

**Description**

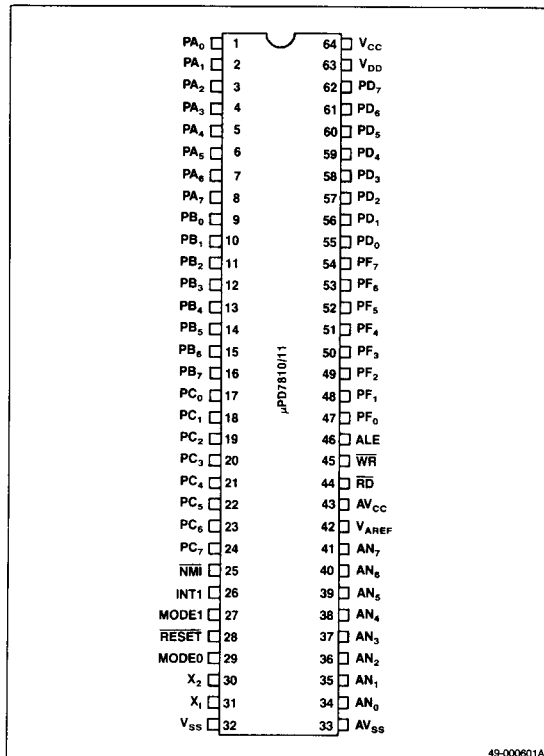
The  $\mu$ PD7810 and  $\mu$ PD7811 single-chip micro-computers integrate sophisticated on-chip peripheral functionality normally provided by external components. The device's internal 16-bit ALU and data paths, combined with a powerful instruction set and addressing, make the  $\mu$ PD7810/11 appropriate in data processing as well as control applications. The devices integrate a 16-bit ALU, 4K-ROM, 256-byte RAM with an 8-channel A/D converter, a multifunction 16-bit timer/event counter, two 8-bit timers, a USART, and two zero-cross detect inputs on a single die, allowing their use in fast, high end processing applications. This involves analog signal interface and processing.

The  $\mu$ PD7811 is the mask-ROM high volume production device embedded with custom customer program. The  $\mu$ PD7810 is a ROM-less version for prototyping and small volume production. The  $\mu$ PD78PG11E is a piggy-back EPROM version for design development.

**Features**

- NMOS silicon gate technology requiring +5 V power supply
- Complete single-chip microcomputer
  - 16-bit ALU
  - 4K x 8 ROM
  - 256-byte RAM
- 44 I/O lines
- Two zero-cross detect inputs
- Two 8-bit timers
- Expansion capabilities
  - 8085A bus-compatible
  - 60K-byte external memory address range
- 8-channel, 8-bit A/D converter
  - Autoscan mode
  - Channel select mode
- Full duplex USART
  - Synchronous and asynchronous
- 153 instructions
  - 16-bit arithmetic, multiply and divide
- 1  $\mu$ s instruction cycle time (12 MHz operation)
- Prioritized interrupt structure
  - 3 external
  - 8 internal
- Standby function
- On-chip clock generator
- 64-pin plastic QUIP or shrink DIP

**Pin Configuration**



**Ordering Information**

Part Number	Package Type	Max Frequency of Operation
$\mu$ PD7810G-36 $\mu$ PD7811G-36	64-pin plastic QUIP	12 MHz
$\mu$ PD7810CW $\mu$ PD7811CW	64-pin plastic shrink DIP	12 MHz

**Pin Identification**

No.	Symbol	Function
1-8	PA <sub>0</sub> -PA <sub>7</sub>	Port A I/O
9-16	PB <sub>0</sub> -PB <sub>7</sub>	Port B I/O
17	PC <sub>0</sub> /TxD	Port C I/O line 0/Transmit data output
18	PC <sub>1</sub> /RxD	Port C I/O line 1/Receive data input
19	PC <sub>2</sub> /SCK	Port C I/O line 2/Serial clock I/O
20	PC <sub>3</sub> /TI/ INT <sub>2</sub>	Port C I/O line 3/Timer input/Interrupt request 2 input
21	PC <sub>4</sub> /TO	Port C I/O line 4/Timer output
22	PC <sub>5</sub> /CI	Port C I/O line 5/Counter input
23, 24	PC <sub>6</sub> , PC <sub>7</sub> / CO <sub>0</sub> , CO <sub>1</sub>	Port C I/O lines 6, 7/Counter outputs 0, 1
25	NMI	Nonmaskable interrupt input
26	INT1	Interrupt request 1 input
27	MODE1/M <sub>1</sub>	Mode 1 input/Memory cycle 1 output
28	RESET	Reset input
29	MODE0/ I <sub>0</sub> /M	Mode 0 input/I/O/Memory output
30, 31	X2, X1	Crystal connections 1, 2
32	V <sub>SS</sub>	Ground
33	AV <sub>SS</sub>	Port T threshold voltage input
34-41	AN <sub>0</sub> -AN <sub>7</sub>	A/D converter analog inputs 0-7
42	V <sub>AREF</sub>	A/D converter reference voltage
43	AV <sub>CC</sub>	A/D converter power supply
44	R <sub>D</sub>	Read strobe output
45	W <sub>R</sub>	Write strobe output
46	ALE	Address latch enable output
47-54	PF <sub>0</sub> -PF <sub>7</sub>	Port F I/O/Expansion memory address bus (bits 8-15)
55-62	PD <sub>0</sub> -PD <sub>7</sub>	Port D I/O/Expansion memory address/data bus
63	V <sub>DD</sub>	RAM backup power supply
64	V <sub>CC</sub>	5 V power supply

**Pin Functions****PA<sub>0</sub>-PA<sub>7</sub> [Port A]**

Port A is an 8-bit three-state port. Each bit is independently programmable as either input or output. Reset makes all lines of port A inputs.

**PB<sub>0</sub>-PB<sub>7</sub> [Port B]**

Port B is an 8-bit three-state port. Each bit is independently programmable as either input or output. Reset makes all lines of port B inputs.

**PC<sub>0</sub>-PC<sub>7</sub> [Port C]**

Port C is an 8-bit three-state port. Each bit is independently programmable as either input or output. Alternatively, the lines of port C can be used as control lines for the USART and timer. Reset puts all lines of port C in port mode, input.

**TxD [Transmit Data]**. Serial data output terminal.

**RxD [Receive Data]**. Serial data input terminal.

**SCK [Serial Clock]**. Output for the serial clock when internal clock is used. Input for serial clock when external clock is used.

**TI [Timer Input]**. Timer input terminal.

**INT<sub>2</sub> [Interrupt Request 2]**. Falling-edge-triggered, maskable interrupt input terminal and AC-input, zero-cross detection terminal.

**TO [Timer Output]**. The output of TO is a square wave with a frequency determined by the timer/counter.

**CI [Counter Input]**. External pulse input to timer/event counter.

**CO<sub>0</sub>, CO<sub>1</sub> [Counter Outputs 0, 1]**. Programmable rectangular wave outputs based on timer/event counter.

**PD<sub>0</sub>-PD<sub>7</sub> [Port D]**

Port D is an 8-bit three-state port. It can be programmed as either 8 bits of input or 8 bits of output. When external expansion memory is used, port D acts as the multiplexed address/data bus.

**PF<sub>0</sub>-PF<sub>7</sub> [Port F]**

Port F is an 8-bit three-state port. Each bit is independently programmable as an input or output. When external expansion memory is used, port F outputs the high-order address bits.

**AN<sub>0</sub>-AN<sub>7</sub>**

These are the eight analog inputs to the A/D converter. AN<sub>4</sub>-AN<sub>7</sub> can also be used as a digital input for falling edge detection.

**AV<sub>SS</sub> [A/D Converter Power Ground]**

AV<sub>SS</sub> is the ground potential for the A/D converter power supply.

**NMI [Nonmaskable Interrupt]**

Falling-edge-triggered nonmaskable interrupt input.

**INT1 [Interrupt Request 1]**

INT1 is a rising-edge-triggered, maskable interrupt input. It is also an AC-input, zero-cross detection terminal.

 **$\overline{\text{RESET}}$  [Reset]**

When the  $\overline{\text{RESET}}$  input is brought low, it initializes the μPD7810/11.

