−CY When W = 1, AW←AW−imm−CY	×	×	×	×	×	×

Instruction Group	Mnemonic	Operand	Operation Code	Number of Bytes	Number of Clocks		Operation	Flags					
			7 6 5 4 3 2 1 0 / 7 6 5 4 3 2 1 0		μPD70108H	μPD70116H		AC	CY	V	P	S	Z
BCD operation instruction	ADD4S		0 0 0 0 1 1 1 1 / 0 0 1 0 0 0 0 0	2	19×n+7	19×n+7	dst BCD string←dst BCD string + src BCD string*	U	×	U	U	U	×
	SUB4S		0 0 0 0 1 1 1 1 / 0 0 1 0 0 0 1 0	2	19×n+7	19×n+7	dst BCD string←dst BCD string – src BCD string*	U	×	U	U	U	×
	CMP4S		0 0 0 0 1 1 1 1 / 0 0 1 0 0 1 1 0	2	19×n+7	19×n+7	dst BCD string – src BCD string*	U	×	U	U	U	×
	ROL4	reg8	0 0 0 0 1 1 1 1 / 0 0 1 0 1 0 0 0 1 1 0 0 0 reg	3	13	13	reg: AL_L ← Upper | Lower						
		mem8	0 0 0 0 1 1 1 1 / 0 0 1 0 1 0 0 0 mod 0 0 0 mem	3-5	28	28	mem: AL_L ← Upper | Lower						
	ROR4	reg8	0 0 0 0 1 1 1 1 / 0 0 1 0 1 0 1 0 1 1 0 0 0 reg	3	17	17	reg: AL_L → Upper | Lower						
		mem8	0 0 0 0 1 1 1 1 / 0 0 1 0 1 0 1 0 mod 0 0 0 mem	3-5	32	32	mem: AL_L → Upper | Lower						
Increment/decrement instruction	INC	reg8	1 1 1 1 1 1 1 0 / 1 1 0 0 0 reg	2	2	2	reg8←reg8+1	×		×	×	×	×
		mem	1 1 1 1 1 1 1 W / mod 0 0 0 mem	2-4	16/24	16/24	(mem)←(mem)+1	×		×	×	×	×
		reg16	0 1 0 0 0 reg	1	2	2	reg16←reg16+1	×		×	×	×	×
	DEC	reg8	1 1 1 1 1 1 1 0 / 1 1 0 0 1 reg	2	2	2	reg8←reg8–1	×		×	×	×	×
		mem	1 1 1 1 1 1 1 W / mod 0 0 1 mem	2-4	16/24	16/24	(mem)←(mem)–1	×		×	×	×	×
		reg16	0 1 0 0 1 reg	1	2	2	reg16←reg16–1	×		×	×	×	×

n : Half of the number of digits of BCD

* : The number of digits in BCD is specified using the CL register. Its value can range from 1 to 254.

Instruction Group	Mnemonic	Operand	Operation Code																Number of Bytes	Number of Clocks		Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0		μPD70108H	μPD70116H		AC	CY	V	P	S	Z
Multiplication instruction	MULU	reg8	1	1	1	1	0	1	1	0	1	1	1	0	0	reg			2	21-22	21-22	AW←ALxreg8 AH = 0: CY←0,V←0 AH ≠ 0: CY←1,V←1	U	x	x	U	U	U
		mem8	1	1	1	1	0	1	1	0	mod	1	0	0	mem				2-4	27-28	27-28	AW←ALx(mem8) AH = 0: CY←0,V←0 AH ≠ 0: CY←1,V←1	U	x	x	U	U	U
		reg16	1	1	1	1	0	1	1	1	1	1	1	0	0	reg			2	29-30	29-30	DW, AW←AWxreg16 DW = 0: CY←0,V←0 DW ≠ 0: CY←1,V←1	U	x	x	U	U	U
		mem16	1	1	1	1	0	1	1	1	mod	1	0	0	mem				2-4	39-40	35-36/ 39-40	DW,AW←AWx(mem16) DW = 0: CY←0,V←0 DW ≠ 0: CY←1,V←1	U	x	x	U	U	U
	MUL	reg8	1	1	1	1	0	1	1	0	1	1	1	0	1	reg			2	33-39	33-39	AW←ALxreg8 AH = Sign extension for AL: CY←0,V←0 AH ≠ Sign extension for AL: CY←1,V←1	U	x	x	U	U	U
		mem8	1	1	1	1	0	1	1	0	mod	1	0	1	mem				2-4	39-45	39-45	AW←ALx(mem8) AH = Sign extension for AL: CY←0,V←0 AH ≠ Sign extension for AL: CY←1,V←1	U	x	x	U	U	U
		reg16	1	1	1	1	0	1	1	1	1	1	1	0	1	reg			2	41-47	41-47	DW,AW←AWxreg16 DW = Sign extension for AW: CY←0,V←0 DW ≠ Sign extension for AW: CY←1,V←1	U	x	x	U	U	U
		mem16	1	1	1	1	0	1	1	1	mod	1	0	1	mem				2-4	51-57	47-53/ 51-57	DW,AW←AWx(mem16) DW = Sign extension for AW: CY←0,V←0 DW ≠ Sign extension for AW: CY←1,V←1	U	x	x	U	U	U

Instruction Group	Mnemonic	Operand	Operation Code														Number of Bytes	Number of Clocks		Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0		μPD70108H	μPD70116H		AC	CY	V	P	S	Z
Multiplication instruction	MUL (cont'd)	reg16, (reg16',)* imm8	0	1	1	0	1	0	1	1	1	1	reg				reg'		3	28-34	28-34	reg16←reg16'ximm8 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			3-5	38-44	34-40/ 38-44	reg16←(mem16)ximm8 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'		4	36-42	36-42	reg16←reg16'ximm16 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			4-6	46-52	42-48/ 46-52	reg16←(mem16)ximm16 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 omitted, the same register, specified for the first operand, is assumed to be specified.

Instruction Group	Mnemonic	Operand	Operation Code														Number of Bytes	Number of Clocks		Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0		μPD70108H	μPD70116H		AC	CY	V	P	S	Z
Unsigned division instruction	DIVU	reg8	1	1	1	1	0	1	1	0	1	1	1	1	0	reg			2	19	19	temp←AW When temp+reg8 ≤ FFH, AH←temp%reg8, AL←temp+reg8 When temp+reg8 > FFH, TA←(001H,000H), TC←(003H,002H) SP←SP−2,(SP+1,SP)←PSW,IE←0,BRK←0 SP←SP−2,(SP+1,SP)←PS,PS←TC SP←SP−2,(SP+1,SP)←PC,PC←TA	U	U	U	U	U	U
		mem8	1	1	1	1	0	1	1	0	mod	1	1	0	mem				2-4	25	25	temp←AW When temp+(mem8) ≤ FFH, AH←temp%(mem8), AL←temp+(mem8) When temp+(mem8) > FFH, TA←(001H,000H), TC←(003H,002H) SP←SP−2,(SP+1,SP)←PSW,IE←0,BRK←0 SP←SP−2,(SP+1,SP)←PS,PS←TC SP←SP−2,(SP+1,SP)←PC,PC←TA	U	U	U	U	U	U
		reg16	1	1	1	1	0	1	1	1	1	1	1	1	0	reg			2	25	25	temp←DW,AW When temp+reg16 ≤ FFFFH, DW←temp%reg16, AW←temp+reg16 When temp+reg16 > FFFFH, TA←(001H,000H), TC←(003H,002H) SP←SP−2,(SP+1,SP)←PSW,IE←0,BRK←0 SP←SP−2,(SP+1,SP)←PS,PS←TC SP←SP−2,(SP+1,SP)←PC,PC←TA	U	U	U	U	U	U
		mem16	1	1	1	1	0	1	1	1	mod	1	1	0	mem				2-4	34	30/34	temp←DW, AW When temp+(mem16) ≤ FFFFH, DW←temp%(mem16), AW←temp+(mem16) When temp+(mem16) > FFFFH, TA←(001H,000H), TC←(003H,002H) SP←SP−2,(SP+1,SP)←PSW,IE←0,BRK←0 SP←SP−2,(SP+1,SP)←PS,PS←TC SP←SP−2,(SP+1,SP)←PC,PC←TA	U	U	U	U	U	U

Instruction Group	Mnemonic	Operand	Operation Code								Number of Bytes	Number of Clocks		Operation	Flags					
			7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0								μPD70108H	μPD70116H		AC	CY	V	P	S	Z
Signed division instruction	DIV	reg8	1 1 1 1 0 1 1 0	1 1 1 1 1 reg							2	29-34	29-34	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, TA←(001H,000H), TC←(003H,002H) SP←SP–2,(SP+1,SP)←PSW,IE←0,BRK←0 SP←SP–2,(SP+1,SP)←PS,PS←TC SP←SP–2,(SP+1,SP)←PC,PC←TA	U	U	U	U	U	U
		mem8	1 1 1 1 0 1 1 0	mod 1 1 1 mem							2-4	34-39	34-39	temp←AW When temp+(mem8) > 0 and temp+(mem8) ≤ 7FH or when temp+(mem8) < 0 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, TA←(001H,000H), TC←(003H,002H) SP←SP–2,(SP+1,SP)←PSW,IE←0,BRK←0 SP←SP–2,(SP+1,SP)←PS,PS←TC SP←SP–2,(SP+1,SP)←PC,PC←TA	U	U	U	U	U	U
		reg16	1 1 1 1 0 1 1 1	1 1 1 1 1 reg							2	38-43	38-43	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, TA←(001H,000H),TC←(003H,002H) SP←SP–2,(SP+1,SP)←PSW,IE←0,BRK←0 SP←SP–2,(SP+1,SP)←PS,PS←TC SP←SP–2,(SP+1,SP)←PC,PC←TA	U	U	U	U	U	U
		mem16	1 1 1 1 0 1 1 1	mod 1 1 1 mem							2-4	47-52	43-48/ 47-52	temp←DW,AW When temp+(mem16) > 0 and temp+(mem16) ≤ 7FFFH or when temp+(mem16) < 0 and temp+(mem16) > 0–7FFFH–1, DW←temp%(mem16),AW←temp+(mem16) When temp+(mem16) > 0 and temp+(mem16) > 7FFFH or when temp+(mem16) < 0 and temp+(mem16) ≤ 0–7FFFH–1, TA←(001H,000H),TC←(003H,002H) SP←SP–2,(SP+1,SP)←PSW,IE←0,BRK←0 SP←SP–2,(SP+1,SP)←PS,PS←TC SP←SP–2,(SP+1,SP)←PC,PC←TA	U	U	U	U	U	U

Instruction Group	Mnemonic	Operand	Operation Code																Number of Bytes	Number of Clocks		Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0		μPD70108H	μPD70116H		AC	CY	V	P	S	Z
BCD coprocessor instruction	ADJBA		0	0	1	1	0	1	1	1									1	7	7	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	3	3	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	7	7	When AL ∧ 0FH > 9 or AC = 1, AL←AL−6, AC←1 CY←AC, AL←AL ∧ 0FH	×	×	U	U	U	U
	ADJ4S		0	0	1	0	1	1	1	1									1	3	3	When AL ∧ 0FH > 9 or AC = 1, AL←AL−6, AC←1 When AL > 9FH or CY = 1, AL←AL−60H, CY←1	×	×	U	×	×	×
Data conversion	CVTBD		1	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0	2	15	15	AH←AL÷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	7	7	AL←AH×0AH+AL, AH←0	U	U	U	×	×	×
	CVTBW		1	0	0	1	1	0	0	0									1	2	2	When AL < 80H, AH←0. Otherwise, AH←FFH						
	CVTWL		1	0	0	1	1	0	0	1									1	4-5	4-5	When AW < 8000H, DW←0. Otherwise, DW←FFFFH						
Compare instruction	CMP	reg,reg'	0	0	1	1	1	0	1	W	1	1	reg				reg'		2	2	2	reg−reg'	×	×	×	×	×	×
		mem,reg	0	0	1	1	1	0	0	W	mod		reg				mem		2-4	11/15	11/15	(mem)−reg	×	×	×	×	×	×
		reg,mem	0	0	1	1	1	0	1	W	mod		reg				mem		2-4	11/15	11/15	reg−(mem)	×	×	×	×	×	×
		reg,imm	1	0	0	0	0	0	s	W	1	1	1	1	1		reg		3-4	4	4	reg−imm	×	×	×	×	×	×
		mem,imm	1	0	0	0	0	0	s	W	mod		1	1	1		mem		3-6	13/17	13/17	(mem)−imm	×	×	×	×	×	×
		acc,imm	0	0	1	1	1	1	0	W									2-3	4	4	When W = 0, AL−imm When W = 1, AW−imm	×	×	×	×	×	×

Instruction Group	Mnemonic	Operand	Operation Code																Number of Bytes	Number of Clocks		Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0		μPD70108H	μPD70116H		AC	CY	V	P	S	Z
Complements	NOT	reg	1	1	1	1	0	1	1	W	1	1	0	1	0	reg			2	2	2	reg←$\overline{reg}$						
		mem	1	1	1	1	0	1	1	W	mod		0	1	0	mem			2-4	16/24	16/24	(mem)←$\overline{(mem)}$						
	NEG	reg	1	1	1	1	0	1	1	W	1	1	0	1	1	reg			2	2	2	reg←$\overline{reg}$+1	×	×	×	×	×	×
		mem	1	1	1	1	0	1	1	W	mod		0	1	1	mem			2-4	16/24	16/24	(mem)←$\overline{(mem)}$+1	×	×	×	×	×	×
Logical operation instruction	TEST	reg,reg'	1	0	0	0	0	1	0	W	1	1	reg'			reg			2	2	2	reg ∧ reg'	U	0	0	×	×	×
		mem,reg reg,mem	1	0	0	0	0	1	0	W	mod		reg			mem			2-4	10/14	10/14	(mem) ∧ reg	U	0	0	×	×	×
		reg,imm	1	1	1	1	0	1	1	W	1	1	0	0	0	reg			3-4	4	4	reg ∧ imm	U	0	0	×	×	×
		mem,imm	1	1	1	1	0	1	1	W	mod		0	0	0	mem			3-6	11/15	11/15	(mem) ∧ imm	U	0	0	×	×	×
		acc,imm	1	0	1	0	1	0	0	W									2-3	4	4	When W = 0, AL ∧ imm8 When W = 1, AW ∧ imm16	U	0	0	×	×	×
	AND	reg,reg'	0	0	1	0	0	0	1	W	1	1	reg			reg'			2	2	2	reg←reg ∧ reg'	U	0	0	×	×	×
		mem,reg	0	0	1	0	0	0	0	W	mod		reg			mem			2-4	16/24	16/24	(mem)←(mem) ∧ reg	U	0	0	×	×	×
		reg,mem	0	0	1	0	0	0	1	W	mod		reg			mem			2-4	11/15	11/15	reg←reg ∧ (mem)	U	0	0	×	×	×
		reg,imm	1	0	0	0	0	0	0	W	1	1	1	0	0	reg			3-4	4	4	reg←reg ∧ imm	U	0	0	×	×	×
		mem,imm	1	0	0	0	0	0	0	W	mod		1	0	0	mem			3-6	18/26	18/26	(mem)←(mem) ∧ imm	U	0	0	×	×	×
		acc,imm	0	0	1	0	0	1	0	W									2-3	4	4	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	2	2	reg←reg ∨ reg'	U	0	0	×	×	×
		mem,reg	0	0	0	0	1	0	0	W	mod		reg			mem			2-4	16/24	16/24	(mem)←(mem) ∨ reg	U	0	0	×	×	×
		reg,mem	0	0	0	0	1	0	1	W	mod		reg			mem			2-4	11/15	11/15	reg←reg ∨ (mem)	U	0	0	×	×	×
		reg,imm	1	0	0	0	0	0	0	W	1	1	0	0	1	reg			3-4	4	4	reg←reg ∨ imm	U	0	0	×	×	×
		mem,imm	1	0	0	0	0	0	0	W	mod		0	0	1	mem			3-6	18/26	18/26	(mem)←(mem) ∨ imm	U	0	0	×	×	×
		acc,imm	0	0	0	0	1	1	0	W									2-3	4	4	When W = 0, AL←AL ∨ imm8 When W = 1, AW←AW ∨ imm16	U	0	0	×	×	×

