Description

The μPD70008 and μPD70008A are power saving, high performance, general purpose 8-bit microprocessor. It is a CMOS-process part with a standby mode that greatly reduces power consumption.

Features

- ☐ High performance μPD780 instruction set
- ☐ Instruction cycle:
  1 μs at 4 MHz (μPD70008, μPD70008A-4)
  0.66 μs at 6 MHz (μPD70008A-6)
- ☐ Direct addressing of up to 64 K bytes of memory
- ☐ Memory refresh function
- ☐ Interrupt functions:
  – Maskable external interrupt ($\overline{INT}$)
  – Nonmaskable external interrupt ($\overline{NMI}$)
- ☐ Low-power standby mode (HALT)
- ☐ CMOS standby mode (HALT)
- ☐ Single power supply

Ordering Information

Part Number	Package Type	Max Frequency of Operation
μPD70008C	40-pin plastic DIP	4 MHz
μPD70008AC-4	40-pin plastic DIP	4 MHz
μPD70008AC-6	40-pin plastic DIP	6 MHz
μPD70008AG-4	44-pin plastic miniflat	4 MHz
μPD70008AG-6	44-pin plastic miniflat	6 MHz
μPD70008AL-6	44-pin PLCC	6 MHz

Absolute Maximum Ratings

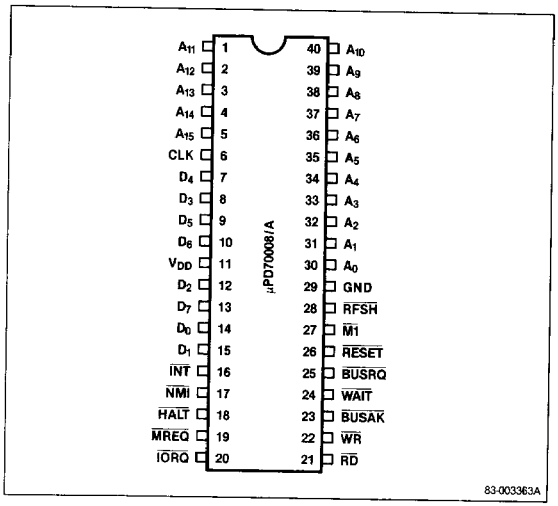
$T_A = 25°C$

Power supply voltage, V_{DD}	−0.3 V to +7 V
Input voltage, V_{IN}	−0.3 V to V_{DD} +0.3 V
Output voltage, V_O	−0.3 V to V_{DD} +0.3 V
Operating temperature, T_{OPT}	
μPD70008	−10°C to +70°C
μPD70008A	−45°C to +85°C
Storage temperature, T_{STG}	−65°C to +150°C

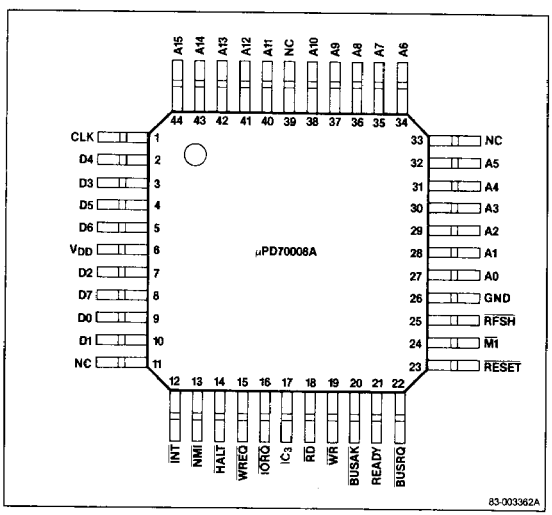
Comment: Exposing the device to stresses above those listed in Absolute Maximum Ratings could cause permanent damage. The device is not meant to be operated under conditions outside the limits described in the operational sections of the specification. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Pin Configurations

40-Pin Plastic DIP



44-Pin Plastic Miniflat



Pin Identification

40-Pin Plastic DIP

No.	Symbol	Function
1–5	A_{11}–A_{15}	Address bus, high bits, outputs
6	CLK	Clock input
7–10	D_3–D_6	Data bus, bits 3–6, inputs/outputs
11	V_{DD}	Power supply
12	D_2	Datra bus, bit 2, input/output
13	D_7	Data bus, bit 7, input/output
14, 15	D_0, D_1	Data bus, bits 0, 1, inputs/outputs
16	$\overline{INT}$	Interrupt input
17	$\overline{NMI}$	Nonmaskable interrupt input
18	$\overline{HALT}$	Halt/standby mode output
19	$\overline{MREQ}$	Memory request output
20	$\overline{IORQ}$	I/O request output
21	$\overline{RD}$	Read strobe output
22	$\overline{WR}$	Write strobe output
23	$\overline{BUSAK}$	Bus acknowledge output
24	$\overline{WAIT}$	Wait input
25	$\overline{BUSRQ}$	Bus request input
26	$\overline{RESET}$	Reset input
27	$\overline{M1}$	Machine cycle 1 output
28	$\overline{RFSH}$	Refresh request output
29	GND	Ground
30–40	A_0–A_{10}	Address bus, low bits, outputs

44-Pin Plastic Miniflat

No.	Symbol	Function
40–44	A_{11}–A_{15}	Address bus, high bits, outputs
1	CLK	Clock input
2–5	D_3–D_6	Data bus, bits 3–6, inputs/outputs
6	V_{DD}	Power supply
7	D_2	Datra bus, bit 2, input/output
8	D_7	Data bus, bit 7, input/output
9–10	D_0, D_1	Data bus, bits 0, 1, inputs/outputs
12	$\overline{INT}$	Interrupt input
13	$\overline{NMI}$	Nonmaskable interrupt input
14	$\overline{HALT}$	Halt/standby mode output
15	$\overline{MREQ}$	Memory request output
16	$\overline{IORQ}$	I/O request output
18	$\overline{RD}$	Read strobe output
19	$\overline{WR}$	Write strobe output
20	$\overline{BUSAK}$	Bus acknowledge output
21	$\overline{WAIT}$	Wait input
22	$\overline{BUSRQ}$	Bus request input
23	$\overline{RESET}$	Reset input
24	$\overline{M1}$	Machine cycle 1 output
25	$\overline{RFSH}$	Refresh request output
26	GND	Ground
28–32, 34–38	A_0–A_{10}	Address bus, low bits, outputs
17	IC	Internally connected
11, 33, 39	NC	Not connected

Pin Functions

A_{15}–A_0 (Address Bus)

These three-state output pins form a 16-bit address bus for addressing memory or peripheral devices. The address bus enters the high impedance state when bus acknowledge is active. In the standby mode, these pins output high- or low-level signals.

D_7–D_0 (Data Bus)

These three-state pins form an 8-bit bidirectional data bus. On this bus data is transferred between the μPD70008/A and memory or peripheral devices. This bus enters the high impedance state when bus acknowledge is active. In the standby mode, these pins are high-level.

$\overline{INT}$ (Interrupt)

This pin is an active-low interrupt input which can be masked by software. $\overline{NMI}$ has a lower priority than $\overline{NMI}$ and $\overline{BUSRQ}$. $\overline{INT}$ releases the standby mode.

$\overline{NMI}$ (Nonmaskable Interrupt)

This pin inputs an interrupt which is not maskable by software. $\overline{NMI}$ is active-low in the μPD70008, and is falling edge triggered in the μPD70008A. $\overline{NMI}$ has a higher priority than $\overline{INT}$, but a lower priority than $\overline{BUSRQ}$ and $\overline{RESET}$. $\overline{NMI}$ releases the standby mode.

$\overline{MREQ}$ (Memory Request)

This three-state pin is an active-low output. The μPD70008/A asserts $\overline{MREQ}$ to indicate that the information on the address bus is a memory address. This pin enters the high impedance state when bus acknowledge is active. $\overline{MREQ}$ is inactive (high) in the standby mode.

Pin Functions (cont)

$\overline{IORQ}$ (I/O Request)

This three-state pin is an active-low output. The μPD70008/A asserts $\overline{IORQ}$ to indicate that the information on the address bus is a peripheral device address. $\overline{IORQ}$ is also asserted during a maskable interrupt service to request the interrupting device to output its interrupt vector to the data bus. This pin enters the high impedance state when bus acknowledge is active. $\overline{IORQ}$ is inactive (high) in the standby mode.

$\overline{RD}$ (Read Strobe)

This three-state active-low output provides a read strobe for the memory and peripheral devices. The pin enters the high impedance state when the bus acknowledge is active. $\overline{RD}$ is inactive (high) in the standby mode.

$\overline{WR}$ (Write Strobe)

This three-state active-low output provides a write strobe for the memory and peripheral devices. This pin enters the high impedance state when bus acknowledge is active. $\overline{WR}$ is inactive (high) in the standby mode.

$\overline{BUSRQ}$ (Bus Request)

This is an active-low input. Peripheral devices assert $\overline{BUSRQ}$ to request the μPD70008/A to release control of the address bus (A_{15}-A_0), data bus (D_7-D_0) and control bus ($\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$ and $\overline{WR}$) and assert bus acknowledge. $\overline{BUSRQ}$ has a higher priority than either $\overline{INT}$ or $\overline{NMI}$, but is lower in priority than $\overline{RESET}$. $\overline{BUSRQ}$ will temporarily suspend the standby mode. The μPD70008/A leaves standby mode when $\overline{BUSRQ}$ is asserted, but returns to the standby mode when $\overline{BUSRQ}$ is released.

$\overline{BUSAK}$ (Bus Acknowledge)

This active-low output indicates that the data bus (D_7-D_0), address bus (A_{15}-A_0), and control bus ($\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$ and $\overline{WR}$) have entered the high impedance state. This releases the buses from CPU control and makes them available to the peripheral devices for data exchange. This state cannot be released by $\overline{NMI}$ or $\overline{INT}$, but responds only to $\overline{RESET}$ or the release of $\overline{BUSRQ}$.

$\overline{HALT}$ (Halt/Standby Mode)

This active-low output is asserted after the halt command has been executed and indicates that the μPD70008/A has entered the standby mode.

$\overline{WAIT}$ (Wait)

This pin is an active-low input. Memory and peripheral devices assert this signal to increase read or write access time. When $\overline{WAIT}$ is asserted, the μPD70008/A inserts wait states (TW) into the machine cycle until $\overline{WAIT}$ is released.

$\overline{RESET}$ (Reset)

This active-low input is used to reset the μPD70008/A. The standby mode is released on Reset. Reset has the highest priority.

$$\overline{INT1} < \overline{NMI} < \overline{BUSRQ} < \overline{RESET}$$

$\overline{RFSH}$ (Refresh Request)

This pin is an active-low output. The μPD70008/A asserts $\overline{RFSH}$ to trigger the external memory refresh operation. When $\overline{RFSH}$ is low, the lower seven bits of the address bus (A_6-A_0) are a refresh address. This pin is inactive (high) in the standby mode.

$\overline{M1}$ (Machine Cycle 1)

This pin is an active-low output. When $\overline{M1}$ is asserted, it indicates that the μPD70008/A is in the opcode fetch cycle, $\overline{M1}$.

CLK (Clock)

This pin is the system clock input.