**MODE1, MODE0 [Mode 1, 0]**

The MODE1 and MODE0 inputs select the memory expansion mode. MODE1 also outputs the M1 signal during each opcode fetch. MODE0 outputs the  $\overline{\text{IO/M}}$  signal.

**V<sub>AREF</sub> [A/D Converter Reference]**

V<sub>AREF</sub> sets the upper limit for the A/D converter's conversion range.

**AV<sub>CC</sub> [A/D Converter Power]**

This is the power supply voltage for the A/D converter.

 **$\overline{\text{RD}}$  [Read Strobe]**

The  $\overline{\text{RD}}$  output goes low to gate data from external devices onto the data bus.  $\overline{\text{RD}}$  goes high during reset.

 **$\overline{\text{WR}}$  [Write Strobe]**

The  $\overline{\text{WR}}$  output goes low to indicate that the data bus holds valid data. It is a strobe signal for external memory or I/O write operations.  $\overline{\text{WR}}$  goes high during reset.

**ALE [Address Latch Enable]**

The ALE output latches the address signal to the output of PD<sub>0</sub>-PD<sub>7</sub>.

**X1, X2 [Crystal Connections 1, 2]**

X1 and X2 are the system clock crystal oscillator terminals. X1 is the input for an external clock.

**V<sub>SS</sub> [Ground]**

Ground potential.

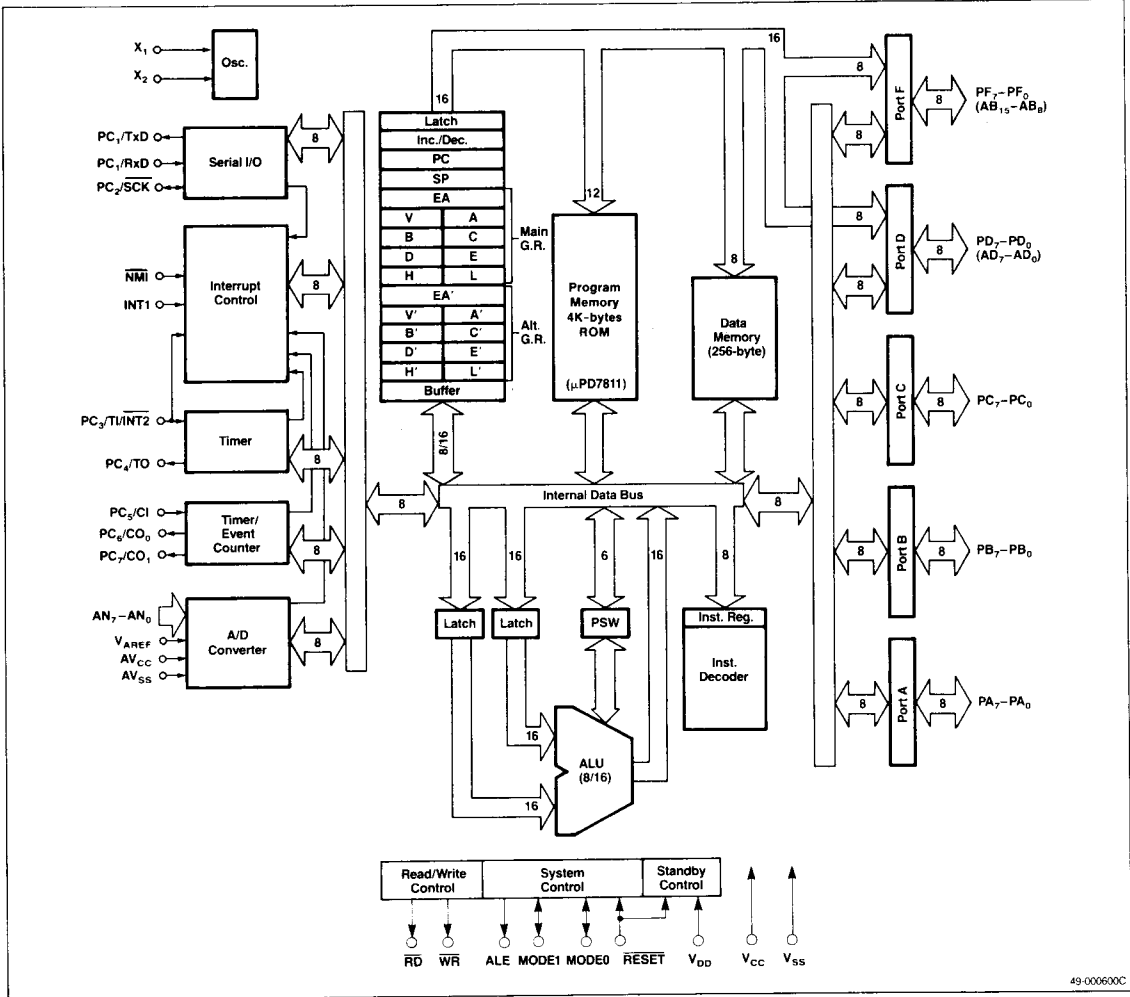
**V<sub>DD</sub> [Backup Power]**

Backup power for on-chip RAM.

**V<sub>CC</sub> [Power Supply]**

+5 V power supply.

## Block Diagram



## Functional Description

### Memory Map

The μPD7811 can directly address up to 64K bytes of memory. Except for the on-chip ROM (0-4095) and RAM (65280-65535), any memory location can be used as ROM or RAM. The memory map, figure 1, defines the 0 to 64K byte memory space for the μPD7811.

### Input/Output

The μPD7810/11 has 8 analog input lines (AN<sub>0</sub>-AN<sub>7</sub>), 44 digital I/O lines, five 8-bit ports (port A, port B, port C, port D, port F), and 4 input lines (AN<sub>4</sub>-AN<sub>7</sub>).

**Analog Input Lines.** AN<sub>0</sub>-AN<sub>7</sub> are configured as analog input lines for on-chip A/D converter.

**Port A, Port B, Port C, Port F.** Each line of these ports can be individually programmed as an input or output. When used as I/O ports, all have latched outputs and high-impedance inputs.

**Port D.** Port D can be programmed as a byte input or a byte output.

**AN<sub>4</sub>-AN<sub>7</sub>.** The high order analog input lines, AN<sub>4</sub>-AN<sub>7</sub>, can be used as digital input lines for falling edge detection.

**Control Lines.** Under software control, each line of port C can be configured individually to provide control lines for the serial interface, timer, and timer/counter.

**Memory Expansion.** In addition to the single-chip operation mode, the μPD7811 has four memory expansion modes. Under software control, port D can provide a multiplexed low-order address and data bus; port F can provide a high-order address bus. Table 1 shows the relation between memory expansion modes and the pin configurations of port D and port F.

**Table 1. Memory Expansion Modes and Port Configurations**

Memory Expansion	Port Configuration	
None	Port D	I/O port
	Port F	I/O port
256 Bytes	Port D	Multiplexed address/data bus
	Port F	I/O port
4K Bytes	Port D	Multiplexed address/data bus
	Port F <sub>0</sub> -F <sub>3</sub>	Address bus
	Port F <sub>4</sub> -F <sub>7</sub>	I/O port
16K Bytes	Port D	Multiplexed address/data bus
	Port F <sub>0</sub> -F <sub>5</sub>	Address bus
	Port F <sub>6</sub> -F <sub>7</sub>	I/O port
60K Bytes	Port D	Multiplexed address/data bus
	Port F	Address bus