Instruction Group	Mnemonic	Operand	Operation Code																Number of Bytes	Number of Clocks		Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0		μPD70108H	μPD70116H		AC	CY	V	P	S	Z
Logical operation instruction	XOR	reg,reg'	0	0	1	1	0	0	1	W	1	1		reg				reg'	2	2	2	reg←reg ∀ reg'	U	0	0	×	×	×
		mem,reg	0	0	1	1	0	0	0	W	mod			reg				mem	2-4	16/24	16/24	(mem)←(mem) ∀ reg	U	0	0	×	×	×
		reg,mem	0	0	1	1	0	0	1	W	mod			reg				mem	2-4	11/15	11/15	reg←reg ∀ (mem)	U	0	0	×	×	×
		reg,imm	1	0	0	0	0	0	0	W	1	1	1	1	0			reg	3-4	4	4	reg←reg ∀ imm	U	0	0	×	×	×
		mem,imm	1	0	0	0	0	0	0	W	mod			1	1	0		mem	3-6	18/26	18/26	(mem)←(mem) ∀ imm	U	0	0	×	×	×
		acc,imm	0	0	1	1	0	1	0	W									2-3	4	4	When W = 0, AL←AL ∀ imm8 When W = 1, AW←AW ∀ imm16	U	0	0	×	×	×

Instruction Group	Mnemonic	Operand	Operation Code																Number of Bytes	Number of Clocks		Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0		μPD70108H	μPD70116H		AC	CY	V	P	S	Z
Bit operation instruction	TEST1	reg8,CL	0	0	0	1	0	0	0	0	1	1	0	0	0	reg			3	3	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				3-5	8	8	(mem8) bit NO.CL = 0 : Z←1 (mem8) bit NO.CL = 1 : Z←0	U	0	0	U	U	×
		reg16,CL					0	0	0	1	1	1	0	0	0	reg			3	3	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				3-5	12	8/12	(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	4	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				4-6	9	9	(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	4	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	1	mod	0	0	0	mem				4-6	13	9/13	(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	4	4	reg8 bit NO.CL←$\overline{\text{reg8 bit NO.CL}}$						
		mem8,CL					0	1	1	0	mod	0	0	0	mem				3-5	13	13	(mem8) bit NO.CL←$\overline{\text{(mem8) bit NO.CL}}$						
		reg16,CL					0	1	1	1	1	1	0	0	0	reg			3	4	4	reg16 bit NO.CL←$\overline{\text{reg16 bit NO.CL}}$						
		mem16,CL					0	1	1	1	mod	0	0	0	mem				3-5	21	13/21	(mem16) bit NO.CL←$\overline{\text{(mem16) bit NO.CL}}$						
		reg8,imm3					1	1	1	0	1	1	0	0	0	reg			4	5	5	reg8 bit NO.imm3←$\overline{\text{reg8 bit NO.imm3}}$						
		mem8,imm3					1	1	1	0	mod	0	0	0	mem				4-6	14	14	(mem8) bit NO.imm3←$\overline{\text{(mem8) bit NO.imm3}}$						
		reg16,imm4					1	1	1	1	1	1	0	0	0	reg			4	5	5	reg16 bit NO.imm4←$\overline{\text{reg16 bit NO.imm4}}$						
		mem16,imm4					1	1	1	1	mod	0	0	0	mem				4-6	22	14/22	(mem16) bit NO.imm4←$\overline{\text{(mem16) bit NO.imm4}}$						
			2nd byte*								3rd byte*																	
	NOT1	CY	1	1	1	1	0	1	0	1									1	2	2	CY←$\overline{\text{CY}}$		×				

* : 1st byte = 0FH

Instruction Group	Mnemonic	Operand	Operation Code																Number of Bytes	Number of Clocks		Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0		μPD70108H	μPD70116H		AC	CY	V	P	S	Z
Bit operation instruction	CLR1	reg8,CL	0	0	0	1	0	0	1	0	1	1	0	0	0	reg			3	5	5	reg8 bit NO.CL←0						
		mem8,CL					0	0	1	0	mod		0	0	0	mem			3-5	14	14	(mem8) bit NO.CL←0						
		reg16,CL					0	0	1	1	1	1	0	0	0	reg			3	5	5	reg16 bit NO.CL←0						
		mem16,CL					0	0	1	1	mod		0	0	0	mem			3-5	22	14/22	(mem16) bit NO.CL←0						
		reg8,imm3					1	0	1	0	1	1	0	0	0	reg			4	6	6	reg8 bit NO.imm3←0						
		mem8,imm3					1	0	1	0	mod		0	0	0	mem			4-6	15	15	(mem8) bit NO.imm3←0						
		reg16,imm4					1	0	1	1	1	1	0	0	0	reg			4	6	6	reg16 bit NO.imm4←0						
		mem16,imm4					1	0	1	1	mod		0	0	0	mem			4-6	23	15/23	(mem16) bit NO.imm4←0						
	SET1	reg8,CL					0	1	0	0	1	1	0	0	0	reg			3	4	4	reg8 bit NO.CL←1						
		mem8,CL					0	1	0	0	mod		0	0	0	mem			3-5	13	13	(mem8) bit NO.CL←1						
		reg16,CL					0	1	0	1	1	1	0	0	0	reg			3	4	4	reg16 bit NO.CL←1						
		mem16,CL					0	1	0	1	mod		0	0	0	mem			3-5	21	13/21	(mem16) bit NO.CL←1						
		reg8,imm3					1	1	0	0	1	1	0	0	0	reg			4	5	5	reg8 bit NO.imm3←1						
		mem8,imm3					1	1	0	0	mod		0	0	0	mem			4-6	14	14	(mem8) bit NO.imm3←1						
		reg16,imm4					1	1	0	1	1	1	0	0	0	reg			4	5	5	reg16 bit NO.imm4←1						
		mem16,imm4					1	1	0	1	mod		0	0	0	mem			4-6	22	14/22	(mem16) bit NO.imm4←1						

2nd byte* 3rd byte*

* : 1st byte = 0FH

	CLR1	CY	1	1	1	1	1	0	0	0									1	2	2	CY←0		0				
		DIR	1	1	1	1	1	1	0	0									1	2	2	DIR←0						
	SET1	CY	1	1	1	1	1	0	0	1									1	2	2	CY←1		1				
		DIR	1	1	1	1	1	1	0	1									1	2	2	DIR←1						

Instruction Group	Mnemonic	Operand	Operation Code	Number of Bytes	Number of Clocks		Operation	Flags					
			7 6 5 4 3 2 1 0 / 7 6 5 4 3 2 1 0		μPD70108H	μPD70116H		AC	CY	V	P	S	Z
Shift instruction	SHL	reg,1	1 1 0 1 0 0 0 W / 1 1 1 0 0 reg	2	6	6	CY←MSB of reg, reg←regx2 When MSB of reg ≠ CY, V←1 When MSB of reg = CY, V←0	U	×	×	×	×	×
		mem,1	1 1 0 1 0 0 0 W / mod 1 0 0 mem	2-4	16/24	16/24	CY←MSB of (mem), (mem)←(mem)x2 When MSB of (mem) ≠ CY, V←1 When MSB of (mem) = CY, V←0	U	×	×	×	×	×
		reg,CL	1 1 0 1 0 0 1 W / 1 1 1 0 0 reg	2	7+n	7+n	temp←CL, repeats following operation, while temp ≠ 0: CY←MSB of reg, reg←regx2 temp←temp–1	U	×	U	×	×	×
		mem,CL	1 1 0 1 0 0 1 W / mod 1 0 0 mem	2-4	19/27+n	19/27+n	temp←CL, repeats following operation, while temp ≠ 0: CY←MSB of (mem), (mem)←(mem)x2 temp←temp–1	U	×	U	×	×	×
		reg,imm8	1 1 0 0 0 0 0 W / 1 1 1 0 0 reg	3	7+n	7+n	temp←imm8, repeats following operation, while temp ≠ 0: CY←MSB of reg, reg←regx2 temp←temp–1	U	×	U	×	×	×
		mem,imm8	1 1 0 0 0 0 0 W / mod 1 0 0 mem	3-5	19/27+n	19/27+n	temp←imm8, repeats following operation, while temp ≠ 0: CY←MSB of (mem), (mem)←(mem)x2 temp←temp–1	U	×	U	×	×	×
	SHR	reg,1	1 1 0 1 0 0 0 W / 1 1 1 0 1 reg	2	6	6	CY←LSB of reg, reg←reg÷2 MSB of reg ≠ next bit of MSB of reg: V←1 MSB of reg = next bit of MSB of reg: V←0	U	×	×	×	×	×
		mem,1	1 1 0 1 0 0 0 W / mod 1 0 0 mem	2-4	16/24	16/24	CY←MSB of (mem), (mem)←(mem)÷2 MSB of reg ≠ next bit of MSB of reg: V←1 MSB of reg = next bit of MSB of reg: CY, V←0	U	×	×	×	×	×

n : Number of shifts

Instruction Group	Mnemonic	Operand	Operation Code	Number of Bytes	Number of Clocks		Operation	Flags					
			7 6 5 4 3 2 1 0 / 7 6 5 4 3 2 1 0		μPD70108H	μPD70116H		AC	CY	V	P	S	Z
Shift instruction	SHR (cont'd)	reg,CL	1 1 0 1 0 0 1 W / 1 1 1 0 0 reg	2	7+n	7+n	temp←CL, repeats following operation, while temp ≠ 0: CY←LSB of reg, reg←reg+2 temp←temp−1	U	×	U	×	×	×
		mem,CL	1 1 0 1 0 0 1 W / mod 1 0 0 mem	2-4	19/27+n	19/27+n	temp←CL, repeats following operation, while temp ≠ 0: CY←LSB of (mem), (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	7+n	7+n	temp←imm8, repeats following operation, while temp ≠ 0: CY←LSB of reg, reg←reg+2 temp←temp−1	U	×	U	×	×	×
		mem,imm8	1 1 0 0 0 0 0 W / mod 1 0 0 mem	3-5	19/27+n	19/27+n	temp←imm8, repeats following operation, while temp ≠ 0: CY←LSB of (mem), (mem)←(mem)+2 temp←temp−1	U	×	U	×	×	×
	SHRA	reg,1	1 1 0 1 0 0 0 W / 1 1 1 1 1 reg	2	6	6	CY←LSB of reg, reg←reg+2, V←0 MSB of operand is not affected.	U	×	U	×	×	×
		mem,1	1 1 0 1 0 0 0 W / mod 1 1 1 mem	2-4	16/24	16/24	CY←LSB of (mem), (mem)←(mem)+2, V←0 MSB of operand is not affected.	U	×	U	×	×	×
		reg,CL	1 1 0 1 0 0 1 W / 1 1 1 1 1 reg	2	7+n	7+n	temp←CL, repeats following operation, while temp ≠ 0: CY←LSB of reg, reg←reg+2 temp←temp−1, MSB of operand is not affected.	U	×	U	×	×	×
		mem,CL	1 1 0 1 0 0 1 W / mod 1 1 1 mem	2-4	19/27+n	19/27+n	temp←CL, repeats following operation, while temp ≠ 0: CY←LSB of (mem), (mem)←(mem)+2 temp←temp−1, MSB of operand is not affected.	U	×	U	×	×	×
		reg,imm8	1 1 0 0 0 0 0 W / 1 1 1 1 1 reg	3	7+n	7+n	temp←imm8, repeats following operation, while temp ≠ 0: CY←LSB of reg, reg←reg+2 temp←temp−1, MSB of operand is not affected.	U	×	U	×	×	×
		mem,imm8	1 1 0 0 0 0 0 W / mod 1 1 1 mem	3-5	19/27+n	19/27+n	temp←imm8, repeats following operation, while temp ≠ 0: CY←LSB of (mem), (mem)←(mem)+2 temp←temp−1, MSB of operand is not affected.	U	×	U	×	×	×

n : Number of shifts

Instruction Group	Mnemonic	Operand	Operation Code																Number of Bytes	Number of Clocks		Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0		μPD70108H	μPD70116H		AC	CY	V	P	S	Z
Rotate ion instruction	ROL	reg,1	1	1	0	1	0	0	0	W	1	1	0	0	0	reg			2	6	6	CY←MSB of reg, reg←regx2+CY When MSB of reg ≠ CY: V←1 When MSB of reg = CY: V←0		x	x			
		mem,1	1	1	0	1	0	0	0	W	mod	0	0	0	mem				2-4	16/24	16/24	CY←MSB of (mem), (mem)←(mem)x2+CY When MSB of (mem) ≠ CY: V←1 When MSB of (mem) = CY: V←0		x	x			
		reg,CL	1	1	0	1	0	0	1	W	1	1	0	0	0	reg			2	7+n	7+n	temp←CL, repeats following operation, while temp ≠ 0: CY←MSB of reg, reg←regx2+CY temp←temp−1		x	U			
		mem,CL	1	1	0	1	0	0	1	W	mod	0	0	0	mem				2-4	19/27+n	19/27+n	temp←CL, repeats following operation, while temp ≠ 0: CY←MSB of (mem), (mem)←(mem)x2+CY temp←temp−1		x	U			
		reg,imm8	1	1	0	0	0	0	0	W	1	1	0	0	0	reg			3	7+n	7+n	temp←imm8, repeats following operation, while temp ≠ 0: CY←MSB of reg, reg←regx2+CY temp←temp−1		x	U			
		mem,imm8	1	1	0	0	0	0	0	W	mod	0	0	0	mem				3-5	19/27+n	19/27+n	temp←imm8, repeats following operation, while temp ≠ 0: CY←MSB of (mem), (mem)←(mem)x2+CY temp←temp−1		x	U			
	ROR	reg,1	1	1	0	1	0	0	0	W	1	1	0	0	1	reg			2	6	6	CY←LSB of reg←reg+2 MSB of reg←CY MSB of reg ≠ next bit of MSB of reg: V←1 MSB of reg = next bit of MSB of reg: V←0		x	x			
		mem,1	1	1	0	1	0	0	0	W	mod	0	0	1	mem				2-4	16/24	16/24	CY←LSB of (mem)←(mem)+2 MSB of (mem)←CY MSB of (mem) ≠ next bit of MSB of reg: V←1 MSB of (mem) = next bit of MSB of reg: V←0		x	x			

n : Number of shifts

Instruction Group	Mnemonic	Operand	Operation Code	Number of Bytes	Number of Clocks		Operation	Flags					
			7 6 5 4 3 2 1 0 / 7 6 5 4 3 2 1 0		μPD70108H	μPD70116H		AC	CY	V	P	S	Z
Rotate ion instruction	ROR (cont'd)	reg,CL	1 1 0 1 0 0 1 W / 1 1 0 0 1 reg	2	7+n	7+n	temp←CL, repeats following operation, while temp ≠ 0: CY←LSB of reg, reg←reg+2 MSB of reg←CY temp←temp–1		x	U			
		mem,CL	1 1 0 1 0 0 1 W / mod 0 0 1 mem	2-4	19/27+n	19/27+n	temp←CL, repeats following operation, while temp ≠ 0: CY←LSB of (mem), (mem)←(mem)+2 MSB of (mem)←CY temp←temp–1		x	U			
		reg,imm8	1 1 0 0 0 0 0 W / 1 1 0 0 1 reg	3	7+n	7+n	temp←imm8, repeats following operation, while temp ≠ 0: CY←LSB of reg, reg←reg+2 MSB of reg←CY temp←temp–1		x	U			
		mem,imm8	1 1 0 0 0 0 0 W / mod 0 0 1 mem	3-5	19/27+n	19/27+n	temp←imm8, repeats following operation, while temp ≠ 0: CY←LSB of (mem), (mem)←(mem)+2 MSB of (mem)←CY temp←temp–1		x	U			
	ROLC	reg,1	1 1 0 1 0 0 0 W / 1 1 0 1 0 reg	2	6	6	tmpcy←CY, CY←MSB of reg reg←regx2+tmpcy When MSB of reg ≠ CY: V←1 When MSB of reg = CY: V←0		x	x			
		mem,1	1 1 0 1 0 0 0 W / mod 0 1 0 mem	2-4	16/24	16/24	tmpcy←CY, CY←MSB of (mem) (mem)←(mem)x2+tmpcy When MSB of (mem) ≠ CY: V←1 When MSB of (mem) = CY: V←0		x	x			
		reg,CL	1 1 0 1 0 0 1 W / 1 1 0 1 0 reg	2	7+n	7+n	temp←CL, repeats following operation, while temp ≠ 0: tmpcy←CY, CY←MBS of reg reg←regx2+tmpcy temp←temp–1		x	U			