V_{DD} (Power Supply)

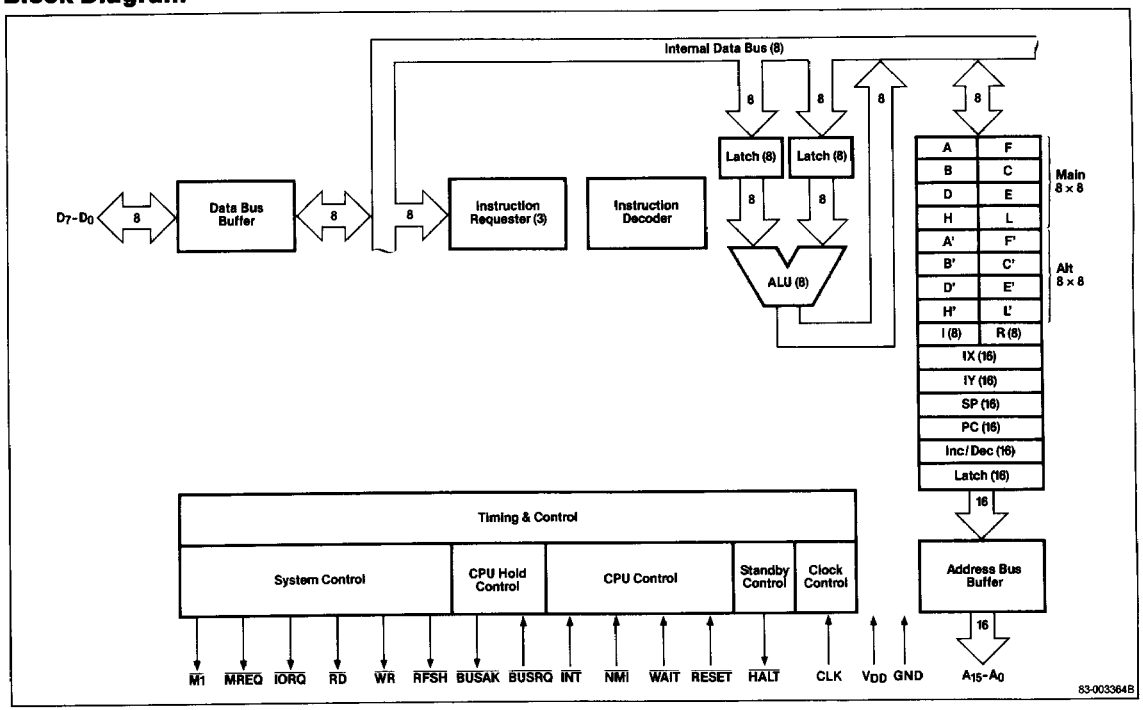
This pin is the +5 V power supply input.

GND (Ground)

This pin is the ground pin.

3

Block Diagram



DC Characteristics

μPD70008: $T_A = -10°C$ to $+70°C$, μPD70008A: $T_A = -40°C$ to $+85°C$, $V_{DD} = +5V \pm 10\%$

Parameter	Symbol	Limits Min	Typ	Max	Unit	Test Conditions
Input voltage high	V_{IH1}	2.2		V_{DD}	V	Except CLK, $\overline{RESET}$
	V_{IH2}	$V_{DD}-0.6$		$V_{DD}+0.3$	V	CLK, $\overline{RESET}$
Input voltage low	V_{IL1}	−0.3		0.8	V	Except CLK, $\overline{RESET}$
	V_{IL2}	−0.3		0.45	V	CLK, $\overline{RESET}$
Output voltage high	V_{OH}	2.4			V	$I_{OH} = -400\,\mu A$
Output voltage low	V_{OL}			0.4	V	$I_{OL} = 2.5\,mA$
Input leakage current high	I_{LIH}			10	μA	$V_I = V_{DD}$
Input leakage current low	I_{LIL}			−10	μA	$V_{IN} = 0\,V$
Output leakage current high	I_{LOH}			10	μA	$V_O = V_{DD}$
Output leakage current low	I_{LOL}			−10	μA	$V_O = 0\,V$

Parameter	Symbol	Limits Min	Typ	Max	Unit	Test Conditions
Supply current (Note 1)						
μPD70008	I_{DD1}		10	30	mA	$t_{CYK} = 0.25\,\mu s$
	I_{DD2}		500		μA	$t_{CYK} = 0.25\,\mu s$
μPD70008A-4	I_{DD1}		9	20	mA	$t_{CYK} = 0.25\,\mu s$
	I_{DD2}		80	240	μA	$t_{CYK} = 0.25\,\mu s$
μPD70008A-6	I_{DD1}		14	30	mA	$t_{CYK} = 0.165\,\mu s$
	I_{DD2}		120	360	μA	$t_{CYK} = 0.165\,\mu s$

Note:
(1) I_{DD1} is normal operating current.
I_{DD2} is standby mode current.

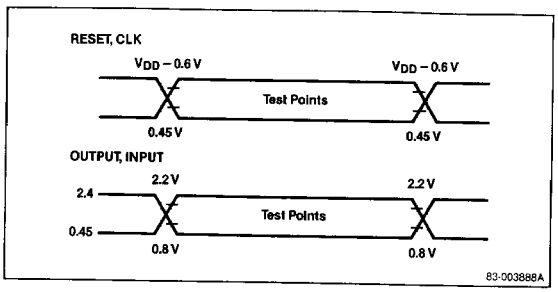
Capacitance

$T_A = 25°C$, $f_C = 1\,MHz$

Parameter	Symbol	Limits Min	Typ	Max	Unit	Test Conditions
CLK input capacitance	C_K			35	pF	(Note 1)
Input capacitance	C_I			5	pF	(Note 1)
Output capacitance	C_O			10	pF	(Note 1)
I/O capacitance	C_{IO}			10	pF	(Note 1)

Note:
(1) All unmeasured pins returned to 0 V.

AC Test Points



AC Characteristics

μPD70008: $T_A = -10°C$ to $+70°C$, μPD70008A: $T_A = -40°C$ to $+85°C$, $V_{CC} = 5\,V \pm 10\%$

Signal	Parameter	Symbol	Limits μPD70008/A-4 Min	Max	μPD70008A-6 Min	Max	Unit	Test Conditions
CLK	Clock period	t_{CYK}	0.25	(Note 1)	0.165	(Note 8)	μs	
	Clock pulse width high	t_{KKH}	0.11	200	0.065	200	μs	
	Clock pulse width low	t_{KKL}	110	2000	65	2000	ns	
	Clock pulse rise and fall time	t_{KR}, t_{KF}		30		20	ns	
A_{15}–A_0	Address output delay	t_{DKA}		110		90	ns	$C_L = 100\,pF$
	Address delay to float	t_{FKA}		90		80	ns	$C_L = 100\,pF$
	Address stable prior to $\overline{MREQ}$, memory cycle	t_{SAM}	(Note 2)		(Note 9)		ns	$C_L = 100\,pF$
	Address stable prior to $\overline{IORQ}$ in I/O cycle	t_{SAI}	$t_{CYK}-70$		(Note 10)		ns	$C_L = 100\,pF$
	Address stable from $\overline{RD}$, $\overline{WR}$, $\overline{IORQ}$, $\overline{MREQ}$	t_{HRA}	(Note 3)		(Note 11)		ns	$C_L = 100\,pF$
	Address stable from $\overline{RD}$, $\overline{WR}$ during float	t_{FCA}	(Note 4)		(Note 12)		ns	$C_L = 100\,pF$
D_7–D_0	Data output delay	t_{DKD}		180		130	ns	$C_L = 100\,pF$
	Delay to float during write cycle	t_{FKD}		90		80	ns	$C_L = 100\,pF$
	Data setup time to CLK during $\overline{M1}$ cycle	t_{SDKR}	35		30		ns	$C_L = 100\,pF$
	Data setup time to CLK during M2 to M5 cycles	t_{SDKF}	50		40		ns	$C_L = 100\,pF$
	Data stable prior to $\overline{WR}$ (memory cycle)	t_{SMDW}	$t_{CYK}-170$		(Note 13)		ns	$C_L = 100\,pF$
	Data stable prior to $\overline{WR}$ (I/O cycle)	t_{SIDW}	$t_{KKL}+t_{KR}-170$		(Note 14)		ns	$C_L = 100\,pF$
	Data stable from $\overline{WR}$	t_{FCD}	(Note 5)		(Note 15)		ns	$C_L = 100\,pF$
$\overline{WR}$	$\overline{WR}$ delay from CLK ↑ to $\overline{WR}$ low	t_{DKRWL}		65		60	ns	
	$\overline{WR}$ delay from CLK ↓ to $\overline{WR}$ low	t_{DKFWL}		80		70	ns	
	$\overline{WR}$ delay from CLK ↓ to $\overline{WR}$ high	t_{DKFWH}		80		70	ns	
	$\overline{WR}$ low pulse width	t_{WWL}	$t_{CYK}-30$		(Note 18)		ns	
$\overline{M1}$	$\overline{M1}$ delay from CLK ↑ to $\overline{M1}$ low	t_{DKM1L}		100		80	ns	$C_L = 100\,pF$
	$\overline{M1}$ delay from CLK ↑ to $\overline{M1}$ high	t_{DKM1H}		100		80	ns	$C_L = 100\,pF$

AC Characteristics (cont)

μPD70008: $T_A = -10°C$ to $+70°C$, μPD70008A: $T_A = -40°C$ to $+85°C$, $V_{CC} = 5V \pm 10\%$

Signal	Parameter	Symbol	Limits μPD70008/A-4 Min	Limits μPD70008/A-4 Max	Limits μPD70008A-6 Min	Limits μPD70008A-6 Max	Unit	Test Conditions
$\overline{RFSH}$	$\overline{RFSH}$ delay from CLK ↑ to $\overline{RFSH}$ low	t_{DKRFL}		130		110	ns	$C_L = 100$ pF
	$\overline{RFSH}$ delay from CLK ↑ to $\overline{RFSH}$ high	t_{DKRFH}		120		100	ns	$C_L = 100$ pF
$\overline{WAIT}$	$\overline{WAIT}$ setup time to CLK ↓	t_{SWTK}	70		60		ns	
$\overline{HALT}$	$\overline{HALT}$ delay from CLK ↓	t_{DKHT}		300		260	ns	$C_L = 100$ pF
$\overline{INT}$	$\overline{INT}$ setup time to CLK ↑	t_{SITK}	80		70		ns	
$\overline{NMI}$	$\overline{NMI}$ low pulse width	t_{NNL}	80		70		ns	
$\overline{BUSRQ}$	$\overline{BUSRQ}$ setup time to CLK ↓	t_{SBQK}	50		50		ns	
$\overline{BUSAK}$	$\overline{BUSAK}$ delay from CLK ↑ to $\overline{BUSAK}$ low	t_{DKRBA}		100		90	ns	$C_L = 100$ pF
	$\overline{BUSAK}$ delay from CLK ↓ to $\overline{BUSAK}$ high	t_{DKFBA}		100		90	ns	$C_L = 100$ pF
$\overline{RESET}$	$\overline{RESET}$ setup to CLK	t_{SRSK}	60		60		ns	
Other	Delay to float ($\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, $\overline{WR}$)	t_{FKC}		80		70	ns	
	$\overline{M1}$ stable prior to $\overline{IORQ}$ (interrupt acknowledge)	t_{SM1I}	(Note 7)		(Note 19)		ns	
	Hold time for setup time	t_H	0		0		ns	
$\overline{MREQ}$	$\overline{MREQ}$ delay from CLK ↓ to $\overline{MREQ}$ low	t_{DKFML}		85		70	ns	$C_L = 100$ pF
	$\overline{MREQ}$ delay from CLK ↑ to $\overline{MREQ}$ high	t_{DKRMH}		85		70	ns	$C_L = 100$ pF
	$\overline{MREQ}$ delay from CLK ↓ $\overline{MREQ}$ high	t_{DKFMH}		85		70	ns	$C_L = 100$ pF
	Pulse width $\overline{MREQ}$ low	t_{MML}	$t_{CYK} - 30$		(Note 16)		ns	$C_L = 100$ pF
	Pulse width $\overline{MREQ}$ high	t_{MMH}	(Note 6)		(Note 17)		ns	$C_L = 100$ pF
$\overline{IORQ}$	$\overline{IORQ}$ delay from CLK ↑ to $\overline{IORQ}$ low	t_{DKRIL}		75		65	ns	$C_L = 100$ pF
	$\overline{IORQ}$ delay from CLK ↓ to $\overline{IORQ}$ low	t_{DKFIL}		85		70	ns	$C_L = 100$ pF
	$\overline{IORQ}$ delay from CLK ↑ to $\overline{IORQ}$ high	t_{DKRIH}		85		70	ns	$C_L = 100$ pF
	$\overline{IORQ}$ delay from CLK ↓ to $\overline{IORQ}$ high	t_{DKFIH}		85		70	ns	$C_L = 100$ pF
$\overline{RD}$	$\overline{RD}$ delay from CLK ↑ to $\overline{RD}$ low	t_{DKRRL}		85		70	ns	$C_L = 100$ pF
	$\overline{RD}$ delay from CLK ↓ to $\overline{RD}$ low	t_{DKRFL}		95		80	ns	$C_L = 100$ pF
	$\overline{RD}$ delay from CLK ↑ to $\overline{RD}$ high	t_{DKRRH}		85		70	ns	$C_L = 100$ pF
	$\overline{RD}$ delay from CLK ↓ to $\overline{RD}$ high	t_{DKFRH}		85		70	ns	$C_L = 100$ pF