### Timers

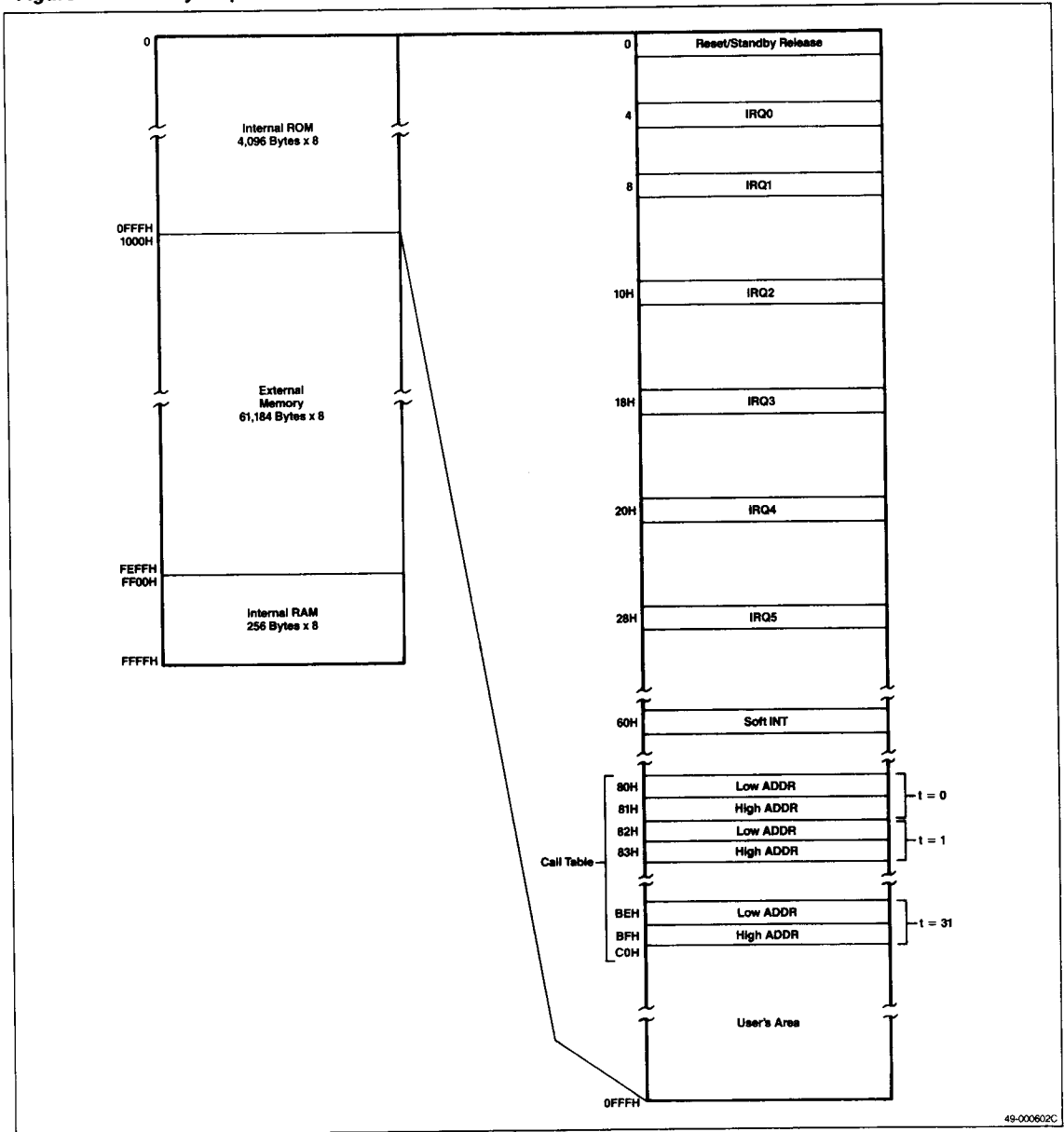
There are two 8-bit timers. The timers may be programmed independently or may be cascaded and used as an 8-bit timer with 8-bit prescaler. The timer can be software set to increment at intervals of four machine cycles (1 μs at 12 MHz operation) or 128 machine cycles (32 μs at 12 MHz), or to increment on receipt of a pulse at T1. Figure 2 shows the block diagram for the timer.

### Timer/Event Counter

The 16-bit multifunctional timer/event counter (figure 3) can be used for the following operations:

- Interval timer
- External event counter
- Frequency measurement
- Pulse width measurement
- Programmable square-wave output

**Figure 1. Memory Map**



49-000602C

Figure 2. Timer Block Diagram

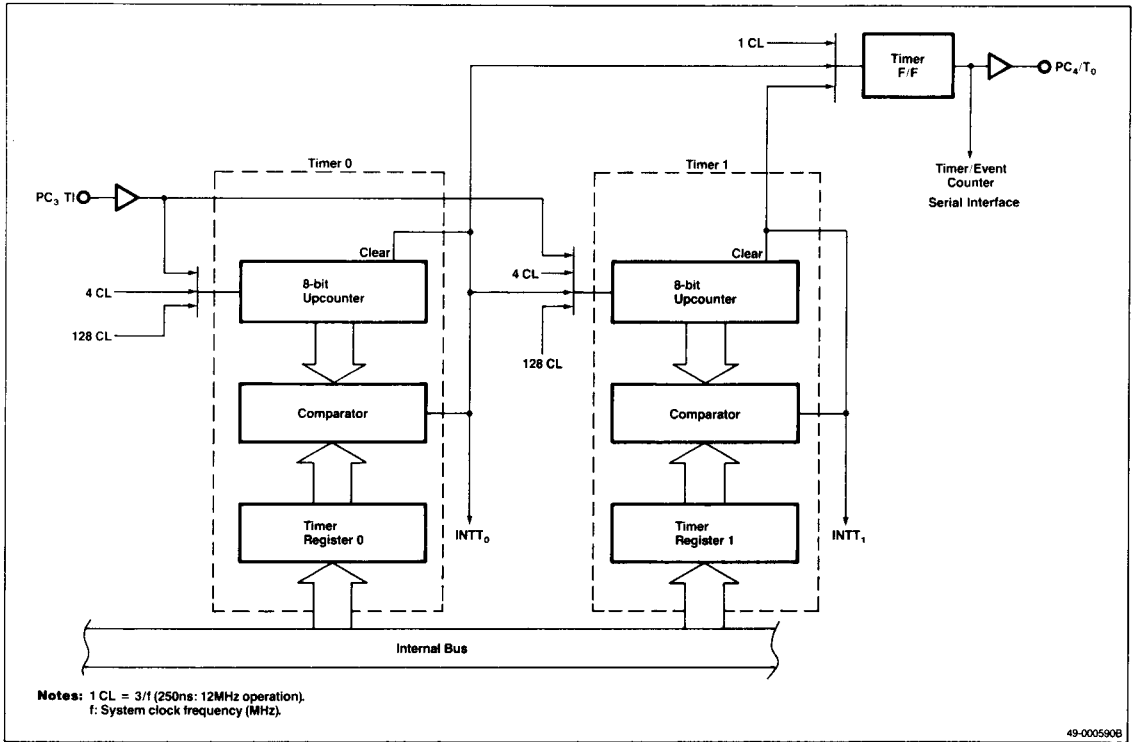
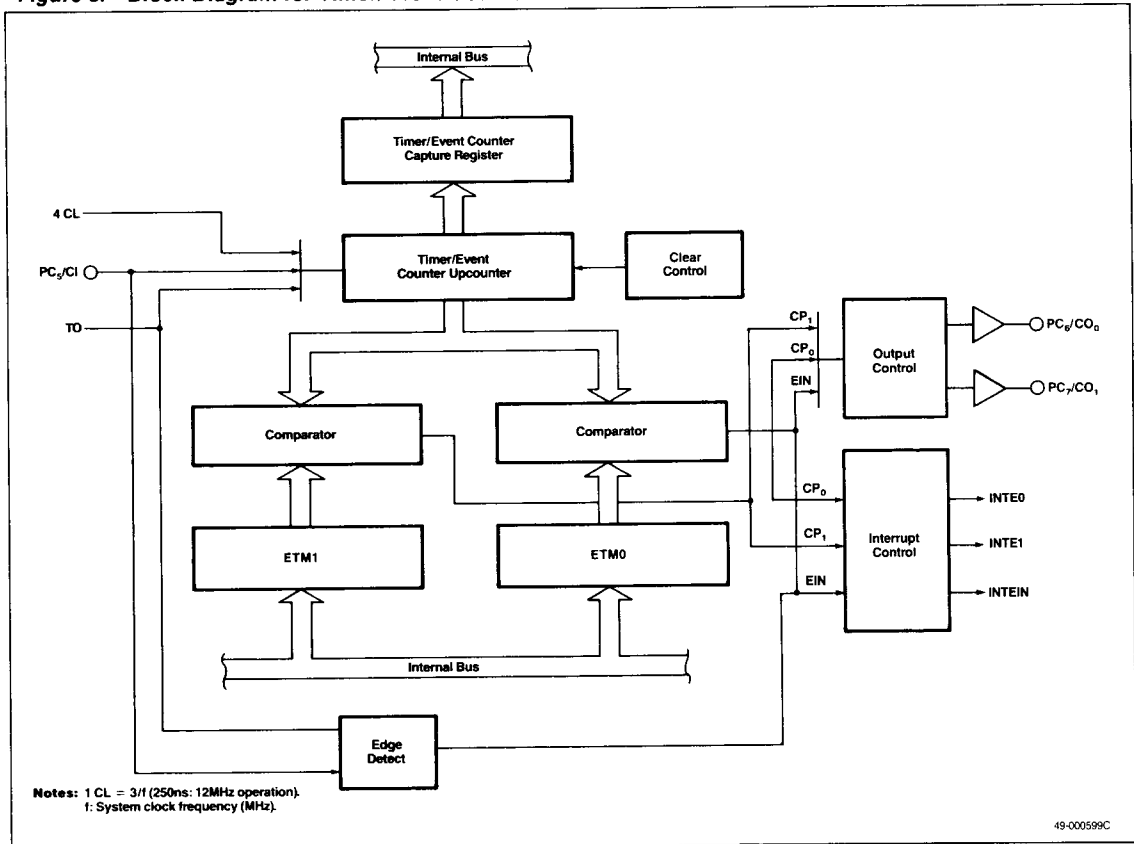