n : Number of shifts

Instruction Group	Mnemonic	Operand	Operation Code																Number of Bytes	Number of Clocks		Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0		μPD70108H	μPD70116H		AC	CY	V	P	S	Z
Rotate ion instruction	ROLC (cont'd)	mem,CL	1	1	0	1	0	0	1	W	mod	0	1	0	mem				2-4	19/27+n	19/27+n	temp←CL, repeats following operation, while temp ≠ 0: tmpcy←CY, CY←MSB of (mem) (mem)←(mem)x2+tmpcy temp←temp−1		x	U			
		reg,imm8	1	1	0	0	0	0	0	W	1	1	0	1	0	reg			3	7+n	7+n	temp←imm8, repeats following operation, while temp ≠ 0: tmpcy←CY, CY←MSB of reg reg←regx2+tmpcy temp←temp−1		x	U			
		mem,imm8	1	1	0	0	0	0	0	W	mod	0	1	0	mem				3-5	19/27+n	19/27+n	temp←imm8, repeats following operation, while temp ≠ 0: tmpcy←CY, CY←MSB of (mem) (mem)←(mem)x2+tmpcy temp←temp−1		x	U			
	RORC	reg,1	1	1	0	1	0	0	0	W	1	1	0	1	1	reg			2	6	6	tmpcy←CY, CY← LSB of reg reg←reg+2 MSB of reg←tmpcy When MSB of reg ≠ next bit of MSB of: V←1 When MSB of reg = next bit of MSB of reg: V←0		x	x			
		mem,1	1	1	0	1	0	0	0	W	mod	0	1	1	mem				2-4	16/24	16/24	tmpcy←CY, CY←LSB of (mem) (mem)←(mem)+2 MSB of (mem)←tmpcy When MSB of (mem) ≠ next bit of MSB of (mem): V←1 When MSB of (mem) = next bit of MSB of (mem): V←0		x	x			
		reg,CL	1	1	0	1	0	0	1	W	1	1	0	1	1	reg			2	7+n	7+n	temp←CL, repeats following operation, while temp ≠ 0: tmpcy←CY, CY←LSB of reg reg←reg+2 MSB of reg←tmpcy temp←temp−1		x	U			

n : Number of shifts

Instruction Group	Mnemonic	Operand	Operation Code																Number of Bytes	Number of Clocks		Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0		μPD70108H	μPD70116H		AC	CY	V	P	S	Z
Rotate ion instruction	RORC (cont'd)	mem,CL	1	1	0	1	0	0	1	W	mod	0	1	1	mem				2-4	19/27+n	19/27+n	temp←CL, repeats following operation, while temp ≠ 0: tmpcy←CY, CY←LSB of (mem) (mem)←(mem)+2 MSB of (mem)←tmpcy temp←temp−1		x	U			
		reg,imm8	1	1	0	0	0	0	0	W	1	1	0	1	1	reg			3	7+n	7+n	temp←imm8, repeats following operation, while temp ≠ 0: tmpcy←CY, CY←LSB of reg reg←reg+2 MSB of reg←tmpcy temp←temp−1		x	U			
		mem,imm8	1	1	0	0	0	0	0	W	mod	0	1	1	mem				3-5	19/27+n	19/27+n	temp←imm8, repeats following operation, while temp ≠ 0: tmpcy←CY, CY←LSB of (mem) (mem)←(mem)+2 MSB of (mem)←tmpcy temp←temp−1		x	U			

n : Number of shifts

Instruction Group	Mnemonic	Operand	Operation Code	Number of Bytes	Number of Clocks		Operation	Flags					
			7 6 5 4 3 2 1 0 / 7 6 5 4 3 2 1 0		μPD70108H	μPD70116H		AC	CY	V	P	S	Z
Subroutine control instruction	CALL	near-proc	1 1 1 0 1 0 0 0	3	20	16/20	SP←SP–2, (SP+1,SP)←PC PC←PC+disp						
		regptr16	1 1 1 1 1 1 1 1 / 1 1 0 1 0 reg	2	18	14/18	SP←SP–2, (SP+1,SP)←PC PC←regptr16						
		memptr16	1 1 1 1 1 1 1 1 / mod 0 1 0 mem	2-4	31	23/31	TA←(memptr16) SP←SP–2, (SP+1,SP)←PC, PC←TA						
		far-proc	1 0 0 1 1 0 1 0	5	29	21/29	SP←SP–2, (SP+1,SP)←PS, PS←seg SP←SP–2, (SP+1,SP)←PC, PC←offset						
		memptr32	1 1 1 1 1 1 1 1 / mod 0 1 1 mem	2-4	47	31/47	TA←(memptr32), TB←(memptr32+2) SP←SP–2, (SP+1,SP)←PS, PS←TB SP←SP–2, (SP+1,SP)←PC, PC←TA						
	RET		1 1 0 0 0 0 1 1	1	19	15/19	PC←(SP+1,SP) SP←SP+2						
		pop-value	1 1 0 0 0 0 1 0	3	24	20/24	PC←(SP+1,SP) SP←SP+2, SP←SP+pop-value						
			1 1 0 0 1 0 1 1	1	29	21/29	PC←(SP+1,SP) PS←(SP+3,SP+2) SP←SP+4						
		pop-value	1 1 0 0 1 0 1 0	3	32	24/32	PC←(SP+1,SP) PS←(SP+3,SP+2) SP←SP+4, SP←SP+pop-value						

Instruction Group	Mnemonic	Operand	Operation Code																Number of Bytes	Number of Clocks		Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0		μPD70108H	μPD70116H		AC	CY	V	P	S	Z
Stack operation instruction	PUSH	mem16	1	1	1	1	1	1	1	1	mod	1	1	0	mem				2-4	26	18/26	SP←SP−2 (SP+1,SP)←(mem16)						
		reg16	0	1	0	1	0	reg											1	12	8/12	SP←SP−2 (SP+1,SP)←reg16						
		sreg	0	0	0	sreg		1	1	0									1	12	8/12	SP←SP−2 (SP+1,SP)←sreg						
		PSW	1	0	0	1	1	1	0	0									1	12	8/12	SP←SP−2 (SP+1,SP)←PSW						
		R	0	1	1	0	0	0	0	0									1	67	35/67	Push registers on the stack						
		imm8	0	1	1	0	1	0	1	0									2	11	7/11	(SP−1,SP−2)←sign expansion of imm8 SP←SP−2						
		imm16	0	1	1	0	1	0	0	0									3	12	8/12	(SP−1,SP−2)←imm16 SP←SP−2						
	POP	mem16	1	0	0	0	1	1	1	1	mod	0	0	0	mem				2-4	25	17/25	SP←SP+2 (mem16)←(SP−1, SP−2)						
		reg16	0	1	0	1	1	reg											1	12	8/12	SP←SP+2 reg16←(SP−1, SP−2)						
		sreg	0	0	0	sreg		1	1	1									1	12	8/12	SP←SP+2 sreg←(SP−1, SP−2) sreg: SS,DS0,DS1						
		PSW	1	0	0	1	1	1	0	1									1	12	8/12	SP←SP+2 PSW←(SP−1, SP−2)	R	R	R	R	R	R
		R	0	1	1	0	0	0	0	1									1	75	43/75	Pop registers from the stack						

Instruction Group	Mnemonic	Operand	Operation Code																Number of Bytes	Number of Clocks		Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0		μPD70108H	μPD70116H		AC	CY	V	P	S	Z
Branch instruction	PREPARE	imm16,imm8	1	1	0	0	1	0	0	0									4	**Note1**	**Note2**	Prepare New Stack Frame						
	DISPOSE		1	1	0	0	1	0	0	1									1	10	6/10	Dispose of Stack Frame						
	BR	near-label	1	1	1	0	1	0	0	1									3	13	13	PC←PC+disp						
		short-label	1	1	1	0	1	0	1	1									2	12	12	PC←PC+ext-disp8						
		regptr16	1	1	1	1	1	1	1	1	1	1	1	0	0	reg			2	11	11	PC←regptr16						
		memptr16	1	1	1	1	1	1	1	1	mod			1	0	0	mem		2-4	24	20/24	PC←(memptr16)						
		far-label	1	1	1	0	1	0	1	0									5	15	15	PS←seg PC←offset						
		memptr32	1	1	1	1	1	1	1	1	mod			1	0	1	mem		2-4	35	27/35	PS←(memptr32+2) PC←(memptr32)						

Note 1. When imm8 = 0 : 16
When imm8 ≥ 1 : 23+16 (imm8-1)

2. When imm8 = 0 : 12/16
When imm8 ≥ 1 : {19+8 (imm8–1)}/{23+16 (imm8–1)}

Instruction Group	Mnemonic	Operand	Operation Code								Number of Bytes	Note Number of Clocks		Operation		Flags					
			7	6	5	4	3	2	1	0		μPD70108H	μPD70116H			AC	CY	V	P	S	Z
Condition branch instruction	BV	short-label	0	1	1	1	0	0	0	0	2	14/4	14/4	if V = 1	PC←PC+ext-disp8						
	BNV	short-label					0	0	0	1	2	14/4	14/4	if V = 0	PC←PC+ext-disp8						
	BC BL	short-label					0	0	1	0	2	14/4	14/4	if CY = 1	PC←PC+ext-disp8						
	BNC BNL	short-label					0	0	1	1	2	14/4	14/4	if CY = 0	PC←PC+ext-disp8						
	BE BZ	short-label					0	1	0	0	2	14/4	14/4	if Z = 1	PC←PC+ext-disp8						
	BNE BNZ	short-label					0	1	0	1	2	14/4	14/4	if Z = 0	PC←PC+ext-disp8						
	BNH	short-label					0	1	1	0	2	14/4	14/4	if CY ∨ Z = 1	PC←PC+ext-disp8						
	BH	short-label					0	1	1	1	2	14/4	14/4	if CY ∨ Z = 0	PC←PC+ext-disp8						
	BN	short-label					1	0	0	0	2	14/4	14/4	if S = 1	PC←PC+ext-disp8						
	BP	short-label					1	0	0	1	2	14/4	14/4	if S = 0	PC←PC+ext-disp8						
	BPE	short-label					1	0	1	0	2	14/4	14/4	if P = 1	PC←PC+ext-disp8						
	BPO	short-label					1	0	1	1	2	14/4	14/4	if P = 0	PC←PC+ext-disp8						
	BLT	short-label					1	1	0	0	2	14/4	14/4	if S ∀ V = 1	PC←PC+ext-disp8						
	BGE	short-label					1	1	0	1	2	14/4	14/4	if S ∀ V = 0	PC←PC+ext-disp8						
	BLE	short-label					1	1	1	0	2	14/4	14/4	if (S ∀ V) ∨ Z = 1	PC←PC+ext-disp8						
	BGT	short-label					1	1	1	1	2	14/4	14/4	if (S ∀ V) ∨ Z = 0	PC←PC+ext-disp8						
	DBNZNE	short-label	1	1	1	0	0	0	0	0	2	14/5	14/5	CW = CW−1 if Z = 0 and CW ≠ 0	PC←PC+ext-disp8						
	DBNZE	short-label	1	1	1	0	0	0	0	1	2	14/5	14/5	CW = CW−1 if Z = 1 and CW ≠ 0	PC←PC+ext-disp8						
	DBNZ	short-label	1	1	1	0	0	0	1	0	2	13/5	13/5	CW = CW−1 if CW ≠ 0	PC←PC+ext-disp8						

Note Condition judgement: true/false

Instruction Group	Mnemonic	Operand	Operation Code																Number of Bytes	Number of Clocks		Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0		μPD70108H	μPD70116H		AC	CY	V	P	S	Z
Interrupt instruction	BCWZ	short-label	1	1	1	0	0	0	1	1									2	13/5[Note1]	13/5[Note1]	if CW = 0 PC←PC+ext-disp8						
	BRK	3	1	1	0	0	1	1	0	0									1	50	38/50	TA←(00DH,00CH), TC←(00FH,00EH) SP←SP–2, (SP+1,SP)←PSW, IE←0, BRK←0 SP←SP–2, (SP+1,SP)←PS, PS←TC SP←SP–2, (SP+1,SP)←PC, PC←TA						
		imm8 (≠3)	1	1	0	0	1	1	0	1									2	50	38/50	TA←(4n+1,4n), TC←(4n+3,4n+2) n = imm8 SP←SP–2, (SP+1,SP)←PSW, IE←0, BRK←0 SP←SP–2, (SP+1,SP)←PS, PS←TC SP←SP–2, (SP+1,SP)←PC, PC←TA						
	BRKV		1	1	0	0	1	1	1	0									1	**Note2**	**Note3**	When V = 1, TA←(011H,010H), TC←(013H,012H) SP←SP–2, (SP+1,SP)←PSW, IE←0, BRK←0 SP←SP–2, (SP+1,SP)←PS, PS←TC SP←SP–2, (SP+1,SP)←PC, PC←TA						
	RETI		1	1	0	0	1	1	1	1									1	39	27/39	PC←(SP+1,SP), PS←(SP+3,SP+2), PSW←(SP+5,SP+4), SP←SP+6	R	R	R	R	R	R

Note 1. Condition judgement: true/false

2. When V = 1 : 52
When V = 0 : 3

3. When V = 1 : 40/52
When V = 0 : 3

Instruction Group	Mnemonic	Operand	Operation Code								Number of Bytes	Number of Clocks		Operation	Flags					
			7	6	5	4	3	2	1	0		μPD70108H	μPD70116H		AC	CY	V	P	S	Z
Interrupt instruction	BRKEM	imm8	0	0	0	0	1	1	1	1	3	50	38/50	TA←(4n+1,4n), TC←(4n+3,4n+2) n = imm8 SP←SP–2, (SP+1,SP)←PSW, MD←0 MD is enabled to write SP←SP–2, (SP+1,SP)←PS, PS←TC SP←SP–2, (SP+1,SP)←PC, PC←TA						
	CHKIND	reg16,mem32	0	1	1	0	0	0	1	0	2-4	**Note1**	**Note2**	When (mem32) > reg16 or (mem32+2) < reg16, TA←(015H,014H), TC←(017H,016H) SP←SP–2, (SP+1,SP)←PSW, IE←0, BRK←0 SP←SP–2, (SP+1,SP)←PS, PS←TC SP←SP–2, (SP+1,SP)←PC, PC←TA						

7	6	5	4	3	2	1	0
mod		reg		mem			

Note 1. 73-76 when interrupt condition is satisfied.
26 when interrupt condition is not satisfied.

2. (53-56)/(73-76) when interrupt condition is satisfied.
18/26 when interrupt condition is not satisfied.