Note:

(1) $t_{CYK} = t_{KKH} + t_{KKL} + t_{KR} + t_{KF}$

(2) $t_{SAM} = t_{KKH} + t_{KF} - 65$

(3) $t_{HRA} = t_{KKL} + t_{HR} - 50$

(4) $t_{FCA} = t_{KKL} + t_{KR} - 45$

(5) $t_{FCD} = t_{KKL} + t_{KR} - 70$

(6) $t_{MMH} = t_{KKH} + t_{KF} - 20$

(7) $t_{SM1I} = 2t_{CYK} + t_{KKH} + t_{KF} - 65$

(8) $t_{CYK} = t_{KKH} + t_{KKL} + t_{KR} + t_{KF}$

(9) $t_{SAM} = t_{KKH} + t_{KF} - 50$

(10) $t_{SAI} = t_{CYK} - 55$

(11) $t_{HRA} = t_{KKL} + t_{KR} - 50$

(12) $t_{FCA} = t_{KKL} + t_{KR} - 40$

(13) $t_{SMDW} = t_{CYK} - 140$

(14) $t_{SIDW} = t_{KKL} + t_{KR} - 140$

(15) $t_{FCD} = t_{KKL} + t_{KR} - 55$

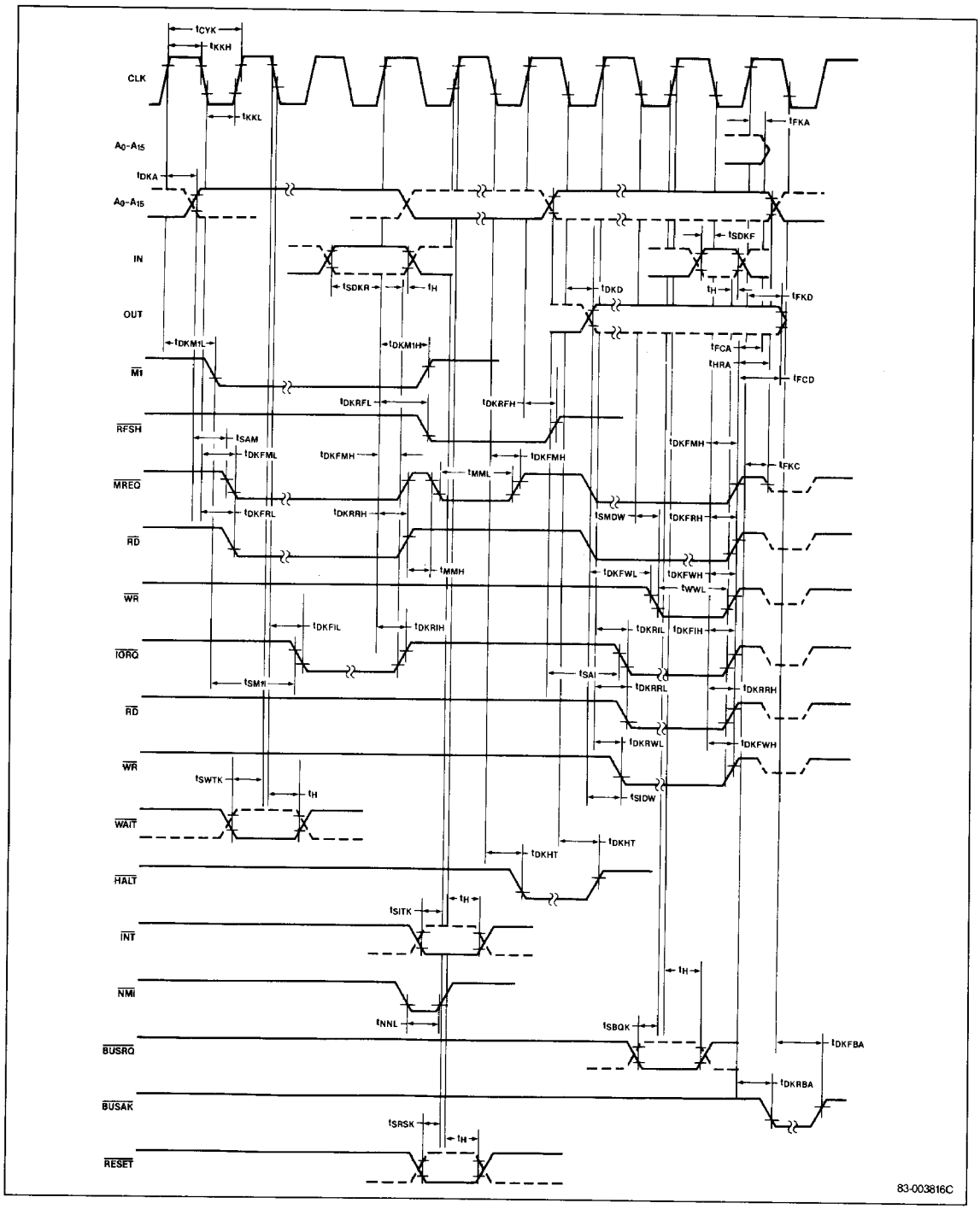
(16) $t_{MML} = t_{CYK} - 30$

(17) $t_{MMH} = t_{KKH} + t_{KF} - 20$

(18) $t_{WWL} = t_{CYK} - 30$

(19) $t_{SM1I} = 2t_{CYK} + t_{KKH} + t_{KF} - 50$

Timing Waveforms



Register Configuration

Program Counter (PC)

The 16-bit program counter contains the address of the next instruction to be fetched and executed. It is set to 0000H at reset.

Stack Pointer (SP)

The 16-bit stack pointer stores the first address of the portion of main memory used as a LIFO stack. SP is decremented when a CALL or PUSH is executed, or when an interrupt occurs. It is incremented when a RET, POP, or interrupt return is executed.

Index Registers (IX, IY)

These two 16-bit registers are used to perform indexed addressing.

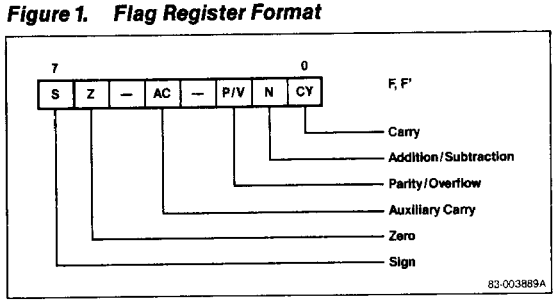
Accumulators (A, A')

The μPD70008/A has two 8-bit accumulators: the main accumulator (A) which is used to perform arithmetic and logic operations, and an alternate accumulator (A'). The contents of the main and alternate accumulators can be exchanged using the (EX) instruction. The alternate accumulator can be used for background operation, or to save the data in the main accumulator when an interrupt is processed.

Flag Registers (F, F')

The μPD70008/A has two 8-bit flag registers: main (F) and alternate (F') of the format shown in figure 1. The main flag register (F) has the status flags resulting from normal operation. The contents of the main and alternate registers can be exchanged using the exchange (EX) instruction. The alternate (F') register can be used for background operation, or to save the state of the main flag register when an interrupt is processed.

Figure 1. Flag Register Format



General Purpose Registers

The μPD70008/A has twelve 8-bit general purpose registers: six main registers (B, C, D, E, H, and L) and six alternate registers (B', C', D', E', H', and L'). Each register can be used individually as an 8-bit register, or can be used in pairs as 16-bit registers (BC, B'C', DE, D'E', HL, and H'L').

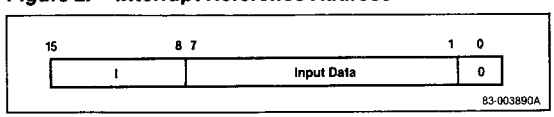
The main registers are used when instructions are executed normally. The contents of the main and alternate registers are exchanged using the EX instruction. The alternate registers may be used for background operation or to save the contents of the main registers when an interrupt is processed.

Interrupt Page Address Register (I)

This 8-bit register is used to generate addresses in maskable interrupt mode 2. See figure 2. These addresses are used with externally input data to reference an interrupt start address table.

This register is cleared to 00H at reset.

Figure 2. Interrupt Reference Address



Memory Refresh Register (R)

This 7-bit register retains the refresh address for the external dynamic memory. The contents of this register are automatically incremented in each opcode fetch (M1) cycle. The contents of this register are output on the lower 7 bits of the address bus (A_6–A_0).

This register is cleared to 00H at reset.

Timing

This section describes read and write timing for memory and I/O devices in connection with CPU operation timing. A single clock cycle (from one leading edge to the next) is defined as one timing state. The nth state is represented as Tn. A single instruction consists of two to six machine cycles. A single machine cycle requires three to six timing states. The nth machine cycle is represented as Mn.

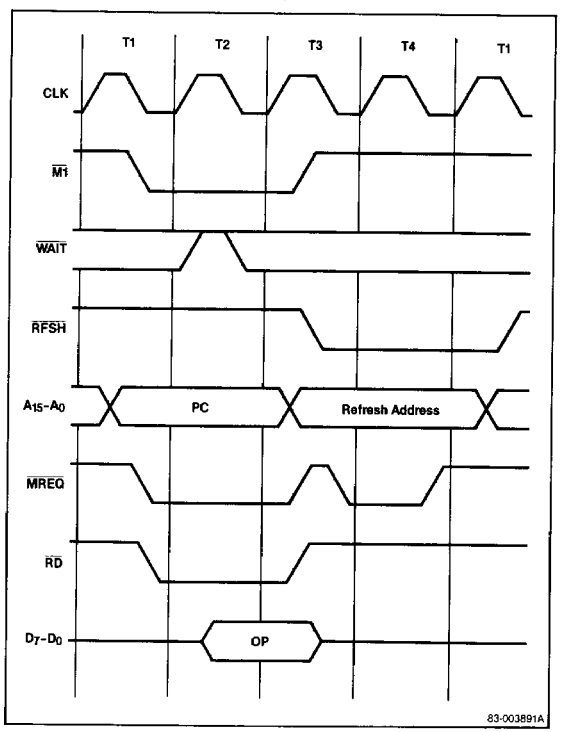
Table 1 lists the number of states normally required by each cycle.