Figure 3. Block Diagram for Timer/Event Counter



### 8-Bit A/D Converter

- 8 input channels
- 4 conversion result registers
- 2 powerful operation modes
  - Autoscan mode
  - Channel select mode
- Successive approximation technique
- Absolute accuracy:  $\pm 1.5$  LSB ( $\pm 0.6\%$ )
- Conversion range: 0 to 5 V
- Conversion time: 48  $\mu$ s
- Interrupt generation

### Analog/Digital Converter

The  $\mu$ PD7810/11 features an 8-bit, high speed, high accuracy A/D converter. The A/D converter is made up of a 256-resistor ladder and a successive approximation register (SAR). There are four conversion result registers (CR<sub>0</sub>-CR<sub>3</sub>). The 8-channel analog input may be operated in either of two modes. In the select mode, the conversion value of one analog input is sequentially stored in CR<sub>0</sub>-CR<sub>3</sub>. In the scan mode, the upper four channels or the lower four channels may be specified. Then those four channels will be consecutively selected and the conversion results stored sequentially in the four conversion result registers. Figure 4 shows the block diagram for the A/D converter.



### Interrupt Structure

There are 11 interrupt sources. Three are external interrupts and eight are internal. The following, table 2, shows 11 interrupt sources divided into six priority levels. See figure 5.

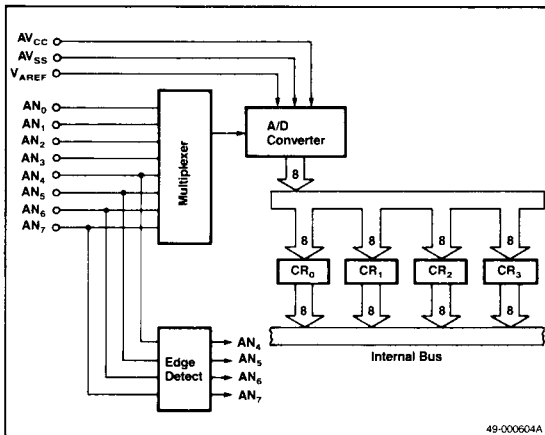
### Standby Function

The standby function saves the top 32 bytes of RAM with backup power ( $V_{DD}$ ) if the main power ( $V_{CC}$ ) fails. On power-up, you can check the standby flag (SB) to determine whether recovery was made from standby mode or from a cold start.

**Table 2. Interrupt Sources**

Interrupt Request	Interrupt Address	Type of Interrupt	Internal/External
IRQ0	4	NMI (Nonmaskable interrupt)	Ext
IRQ1	8	INTT0 (Coincidence signal from timer 0)	Int
		INTT1 (Coincidence signal from timer 1)	
IRQ2	16	INT1 (Maskable interrupt)	Ext
		INT2 (Maskable interrupt)	
IRQ3	24	INTE0 (Coincidence signal from timer/event counter)	Int
		INTE1 (Coincidence signal from timer/event counter)	
IRQ4	32	INTEIN (Falling signal of CI and TO counter)	Int/Ext
		INTAD (A/D converter interrupt)	
IRQ5	40	INTSR (Serial receive interrupt)	Int
		INST (Serial send interrupt)	

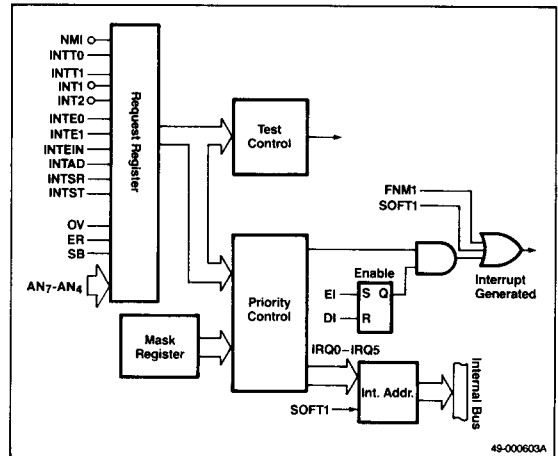
**Figure 4. A/D Converter Block Diagram**



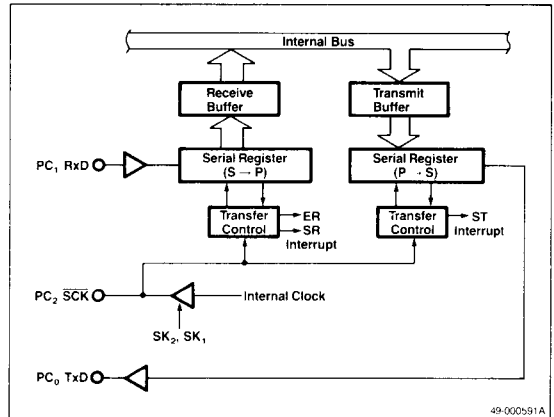
### Universal Serial Interface

The serial interface can operate in one of three modes: synchronous, asynchronous, and I/O interface. The I/O interface mode transfers data MSB first, for easy interfacing to certain NEC peripheral devices. Synchronous and asynchronous modes transfer data LSB first. Synchronous operation offers two modes of data reception: search and nonsearch. In the search mode, data is transferred one bit at a time from the serial register to the receive buffer. This allows a software search for a sync character. In the nonsearch mode, data transfer from the serial register to the transmit buffer occurs eight bits at a time. Figure 6 shows the universal serial interface block diagram.

**Figure 5. Interrupt Structure Block Diagram**



**Figure 6. Universal Signal Interface Block Diagram**



4

### Zero-Crossing Detector

The INT1 and INT2 terminals (used common to T1 and PC<sub>3</sub>) can detect the zero-crossing point of low-frequency AC signals. When driven directly, these pins respond as a normal digital input. Figure 7 shows the zero-crossing detection circuitry.

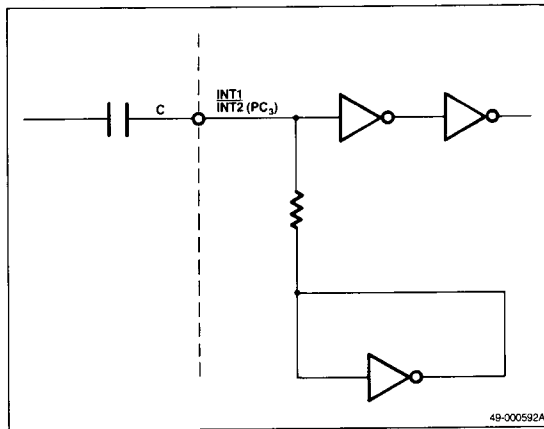
The zero-crossing detection capability allows you to make the 50-60 Hz power signal the basis for system timing and to control voltage phase-sensitive devices.