Instruction Group	Mnemonic	Operand	Operation Code																Number of Bytes	Number of Clocks		Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0		μPD70108H	μPD70116H		AC	CY	V	P	S	Z
CPU control instruction	HALT		1	1	1	1	0	1	0	0									1	2	2	CPU Halt						
	POLL		1	0	0	1	1	0	1	1									1	2+5n	2+5n	Poll and wait n: $\overline{\text{POLL}}$ pin sampling times						
	DI		1	1	1	1	1	0	1	0									1	2	2	IE←0						
	EI		1	1	1	1	1	0	1	1									1	2	2	IE←1						
	BUSLOCK		1	1	1	1	0	0	0	0									1	2	2	Bus Lock Prefix						
	FPO1	fp-op	1	1	0	1	1	X	X	X	1	1	Y	Y	Y	Z	Z	Z	2	2	2	No Operation						
		fp-op,mem	1	1	0	1	1	X	X	X	mod		Y	Y	Y	mem			2-4	15	11/15	data bus←(mem)						
	FPO2	fp-op	0	1	1	0	0	1	1	X	1	1	Y	Y	Y	Z	Z	Z	2	2	2	No Operation						
		fp-op,mem	0	1	1	0	0	1	1	X	mod		Y	Y	Y	mem			2-4	15	11/15	data bus←(mem)						
	NOP		1	0	0	1	0	0	0	0									1	3	3	No Operation						

	*		0	0	1	sreg			1	1	0								1	2	2	Segment override prefix						

*: DS0:, DS1:, PS:, SS:

Instruction Group	Mnemonic	Operand	Operation Code																Number of Bytes	Number of Clocks		Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0		μPD70108H	μPD70116H		AC	CY	V	P	S	Z
8080	RETEM		1	1	1	0	1	1	0	1	1	1	1	1	1	1	0	1	2	39	27/39	PC←(SP+1,SP), PC←(SP+3,SP+2), PSW←(SP+5,SP+4), SP←SP+6, MD is disabled to write.	R	R	R	R	R	R
	CALLN	imm8	1	1	1	0	1	1	0	1	1	1	1	0	1	1	0	1	3	58	38/58	TA←(4n+1,4n), TC←(4n+3,4n+2) n = imm8 SP←SP-2, (SP+1,SP)←PSW, MD←1 SP←SP-2, (SP+1,SP)←PS, PS←TC SP←SP-2, (SP+1,SP)←PC, PC←TA						

9. ELECTRICAL SPECIFICATIONS

9.1 WHEN V_{DD} = 5 V ± 10%

ABSOLUTE MAXIMUM RATINGS (T_A = 25°C)

Item	Symbol	Conditions	Ratings	Unit
Power supply voltage	V_{DD}		−0.5 to +7.0	V
Input voltage	V_I	V_{DD} = 5 V ± 10 %	−0.5 to V_{DD}+0.5	V
Clock input voltage	V_K		−0.5 to V_{DD}+1.0	V
Output voltage	V_O		−0.5 to V_{DD}+0.5	V
Operating ambient temperature	T_A		−40 to +85	°C
Storage temperature	T_{stg}		−65 to +150	°C

Caution 1. Be sure to avoid direct coupling between IC-product output (or input/output) pins or between V_{DD} or Vcc and GND. However, direct coupling between open-drain pins or open-collector pins is O.K. Direct coupling is also possible between pins which become high-impedance if they are installed on external circuitry for which the timing is set to avoid output conflicts.

2. If the absolute maximum ratings of even one of the items above are exceeded for a moment, the product quality may be damaged. In other words, the absolute maximum ratings are the values above which physical damage may result. Be sure to use the product in a way in which these rating values are not exceeded.

The specifications and conditions described in the DC and AC characteristics are the limits for normal operation and quality assurance of the product.

DC CHARACTERISTICS (T_A = −40 to +85°C, V_{DD} = 5 V ± 10%)

Item	Symbol	Conditions	MIN.	TYP.	MAX.	Unit
Input voltage, high	V_{IH}	Other than (1), (2)	2.2		V_{DD}+0.5	V
		(1) READY	0.6V_{DD}		V_{DD}+0.5	
		(2) HLDRQ (RQ0, RQ1) : DIP only				
Input voltage, low	V_{IL}		−0.5		0.8	V
Clock input voltage, high	V_{KH}		0.8V_{DD}		V_{DD}+1.0	V
Clock input voltage, low	V_{KL}		−0.5		0.15V_{DD}	V
Output voltage, high	V_{OH}	I_{OH} = −2.5 mA	0.7V_{DD}			V
		I_{OH} = −100 μA	V_{DD}−0.4			
Output voltage, low	V_{OL}	I_{OL} = 2.5 mA			0.4	V
Input leakage current, high	I_{LIH}	V_I = V_{DD}			10	μA
Input leakage current, low	I_{LIL}	V_I = 0 V			−10	μA
Output leakage current, high	I_{LOH}	V_O = V_{DD}			10	μA
Output leakage current, low	I_{LOL}	V_O = 0 V			−10	μA
$\overline{RQ}$ input current, high	I_{HQH}	V_I = V_{DD}			10	μA
$\overline{RQ}$ input current, low	I_{HQL}	V_I = 0 V			−0.5	mA
Latch leakage current, high	I_{LLH}	V_I = 3.0 V	−50		−300	μA
Latch leakage current, low	I_{LLL}	V_I = 0.8 V	50		300	μA
Latch inverse current, (L → H)	I_{ILH}				400	μA
Latch inverse current, (H → L)	I_{ILL}				−400	μA
★ Supply current [Note]	I_{DD}	Operating: V_{IH} = V_{DD}, V_{IL} = 0V, t_{CYK} ≤ 2 μs		3.5f_x	5f_x	mA
★		Standy: (HALT), V_{IH} = V_{DD}, V_{IL} = 0 V, t_{CYK} ≤ 2 μs		0.5f_x	0.8f_x	
		Standby: (Clock input stops) V_{IH} = V_{DD}, V_{IL} = 0 V			50	μA

Note The constants 3.5, 5, 0.5, and 0.8 of the value are in mA/MHz. f_x is the frequency of CLK input.

For the supply current for the L and F rated products, please consult your NEC dealer.

CAPACITANCE (T_A = 25°C, V_{DD} = 0 V)

Item	Symbol	Conditions	MIN.	TYP.	MAX.	Unit
Input capacitance	C_I	f_C = 1MHz Unmeasured pins returned to 0 V			10	pF
I/O capacitance	C_{IO}				15	pF

AC CHARACTERISTICS (T_A = −40 to +85°C, V_{DD} = 5 V ± 10%)

(1) In both small-scale/large-scale mode

Loading capacity of output pins: C_L = 100pF

Item	Symbol		μPD70108H-10, μPD70116H-10		μPD70108H-12, μPD70116H-12		μPD70108H-16, μPD70116H-16		Unit
			MIN.	MAX.	MIN.	MAX.	MIN.	MAX.	
Clock cycle	①	t_{CYK}	100		80		62.5		ns
Clock pulse high-level width (V_{KH} = 3.0 V)	②	t_{KKH}	39		35		25		ns
Clock pulse low-level width (V_{KL} = 1.7 V)	③	t_{KKL}	45		35		25		ns
Clock rise time (1.7 → 3.0 V)	④	t_{KR}		5		5		5	ns
Clock fall time (3.0 → 1.7 V)	⑤	t_{KF}		5		5		5	ns
RESET release delay time (V_{DD} = 5 V to 10 %)	⑥	t_{DVRST}	1		1		1		μs
RESET set time (vs. CLK ↑)	⑦	t_{SRSTK}	15		10		10		ns
RESET hold time (vs. CLK ↑)	⑧	t_{HKRST}	15		15		15		ns
RESET high-level width	⑨	t_{WRSTH}	$4t_{CYK}$		$4t_{CYK}$		$4t_{CYK}$		ns
READY inactive set time (vs. CLK ↓)	⑩	t_{SRYLK}	−8		−8		−8		ns
READY inactive hold time (vs. CLK ↑)	⑪	t_{HKRYH}	20		20		15		ns
READY active set time (vs. CLK ↑)	⑫	t_{SRYHK}	t_{KKL}−8		t_{KKL}−8		t_{KKL}−8		ns
READY active hold time (vs. CLK ↑)	⑬	t_{HKRYL}	20		20		15		ns
Data set time (vs. CLK ↓)	⑭	t_{SDK}	10		10		5		ns
Data hold time (vs. CLK ↓)	⑮	t_{HKD}	10		10		10		ns
NMI, INT, $\overline{POLL}$ set time (vs. CLK ↑)	⑯	t_{SIK}	15		15		10		ns
Input rise time [Note1] (0.8 → 2.2 V)	⑰	t_{IR}		20		20		20	ns
Input fall time [Note1] (2.2 → 0.8 V)	⑱	t_{IF}		12		12		12	ns
Output rise time (0.8 → 2.2 V)	⑲	t_{OR}		20		15		10	ns
Output fall time (2.2 → 0.8 V)	⑳	t_{OF}		12		10		8	ns
CLK ↓ → address delay time	㉑	t_{DKA}	5	48	5	40	5	30	ns
CLK ↓ → address hold time	㉒	t_{HKA}	5		5		5		ns
CLK ↓ → PS delay time	㉓	t_{DKP}	5	50	5	40	5	30	ns
CLK ↑ → PS float delay time	㉔	t_{FKP}	5	50	5	40	5	30	ns
CLK ↓ → address float delay time	㉕	t_{FKA}	t_{HKA}	50	t_{HKA}	40	t_{HKA}	30	ns
CLK ↓ → $\overline{RD}$ ↓ delay time	㉖	t_{DKRL}	0	50	0	40	0	30	ns
CLK ↓ → $\overline{RD}$ ↑ delay time[Note2]	㉗	t_{DKRH}	0	50	0	40	0	30	ns
$\overline{RD}$ ↑ → address delay time	㉘	t_{DRHA}	t_{CYK}−35		t_{CYK}−25		t_{CYK}−10		ns
$\overline{RD}$ low-level width	㉙	t_{RR}	$2t_{CYK}$−40		$2t_{CYK}$−25		$2t_{CYK}$−20		ns
CLK ↓ → data output delay time	㉚	t_{DKD}	5	50	5	40	5	30	ns
CLK ↓ → data float delay time	㉛	t_{FKD}	0	50	0	40	0	30	ns

Note 1. CLK excluded

2. When $\overline{RD}$ becomes inactive in read cycle, the data has been read.

(2) Small-scale mode

Loading capacity of output pins: C_L = 100pF

Item	Symbol		μPD70108H-10, μPD70116H-10		μPD70108H-12, μPD70116H-12		μPD70108H-16, μPD70116H-16		Unit
			MIN.	MAX.	MIN.	MAX.	MIN.	MAX.	
Address set time (vs. ASTB ↓)	㉜	t_{SAST}	t_{KKL}–30		t_{KKL}–20		t_{KKL}–15		ns
CLK ↓ → ASTB ↑ delay time	㉝	t_{DKSTH}	0	40	0	35	0	30	ns
CLK ↑ → ASTB ↓ delay time	㉞	t_{DKSTL}	0	45	0	40	0	30	ns
ASTB high-level width	㉟	t_{STST}	t_{KKL}–10		t_{KKL}–10		t_{KKL}–5		ns
ASTB ↓ → address hold time	㊱	t_{HSTA}	t_{KKH}–10		t_{KKH}–10		t_{KKH}–5		ns
CLK → control delay time**Note**	㊲	t_{DKCT}	0	55	0	40	0	30	ns
Address float → $\overline{RD}$ ↓ delay time	㊳	t_{AFRL}	0		0		0		ns
$\overline{WR}$ low-level width	㊴	t_{WW}	2t_{CYK}–35		2t_{CYK}–25		2t_{CYK}–20		ns
HLDRQ set time (vs. CLK ↑)	㊵	t_{SHQK}	20		10		5		ns
CLK ↓ → HLDAK delay time	㊶	t_{DKHA}	0	60	0	40	0	30	ns
$\overline{WR}$ ↑ → $\overline{BUFEN}$ ↑	㊷	t_{WCT}	t_{KKL}–20		t_{KKL}–15		t_{KKL}–10		ns

Note When $\overline{BUFEN}$ becomes inactive in read cycle, the data has been read.

(3) Large-scale mode

Loading capacity of output pins: C_L = 100pF

Item	Symbol		μPD70108H-10, μPD70116H-10		μPD70108H-12, μPD70116H-12		μPD70108H-16, μPD70116H-16		Unit
			MIN.	MAX.	MIN.	MAX.	MIN.	MAX.	
CLK ↑ → BS ↓ delay time	㊸	t_{DKBL}	0	50	0	40	0	30	ns
CLK ↓ → BS ↑ delay time	㊹	t_{DKBH}	0	50	0	40	0	35	ns
Address float → $\overline{RD}$ ↓ delay time	㊺	t_{DAFRL}	0		0		0		ns
CLK ↓ → QS delay time	㊻	t_{DKQS}	0	50	0	40	0	30	ns
CLK ↓ → $\overline{AK}$ delay time	㊼	t_{DKAK}	0	40	0	40	0	30	ns
$\overline{RQ}$ set time (vs. CLK ↑)	㊽	t_{SRQK}	9		7		5		ns
$\overline{RQ}$ hold time (vs. CLK ↓)	㊾	t_{HKRQ1}	0		0		0		ns
$\overline{RQ}$ hold time (vs. CLK ↑)	㊿	t_{HKRQ2}	20		15		10		ns

9.2 WHEN V_{DD} = 3 V ± 10%

ABSOLUTE MAXIMUM RATINGS (T_A = 25°C)

Item	Symbol	Conditions	Ratings	Unit
Power supply voltage	V_{DD}		−0.5 to +7.0	V
Input voltage	V_I	V_{DD} = 3 V ± 10 %	−0.5 to V_{DD}+0.5	V
CLK input voltage	V_K		−0.5 to V_{DD}+1.0	V
Output voltage	V_O		−0.5 to V_{DD}+0.5	V
Operating ambient temperature	T_A		−40 to +85	°C
Storage temperature	T_{stg}		−65 to +150	°C

Caution 1. Be sure to avoid direct coupling between IC-product output (or input/output) pins or between V_{DD} or V_{CC} and GND. However, direct coupling between open-drain pins or open-collector pins is O.K. Direct coupling is also possible between pins which become high-impedance if they are installed on external circuitry for which the timing is set to avoid output conflicts.

2. If the absolute maximum ratings of even one of the items above are exceeded for a moment, the product quality may be damaged. In other words, the absolute maximum ratings are the values above which physical damage may result. Be sure to use the product in a way in which these rating values are not exceeded.

The specifications and conditions described in the DC and AC characteristics are the limits for normal operation and quality assurance of the product.

DC CHARACTERISTICS (T_A = −40 to +85°C, V_{DD} = 3 V ± 10%)

Item	Symbol	Conditions	MIN.	TYP.	MAX.	Unit
Input voltage, high	V_{IH}		0.7V_{DD}		V_{DD}+0.3	V
Input voltage, low	V_{IL}		−0.5		0.2V_{DD}	V
Clock input voltage, high	V_{KH}		0.8V_{DD}		V_{DD}+0.5	V
Clock input voltage, low	V_{KL}		−0.5		0.2V_{DD}	V
Output voltage, high	V_{OH}	I_{OH} = −2.5 mA	0.7V_{DD}			V
		I_{OH} = −100 μA	V_{DD}−0.4			
Output voltage, low	V_{OL}	I_{OL} = 2.5 mA			0.4	V
Input leakage current, high	I_{LIH}	V_I = V_{DD}			10	μA
Input leakage current, low	I_{LIL}	V_I = 0 V			−10	μA
Output leakage current, high	I_{LOH}	V_O = V_{DD}			10	μA
Output leakage current, low	I_{LOL}	V_O = 0 V			−10	μA
$\overline{RQ}$ input current, high	I_{HQH}	V_I = V_{DD}			10	μA
$\overline{RQ}$ input current, low	I_{HQL}	V_I = 0 V			−0.5	mA
Latch leakage current, high	I_{LLH}	V_I = 2.0 V	−15		−200	μA
Latch leakage current, low	I_{LLL}	V_I = 0.2 V	15		200	μA
Latch inverse current, (L → H)	I_{ILH}				300	μA
Latch inverse current, (H → L)	I_{ILL}				−300	μA
★ Supply current [Note]	I_{DD}	Operating: V_{IH} = V_{DD}, V_{IL} = 0 V, t_{CYK} ≤ 2 μs		2f_X	3.5f_X	mA
★		Standy: (HALT), V_{IH} = V_{DD}, V_{IL} = 0 V, t_{CYK} ≤ 2 μs		0.2f_X	0.4f_X	
		Standby (Clock input stops): V_{IH} = V_{DD}, V_{IL} = 0 V			30	μA

Note The constants 2, 3.5, 0.2, and 0.4 of the value are in mA/MHz. f_X is the frequency of clock input.