Table 1. Timing States per Cycle

Cycle	Number of States per Machine Cycle
Opcode fetch	4
Memory read	3
Memory write	3
I/O read	4
I/O write	4

The four states for I/O read and write include a single wait state (TW). The μPD70008/A inserts one wait state in every I/O read or write. Slower external devices may assert the $\overline{WAIT}$ signal to request longer read and write access times. This time will be added to the original number of clock states. The $\overline{WAIT}$ signal is monitored on the trailing edge of clock state T2. If $\overline{WAIT}$ is asserted, a wait state (TW) is generated. The μPD70008/A continues to monitor $\overline{WAIT}$ on the clock's trailing edge, and supplies additional wait states a long as that signal is asserted. When $\overline{WAIT}$ is released the μPD70008/A proceeds to the T3 state.

Figure 3. Opcode Fetch Cycle



Opcode Fetch Cycle

The first machine cycle of each instruction, M1, is the opcode fetch cycle. See figure 3. The opcode is fetched from memory during the first half of this cycle, and the dynamic memory is refreshed during the latter half.

The memory outputs the opcode to the data bus when $\overline{MREQ}$, $\overline{RD}$, or $\overline{M1}$ is asserted. It is then read into the CPU at the leading edge of clock state T3.

The CPU outputs a refresh address onto A_6-A_0 during T3. It is applied to the dynamic memory when $\overline{RFSH}$ or $\overline{MREQ}$ are asserted.

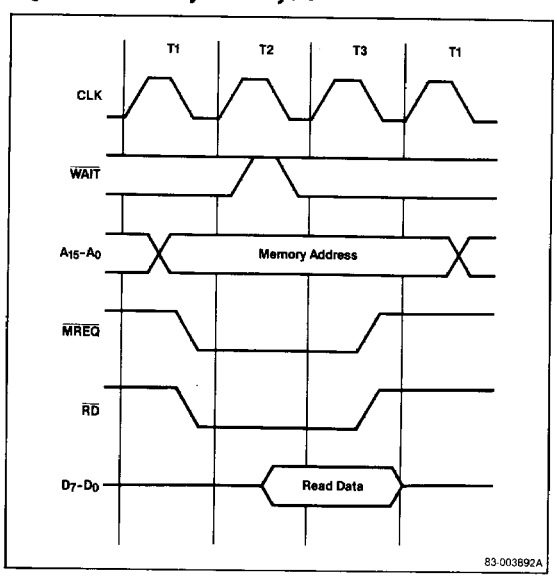
Memory Read Cycle

The memory contents are read out to the data bus when $\overline{MREQ}$ or $\overline{RD}$ is asserted. The μPD70008/A reads data from the data bus on the trailing edge of T3. See figure 4.

Memory Write Cycle

Write data is output to the data bus between the last half of state T1 of the current machine cycle and the first half of state T1 of the next cycle. It is written to memory when $\overline{WR}$ or $\overline{MREQ}$ is asserted. See figure 5.

Figure 4. Memory Read Cycle



3

Figure 5. Memory Write Cycle

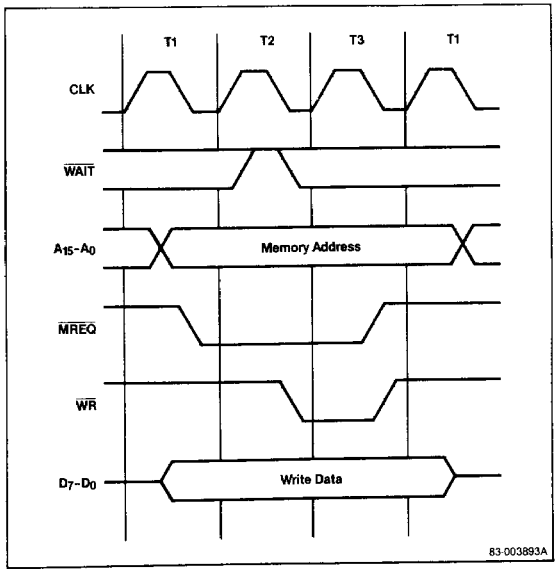
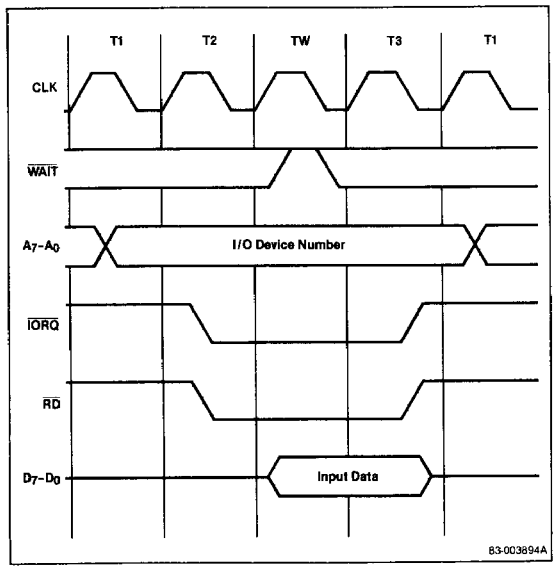


Figure 6. I/O Read Cycle



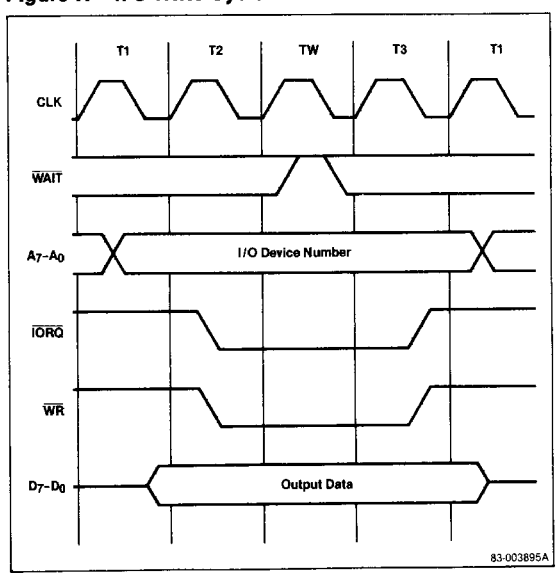
I/O Read Cycle

The contents of an I/O device are read out to the data bus when $\overline{IORQ}$ or $\overline{RD}$ is asserted. The μPD70008/A reads the data bus on the trailing edge of the T3 clock state. See figure 6. To compensate for I/O devices with longer access times, the μPD70008/A generates one wait state (TW) regardless of the condition of the $\overline{WAIT}$ signal. To extend the access time, the CPU must detect the $\overline{WAIT}$ signal asserted at the falling edge of TW.

I/O Write Cycle

Write data is output to the data bus between the last half of state T1 of the current machine cycle and the first half of T1 of the next machine cycle. It is written to an I/O device when $\overline{IORQ}$ or $\overline{WR}$ is asserted. As in the I/O read cycle, one wait state is automatically inserted in the I/O write cycle. See figure 7. The $\overline{WAIT}$ signal is used to insert additional wait states in the I/O write cycle in exactly the same way as in the read cycle.

Figure 7. I/O Write Cycle



Bus Request State

The bus request causes the μPD70008/A address bus (A_{15}-A_0), data bus (D_7-D_0), and control bus ($\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, and $\overline{WR}$) pins to enter the high impedance state.This makes the buses available to external devices for DMA access.

The bus request state is controlled by the bus request ($\overline{BUSRQ}$) signal. See figure 8. The μPD70008/A detects $\overline{BUSRQ}$ at the rising edge of the last state of each machine cycle. If it is active (low) the μPD70008/A does not move on the next machine cycle, but enters the bus request state. The μPD70008/A asserts $\overline{BUSAK}$ to indicate that the $\overline{BUSRQ}$ signal has been received, and the three buses have entered the high impedance state.

$\overline{BUSRQ}$ is checked at the rising edge of all clock states. When it becomes inactive the μPD70008/A leaves the bus request state, and proceeds to the next cycle.

$\overline{BUSRQ}$ temporarily suspends the standby mode. When $\overline{BUSRQ}$ is asserted, the μPD70008/A leaves the standby mode and enters the bus request state. When $\overline{BUSRQ}$ is released the μPD70008/A returns to the standby mode.

Interrupts are disabled during the bus request state.

Interrupts

The μPD70008/A has two types of interrupt: maskable ($\overline{INT}$) and nonmaskable ($\overline{NMI}$). The nonmaskable interrupt request cannot be masked by software. It will be acknowledged unless the μPD70008/A is in the bus request state. The maskable interrupt can be masked by software. It is controlled by setting or resetting the interrupt enable flip-flop (IFF) using the EI or DI instructions. $\overline{INT}$ has a lower priority than the nonmaskable interrupt. The maskable interrupt will therefore not be acknowledged if there is a nonmaskable interrupt, or if the μPD70008/A is in the bus request state.

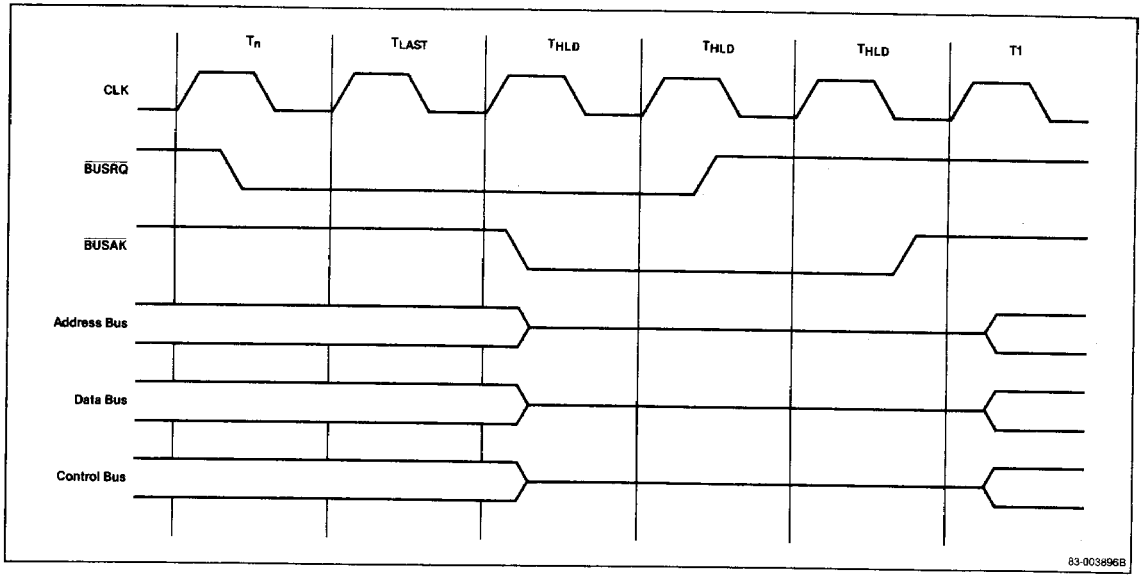
$$\overline{INTI} < \overline{NMI} < \overline{BUSRQ} < \overline{RESET}$$

Nonmaskable Interrupt Operation

The falling edge of $\overline{NMI}$ always sets the nonmaskable interrupt flip-flop. The μPD70008/A checks the flip-flop at the rising edge of the last clock state of an instruction. If it is set, the μPD70008/A transfers control to the nonmaskable interrupt service routine. The interrupt process starts at the opcode fetch cycle, (M1, 5 states) but the opcode fetched at this point is ignored. The contents of the PC are stored on the stack in the next two machine cycles (M2, 3 states and M3, 3 states). At the same time, the address 0066H is loaded into the PC, and the state of the interrupt enable flip-flop is saved to an exclusive flip-flop. The entire interrupt routine requires 3 machine cycles (11 states). The contents of the PC and IFF are restored by the execution of the RETN instruction at the end of the interrupt procedure.