To use the zero-cross detection mode, an AC signal of approximately 1-3 V AC (peak-to-peak) and a maximum frequency of 1 kHz is coupled through an external capacitor to the INT1 and INT2 pins.

For the INT1 pin, the internal digital state is sensed as a 0 until the rising edge crosses the average DC level, when it becomes a 1 and INT1 interrupt is generated.

For the INT2 pin, the state is sensed as a 1 until the falling edge crosses the average DC level, when it becomes a 0 and INT2 interrupt is generated.

Figure 7. Zero-Crossing Detection Circuit



### Absolute Maximum Ratings

Power supply voltages, V <sub>CC</sub>	-0.5 V to +7.0 V
V <sub>DD</sub>	-0.5 V to +7.0 V
AV <sub>CC</sub>	-0.5 V to +7.0 V
AV <sub>SS</sub>	-0.5 V to +0.5 V
Input voltage, V <sub>I</sub>	-0.5 V to +7.0 V
Output voltage, V <sub>O</sub>	-0.5 V to +7.0 V
Reference input voltage, V <sub>AREF</sub>	-0.5 V to V <sub>CC</sub>
Operating temperature, T <sub>OPR</sub> 10 MHz ≤ f <sub>XTAL</sub> ≤ 12 MHz	-10°C to +70°C
f <sub>XTAL</sub> ≤ 10 MHz	-40°C to -85°C
Storage temperature, T <sub>STG</sub>	-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 this specification. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

### Operating Conditions

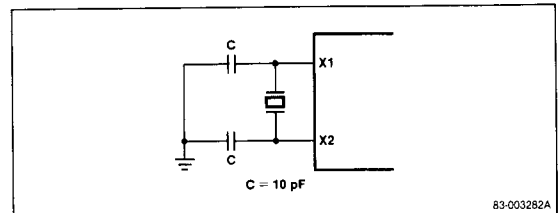
Oscillating Frequency	T <sub>A</sub>	V <sub>CC</sub> , AV <sub>CC</sub>
f <sub>XTAL</sub> ≤ 10 MHz	-40°C to +85°C	+5.0 V ±10%
10 MHz ≤ f <sub>XTAL</sub> ≤ 12 MHz	-10°C to +70°C	+5.0 V ±5%

### Capacitance

T<sub>A</sub> = 25°C; V<sub>CC</sub> = V<sub>DD</sub> = V<sub>SS</sub> = 0 V

Parameter	Symbol	Limits			Unit	Test Conditions
		Min	Typ	Max		
Capacitance	C <sub>I</sub>			10	pF	Af <sub>c</sub> = 1 MHz. Unmeasured pins returned to 0 V.
Output capacitance	C <sub>O</sub>			20	pF	
I/O capacitance	C <sub>I/O</sub>			20	pF	

### Recommended XTAL Oscillation Circuit



**DC Characteristics**

$T_A = -10^{\circ}\text{C}$  to  $+70^{\circ}\text{C}$ ;  $V_{CC} = +5.0\text{ V} \pm 5\%$ ;  $V_{SS} = 0\text{ V}$ ;  $V_{DD} = V_{CC} - 0.8\text{ V}$  to  $V_{CC}$

Parameter	Symbol	Limits			Unit	Test Conditions
		Min	Typ	Max		
Input low voltage	$V_{IL}$	0		0.8	V	
Input high voltage	$V_{IH1}$	2.0		$V_{CC}$	V	All except SCK, RESET, X1 and X2
	$V_{IH2}$	$0.8 V_{CC}$		$V_{CC}$	V	SCK, X1, X2
	$V_{IH3}$	$0.8 V_{DD}$		$V_{CC}$	V	RESET
Output low voltage	$V_{OL}$			0.45	V	$I_{OL} = 2.0\text{ mA}$
Output high voltage	$V_{OH}$	2.4			V	$I_{OH} = -200\text{ }\mu\text{A}$
Data retention voltage	$V_{DDDR}$	3.2			V	$V_{CC} = 0\text{ V}$ ; RESET = $V_{IL}$
Input current	$I_I$			$\pm 200$	$\mu\text{A}$	INT1, TI(PC3); $+0.45\text{ V} \leq V_I < V_{CC}$
Input leakage current	$I_{LI}$			$\pm 10$	$\mu\text{A}$	All except INT, TI(PC3) $0\text{ V} \leq V_I \leq V_{CC}$
Output leakage current	$I_{LO}$			$\pm 10$	$\mu\text{A}$	$+0.45\text{ V} \leq V_O \leq V_{CC}$
$V_{CC}$ supply current	$I_{CC}$	6	12		mA	
$V_{DD}$ supply current	$I_{DD}$	1.5	3.5		mA	$T_A = -40$ to $+85^{\circ}\text{C}$
				3.2	mA	$V_{CC} = V_{DD} = 5\text{ V}$ $T_A = -10$ to $+70^{\circ}\text{C}$
$V_{CC}$ supply current	$I_{CC}$	150	220		mA	$T_A = -40$ to $+85^{\circ}\text{C}$ ; $V_{CC} = V_{DD} = 5\text{ V}$

**Serial Operation**

Parameter	Symbol	Limits		Unit	Test Conditions
		Min	Max		
SCK cycle time	$t_{CYK}$	1		$\mu\text{s}$	SCK input (1)
			500	ns	(2)
			2	$\mu\text{s}$	SCK output
SCK width low	$t_{KKL}$	750		ns	SCK input(1)
			200	ns	SCK input (2)
			900	ns	SCK output
SCK width high	$t_{KKH}$	750		ns	SCK input (1)
			200	ns	SCK input (2)
			900	ns	SCK output
RxD set-up time to SCK $\uparrow$	$t_{RXK}$	80		ns	(1)
RxD hold time after SCK $\uparrow$	$t_{KRX}$		80	ns	(1)
SCK $\downarrow$ TxD delay time	$t_{KTX}$		210	ns	(1)

**Note:**

- (1) 1x baud rate in asynchronous, synchronous, or I/O interface mode.
- (2) 16x baud rate or 64x baud rate in asynchronous mode.

**Zero-Cross Characteristics**

Parameter	Symbol	Limits		Unit	Test Conditions
		Min	Max		
Zero-cross detection input	$V_{ZX}$	1	3	V ac, p-p	Ac coupled
Zero-cross accuracy	$A_{ZX}$		$\pm 135$	mV	60-Hz sine wave
Zero-cross detection input frequency	$f_{ZX}$	0.05	1	kHz	



**AC Characteristics****Read/Write Operation**

$$V_{SS} = 0 \text{ V}, V_{CC} - 0.8 \text{ V} \leq V_{DD} \leq V_{CC}$$

Parameter	Symbol	Limits				Unit	Test Conditions (1)
		$f_{XTAL} = 10 \text{ MHz}$		$f_{XTAL} = 12 \text{ MHz}$			
		Min	Max	Min	Max		
RESET pulse width	$t_{RP}$	6.0		5.0		$\mu\text{s}$	
Interrupt pulse width	$t_{IP}$	3.6		3.0		$\mu\text{s}$	
Counter input pulse width	$t_{CI}$	600		500		ns	Event counter mode
	$t_{CI}$	4.8		4.0		$\mu\text{s}$	Pulse width measurement mode
Timer input pulse width	$t_{TI}$	600		500		ns	
X1 Input cycle time	$t_{CYC}$	100	250	83	250	ns	
Address set-up to ALE ↓	$t_{AL}$	100		65		ns	
Address hold after ALE ↓	$t_{LA}$	70		50		ns	
Address to RD ↓ delay time	$t_{AR}$	200		150		ns	
$\overline{RD}$ ↓ to address floating	$t_{AFR}$		20		20	ns	
Address to data input	$t_{AD}$		480		360	ns	
ALE ↓ to data input	$t_{LDR}$		300		215	ns	
$\overline{RD}$ ↓ to data input	$t_{RD}$		250		180	ns	
ALE ↓ to RD ↓ delay time	$t_{LR}$	50		35		ns	
Data hold time to $\overline{RD}$ ↑	$t_{RDH}$	0		0		ns	
$\overline{RD}$ ↑ to ALE ↑ delay time	$t_{RL}$	150		115		ns	
$\overline{RD}$ width low	$t_{RR}$	350		280		ns	Data read
		650		530		ns	Opcode fetch
ALE width high	$t_{LL}$	160		125		ns	
$\overline{Mf}$ setup time to ALE ↓	$t_{ML}$	100		65		ns	
$\overline{Mf}$ hold time after ALE ↓	$t_{LM}$	70		50		ns	
$\overline{IO}/\overline{M}$ setup time to ALE ↓	$t_{IL}$	100		65		ns	
$\overline{IO}/\overline{M}$ hold time after ALE ↓	$t_{LI}$	70		50		ns	
Address to $\overline{WR}$ ↓ delay	$t_{AW}$	200		150		ns	
ALE ↓ to data output	$t_{LDW}$		210		195	ns	
$\overline{WR}$ ↓ to data output	$t_{WD}$		100		100	ns	
ALE ↓ to $\overline{WR}$ ↓ delay	$t_{LW}$	50		35		ns	
Data set-up time to $\overline{WR}$ ↑	$t_{DW}$	300		230		ns	
Data hold time to $\overline{WR}$ ↑	$t_{WDH}$	130		95		ns	
$\overline{WR}$ ↑ to ALE ↑ delay time	$t_{WL}$	150		115		ns	
$\overline{WR}$ width low	$t_{WW}$	350		280		ns	