For the supply current for the L and F rated products, please consult your NEC dealer.

CAPACITANCE (T_A = 25°C, V_{DD} = 0 V)

Item	Symbol	Conditions	MIN.	TYP.	MAX.	Unit
Input capacitance	C_I	f_C = 1MHz Unmeasured pins returned to 0 V			10	pF
I/O capacitance	C_{IO}				15	pF

AC CHARACTERISTICS (T_A = −40 to +85°C, V_{DD} = 3 V ± 10%)

(1) In both small-scale/large-scale mode

Loading capacity of output pins: C_L = 100pF

Item	Symbol		μPD70108H-10, μPD70116H-10		μPD70108H-12, μPD70116H-12		μPD70108H-16, μPD70116H-16		Unit
			MIN.	MAX.	MIN.	MAX.	MIN.	MAX.	
Clock cycle	①	t_{CYK}	200		160		125		ns
Clock pulse high-level width (V_{KH} = 2.5 V)	②	t_{KKH}	69		60		44		ns
Clock pulse low-level width (V_{KL} = 1.0 V)	③	t_{KKL}	90		70		60		ns
Clock rise time (1.0 → 2.5 V)	④	t_{KR}		10		10		10	ns
Clock fall time (2.5 → 1.0 V)	⑤	t_{KF}		10		10		10	ns
RESET release delay time (V_{DD} = 3 V ± 10 %)	⑥	t_{DVRST}	1		1		1		μs
RESET set time (CLK ↑)	⑦	t_{SRSTK}	15		15		15		ns
RESET hold time (CLK ↑)	⑧	t_{HKRST}	15		15		15		ns
RESET high-level width	⑨	t_{WRSTH}	$4t_{CYK}$		$4t_{CYK}$		$4t_{CYK}$		ns
READY inactive set time (vs. CLK ↓)	⑩	t_{SRYLK}	−8		−8		−8		ns
READY inactive hold time (vs. CLK ↑)	⑪	t_{HKRYH}	30		25		20		ns
READY active set time (vs. CLK ↑)	⑫	t_{SRYHK}	t_{KKL}−8		t_{KKL}−8		t_{KKL}−8		ns
READY active hold time (vs. CLK ↑)	⑬	t_{HKRYL}	30		25		20		ns
Data set time (vs. CLK ↓)	⑭	t_{SDK}	30		25		20		ns
Data hold time (vs. CLK ↓)	⑮	t_{HKD}	10		10		10		ns
NMI, INT, $\overline{POLL}$ set time (vs. CLK ↑)	⑯	t_{SIK}	30		20		15		ns
Input rise time[Note1] (0.2 V_{DD} → 0.7 V_{DD})	⑰	t_{IR}		20		20		20	ns
Input fall time[Note1] (0.7 V_{DD} → 0.2 V_{DD})	⑱	t_{IF}		12		12		12	ns
Output rise time (0.2 V_{DD} → 0.7 V_{DD})	⑲	t_{OR}		20		20		20	ns
Output fall time (0.7 V_{DD} → 0.2 V_{DD})	⑳	t_{OF}		12		12		12	ns
CLK ↓ → address delay time	㉑	t_{DKA}	10	90	10	70	10	60	ns
CLK ↓ → address hold time	㉒	t_{HKA}	10		10		10		ns
CLK ↓ → PS delay time	㉓	t_{DKP}	10	90	10	70	10	60	ns
CLK ↑ → PS float delay time	㉔	t_{FKP}	10	80	10	70	10	60	ns
CLK ↓ → address float delay time	㉕	t_{FKA}	t_{HKA}	80	t_{HKA}	70	t_{HKA}	60	ns
CLK ↓ → $\overline{RD}$ ↓ delay time	㉖	t_{DKRL}	10	165	10	100	10	80	ns
CLK ↓ → $\overline{RD}$ ↑ delay time[Note2]	㉗	t_{DKRH}	10	150	10	100	10	80	ns
$\overline{RD}$ ↑ → address delay time	㉘	t_{DRHA}	t_{CYK}−35		t_{CYK}−25		t_{CYK}−10		ns
$\overline{RD}$ low-level width	㉙	t_{RR}	$2t_{CYK}$−75		$2t_{CYK}$−60		$2t_{CYK}$−50		ns
CLK ↓ → data output delay time	㉚	t_{DKD}	10	90	10	70	10	60	ns
CLK ↓ → data float delay time	㉛	t_{FKD}	10	80	10	70	10	60	ns

Note 1. CLK excluded

2. When $\overline{RD}$ becomes inactive in read cycle, the data has been read.

(2) Small-scale mode

Loading capacity of output pins: C_L = 100pF

Item	Symbol		μPD70108H-10, μPD70116H-10		μPD70108H-12, μPD70116H-12		μPD70108H-16, μPD70116H-16		Unit
			MIN.	MAX.	MIN.	MAX.	MIN.	MAX.	
Address set time (vs. ASTB ↓)	㉜	t_{SAST}	t_{KKL}–60		t_{KKL}–30		t_{KKL}–30		ns
CLK ↓ → ASTB ↑ delay time	㉝	t_{DKSTH}	5	80	5	70	5	50	ns
CLK ↑ → ASTB ↓ delay time	㉞	t_{DKSTL}	5	85	5	70	5	55	ns
ASTB high-level width	㉟	t_{STST}	t_{KKL}–20		t_{KKL}–15		t_{KKL}–10		ns
ASTB ↓ → address hold time	㊱	t_{HSTA}	t_{KKH}–10		t_{KKH}–10		t_{KKL}–10		ns
CLK → control delay time[Note]	㊲	t_{DKCT}	10	110	10	80	10	65	ns
Address float → $\overline{RD}$ ↓ delay time	㊳	t_{AFRL}	0		0		0		ns
$\overline{WR}$ low-level width	㊴	t_{WW}	2t_{CYK}–60		2t_{CYK}–50		2t_{CYK}–40		ns
HLDRQ set time (vs. CLK ↑)	㊵	t_{SHQK}	35		30		20		ns
CLK ↓ → HLDAK delay time	㊶	t_{DKHA}	10	160	10	120	10	100	ns
$\overline{WR}$ ↑ → $\overline{BUFEN}$ ↑	㊷	t_{WCT}	t_{KKL}–20		t_{KKL}–20		t_{KKL}–20		ns

Note When $\overline{BUFEN}$ becomes inactive in read cycle, the data has been read.

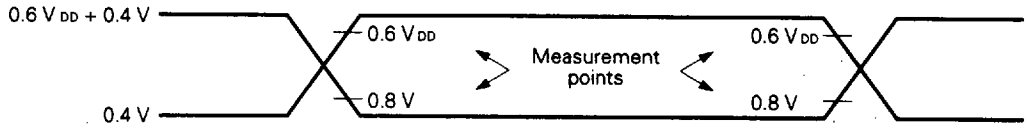
(3) Large-scale mode

Loading capacity of output pins: C_L = 100pF

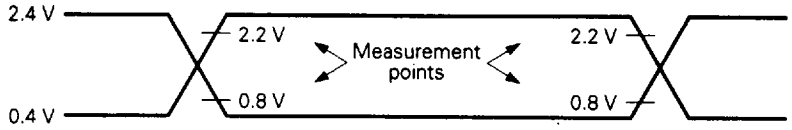
Item	Symbol		μPD70108H-10, μPD70116H-10		μPD70108H-12, μPD70116H-12		μPD70108H-16, μPD70116H-16		Unit
			MIN.	MAX.	MIN.	MAX.	MIN.	MAX.	
CLK ↑ → BS ↓ delay time	㊸	t_{DKBL}	10	110	10	70	10	60	ns
CLK ↓ → BS ↑ delay time	㊹	t_{DKBH}	10	130	10	70	10	70	ns
Address float → $\overline{RD}$ ↓ delay time	㊺	t_{DAFRL}	0		0		0		ns
CLK ↓ → QS delay time	㊻	t_{DKQS}	5	80	5	70	5	50	ns
CLK ↓ → $\overline{AK}$ delay time	㊼	t_{DKAK}	5	70	5	65	5	50	ns
$\overline{RQ}$ set time (vs. CLK ↑)	㊽	t_{SRQK}	20		15		10		ns
$\overline{RQ}$ hold time (vs. CLK ↓)	㊾	t_{HKRQ1}	0		0		0		ns
$\overline{RQ}$ hold time (vs. CLK ↑)	㊿	t_{HKRQ2}	40		35		30		ns

AC TEST INPUT WAVEFORMS (except CLK) (V_{DD} = 5 V ± 10%)

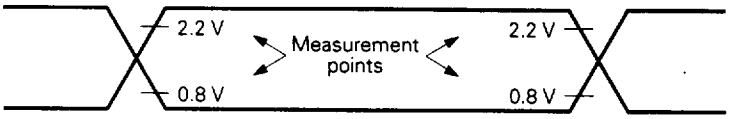
(1) READY, HLDRQ ($\overline{RQ0}$, $\overline{RQ1}$): DIP only



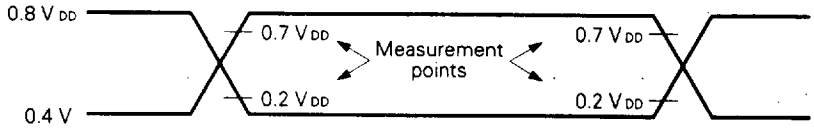
(2) Others



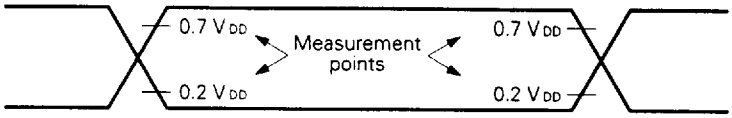
AC TEST OUTPUT MEASUREMENT POINTS (V_{DD} = 5 V ± 10%)



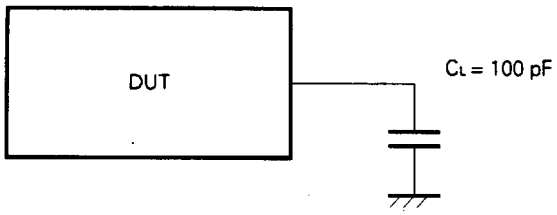
AC TEST INPUT WAVEFORM (except CLK) (V_{DD} = 3 V ± 10%)



AC TEST OUTPUT MEASUREMENT POINTS (V_{DD} = 3 V ± 10%)

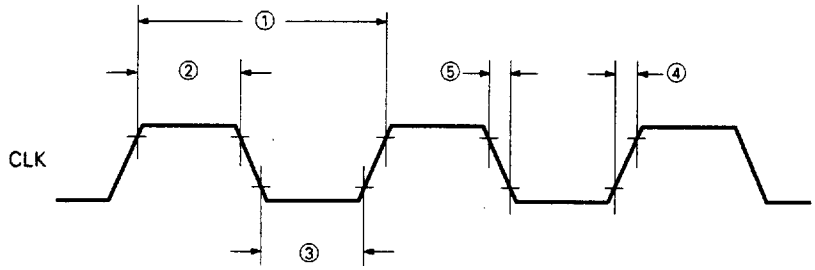


LOAD CONDITION

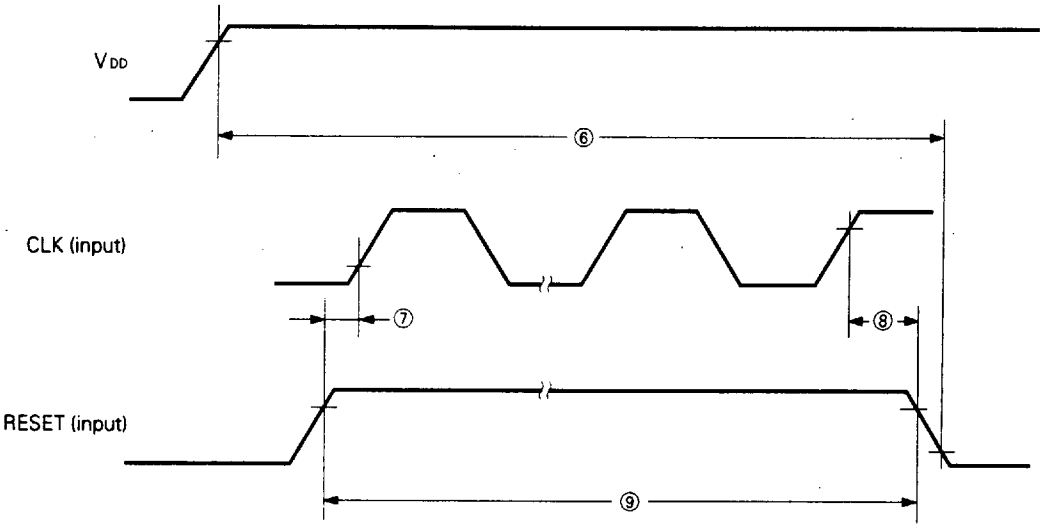


Caution If load capacitance exceeds 100pF due to circuit configuration, reduce the load capacitance of this device down to 100pF or less by inserting a buffer, etc.

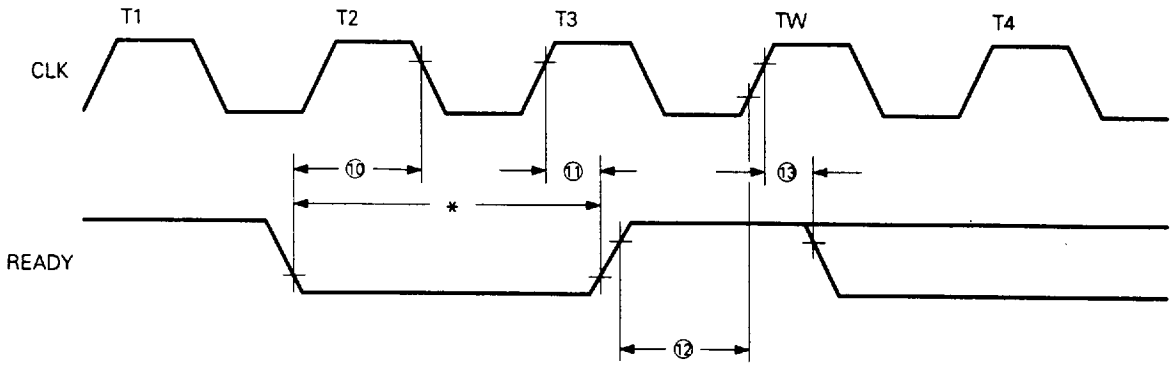
CLOCK TIMING WAVEFORMS



RESET TIMING

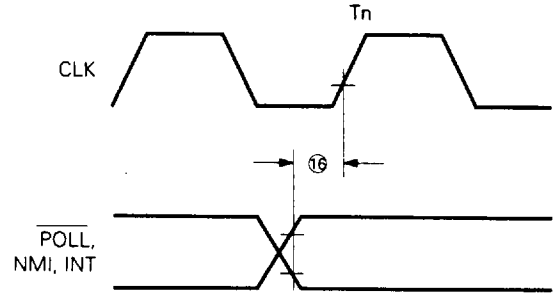


WAIT (READY) TIMING WAVEFORMS

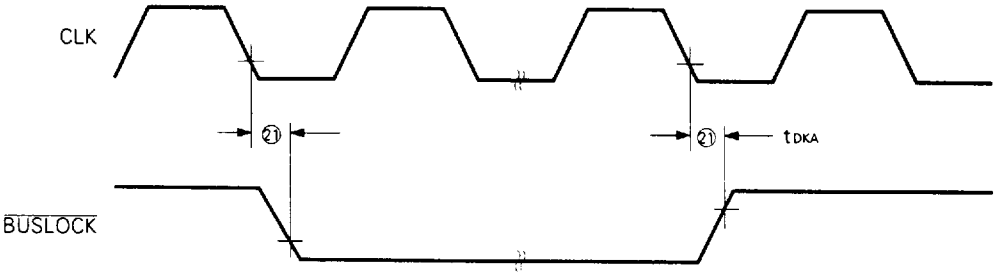


*: In this period, the READY signal must be fixed to low or high.