3

Figure 8. Bus Request State



Maskable Interrupt Operation

Maskable interrupts are processed in three modes. In each mode, the $\overline{INT}$ signal is detected at the rising edge of the last clock state of each instruction. The M1 instruction specifies which mode is to be used.

Mode 0. In this mode, the data placed on the bus by the interrupting device is treated as an instruction. It is fetched in the opcode fetch cycle (M1, 7 states) and executed. The instruction used in this mode is usually a CALL (3 bytes) or RST (1 byte).

If a 1-byte RST instruction is executed, the contents of the PC are saved to the stack. A fixed address specified by the opcode is loaded into the PC during the next M2 (3 states) and M3 (3 states). The execution of this interrupt requires 3 machine cycles or 13 states.

If a 3-byte CALL instruction is executed, the second and third bytes are fetched during the M2 and M3 cycles (3 states each). During M4 and M5 (3 states each), the contents of the PC are saved to the stack and the second and third bytes of the CALL instruction are loaded into the PC. This interrupt requires 5 machine cyces and a total of 19 clock states.

Mode 1. In this mode, the data fetched during M1 (7 states) is ignored, and the μPD70008/A proceeds to the next cycle. During the M2 and M3 machine cycles (3 states each), the contents of the PC are saved to the stack and replaced by the interrupt address 0038H. This interrupt requires 3 machine cycles or 13 states.

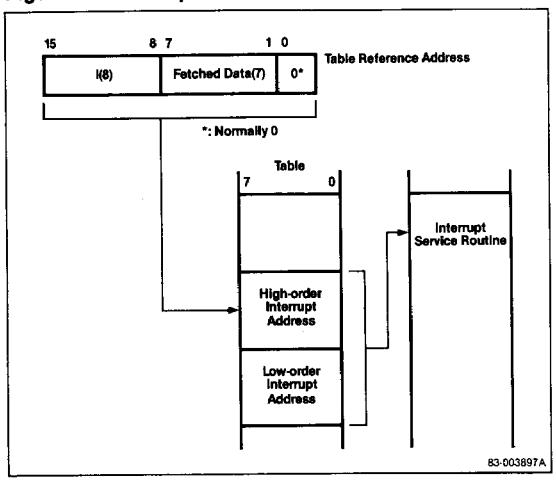
Mode 2. In this mode, the data fetched from the interrupting device and the contents of the interrupt page register (I) are used to reference an interrupt start address table. Program execution jumps to the 16-bit address referenced by the table. See figure 9. The data is fetched during the opcode fetch cycle (M1, 7 states). During M2 and M3 (3 states each) the contents of the PC are saved to the stack. During the M4 and M5 cycles, (3 states each) the table is referenced and the contents of the table location are loaded into the PC. This interrupt requires 5 machine cycles or 19 states.

Standby Mode

The μPD70008/A is provided with a standby mode (HALT). In the standby mode, power consumption is approximately 2% of normal operating power consumption. The standby mode is set by executing the HALT instruction.

In the standby mode, the state of the μPD70008/A is retained. The contents of all registers and the state of all flags are retained as well. Clock signals are supplied only to indispensable circuits in the μPD70008/A to minimize power consumption.

Figure 9. Interrupt Address Table



External operations such as memory access and memory refresh are not performed in the standby mode.

Table 2 shows the state of each output pin in the standby mode.

Table 2. Standby Mode

Pin	Status
Data bus D_7–D_0	High level, pulled up through internal resistance
Address bus A_{15}–A_0	High or low level signals
Control bus $\overline{RD}$, $\overline{WR}$, $\overline{MREQ}$, $\overline{IORQ}$, $\overline{M1}$	High level (inactive)
$\overline{RFSH}$	High level (inactive)
$\overline{HALT}$	Low level (active)

The standby mode is released when a reset or an interrupt occurs. The standby mode is temporarily suspended by a bus request, but not released.

$\overline{RESET}$ in Standby Mode

When the $\overline{RESET}$ signal becomes active (low) the standby mode is released and a normal reset is performed.

$\overline{NMI}$ in Standby Mode

When the $\overline{NMI}$ signal is asserted (low) the standby mode is released and normal nonmaskable interrupt processing is performed. The interrupt is not performed in the bus request state.

$\overline{INT}$ in Standby Mode

When the $\overline{INT}$ signal is asserted (low) the standby mode is released. If the interrupt is enabled, normal interrupt processing is performed. If the interrupt is disabled, execution resumes at the instruction following the HALT instruction.

$\overline{BUSRQ}$ in Standby Mode

The $\overline{BUSRQ}$ signal is detected at the rising edge of each clock in the standby mode. If the $\overline{BUSRQ}$ signal is active (low) the μPD70008/A leaves the standby mode, and enters the bus request state. When $\overline{BUSRQ}$ is released, the standby mode is resumed. The standby mode is not released by the $\overline{BUSRQ}$ signal.

$\overline{RESET}$

The $\overline{RESET}$ signal must be asserted (low) for over 3 clock cycles to be recognized. The following steps are the reset initialization process:

- The program counter (PC) is cleared to 0000H.
- The interrupt enable flip-flop (IFF) is reset to 0, disabling maskable interrupts.The interrupt mode is set to 0.
- The interrupt page address register (I) is cleared to 00H.
- The memory refresh register (R) is cleared to 00H.
- The address bus (A_{15}-A_0) and data bus (D_7-D_0) are set to high impedance.
- All control outputs are set in their inactive state.
- The standby mode is released.

The following registers are undefined at reset:

Stack pointer (SP)
Accumulators (A,A')
Flag registers (F,F')
General purpose registers
(B, B', C, C', D, D', E, E', H, H', L, L')
Index registers (IX,IY)

When $\overline{RESET}$ is released the program will begin execution from location 0000H.

Instruction Set

Each operand should be written in the operand column of an instruction according to the description in table 3. Capital letters are keywords and should be written as they appear.

Table 3. Operand Description

Indentifier	Description
addr	16-bit immediate data or label
faddr	00H, 08H, 10H, 18H, 20H, 28H, 30H, 38H immediate data or label
word	16-bit immediate data or label
byte	8-bit immediate data or label
bit	3-bit immediate data or label (bit specification of 8-bit register / memory)
d	8-bit displacement (signed 2's complement)
r	A, B, C, D, E, H, L
r'	A', B', C', D', E', H', L'
rp	BC, DE, HL, AF
rp1	BC, DE, HL, SP
rp2	BC, DE, IX, SP
rp3	BC, DE, IY, SP
e	Displacement for relative jump (signed 2's complement)

Selection of Register and Condition

rp	qq	rp1	ss, dd	rp2	pp	rp3	rr
BC	00	BC	00	BC	00	BC	00
DE	01	DE	01	DE	01	DE	01
HL	10	HL	10	IX	10	IY	10
AF	11	SP	11	SP	11	SP	11

r, r'	r, r'	bit	b	faddr	t
B B'	000	0	000	00H	000
C C'	001	1	001	08H	001
D D'	010	2	010	10H	010
E E'	011	3	011	18H	011
H H'	100	4	100	20H	100
L L'	101	5	101	28H	101
A A'	111	6	110	30H	110
		7	111	38H	111

Flag Operation

(Blank): Flag not affected

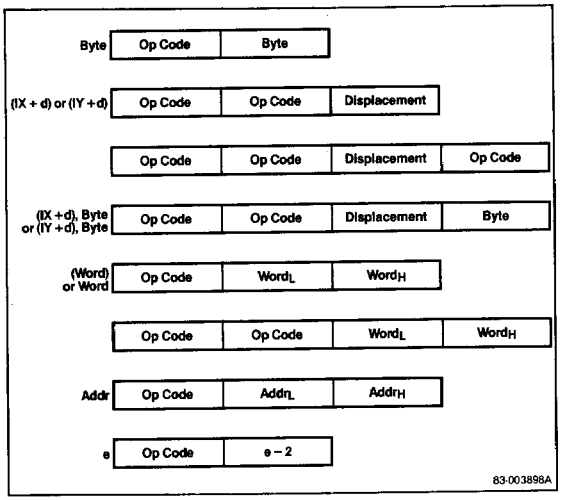
0: Flag reset

1: Flag set

X: Flag affected according to result of operation

U: Flag unknown

Structure of Instruction Byte for Addressing



Instruction Set

Mnemonic	Operands	Operation	Operation Code																No. of Clocks	No. of Bytes	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			S	Z	H	P/V	N	C
8-Bit Transfer Instructions																										
LD	r, r'	r ← r'	0	1		r			r'										4	1						
	r, byte	r ← byte	0	0		r		1	1	0									7	2						
	r, (HL)	r ← (HL)	0	1		r		1	1	0									7	1						
	r, (IX + d)	r ← (IX + disp)	1	1	0	1	1	1	0	1	0	1		r		1	1	0	19	3						
			disp																							
	r, (IY = d)	r ← (IY + disp)	1	1	1	1	1	1	0	1	0	1		r		1	1	0	19	3						
			disp																							
	(HL), r	(HL) ← r	0	1	1	1	0		r										7	1						
	(IX + d), r	(IX + disp) ← r	1	1	0	1	1	1	0	1	0	1	1	1	0		r		19	3						
			disp																							
	(IY + d), r	(IY + disp) ← r	1	1	1	1	1	1	0	1	0	1	1	1	0		r		19	3						
			disp																							
	(HL), byte	(HL) ← byte	0	0	1	1	0	1	1	0									10	2						
	(IX + d), byte	(IX + disp) ← byte	1	1	0	1	1	1	0	1	0	0	1	1	0	1	1	0	19	4						
			disp																							
	(IY + d), byte	(IY + disp) ← byte	1	1	1	1	1	1	0	1	0	0	1	1	0	1	1	0	19	4						
			disp																							
	A, (BC)	A ← (BC)	0	0	0	0	1	0	1	0									7	1						
	A, (DE)	A ← (DE)	0	0	0	1	1	0	1	0									7	1						
	A, (word)	A ← (word)	0	0	1	1	1	0	1	0									13	3						
	(BC), A	(BC) ← A	0	0	0	0	0	0	1	0									7	1						
	(DE), A	(DE) ← A	0	0	0	1	0	0	1	0									7	1						
	(word), A	(word) ← A	0	0	1	1	0	0	1	0									13	3						
	A, I	A ← I	1	1	1	0	1	1	0	1	0	1	0	1	0	1	1	1	9	2	x	x	0	IFF	0	
	A, R	A ← R	1	1	1	0	1	1	0	1	0	1	0	1	1	1	1	1	9	2	x	x	0	IFF	0	
	I, A	I ← A	1	1	1	0	1	1	0	1	0	1	0	0	0	1	1	1	9	2						
	R, A	R ← A	1	1	1	0	1	1	0	1	0	1	0	0	1	1	1	1	9	2						
	rp1, word	rp1 ← word	0	0	d	d	0	0	0	1									10	3						
	IX, word	IX ← word	1	1	0	1	1	1	0	1	0	0	1	0	0	0	0	1	14	4						
	IY, word	IY ← word	1	1	1	1	1	1	0	1	0	0	1	0	0	0	0	1	14	4						
	HL, (word)	H ← (word + 1), L ← (word)	0	0	1	0	1	0	1	0									16	3						