**Note:**(1) Load capacitance:  $C_L = 150 \text{ pF}$ .

**A/D Converter Characteristics**

$T_A = -10^\circ\text{C}$  to  $+70^\circ\text{C}$ ;  $V_{CC} = AV_{CC} = 5.0\text{ V} \pm 5\%$ ;  $V_{SS} = AV_{SS} = 0\text{ V}$ ;  $V_{AREF} = AV_{CC} - 0.5\text{ V}$  to  $AV_{CC}$ .

Parameter	Symbol	Limits			Unit	Test Conditions
		Min	Typ	Max		
Resolution		8			Bits	
Absolute accuracy				0.4%	LSB	$T_A = -10^\circ\text{C}$ to $+50^\circ\text{C}$
				0.6%	LSB	$T_A = -10^\circ\text{C}$ to $+70^\circ\text{C}$ (Note 1)
Conversion time	$t_{CONV}$	576			$t_{CYC}$	$83\text{ ns} \leq t_{CYC} \leq 110\text{ ns}$
		432			$t_{CYC}$	$110\text{ ns} \leq t_{CYC} \leq 170\text{ ns}$
Sampling time	$t_{SAMP}$	96			$t_{CYC}$	$83\text{ ns} \leq t_{CYC} \leq 110\text{ ns}$
		72			$t_{CYC}$	$110\text{ ns} \leq t_{CYC} \leq 170\text{ ns}$
Analog input voltage	$V_{IA}$	0		$V_{AREF}$	V	
Analog resistance	$R_{AN}$		1000		MΩ	
Analog reference current	$I_{AREF}$	0.2	0.5	1.5	mA	

**Note:**

(1) In case of  $f_{XTAL} \leq 10\text{ MHz}$ ,  $T_A = -40^\circ\text{C}$  to  $+85^\circ\text{C}$ .

**Bus Timing Depending on  $t_{CYC}$**

Symbol	Calculating Expression	Min/Max
$t_{RP}$	60T	Min
$t_{TI}$	6T	Min
$t_{CI(2)}$	6T	Min
$t_{CI(3)}$	48T	Min
$t_{IP}$	36T	Min
$t_{AL}$	2T - 100	Min
$t_{LA}$	T - 30	Min
$t_{AR}$	3T - 100	Min
$t_{AD}$	7T - 220	Max
$t_{LDR}$	5T - 200	Max
$t_{RD}$	4T - 150	Max
$t_{LR}$	T - 50	Min
$t_{RL}$	2T - 50	Min
$t_{RR}$	4T - 50 (Data Read)	Min
	7T - 50 (Opcode Fetch)	
$t_{LL}$	2T - 40	Min
$t_{AW}$	3T - 100	Min
$t_{LDW}$	T + 110	Max
$t_{LW}$	T - 50	Min
$t_{DW}$	4T - 100	Min
$t_{WDH}$	2T - 70	Min
$t_{WL}$	2T - 50	Min
$t_{WW}$	4T - 50	Min
$t_{CYK}$	20T (SCK input)(1)	Min
	24T (SCK output)	
$t_{KKL}$	10T - 80 (SCK input)(1)	Min
	12T - 100 (SCK output)	
$t_{KKH}$	10T - 80 (SCK input)(1)	Min
	12T - 100 (SCK output)	

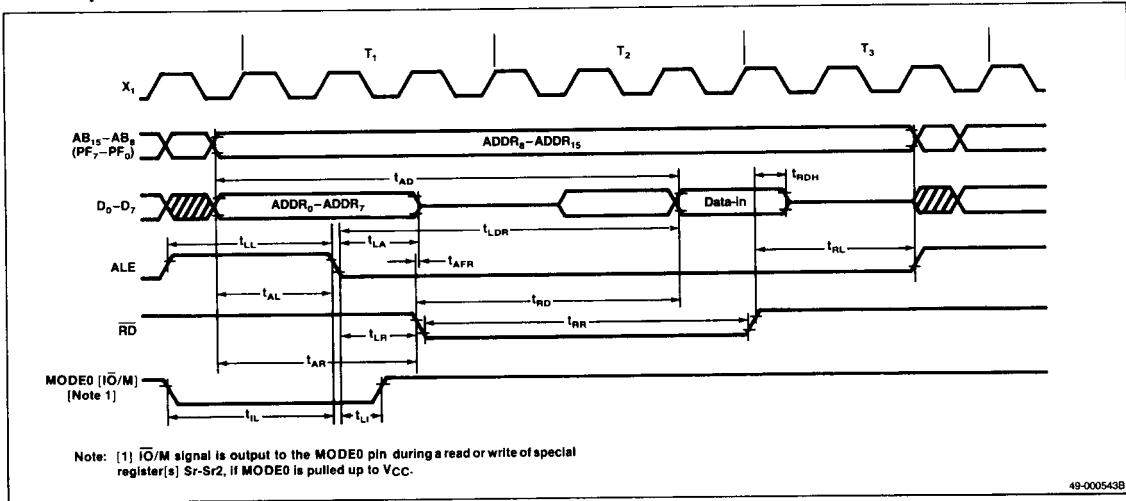
**Note:**

- (1) 1x Baud rate in asynchronous, synchronous, or I/O interface mode.  
 $T = t_{CYC} = 1/f_{XTAL}$ .  
 The items not included in this list are independent of oscillator frequency ( $f_{XTAL}$ ).
- (2) Event counter mode.
- (3) Pulse width measurement mode.