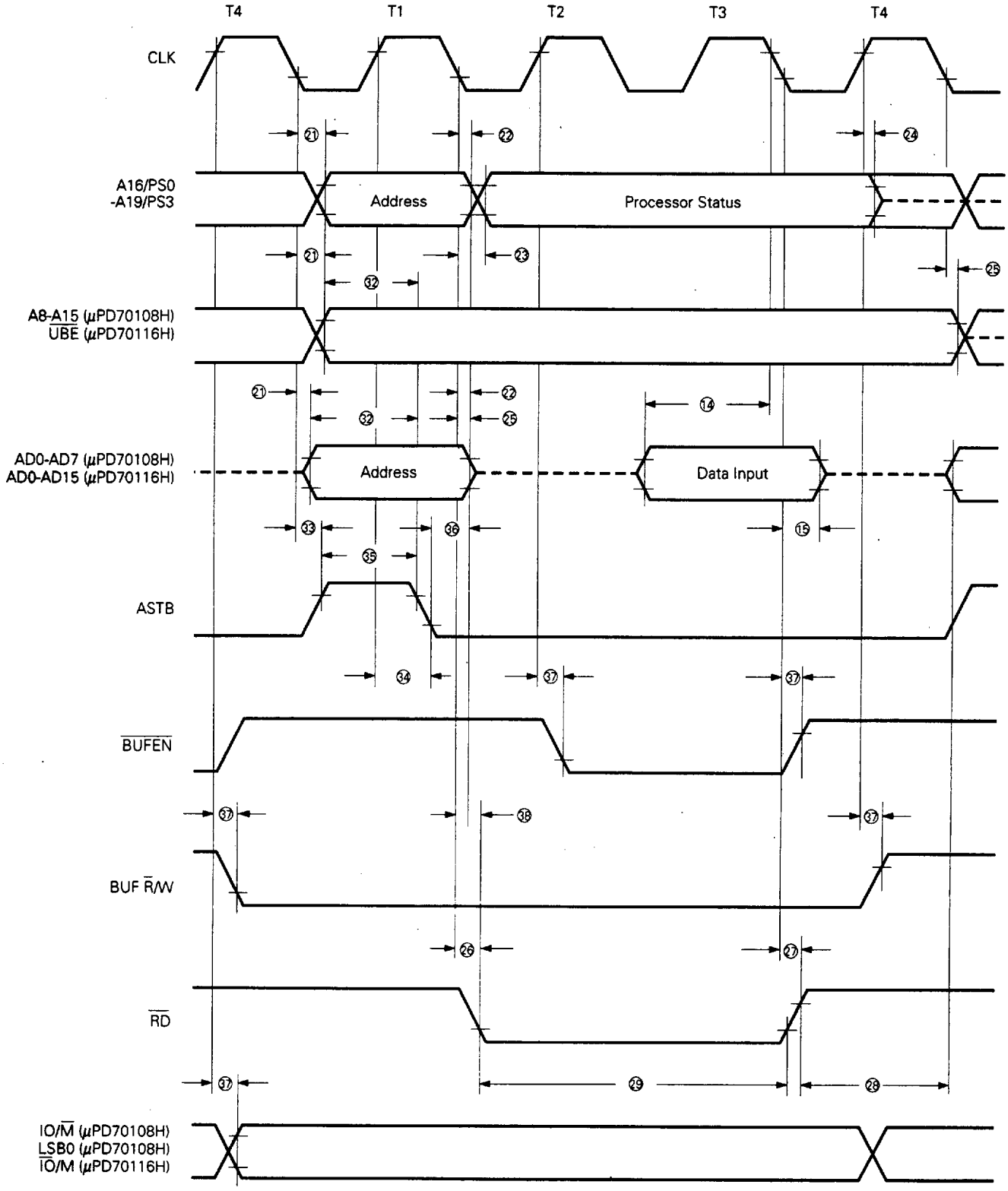
$\overline{\text{POLL}}$, NMI, INT INPUT TIMING WAVEFORMS



$\overline{\text{BUSLOCK}}$ OUTPUT TIMING WAVEFORMS

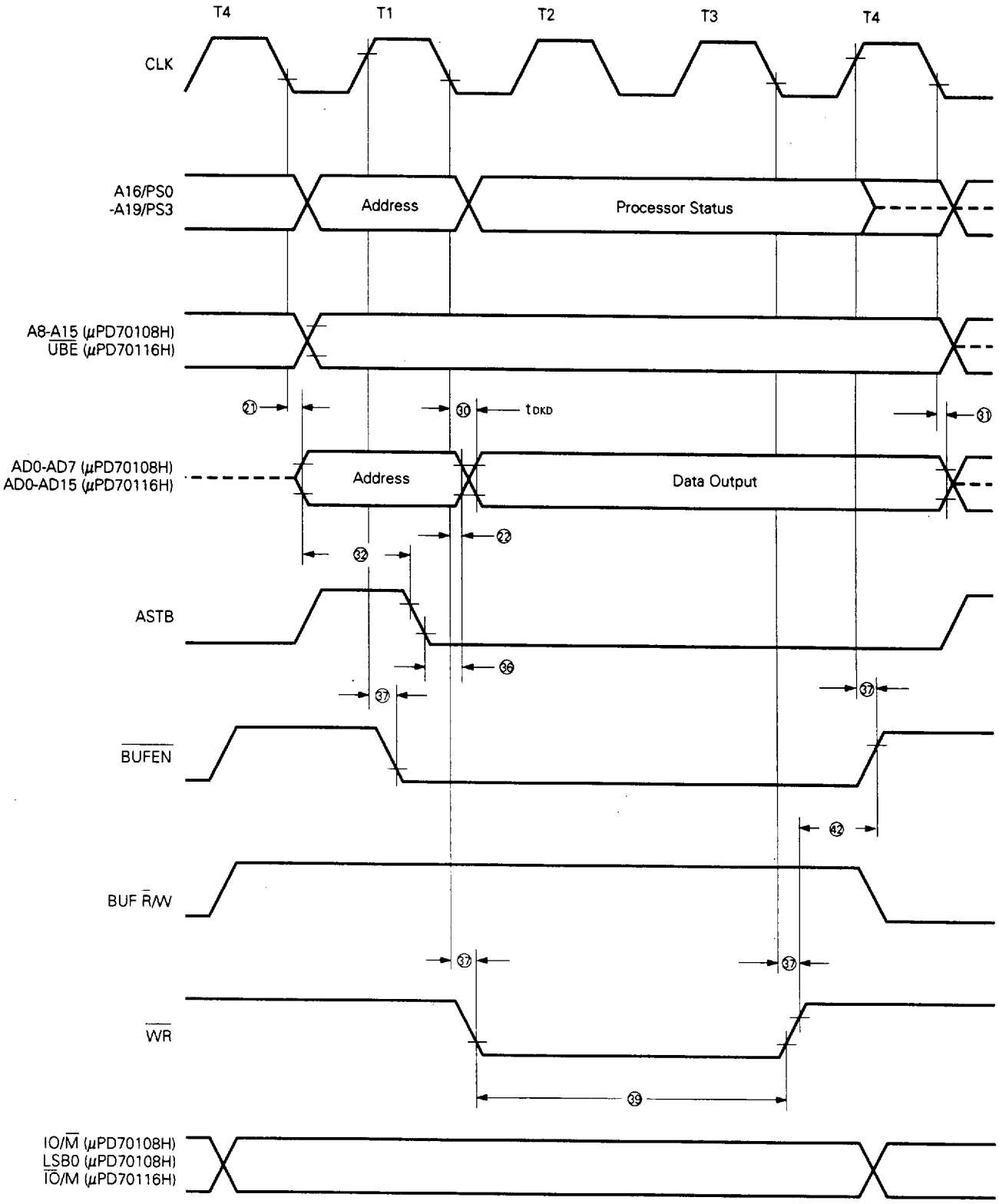


READ TIMING WAVEFORMS (SMALL-SCALE MODE)



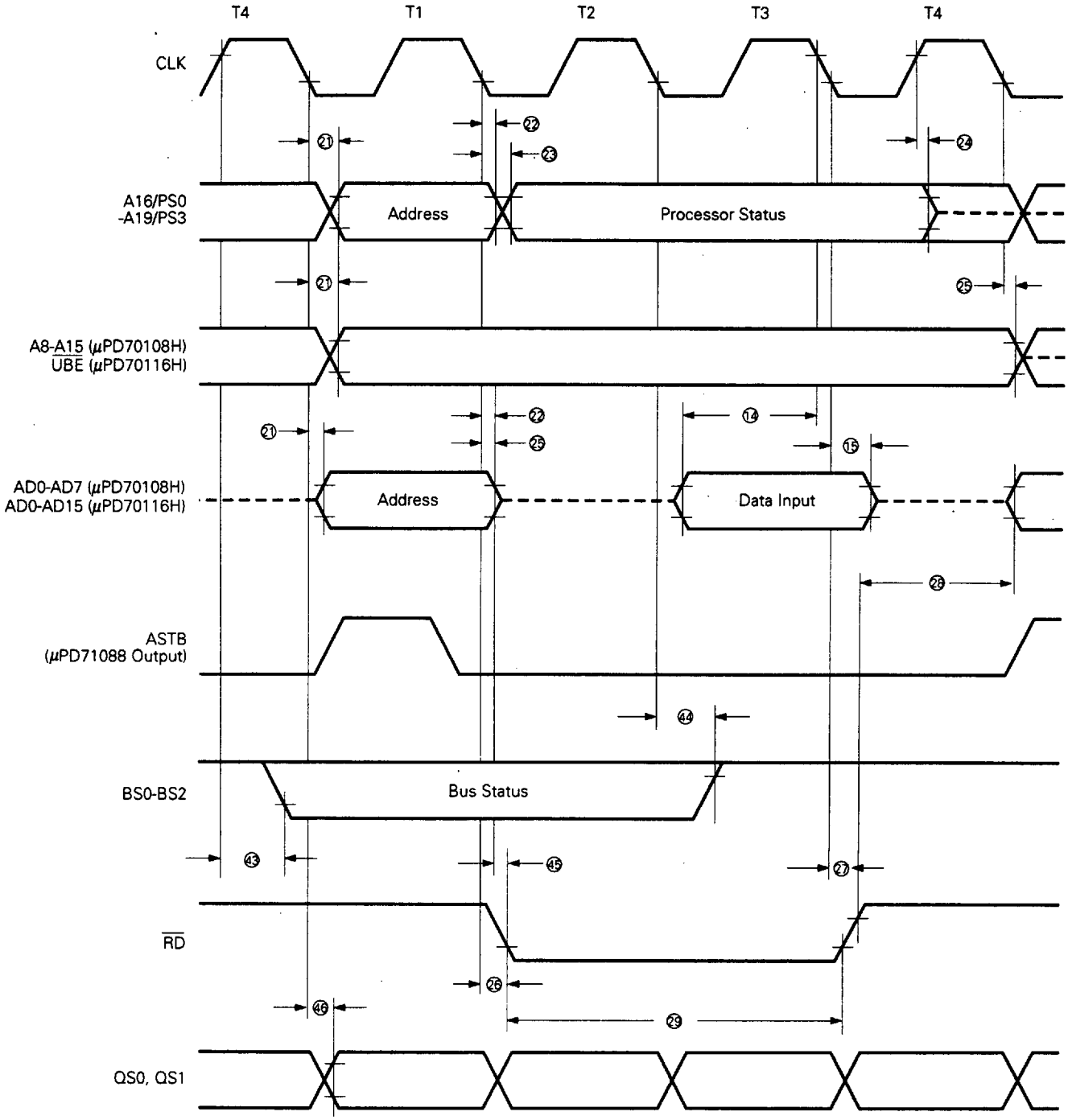
Remark Dashed line indicates high impedance.

WRITE TIMING WAVEFORMS (SMALL-SCALE MODE)



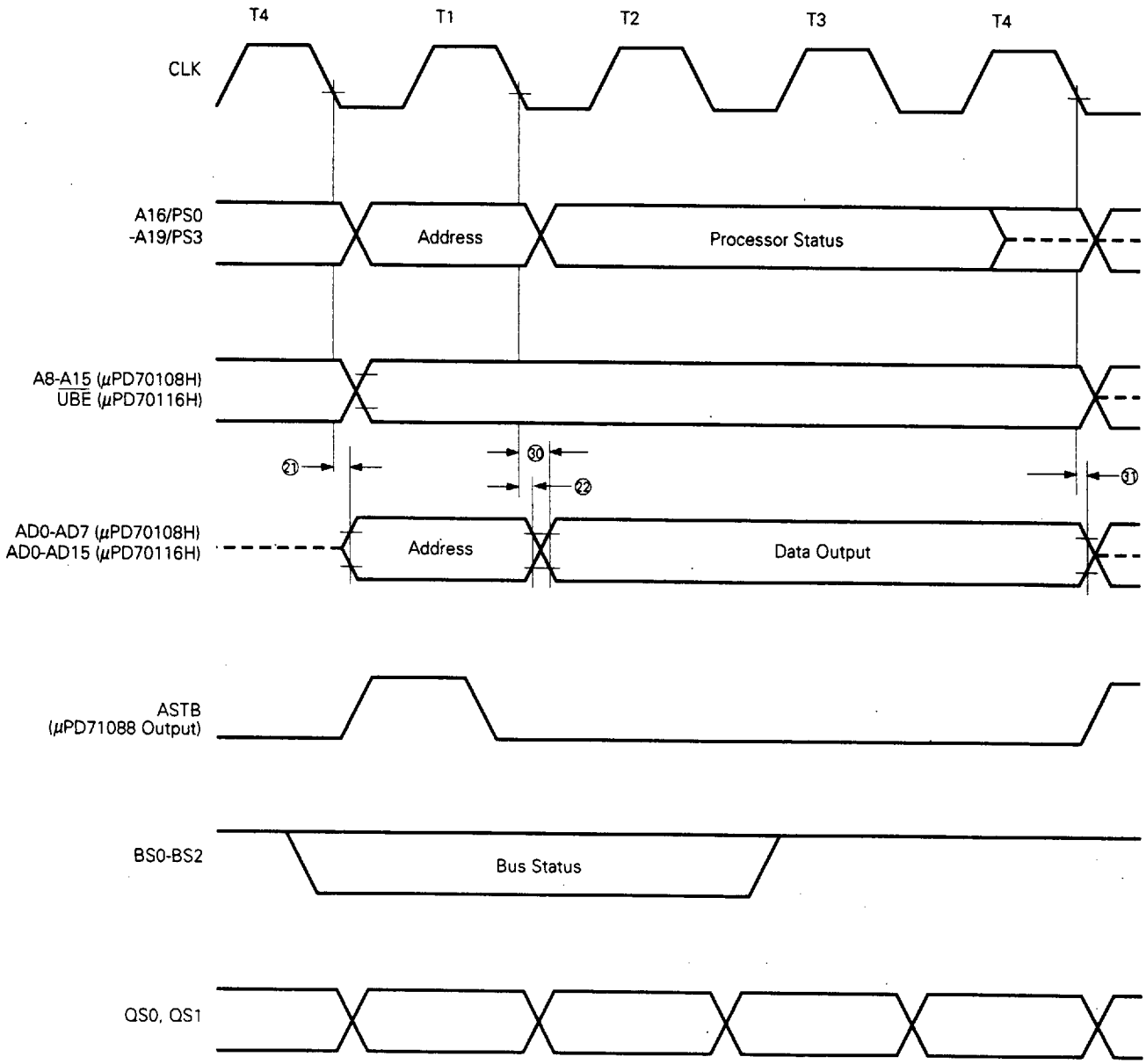
Remark Dashed line indicates high impedance.