Instruction Set (cont)

Mnemonic	Operands	Operation	Operation Code 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	No. of Clocks	No. of Bytes	Flags S	Z	H	P/V	N	C
Sixteen-Bit Transfer Instructions																										
LD	rp1, (word)	$rp1_H \leftarrow (word + 1)$, $rp1_L \leftarrow (word)$	1	1	1	0	1	1	0	1	0	1	d	d	1	0	1	1	20	4						
	IX, (word)	$IX_H \leftarrow (word + 1)$, $IX_L \leftarrow (word)$	1	1	0	1	1	1	0	1	0	0	1	0	1	0	1	0	20	4						
	IY, (word)	$IY_H \leftarrow (word + 1)$, $IY_L \leftarrow (word)$	1	1	1	1	1	1	0	1	0	0	1	0	1	0	1	0	20	4						
	(word), HL	$(word + 1) \leftarrow H$, $(word) \leftarrow L$	0	0	1	0	0	0	1	0									16	3						
	(word), rp1	$(word + 1) \leftarrow rp1_H$, $(word) \leftarrow rp1_L$	1	1	1	0	1	1	0	1	0	1	d	d	0	0	1	1	20	4						
	(word), IX	$(word + 1) \leftarrow IX_H$, $(word) \leftarrow IX_L$	1	1	0	1	1	1	0	1	0	0	1	0	0	0	1	0	20	4						
	(word), IY	$(word + 1) \leftarrow IY_H$, $(word) \leftarrow IY_L$	1	1	1	1	1	1	0	1	0	0	1	0	0	0	1	0	20	4						
	SP, HL	$SP \leftarrow HL$	1	1	1	1	1	0	0	1									6	1						
	SP, IX	$SP \leftarrow IX$	1	1	0	1	1	1	0	1	1	1	1	1	1	0	0	1	10	2						
	SP, IY	$SP \leftarrow IY$	1	1	1	1	1	1	0	1	1	1	1	1	1	0	0	1	10	2						
PUSH	rp	$(SP - 1) \leftarrow rp_L$, $(SP - 2) \leftarrow rp_H$, $SP \leftarrow SP - 2$	1	1	q	q	0	1	0	1									11	1						
	IX	$(SP - 1) \leftarrow IX_L$, $(SP - 2) \leftarrow IX_H$, $SP \leftarrow SP - 2$	1	1	0	1	1	1	0	1	1	1	1	0	0	1	0	1	15	2						
	IY	$(SP - 1) \leftarrow IY_L$, $(SP - 2) \leftarrow IY_H$, $SP \leftarrow SP - 2$	1	1	1	1	1	1	0	1	1	1	1	0	0	1	0	1	15	2						
POP	rp	$rp_L \leftarrow (SP)$, $rp_H \leftarrow (SP + 1)$, $SP \leftarrow SP = 2$	1	1	q	q	0	0	0	1									10	1						
	IX	$IX_L \leftarrow (SP)$, $IX_H \leftarrow (SP + 1)$, $SP \leftarrow SP + 2$	1	1	0	1	1	1	0	1	1	1	1	0	0	0	0	1	14	2						
	IY	$IY_L \leftarrow (SP)$, $IY_H \leftarrow (SP + 1)$, $SP \leftarrow SP + 2$	1	1	1	1	1	1	0	1	1	1	1	0	0	0	0	1	14	2						

Instruction Set (cont)

Mnemonic	Operands	Operation	Operation Code								No. of Clocks	No. of Bytes	Flags					
			7	6	5	4	3	2	1	0			S	Z	H	P/V	N	C
Data Conversion Instructions																		
EX	DE, HL	DE ↔ HL	1	1	1	0	1	0	1	1	4	1						
	AF, AF′	A ↔ A′, F ↔ F′	0	0	0	0	1	0	0	0	4	1						
EXX		BC ↔ BC′, DE ↔ DE′, HL ↔ HL′	1	1	0	1	1	0	0	1	4	1						
EX	(SP), HL	(SP) ↔ L, (SP + 1) ↔ H, SP → SP + 2	1	1	1	0	0	0	1	1	19	1						
	(SP), IX	(SP) ↔ IX_L, (SP + 1) ↔ IX_H, SP → SP + 2	1	1	0	1	1	1	0	1	23	2						
	(SP), IY	(SP) ↔ IY_L, (SP + 1) ↔ IY_H, SP ← SP + 2	1	1	1	1	1	1	0	1	23	2						
Block Transfer Instructions																		
LDI		(DE) ← (HL), DE ← DE + 1, HL ← HL + 1, BC ← BC − 1	1	1	1	0	1	1	0	1	16	2			0	x	0	
LDIR		(DE) ← (HL), DE ← DE + 1, HL ← HL + 1, BC ← BC − 1, End if BC = 0	1	1	1	0	1	1	0	1	21/16(1)	2			0	0	0	
LDD		(DE) ← (HL), DE ← DE − 1, HL ← HL − 1, BC ← BC − 1	1	1	1	0	1	1	0	1	16	2			0	x	0	
LDDR		(DE) ← (HL), DE ← DE − 1, HL ← HL − 1, BC ← BC − 1, End if BC = 0	1	1	1	0	1	1	0	1	21/16(1)	2			0	0	0	
Block Search Instructions																		
CPI		A − (HL), HL ← HL + 1, BC ← BC − 1	1	1	1	0	1	1	0	1	16	2	x	x	x	x	1	
CPIR		A − (HL), HL ← HL + 1, BC ← BC − 1, End if A = (HL) or BC = 0	1	1	1	0	1	1	0	1	21/16(2)	2	x	x	x	x	1	

Instruction Set (cont)

Mnemonic	Operands	Operation	Operation Code								No. of Clocks	No. of Bytes	Flags					
			7	6	5	4	3	2	1	0			S	Z	H	P/V	N	C
Block Search Instructions (cont)																		
CPD		A – (HL), HL ← HL – 1, BC ← BC – 1	1	1	1	0	1	1	0	1	16	2	x	x	x	x	1	
CPDR		A – (HL), HL ← HL – 1, BC ← BC – 1, End if A = (HL) or BC = 0	1	1	1	0	1	1	0	1	21/16(2)	2	x	x	x	x	1	
Eight-Bit Arithmetic Operation Instructions																		
ADD	A, r	A ← A + r	1	0	0	0	0		r		4	1	x	x	x	V	0	x
	A, byte	A ← A + byte	1	1	0	0	0	1	1	0	7	2	x	x	x	V	0	x
	A, (HL)	A ← A + (HL)	1	0	0	0	0	1	1	0	7	1	x	x	x	V	0	x
	A, (IX + d)	A ← A + (IX + disp)	1	1	0	1	1	1	0	1	19	3	x	x	x	V	0	x
			disp															
	A, (IY + d)	A ← A + (IY + disp)	1	1	1	1	1	1	0	1	19	3	x	x	x	V	0	x
			disp															
ADC	A, r	A ← A + r + CY	1	0	0	0	1		r		4	1	x	x	x	V	0	x
	A, byte	A ← A + byte + CY	1	1	0	0	1	1	1	0	7	2	x	x	x	V	0	x
	A, (HL)	A ← A + (HL) + CY	1	0	0	0	1	1	1	0	7	1	x	x	x	V	0	x
	A, (IX + d)	A ← A + (IX + disp) + CY	1	1	0	1	1	1	0	1	19	3	x	x	x	V	0	x
			disp															
	A, (IY + d)	A ← A + (IY + disp) + CY	1	1	1	1	1	1	0	1	19	3	x	x	x	V	0	x
			disp															
SUB	A, r	A ← A – r	1	0	0	1	0		r		4	1	x	x	x	V	1	x
	A, byte	A ← A – byte	1	1	0	1	0	1	1	0	7	2	x	x	x	V	1	x
	A, (HL)	A ← A – (HL)	1	0	0	1	0	1	1	0	7	1	x	x	x	V	1	x
	A, (IX + d)	A ← A – (IX + disp)	1	1	0	1	1	1	0	1	19	3	x	x	x	V	1	x
			disp															
	A, (IY + d)	A ← A – (IY + disp)	1	1	1	1	1	1	0	1	19	3	x	x	x	V	1	x
			disp															

Instruction Set (cont)

Mnemonic	Operands	Operation	Operation Code																No. of Clocks	No. of Bytes	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			S	Z	H	P/V	N	C
Eight-Bit Arithmetic Operation Instructions (cont)																										
SBC	A, r	A ← A − r − CY	1	0	0	1	1		r										1	1	x	x	V	1	x	
	A, byte	A ← A − byte − CY	1	1	0	1	1	1	1	0									7	2	x	x	x	V	1	x
	A, (HL)	A ← A − (HL) − CY	1	0	0	1	1	1	1	0									7	1	x	x	x	V	1	x
	A, (IX + d)	A ← A − (IX + disp) − CY	1	1	0	1	1	1	0	1	1	0	0	1	1	1	1	0	19	3	x	x	x	V	1	x
			disp																							
	A, (IY + d)	A ← A − (IY + disp) − CY	1	1	1	1	1	1	0	1	1	0	0	1	1	1	1	0	19	3	x	x	x	V	1	x
			disp																							
Eight-Bit Logical Operation Instructions																										
AND	A, r	A ← A AND r	1	0	1	0	0		r										4	1	x	x	1	P	0	0
	A, byte	A ← A AND byte	1	1	1	0	0	1	1	0									7	2	x	x	1	P	0	0
	A, (HL)	A ← A AND (HL)	1	0	1	0	0	1	1	0									7	1	x	x	1	P	0	0
	A, (IX + d)	A ← A AND (IX + disp) − CY	1	1	0	1	1	1	0	1	1	0	1	0	0	1	1	0	19	3	x	x	1	P	0	0
			disp																							
	A, (IY + d)	A ← A AND (IY + disp)	1	1	1	1	1	1	0	1	1	0	1	0	0	1	1	0	19	3	x	x	1	P	0	0
			disp																							
OR	A, r	A ← A OR r	1	0	1	1	0		r										4	1	x	x	0	P	0	0
	A, byte	A ← A OR byte	1	1	1	1	0	1	1	0									7	2	x	x	0	P	0	0
	A, (HL)	A ← A OR (HL)	1	0	1	1	0	1	1	0									7	1	x	x	0	P	0	0
	A, (IX + d)	A ← A OR (IX + disp)	1	1	0	1	1	1	0	1	1	0	1	1	0	1	1	0	19	3	x	x	0	P	0	0
			disp																							
	A, (IY + d)	A ← A OR (IYJ + disp)	1	1	1	1	1	1	0	1	1	0	1	1	0	1	1	0	19	3	x	x	0	P	0	0
			disp																							
XOR	A, r	A ← A XOR r	1	0	1	0	1		r										4	1	x	x	0	P	0	0
	A, byte	A ← A XOR byte	1	1	1	0	1	1	1	0									7	2	x	x	0	P	0	0
	A, (HL)	A ← A XOR (HL)	1	0	1	0	1	1	1	0									7	1	x	x	0	P	0	0
	A, (IX + d)	A ← A XOR (IX + disp)	1	1	0	1	1	1	0	1	1	0	1	0	1	1	1	0	19	3	x	x	0	P	0	0
			disp																							
	A, (IY + d)	A ← A XOR (IY + disp)	1	1	1	1	1	1	0	1	1	0	1	0	1	1	1	0	19	3	x	x	0	P	0	0
			disp																							

Instruction Set (cont)