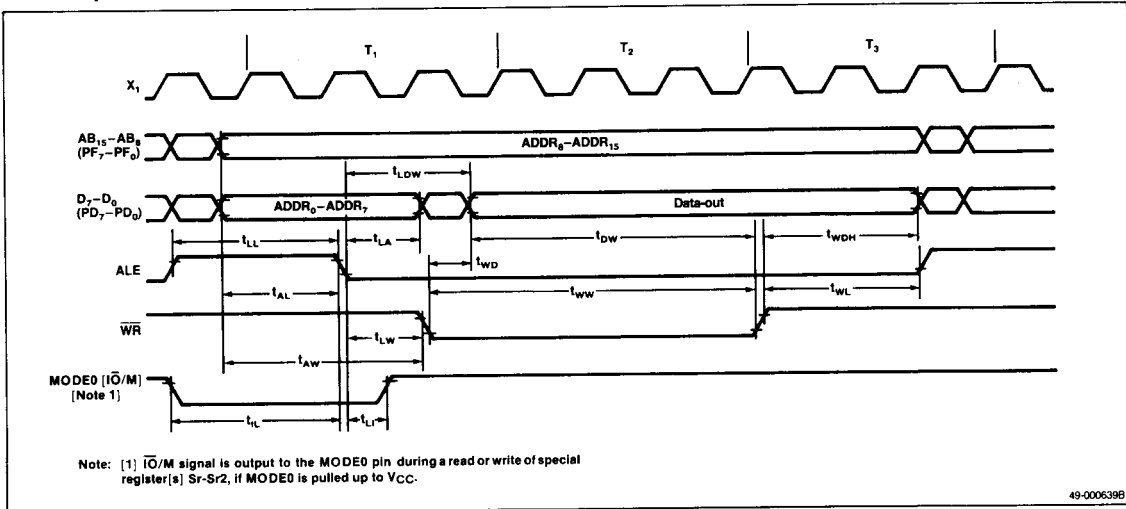


**Timing Waveforms**

**Read Operation**

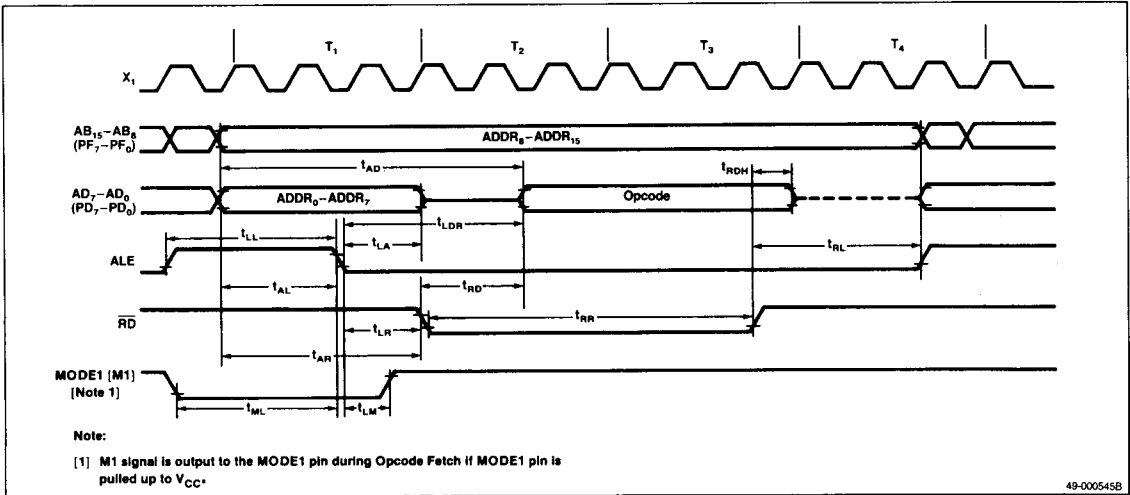


**Write Operation**

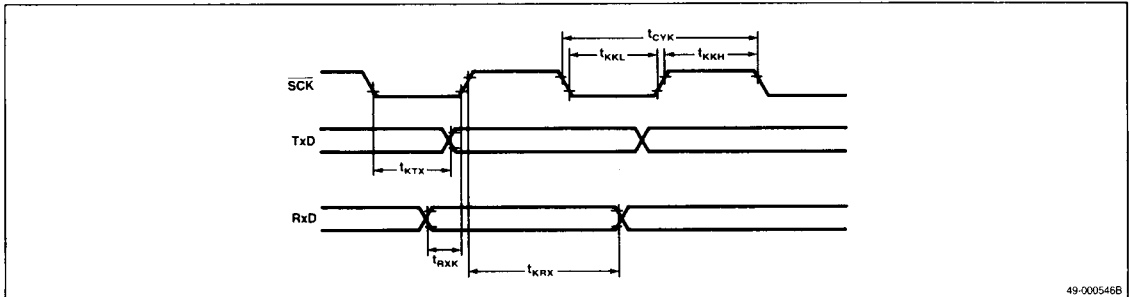


### Timing Waveforms (cont)

#### Opcode Fetch Operation



#### Serial Operation Transmit/Receive Timing



## Operand Format/Description

Format	Description
r	V, A, B, C, D, E, H, L
r1	EAH, EAL, B, C, D, E, H, L
r2	A, B, C
sr	PA, PB, PC, PD, PF, MKH, MKL, ANM, SMH, SML, EOM, ETMM, TMM, MM, MCC, MA, MB, MC, MF, TxB, TM <sub>0</sub> , TM <sub>1</sub>
sr1	PA, PB, PC, PD, PF, MKH, MKL, ANM, SMH, EOM, TMM, RXB, CR0, CR1, CR2, CR3
sr2	PA, PB, PC, PD, PF, MKH, ANM, MKL, SMH, EOM, TMM
sr3	ETM <sub>0</sub> , ETM <sub>1</sub>
sr4	ECNT, ECPT
rp	SP, B, D, H
rp1	V, B, D, H, EA
rp2	SP, B, D, H, EA
rp3	B, D, H
rpa	B, D, H, D+, H+, D-, H-
rpa1	B, D, H
rpa2	B, D, H, D+, H+, D-, H-, D + byte, H + A, H + B, H + EA, H + byte
rpa3	D, H, D + +, H + +, D + byte, H + A, H + B, H + EA, H + byte
wa	8-Bit immediate data
word	16-Bit immediate data
byte	8-Bit immediate data
bit	3-Bit immediate data
f	CY, HC, Z
irf	FNMI, FT0, FT1, F1, F2, FE0, FE1, FEIN, FAD, FSR, FST, ER, OV, AN <sub>4</sub> , AN <sub>5</sub> , AN <sub>6</sub> , AN <sub>7</sub> , SB

## Instruction Set Symbol Definitions

Symbol	Description
←	Transfer direction, result
∧	Logical product (logical AND)
∨	Logical sum (logical OR)
⊕	Exclusive OR
—	Complement
•	Concatenation

## Remarks

### 1. sr-sr4 (special register)

PA = Port A	ECNT = Timer/Event Counter Upcounter
PB = Port B	ECPT = Timer/Event Counter Capture
PC = Port C	
PD = Port D	
PF = Port F	
MA = Mode A	ETMM = Timer/Event Counter Mode
MB = Mode B	EOM = Timer/Event Counter Output Mode
MC = Mode C	
MCC = Mode Control C	
MF = Mode F	
	TxB = TX Buffer
	RxB = RX Buffer
MM = Memory Mapping	SMH = Serial Mode High
TM <sub>0</sub> = Timer Register 0	SML = Serial Mode Low
TM <sub>1</sub> = Timer Register 1	MKH = Mask High
TMM = Timer Mode	MKL = Mask Low
ETM <sub>0</sub> = Timer/Event Counter Register 0	ANM = A/D Channel Mode
ETM <sub>1</sub> = Timer/Event Counter Register 1	CR <sub>0</sub> = A/D Conversion Result 0-3 to CR <sub>3</sub>

### 2. rp-rp3 (register pair)

SP = Stack Pointer	H = HL
B = BC	V = VA
D = DE	EA = Extended Accumulator

### 3. rpa-rpa3 (rp addressing)

B = (BC)	D + + = (DE) ++
D = (DE)	H + + = (HL) ++
H = (HL)	D + byte = (DE) + byte
D + = (DE) +	H + A = (HL) + (A)
H - = (HL) +	H + B = (HL) + (B)
D - = (DE) -	H + EA = (HL) + (EA)
H - = (HL) -	H + byte = (HL) + byte

### 4. f (flag)

CY = Carry	HC = Half Carry	Z = Zero
------------	-----------------	----------

### 5. irf (interrupt flag)

NMI = NMI* Input	FEIN = INTFEIN
	FAD = INTFAD
FT0 = INTFT0	FSR = INTFSR
FT1 = INTFT1	FST = INTFST
F1 = INTF1	ER = Error
F2 = INTF2	OV = Overflow
FE0 = INTFE0	AN <sub>4</sub> to AN <sub>7</sub> = Analog Input 4-7
FE1 = INTFE1	SB = Standby