READ TIMING WAVEFORMS (LARGE-SCALE MODE)



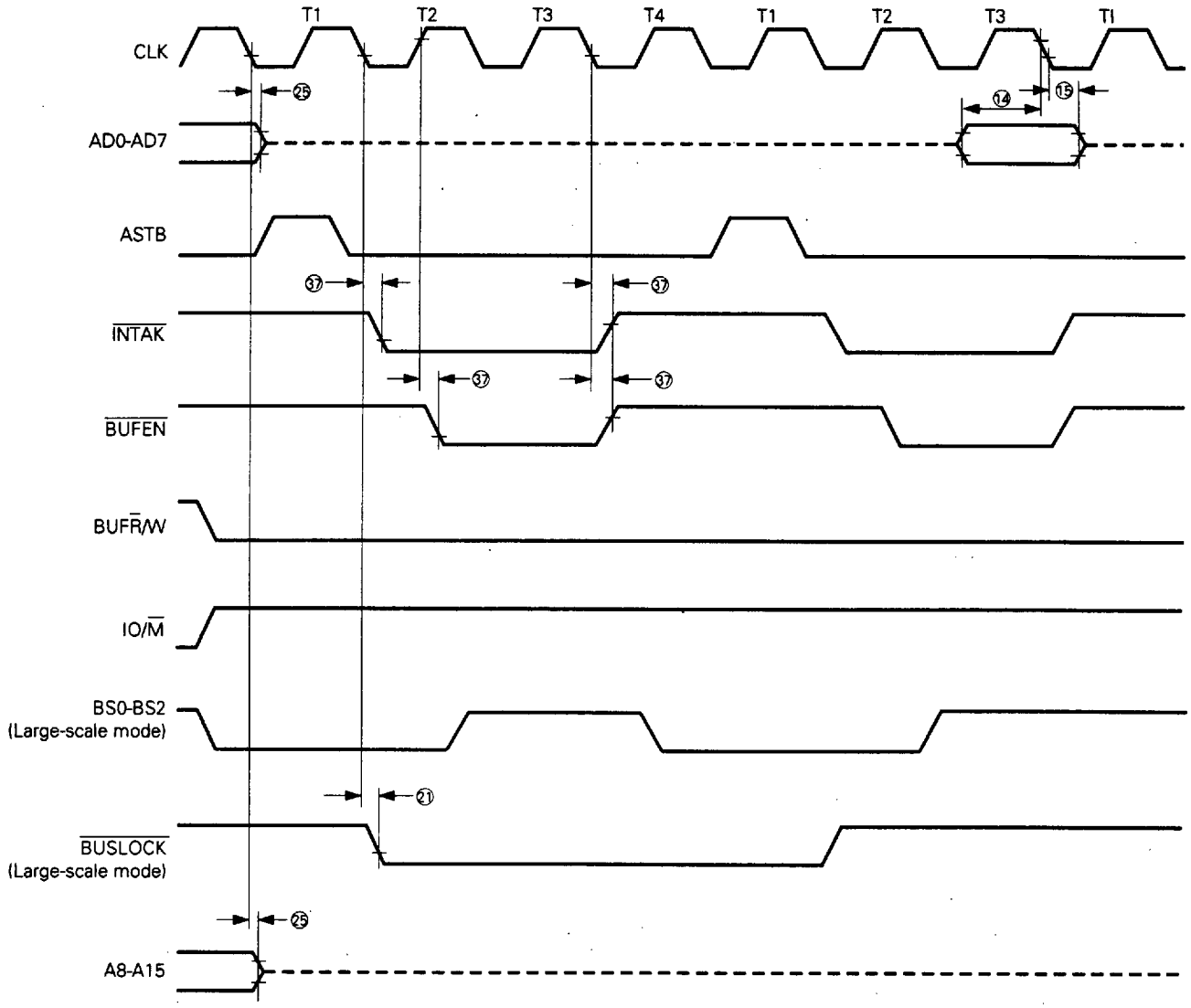
Remark Dashed line indicates high impedance.

WRITE TIMING WAVEFORMS (LARGE-SCALE MODE)



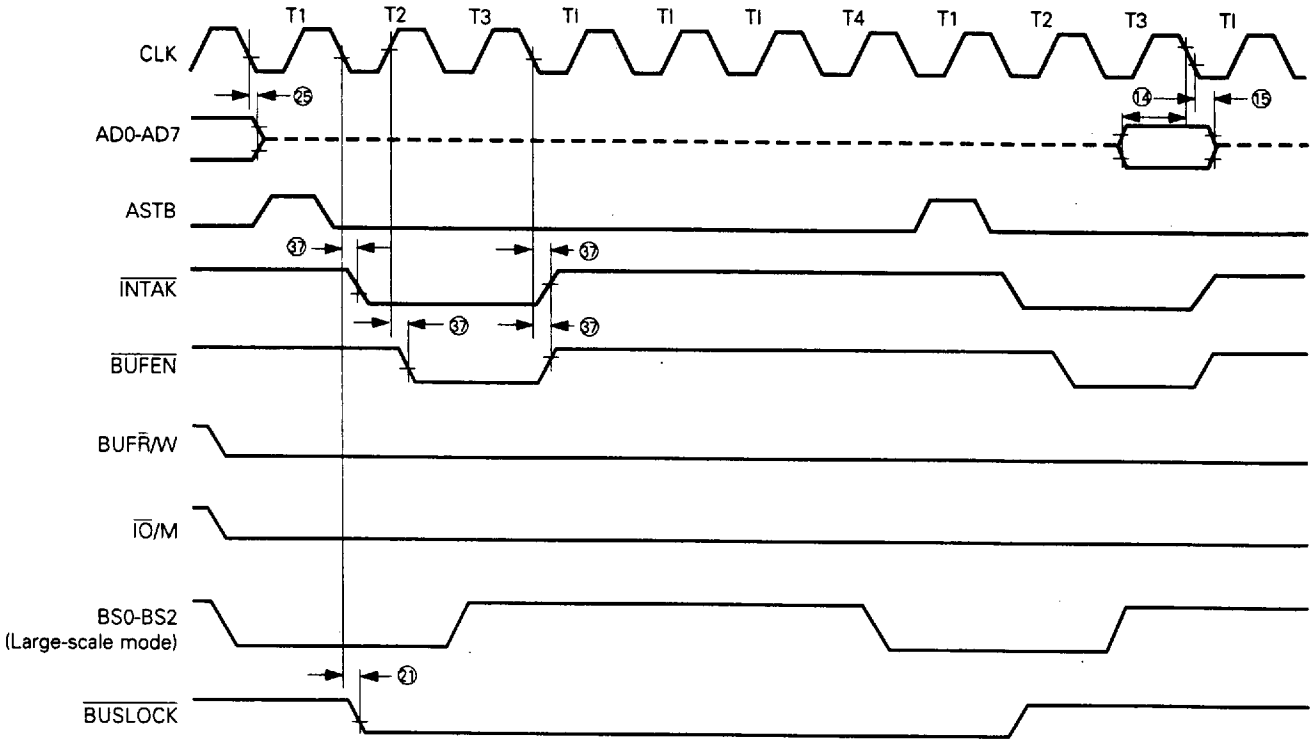
Remark Dashed line indicates high impedance.

INTERRUPT ACKNOWLEDGE TIMING WAVEFORMS (μPD70108H)



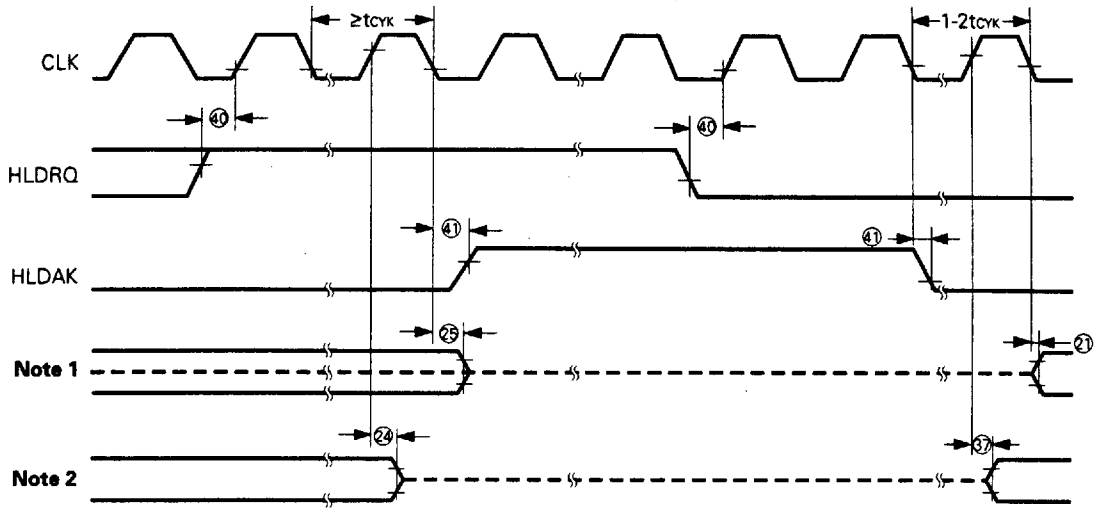
Remark Dashed line indicates high impedance.

INTERRUPT ACKNOWLEDGE TIMING WAVEFORMS (μPD70116H)



Remark Dashed line indicates high impedance.

HOLD REQUEST/ACKNOWLEDGE TIMING WAVEFORMS (SMALL-SCALE MODE)

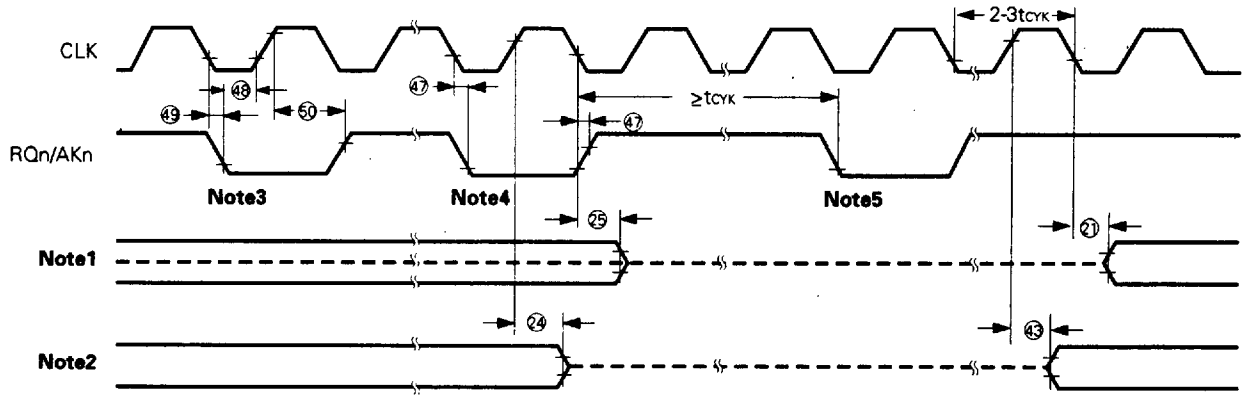


Note 1. μPD70108H: AD0 - AD7, A8 - A15
μPD70116H: AD0 - AD15

2. μPD70108H: A16/PS0 - A19/PS3, $\overline{RD}$, $\overline{WR}$, IO/$\overline{M}$, BUF$\overline{R}$/W, $\overline{BUFEN}$, LBS0
μPD70116H: A16/PS0 - A19/PS3, $\overline{RD}$, $\overline{WR}$, $\overline{IO}$/M, BUF$\overline{R}$/W, $\overline{BUFEN}$, $\overline{UBE}$

Remark Dashed line indicates high impedance.

BUS REQUEST/ACKNOWLEDGE TIMING WAVEFORMS (LARGE-SCALE MODE)



Note 1. μPD70108H: AD0 - AD7, A8 - A15
μPD70116H: AD0 - AD15

2. A16/PS0 - A19/PS3, $\overline{RD}$, BS0 - BS2, $\overline{BUSLOCK}$

3. $\overline{RQn}$ (input) : Request pulse

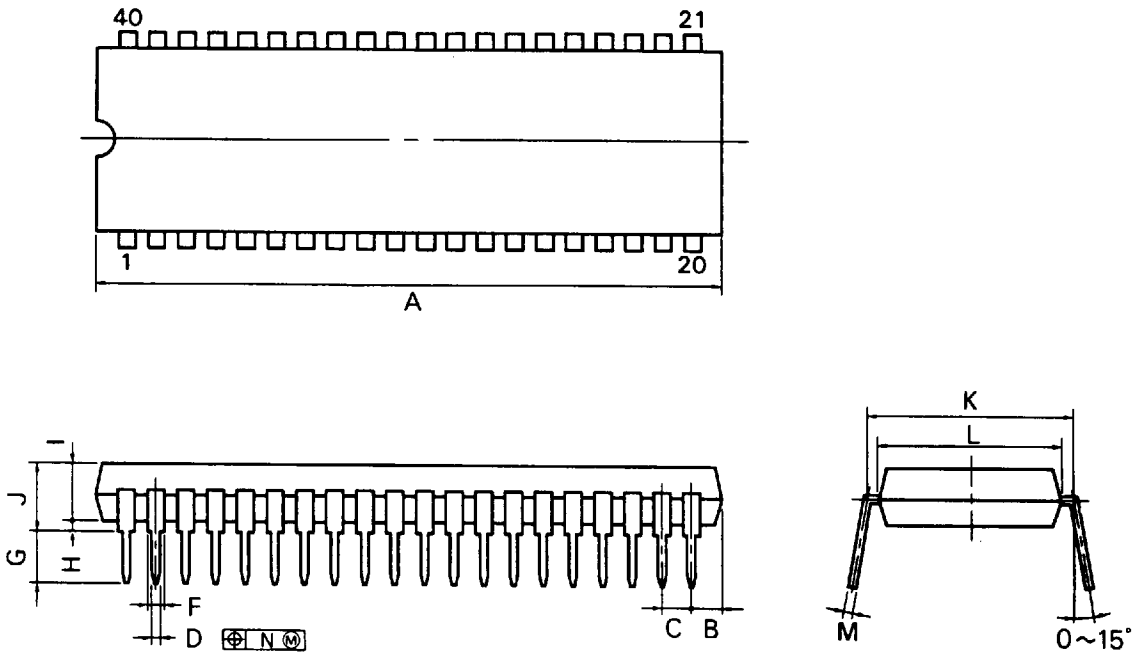
4. $\overline{AKn}$ (output) : Acknowledge pulse

5. $\overline{RQn}$ (input) : Release pulse

Remark Dashed line indicates high impedance.