Mnemonic	Operands	Operation	Operation Code																No. of Clocks	No. of Bytes	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			S	Z	H	P/V	N	C
Eight-Bit Logical Operation Instructions (cont)																										
CP	A, r	A − r	1	0	1	1	1		r										4	1	x	x	x	V	1	x
	A, byte	A − byte	1	1	1	1	1	1	1	0									7	2	x	x	x	V	1	x
	A, (HL)	A − (HL)	1	0	1	1	1	1	1	0									7	1	x	x	x	V	1	x
	A, (IX + d)	A − (IX + disp)	1	1	0	1	1	1	0	1	1	0	1	1	1	1	1	0	19	3	x	x	x	V	1	x
							disp																			
	A, (IY + d)	A − (IY + disp)	1	1	1	1	1	1	0	1	1	0	1	1	1	1	1	0	19	3	x	x	x	V	1	x
							disp																			
Eight-Bit Increment / Decrement Instructions																										
INC	r	r ← r + 1	0	0		r			1	0	0								4	1	x	x	x	V	0	
	(HL)	(HL) ← (HL) + 1	0	0	1	1	0	1	0	0									11	1	x	x	x	V	0	
	(IX + d)	(IX + disp) ← (IX + disp) + 1	1	1	0	1	1	1	0	1	0	0	1	1	0	1	0	0	23	3	x	x	x	V	0	
							disp																			
	(IY + d)	(IY + disp) ← (IX + disp) + 1	1	1	1	1	1	1	0	1	0	0	1	1	0	1	0	0	23	3	x	x	x	V	0	
							disp																			
DEC	r	r ← r − 1	0	0		r			1	0	1								4	1	x	x	x	V	1	
	(HL)	(HL) ← (HL) − 1	0	0	1	1	0	1	0	1									11	1	x	x	x	V	1	
	(IX + d)	(IX + disp) ← (IX + disp) − 1	1	1	0	1	1	1	0	1	0	0	1	1	0	1	0	1	23	3	x	x	x	V	1	
							disp																			
	(IY + d)	(IY + disp) ← (IY + disp) − 1	1	1	1	1	1	1	0	1	0	0	1	1	0	1	0	1	23	3	x	x	x	V	1	
							disp																			
Sixteen-Bit Arithmetic Operation Instructions																										
ADD	HL, rp1	HL ← HL + rp1	0	0	s	s	1	0	0	1									11	1			U		0	x
ADC	HL, rp1	HL ← HL + rp1 + CY	1	1	1	0	1	1	0	1	0	1	s	s	1	0	1	0	15	2	x	x	U	V	0	x
SBC	HL, rp1	HL ← HL − rp1 − CY	1	1	1	0	1	1	0	1	0	1	s	s	0	0	1	0	15	2	x	x	U	V	1	x
ADD	IX, rp2	IX ← IX + rp2	1	1	0	1	1	1	0	1	0	0	p	p	1	0	0	1	15	2			U		0	x
	IY, rp3	IY ← IY + rp3	1	1	1	1	1	1	0	1	0	0	r	r	1	0	0	1	15	2			U		0	x
Sixteen-Bit Increment / Decrement Instructions																										
INC	rp1	rp1 ← rp1 + 1	0	0	s	s	0	0	1	1									6	1						
	IX	IX ← IX + 1	1	1	0	1	1	1	0	1	0	0	1	0	0	0	1	1	10	2						
	IY	IY ← IY + 1	1	1	1	1	1	1	0	1	0	0	1	0	0	0	1	1	10	2						
DEC	rp1	rp1 ← rp1 − 1	0	0	s	s	1	0	1	1									6	1						
	IX	IX ← IX − 1	1	1	0	1	1	1	0	1	0	0	1	0	1	0	1	1	10	2						
	IY	IY ← IY − 1	1	1	1	1	1	1	0	1	0	0	1	0	1	0	1	1	10	2						

Instruction Set (cont)

Mnemonic	Operands	Operation	Operation Code																No. of Clocks	No. of Bytes	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			S	Z	H	P/V	N	C
Accumulator Operation Instructions																										
DAA		Decimal adjust accumulator	0	0	1	0	0	1	1	1									4	1	x	x	x	P		x
CPL		$A \leftarrow \overline{A}$	0	0	1	0	1	1	1	1									4	1			1		1	
NEG		$A \leftarrow \overline{A} + 1$	1	1	1	0	1	1	0	1	0	1	0	0	0	1	0	0	8	2	x	x	x	V	1	x
CCF		$CY \leftarrow \overline{CY}$	0	0	1	1	1	1	1	1									4	1			U		0	x
SCF		CY ← 1	0	0	1	1	0	1	1	1									4	1			0		0	1
Rotate Instructions																										
RLCA			0	0	0	0	0	1	1	1									4	1			0		0	x
RLA			0	0	0	1	0	1	1	1									4	1			0		0	x
RRCA			0	0	0	0	1	1	1	1									4	1			0		0	x
RRA			0	0	0	1	1	1	1	1									4	1			0	0	x	
RLC	r		1	1	0	0	1	0	1	1	0	0	0	0	0		r		8	2	x	x	0	P	0	x
	(HL)		1	1	0	0	1	0	1	1	0	0	0	0	0	1	1	0	15	2	x	x	0	P	0	x
	(IX + d)		1	1	0	1	1	1	0	1	1	1	0	0	1	0	1	1	23	4	x	x	0	P	0	x
			disp								0	0	0	0	0	1	1	0								
	(IY + d)	r, (HL), (IX + disp), (IY + disp)	1	1	1	1	1	1	0	1	1	1	0	0	1	0	1	1	23	4	x	x	0	P	0	x
			disp								0	0	0	0	0	1	1	0								
RL	r		1	1	0	0	1	0	1	1	0	0	0	1	0		r		8	2	x	x	0	P	0	x
	(HL)		1	1	0	0	1	0	1	1	0	0	0	1	0	1	1	0	15	2	x	x	0	P	0	x
	(IX + d)		1	1	0	1	1	1	0	1	1	1	0	1	1	0	1	1	23	4	x	x	0	P	0	x
			disp								0	0	0	1	0	1	1	0								
	(IY + d)	r, (HL), (IX + disp), (IY + disp)	1	1	1	1	1	1	0	1	1	1	0	0	1	0	1	1	23	4	x	x	0	P	0	x
			disp								0	0	0	1	0	1	1	0								
RRC	r		1	1	0	0	1	0	1	1	0	0	0	0	1		r		8	2	x	x	0	P	0	x
	(HL)		1	1	0	0	1	0	1	1	0	0	0	0	1	1	1	0	15	2	x	x	0	P	0	x
	(IX + d)		1	1	0	1	1	1	0	1	1	1	0	0	1	0	1	1	23	4	x	x	0	P	0	x
			disp								0	0	0	0	1	1	1	0								
	(IY + d)	r, (HL), (IX + disp), (IY + disp)	1	1	1	1	1	1	0	1	1	1	0	0	1	0	1	1	23	4	x	x	0	P	0	x
			disp								0	0	0	0	1	1	1	0								

Instruction Set (cont)

Mnemonic	Operands	Operation	Operation Code																No. of Clocks	No. of Bytes	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			S	Z	H	P/V	N	C
Rotate Instructions (cont)																										
RR	r		1	1	0	0	1	0	1	1	0	0	0	1	1		r		8	2	x	x	0	P	0	x
	(HL)		1	1	0	0	1	0	1	1	0	0	0	1	1	1	1	0	15	2	x	x	0	P	0	x
	(IX + d)		1	1	0	1	1	1	0	1	1	1	0	0	1	0	1	1	23	4	x	x	0	P	0	x
							disp				0	0	0	1	1	1	1	0								
	(IY + d)	r, (HL), (IX + disp), (IY + disp)	1	1	1	1	1	1	0	1	1	1	0	0	1	0	1	1	23	4	x	x	0	P	0	x
							disp				0	0	0	1	1	1	1	0								
RLD			1	1	1	0	1	1	0	1	0	1	1	0	1	1	1	1	18	2	x	x	0	P	0	
RRD			1	1	1	0	1	1	0	1	0	1	1	0	0	1	1	1	18	2	x	x	0	P	0	
Shift Instructions																										
SLA	r		1	1	0	0	1	0	1	1	0	0	1	0	0		r		8	2	x	x	0	P	0	x
	(HL)		1	1	0	0	1	0	1	1	0	0	1	0	0	1	1	0	15	2	x	x	0	P	0	x
	(IX + d)		1	1	0	1	1	1	0	1	1	1	0	0	1	0	1	1	23	4	x	x	0	P	0	x
							disp				0	0	1	0	0	1	1	0								
	(IY + d)	r, (HL), (IX + disp), (IY + disp)	1	1	1	1	1	1	0	1	1	1	0	0	1	0	1	1	23	4	x	x	0	P	0	x
							disp				0	0	1	0	0	1	1	0								
SRA	r		1	1	0	0	1	0	1	1	0	0	1	0	1		r		8	2	x	x	0	P	0	x
	(HL)		1	1	0	0	1	0	1	1	0	0	1	0	1	1	1	0	15	2	x	x	0	P	0	x
	(IX + d)		1	1	0	1	1	1	0	1	1	1	0	0	1	0	1	1	23	4	x	x	0	P	0	x
							disp				0	0	1	0	1	1	1	0								
	(IY + d)	r, (HL), (IX + disp), (IY + disp)	1	1	1	1	1	1	0	1	1	1	0	0	1	0	1	1	23	4	x	x	0	P	0	x
							disp				0	0	1	0	1	1	1	0								
SRL	r		1	1	0	0	1	0	1	1	0	0	1	1	1		r		8	2	x	x	0	P	0	x
	(HL)		1	1	0	0	1	0	1	1	0	0	1	1	1	1	1	0	15	2	x	x	0	P	0	x
	(IX + d)		1	1	0	1	1	1	0	1	1	1	0	0	1	0	1	1	23	4	x	x	0	P	0	x
							disp				0	0	1	1	1	1	1	0								
	(IY + d)	r, (HL), (IX + disp), (IY + disp)	1	1	1	1	1	1	0	1	1	1	0	0	1	0	1	1	23	4	x	x	0	P	0	x
							disp				0	0	1	1	1	1	1	0								

Instruction Set (cont)

Mnemonic	Operands	Operation	Operation Code																No. of Clocks	No. of Bytes	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			S	Z	H	P/V	N	C
Bit Operation Instructions																										
BIT	bit, r	$Z \leftarrow \overline{r_b}$	1	1	0	0	1	0	1	1	0	1		b			r		8	2	U	x	1	U	0	
	bit, (HL)	$Z \leftarrow \overline{(HL)_b}$	1	1	0	0	1	0	1	1	0	1		b		1	1	0	12	2	U	x	1	U	0	
	bit, (IX + d)	$Z \leftarrow \overline{(IX + disp)_b}$	1	1	0	1	1	1	0	1	1	1	0	0	1	0	1	1	20	4	U	x	1	U	0	
							disp				0	1		b		1	1	0								
	bit, (IY + d)	$Z \leftarrow \overline{(IY + disp)_b}$	1	1	1	1	1	1	0	1	1	1	0	0	1	0	1	1	20	4	U	x	1	U	0	
							disp				0	1		b		1	1	0								
SET	bit, r	$r_b \leftarrow 1$	1	1	0	0	1	0	1	1	1	1		b			r		8	2						
	bit, (HL)	$(HL)_b \leftarrow 1$	1	1	0	0	1	0	1	1	1	1		b		1	1	0	15	2						
	bit (IX + d)	$(IX + disp)_b \leftarrow 1$	1	1	0	1	1	1	0	1	1	1	0	0	1	0	1	1	23	4						
							disp				1	1		b		1	1	0								
	bit (IY + d)	$(IY + disp)_b \leftarrow 1$	1	1	1	1	1	1	0	1	1	1	0	0	1	0	1	1	23	4						
							disp				1	1		b		1	1	0								
RES	bit, r	$r_b \leftarrow 0$	1	1	0	0	1	0	1	1	1	0		b			r		8	2						
	bit, (HL)	$(HL)_b \leftarrow 0$	1	1	0	0	1	0	1	1	1	0		b		1	1	0	15	2						
	bit (IX + d)	$(IX + disp)_b \leftarrow 0$	1	1	0	1	1	1	0	1	1	1	0	0	1	0	1	1	23	4						
							disp				1	0		b		1	1	0								
	bit (IY + d)	$(IY + disp)_b \leftarrow 0$	1	1	1	1	1	1	0	1	1	1	0	0	1	0	1	1	23	4						
							disp				1	0		b		1	1	0								
Jump Instructions																										
JP	addr	PC ← addr	1	1	0	0	0	0	1	1									10	3						
	NZ, addr	If Z = 0, PC ← addr	1	1	0	0	0	0	1	0									10	3						
	Z, addr	If Z = 1, PC ← addr	1	1	0	0	1	0	1	0									10	3						
	NC, addr	If C = 0, PC ← addr	1	1	0	1	0	0	1	0									10	3						
JP	C, addr	If C = 1, PC ← addr	1	1	0	1	1	0	1	0									10	3						
	PO, addr	If P = 0, PC ← addr	1	1	1	0	0	0	1	0									10	3						
	PE, addr	If P = 1, PC ← addr	1	1	1	0	1	0	1	0									10	3						
	P, addr	If S = 0, PC ← addr	1	1	1	1	0	0	1	0									10	3						
	M, addr	If S = 1, PC ← addr	1	1	1	1	1	0	1	0									10	3						

Instruction Set (cont)

Mnemonic	Operands	Operation	Operation Code																No. of Clocks	No. of Bytes	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			S	Z	H	P/V	N	C
Jump Instructions (cont)																										
JR	e	PC ← PC + e	0	0	0	1	1	0	0	0									12	2						
	NZ, e	If Z = 0, PC ← PC + e	0	0	1	0	0	0	0	0									12 / 7(3)	2						
	Z, e	If Z = 1, PC ← PC + e	0	0	1	0	1	0	0	0									12 / 7(3)	2						
	NC, e	If C = 0, PC ← PC + e	0	0	1	1	0	0	0	0									12 / 7(3)	2						
	C, e	If C = 1, PC ← PC + e	0	0	1	1	1	0	0	0									12 / 7(3)	2						
JP	(HL)	PC ← HL	1	1	1	0	1	0	0	1									4	1						
	(IX)	PC ← IX	1	1	0	1	1	1	0	1	1	1	1	0	1	0	0	1	8	2						
	(IY)	PC ← IY	1	1	1	1	1	1	0	1	1	1	1	0	1	0	0	1	8	2						
DJNZ	e	B ← B − 1; if B ≠ 0, PC ← PC + e	0	0	0	1	0	0	0	0									8 / 13(4)	2						
Call Instructions																										
CALL	addr	(SP − 1) ← PC_H, (SP − 2) ← PC_L, SP ← SP − 2, PC ← addr	1	1	0	0	1	1	0	1									17	3						
	NZ, addr	If conditions met, (SP − 1) ← PC_H, (SP − 2) ← PC_L, SP ← SP − 2, PC ← addr	1	1	0	0	0	1	0	0									17 / 10(5)	3						
	Z, addr	If conditions met, (SP − 1) ← PC_H, (SP − 2) ← PC_L, SP ← SP − 2, PC ← addr	1	1	0	0	1	1	0	0									17 / 10(5)	3						
	NC, addr	If conditions met, (SP − 1) ← PC_H, (SP − 2) ← PC_L, SP ← SP − 2, PC ← addr	1	1	0	1	0	1	0	0									17 / 10(5)	3						
	C, addr	If conditions met, (SP − 1) ← PC_H, (SP − 2) ← PC_L, SP ← SP − 2, PC ← addr	1	1	0	1	1	1	0	0									17 / 10(5)	3						
	PO, addr	If conditions met, (SP − 1) ← PC_H, (SP − 2) ← PC_L, SP ← SP − 2, PC ← addr	1	1	1	0	0	1	0	0									17 / 10(5)	3						

Instruction Set (cont)

Mnemonic	Operands	Operation	Operation Code 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	No. of Clocks	No. of Bytes	Flags S	Z	H	P/V	N	C
Call Instructions (cont)																										
CALL	PE, addr	If conditions met, (SP – 1) ← PC_H, (SP – 2) ← PC_L, SP ← SP – 2, PC ← addr	1	1	1	0	1	1	0	0									17 / 10(5)	3						
	P, addr	If conditions met, (SP – 1) ← PC_H, (SP – 2) ← PC_L, SP ← SP – 2, PC ← addr	1	1	1	1	0	1	0	0									17 / 10(5)	3						
	M, addr	If conditions met, (SP – 1) ← PC_H, (SP – 2) ← PC_L, SP ← SP – 2, PC ← addr	1	1	1	1	1	1	0	0									17 / 10(5)	3						
RST	faddr	(SP – 1) ← PC_H, (SP – 2) ← PC_L, SP ← SP – 2, PC_H ← 0, PC_L – faddr	1	1		t		1	1	1									11	1						
Return Instructions																										
RET		PC_L ← (SP), PC_H ← (SP + 1), SP ← SP + 2	1	1	0	0	1	0	0	1									10	1						
	NZ	If conditions met, PC_L ← (SP), PC_H ← (SP + 1), SP ← SP + 2	1	1	0	0	0	0	0	0									11 / 5(6)	1						
	Z	If conditions met, PC_L ← (SP), PC_H ← (SP + 1), SP ← SP + 2	1	1	0	0	1	0	0	0									11 / 5(6)	1						
	NC	If conditions met, PC_L ← (SP), PC_H ← (SP + 1), SP ← SP + 2	1	1	0	1	0	0	0	0									11 / 5(6)	1						
	C	If conditions met, PC_L ← (SP), PC_H ← (SP + 1), SP ← SP + 2	1	1	0	1	1	0	0	0									11 / 5(6)	1						
	PO	If conditions met, PC_L ← (SP), PC_H ← (SP + 1), SP ← SP + 2	1	1	1	0	0	0	0	0									11 / 5(6)	1						
	PE	If conditions met, PC_L ← (SP), PC_H ← (SP + 1), SP ← SP + 2	1	1	1	0	1	0	0	0									11 / 5(6)	1						
	P	If conditions met, PC_L ← (SP), PC_H ← (SP + 1), SP ← SP + 2	1	1	1	1	0	0	0	0									11 / 5(6)	1						
	M	If conditions met, PC_L ← (SP), PC_H ← (SP + 1), SP ← SP + 2	1	1	1	1	1	0	0	0									11 / 5(6)	1						
RETI		Return from interrupt	1	1	1	0	1	1	0	1	0	1	0	0	1	1	0	1	14	2						
RETN		Return from interrupt, nonmaskable	1	1	1	0	1	1	0	1	0	1	0	0	0	1	0	1	14	2						

Instruction Set (cont)

Mnemonic	Operands	Operation	Operation Code								No. of Clocks	No. of Bytes	Flags					
			7	6	5	4	3	2	1	0			S	Z	H	P/V	N	C
Return Instructions (cont)																		
IN	A, byte	A ← (byte), A_7–A_0 ← byte, A_{15}–A_8 ← byte	1	1	0	1	1	0	1	1			11	2				
	r, (C)	r ← (C), A_7–A_0 ← C, A_{15}–A_8 ← B	1	1	1	0	1	1	0	1	0	1	r		0	0	0	12	2	x	x	x	P	0
INI		(HL) ← (C), B ← B – 1, HL ← HL + 1, A_7–A_0 ← C, A_{15}–A_8 ← B	1	1	1	0	1	1	0	1	1	0	1	0	0	0	1	0	16	2	U	x	U	U	1	
IND		(HL) ← (C), B ← B – 1, HL ← HL – 1, A_7–A_0 ← C, A_{15}–A_8 ← B	1	1	1	0	1	1	0	1	1	0	1	0	1	0	1	0	16	2	U	x	U	U	1	
INIR		(HL) ← (C), B ← B – 1, HL ← HL + 1, A_7–A_0 ← C, A_{15}–A_8 ← B, End if B = 0	1	1	1	0	1	1	0	1	1	0	1	1	0	0	1	0	21/16(7)	2	U	1	U	U	1	
INDR		(HL) ← (C), B ← B – 1, HL ← HL – 1, A_7–A_0 ← C, A_{15}–A_8 ← B, End if B = 0	1	1	1	0	1	1	0	1	1	0	1	1	1	0	1	0	21/16(7)	2	U	1	U	U	1	
OUT	byte, A	(byte) ← A, A_7–A_0 ← byte, A_{15}–A_8 ← B	1	1	0	1	0	0	1	1			11	2				
	(C), r	(C) ← r, A_7–A_0 ← C, A_{15}–A_8 ← B	1	1	1	0	1	1	0	1	0	1	r		0	0	1	12	2						
OUTI		(C) ← (HL), B ← B – 1, HL ← HL + 1, A_7–A_0 ← C, A_{15}–A_8 ← B	1	1	1	0	1	1	0	1	1	0	1	0	0	0	1	1	16	2	U	x	U	U	1	
OUTD		(C) ← (HL), B ← B – 1, HL ← HL – 1, A_7–A_0 ← C, A_{15}–A_8 ← B	1	1	1	0	1	1	0	1	1	0	1	0	1	0	1	1	16	2	U	x	U	U	1	
OUTIR		(C) ← (HL), B ← B – 1, HL ← HL + 1, A_7–A_0 ← C, A_{15}–A_8 ← B, End if B = 0	1	1	1	0	1	1	0	1	1	0	1	1	0	0	1	1	21/16(7)	2	U	1	U	U	1	
OUTDR		(C) ← (HL), B ← B – 1, HL ← HL – 1, A_7–A_0 ← C, A_{15}–A_8 ← B, End if B = 0	1	1	1	0	1	1	0	1	1	0	1	1	1	0	1	1	16	2	U	1	U	U	1	

Instruction Set (cont)

Mnemonic	Operands	Operation	Operation Code								No. of Clocks	No. of Bytes	Flags					
			7	6	5	4	3	2	1	0			S	Z	H	P/V	N	C
CPU Control Instructions																		
NOP		No operation	0	0	0	0	0	0	0	0	4	1						
HALT		Halt	0	1	1	1	0	1	1	0	4	1						
DI		Disable interrupts (IFF ← 0)	1	1	1	1	0	0	1	1	4	1						
EI		Enable interrupts (IFF ← 1)	1	1	1	1	1	0	1	1	4	1						
IM	0	Set interrupt mode 0	1	1	1	0	1	1	0	1	8	2						
	1	Set interrupt mode 1	1	1	1	0	1	1	0	1	8	2						
	2	Set interrupt mode 2	1	1	1	0	1	1	0	1	8	2						

Note:

(1) 21 if BC ≠ 0, 16 if BC = 0

(2) 21 if BC ≠ 0 and A ≠ (HL), 16 if BC = 0 or A = (HL)

(3) 12 if condition is met, 7 if not

(4) 8 if B = 0, 13 if B ≠ 0

(5) 17 if condition is met, 10 if not

(6) 11 if condition is met, 5 if not

(7) 21 if B = 0, 16 if B ≠ 0