10. PACKAGE DRAWINGS

40PIN PLASTIC SHRINK DIP (600 mil)



P40C-70-600A

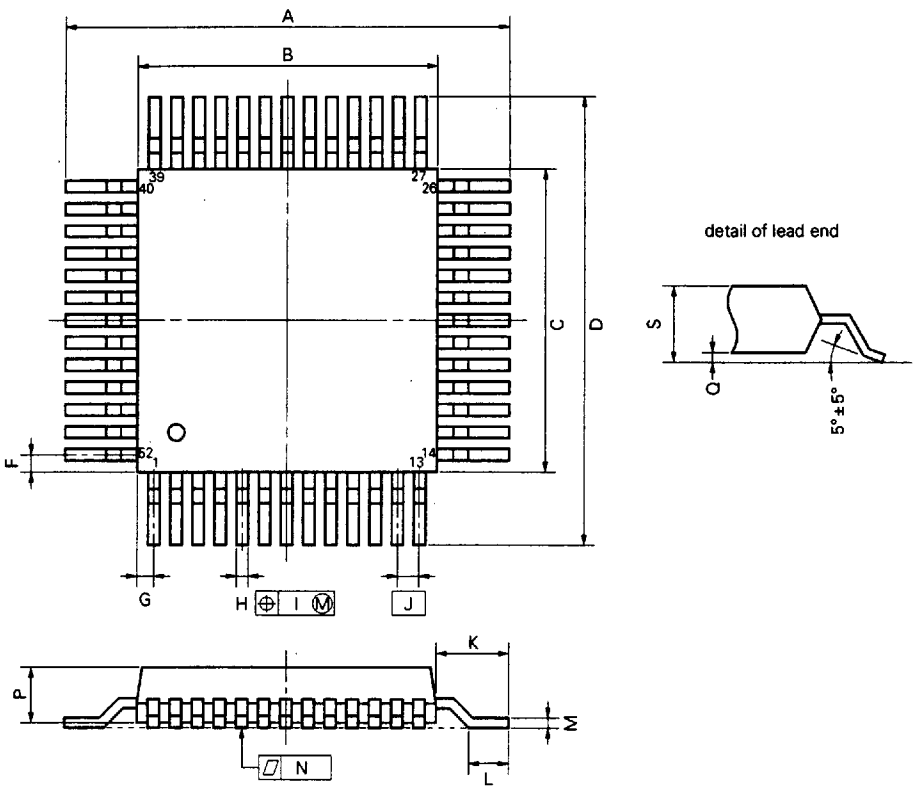
NOTES

1) Each lead centerline is located within 0.17 mm (0.007 inch) of its true position (T.P.) at maximum material condition.

2) Item "K" to center of leads when formed parallel.

ITEM	MILLIMETERS	INCHES
A	39.13 MAX.	1.541 MAX.
B	2.67 MAX.	0.106 MAX.
C	1.778 (T.P.)	0.070 (T.P.)
D	$0.50^{\pm 0.10}$	$0.020^{+0.004}_{-0.005}$
F	0.9 MIN.	0.035 MIN.
G	$3.2^{\pm 0.3}$	$0.126^{\pm 0.012}$
H	0.51 MIN.	0.020 MIN.
I	4.31 MAX.	0.170 MAX.
J	5.08 MAX.	0.200 MAX.
K	15.24 (T.P.)	0.600 (T.P.)
L	13.2	0.520
M	$0.25^{+0.10}_{-0.05}$	$0.010^{+0.004}_{-0.003}$
N	0.17	0.007

52 PIN PLASTIC QFP (□14)



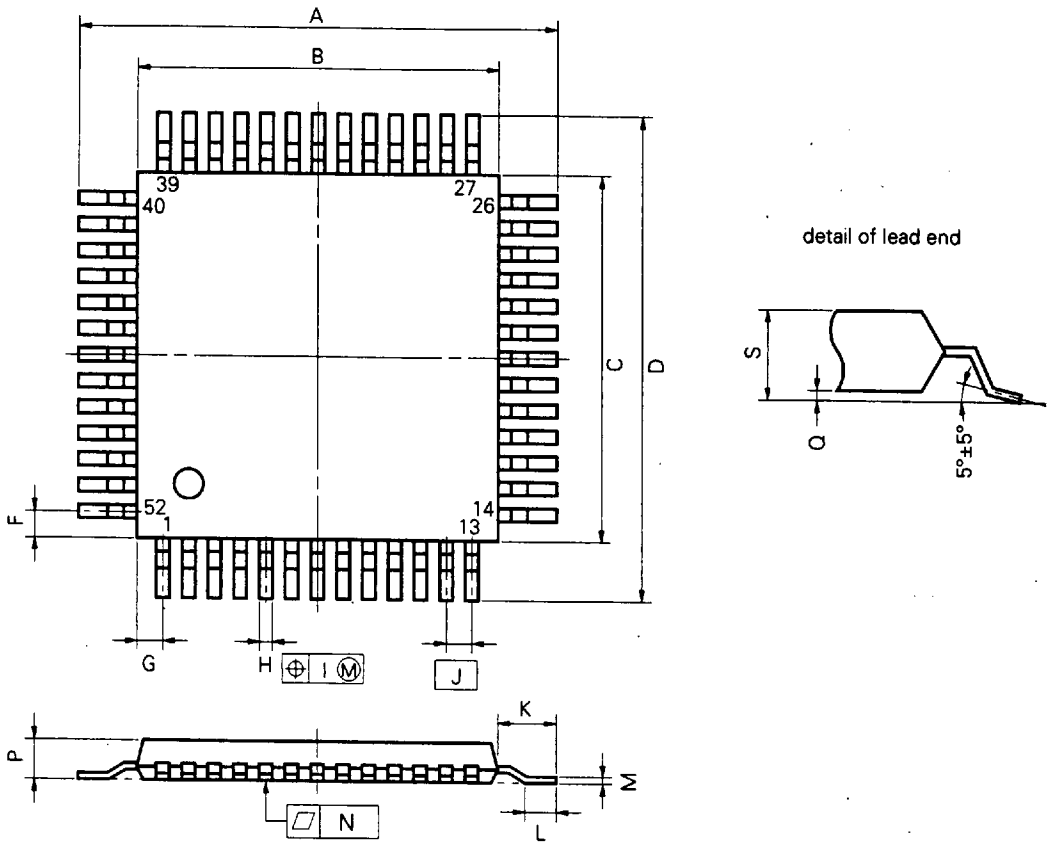
P52GC-100-3B6,3BH-2

NOTE

Each lead centerline is located within 0.20 mm (0.008 inch) of its true position (T.P.) at maximum material condition.

ITEM	MILLIMETERS	INCHES
A	17.6±0.4	0.693±0.016
B	14.0±0.2	$0.551^{+0.009}_{-0.008}$
C	14.0±0.2	$0.551^{+0.009}_{-0.008}$
D	17.6±0.4	0.693±0.016
F	1.0	0.039
G	1.0	0.039
H	0.40±0.10	$0.016^{+0.004}_{-0.005}$
I	0.20	0.008
J	1.0 (T.P.)	0.039 (T.P.)
K	1.8±0.2	$0.071^{+0.008}_{-0.009}$
L	0.8±0.2	$0.031^{+0.009}_{-0.008}$
M	$0.15^{+0.10}_{-0.05}$	$0.006^{+0.004}_{-0.003}$
N	0.10	0.004
P	2.7	0.106
Q	0.1±0.1	0.004±0.004
S	3.0 MAX.	0.119 MAX.

52 PIN PLASTIC QFP (□14)



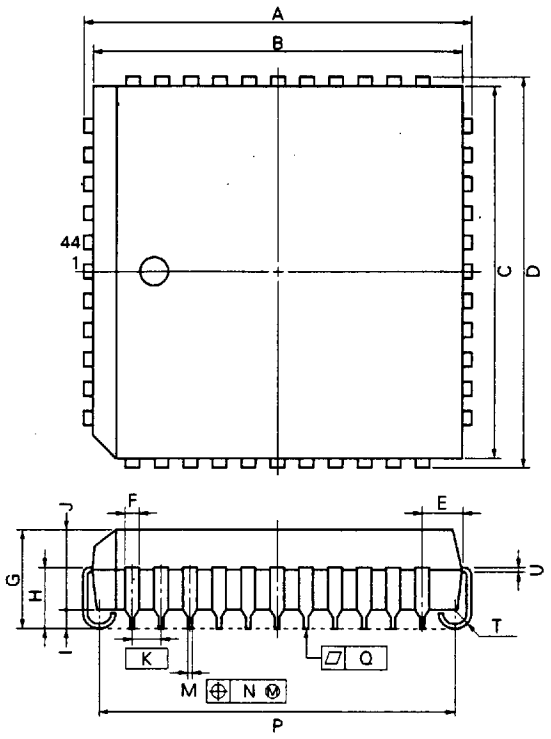
P52G-100-22-2

ITEM	MILLIMETERS	INCHES
A	18.4±0.4	$0.724^{+0.017}_{-0.016}$
B	14.0±0.2	$0.551^{+0.009}_{-0.008}$
C	14.0±0.2	$0.551^{+0.009}_{-0.008}$
D	18.4±0.4	$0.724^{+0.017}_{-0.016}$
F	1.0	0.039
G	1.0	0.039
H	0.40±0.10	$0.016^{+0.004}_{-0.005}$
I	0.20	0.008
J	1.0 (T.P.)	0.039 (T.P.)
K	2.2±0.2	$0.087^{+0.008}_{-0.009}$
L	1.0±0.2	$0.039^{+0.009}_{-0.008}$
M	$0.15^{+0.10}_{-0.05}$	$0.006^{+0.004}_{-0.003}$
N	0.15	0.006
P	1.5±0.1	0.059±0.004
Q	0.1±0.1	0.004±0.004
S	1.7 MAX.	0.067 MAX.

NOTE

Each lead centerline is located within 0.20 mm (0.008 inch) of its true position (T.P.) at maximum material condition.

44 PIN PLASTIC QFJ (□650 mil)



P44L-50A1-2

NOTE

Each lead centerline is located within 0.12 mm (0.005 inch) of its true position (T.P.) at maximum material condition.

ITEM	MILLIMETERS	INCHES
A	17.5±0.2	0.689±0.008
B	16.58	0.653
C	16.58	0.653
D	17.5±0.2	0.689±0.008
E	1.94±0.15	$0.076^{+0.007}_{-0.006}$
F	0.6	0.024
G	4.4±0.2	$0.173^{+0.009}_{-0.008}$
H	2.8±0.2	$0.110^{+0.009}_{-0.008}$
I	0.9 MIN.	0.035 MIN.
J	3.4	0.134
K	1.27 (T.P.)	0.050 (T.P.)
M	0.40±0.10	$0.016^{+0.004}_{-0.005}$
N	0.12	0.005
P	15.50±0.20	$0.610^{+0.009}_{-0.008}$
Q	0.15	0.006
T	R 0.8	R 0.031
U	$0.20^{+0.10}_{-0.05}$	$0.008^{+0.004}_{-0.002}$

11. RECOMMENDED SOLDERING CONDITIONS

For the μPD70108H and μPD70116H, soldering must be performed under the following conditions.

For details of recommended conditions for surface mounting, refer to information document "Semiconductor device mounting technology manual" (IEI-1207).

For other soldering methods, please consult with NEC sales personnel.

Table 11-1 Surface Mounting Type Soldering Conditions (1/2)

(1) μPD70108HGC-XX-3B6: 52-pin plastic QFP (□14 mm) (resin thick 2.70 mm)
μPD70116HGC-XX-3B6: 52-pin plastic QFP (□14 mm) (resin thick 2.70 mm)

(a) V, F rated products ★

Soldering Method	Soldering Conditions	Recommended Conditions Symbol
Infrared reflow	Package peak temperature: 235°C, Duration: 30 sec. max. (at 210°C or above), Number of times: 2 max. < precautions > (1) The second reflow should be started after the first reflow device temperature has returned to the ordinary state. (2) Flux washing must not be performed by the use of water after the first reflow.	IR35-00-2
VPS	Package peak temperature: 215°C, Duration: 40 sec. (at 200°C or above), Number of times: 2 max. < precautions > (1) The second reflow should be started after the first reflow device temperature has returned to the ordinary state. (2) Flux washing must not be performed by the use of water after the first reflow.	VP15-00-2
Wave soldering	Solder bath temperature: 260°C max., Duration: 10 sec. max., Number of times: 1, Preliminary heat temperature: 120°C max. (Package surface temperature)	WS60-00-1
Pin part heating	Pin temperature: 300°C max., Duration: 3 sec. max. (per device side)	——

(b) L rated products

Soldering Method	Soldering Conditions	Recommended Conditions Reference Code
Infrared Reflow	Package peak temperature: 230°C, Time: 30 sec. max. (210°C min.), Number of times: 1, Number of days[Note]: 7 days (after this, prebaking is necessary at 125°C for 10 hours)	IR30-107-1
VPS	Package peak temperature: 215°C, Time: 40 sec. max. (200°C min.), Number of times: 1, Number of days[Note]: 7 days (after this, prebaking is necessary at 125°C for 10 hours)	VP15-107-1
Wave Soldering	Solder bath temperature: 260°C max., Time: 10 sec. max., Number of times: 1, Number of days[Note]: 7 days (after this, prebaking is necessary at 125°C for 10 hours), Preheating temperature: 120°C max. (Package surface temperature)	WS60-107-1
Pin Partial Heating	Pin temperature: 300°C max., Time: 3 sec. max. (per side)	——

Note This means the number of days after unpacking the dry pack. Storage conditions are 25°C and 65% RH max.

Caution Do not use one soldering method in combination with another. (however, pin partial heating can be performed with other soldering methods).

Table 11-1 Surface Mounting Type Soldering Conditions (2/2)

(2) μPD70108HG-XX-22: 52-pin plastic QFP (□14 mm) (resin thick 1.50 mm)
μPD70116HG-XX-22: 52-pin plastic QFP (□14 mm) (resin thick 1.50 mm)

Soldering Method	Soldering Conditions	Recommended Conditions Reference Code
Infrared Reflow	Package peak temperature: 230°C, Time: 30 seconds max. (210°C min.), Number of times: 1, Number of days[Note]:1 day (after this, prebaking is necessary at 125°C for 10 hours)	IR30-101-1
VPS	Package peak temperature: 215°C, Time: 40 seconds max. (200°C min.), Number of times: 1, Number of days[Note]: 1 day (after this, prebaking is necessary at 125°C for 10 hours)	VP15-101-1
Pin Partial Heating	Pin temperature: 300°C max., Time: 3 seconds max. (per side)	——

Note This means the number of days after unpacking the dry pack. Storage conditions are 25°C and 65% RH max.

Caution Do not use one soldering method in combination with another. (however, pin partial heating can be performed with other soldering methods).

(3) μPD70108HLM-XX: 44-pin Plastic QFJ (□650 mil)
μPD70116HLM-XX: 44-pin Plastic QFJ (□650 mil)

★ (a) F rated products

Soldering Method	Soldering Conditions	Recommended Conditions Symbol
Infrared reflow	Package peak temperature: 230°C, Duration: 30 sec. max. (at 210°C or above), Number of times: 1	IR30-00-1
VPS	Package peak temperature: 215°C, Duration: 40 sec. max. (at 200°C or above), Number of times: 1	VP15-00-1
Pin part heating	Pin temperature: 300°C max., Duration: 3 sec. max. (per device side)	——

(b) V, L rated products

Soldering Method	Soldering Conditions	Recommended Conditions Reference Code
Infrared Reflow	Package peak temperature: 230°C, Time: 30 seconds max. (210°C min.), Number of times: 1, Number of days[Note]: 7 days (after this, prebaking is necessary at 125°C for 10 hours)	IR30-107-1
VPS	Package peak temperature: 215°C, Time: 40 seconds max. (200°C min.), Number of times: 1, Number of days[Note]: 7 days (after this, prebaking is necessary at 125°C for 10 hours)	VP15-107-1
Pin Partial Heating	Pin temperature: 300°C max., Time: 3 seconds max. (per side)	——

Note This means the number of days after unpacking the dry pack. Storage conditions are 25°C and 65% RH max.

Caution Do not use one soldering method in combination with another. (however, pin partial heating can be performed with other soldering methods).

Table 11-2 Insertion Type Soldering Conditions

μPD70108HCZ-XX: 40-pin plastic DIP (600 mil)
μPD70116HCZ-XX: 40-pin plastic DIP (600 mil)

Soldering Method	Soldering Conditions
Wave Soldering (Only for pin)	Solder bath temperature: 260°C max., Time: 10 seconds max.
Pin Partial Heating	Pin temperature: 300°C max., Time: 3 seconds max. (per pin)

Caution The wave soldering must be performed at the pin only. Note that the solder must not be directly contacted to the package body.