**Instruction Set**

Mnemonic	Operand	Operation	Operation Code																Bytes	State(1)	Skip Condition				
			B1	B2	B3	B4	7	6	5	4	3	2	1	0	High addr	Low addr	Data	Offset							
<b>8-Bit Data Transfer</b>																									
MOV	r1,A (r1) ← (A)		0	0	0	1	1	T <sub>2</sub>	T <sub>1</sub>	T <sub>0</sub>	4	1	A, r1 (A) ← (r1)		0	0	0	0	1	T <sub>2</sub>	T <sub>1</sub>	T <sub>0</sub>	4	1	
	*sr,A (sr) ← (A)		0	1	0	0	1	1	0	1	10	2	*A, sr1 (A) ← (sr1)		0	1	0	0	1	1	0	0	10	2	
	r, word (r) ← (word)		0	1	1	0	0	0	0	0	17	4		Low addr	0	1	1	0	0	0	0	0	17	4	
	word, r (word) ← (r)		0	1	1	0	0	0	0	0	17	4		High addr	0	1	1	0	0	0	0	0	17	4	
MVI	*r, byte (r) ← byte set L1 if r = A set L0 if r = L		0	1	1	0	1	R <sub>2</sub>	R <sub>1</sub>	R <sub>0</sub>	7	2		Low addr	0	1	1	0	1	R <sub>2</sub>	R <sub>1</sub>	R <sub>0</sub>	7	2	L1 = 1 and r = A L0 = 1 and r = L
MOVW	*wa, byte ((V) ← (wa)) ← byte		0	1	1	0	0	0	0	1	13	3	sr2, byte (sr2) ← byte		0	1	1	0	0	0	0	1	14	3	
	*rpa1, byte (rpa1) ← byte		0	1	0	0	1	A <sub>1</sub>	A <sub>0</sub>	10	2		Data	0	1	1	0	0	0	1	13	3			
STAW	*wa ((V) ← (wa)) ← A		0	1	1	0	0	0	1	1	10	2		Data	0	1	0	0	1	0	1	10	2		
LDAW	*wa (A) ← ((V) ← (wa))		0	0	0	0	0	0	0	1	10	2		Offset	0	0	0	0	0	0	1	10	2		
STAX	*rpa2 (rpa2) ← (A)		A <sub>3</sub>	0	1	1	A <sub>2</sub>	A <sub>1</sub>	A <sub>0</sub>	7/13(3)	2		Data (2)	A <sub>3</sub>	0	1	1	A <sub>2</sub>	A <sub>1</sub>	A <sub>0</sub>	7/13(3)	2			
LDAX	*rpa2 (A) ← ((rpa2))		A <sub>3</sub>	0	1	0	A <sub>2</sub>	A <sub>1</sub>	A <sub>0</sub>	7/13(3)	2		Data (2)	A <sub>3</sub>	0	1	0	A <sub>2</sub>	A <sub>1</sub>	A <sub>0</sub>	7/13(3)	2			
EXX	(B) ↔ (B'), (C) ↔ (C'), (D) ↔ (D') (E) ↔ (E'), (H) ↔ (H'), (L) ↔ (L')		0	0	0	1	0	0	0	1	4	1		Offset	0	0	0	1	0	0	0	1	4	1	
EXA	(V) ↔ (V'), (A) ↔ (A'), (EA) ↔ (EA')		0	0	0	1	0	0	0	0	4	1		Offset	0	0	0	1	0	0	0	0	4	1	
EXH	(H) ↔ (H'), (L) ↔ (L')		0	1	0	1	0	0	0	0	4	1		Offset	0	1	0	1	0	0	0	0	4	1	
<b>16-Bit Data Transfer</b>																									
BLOCK	D ((DE) ← ((HL)), (DE) ← (DE + 1), (HL) ← (HL) + 1, (C) ← (C) - 1 End if borrow		0	0	1	1	0	0	0	1	13 x (C + 1)	1		Offset	0	0	1	1	0	0	0	1	13 x (C + 1)	1	
DMOV	rp3, EA (rp3) ← (EA), (rp3H) ← (EAH) EA, rp3 (EA) ← (rp3L), (EAH) ← (rp3H)		1	0	1	1	0	1	P <sub>1</sub>	P <sub>0</sub>	4	1		Offset	1	0	1	1	0	1	P <sub>1</sub>	P <sub>0</sub>	4	1	
	EA, rp3 (EA) ← (rp3L), (EAH) ← (rp3H)		1	0	1	0	0	1	P <sub>1</sub>	P <sub>0</sub>	4	1		Offset	1	0	1	0	0	1	P <sub>1</sub>	P <sub>0</sub>	4	1	

## Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code																Bytes	Skip Condition
			B1				B2				B3				B4					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0		
<b>16-Bit Data Transfer (cont)</b>																				
DMOV	sr3, EA (sr3) ← (EA) EA, sr4 (EA) ← (sr4)		0	1	0	0	1	0	0	0	1	1	0	1	0	0	1	U <sub>0</sub>	14	2
SBCD	word (word) ← (C), (word + 1) ← (B)		0	1	1	1	0	0	0	0	0	0	0	1	1	1	0	20	4	
		Low addr	High addr																	
SOED	word (word) ← (E), (word + 1) ← (D)		0	1	1	1	0	0	0	0	0	0	1	0	1	1	1	0	20	4
		Low addr	High addr																	
SHLD	word (word) ← (L), (word + 1) ← (H)		0	1	1	1	0	0	0	0	0	0	1	1	1	1	0	20	4	
		Low addr	High addr																	
SSPD	word (word) ← (SP <sub>L</sub> ), (word + 1) ← (SP <sub>H</sub> )		0	1	1	1	0	0	0	0	0	0	0	0	1	1	1	0	20	4
		Low addr	High addr																	
STEAX	rpa3 ((rpa3)) ← (EAL), ((rpa3) + 1) ← (EAH)		0	1	0	0	1	0	0	0	1	0	0	1	C <sub>3</sub>	C <sub>2</sub>	C <sub>1</sub>	C <sub>0</sub>	14/20(3)	3
		Data(4)																		
LBCD	word (C) ← (word), (B) ← (word + 1)		0	1	1	1	0	0	0	0	0	0	0	1	1	1	1	1	20	4
		Low addr	High addr																	
LDED	word (E) ← (word), (D) ← (word + 1)		0	1	1	1	0	0	0	0	0	0	1	0	1	1	1	1	20	4
		Low addr	High addr																	
LHLD	word (L) ← (word), (H) ← (word + 1)		0	1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	20	4
		Low addr	High addr																	
LSPD	word (SP <sub>L</sub> ) ← (word), (SP <sub>H</sub> ) ← ((word) + 1)		0	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	20	4
		Low addr	High addr																	
LDEAX	rpa3 (EAL) ← ((rpa3)), (EAH) ← ((rpa3) + 1)		0	1	0	0	1	0	0	0	1	0	0	0	C <sub>3</sub>	C <sub>2</sub>	C <sub>1</sub>	C <sub>0</sub>	14/20(3)	3
		Data(4)																		
PUSH	rp1 ((SP) - 1) ← (rp1 <sub>H</sub> ), ((SP) - 2) ← (rp1 <sub>L</sub> ) (SP) ← (SP) - 2		1	0	1	1	0	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>								13	1	
POP	rp1 (rp1 <sub>L</sub> ) ← ((SP)), (rp1 <sub>H</sub> ) ← ((SP) + 1) (SP) ← (SP) + 2		1	0	1	0	0	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>								10	1	
LXI	*rp2, word (rp2) ← (word) set L0 if rp2 = H		0	P <sub>2</sub>	P <sub>1</sub>	P <sub>0</sub>	0	1	0	0	Low byte							10	3	
		High byte																	L0 = 1 and rp2 = H	
TABLE	(C) ← ((PC)+3+(A)), B ← ((PC)+3+(A)+1)		0	1	0	0	1	0	0	0	1	0	1	0	0	0	0	17	2	
<b>8-Bit Arithmetic (Register)</b>																				
ADD	A, r (A) ← (A) + (r)		0	1	1	0	0	0	0	0	1	1	0	0	R <sub>2</sub>	R <sub>1</sub>	R <sub>0</sub>	8	2	
	r, A (r) ← (r) + (A)		0	1	1	0	0	0	0	0	0	1	0	0	R <sub>2</sub>	R <sub>1</sub>	R <sub>0</sub>	8	2	
ADC	A, r (A) ← (A) + (r) + (CY)		0	1	1	0	0	0	0	0	1	1	0	1	R <sub>2</sub>	R <sub>1</sub>	R <sub>0</sub>	8	2	
	r, A (r) ← (r) + (A) + (CY)		0	1	1	0	0	0	0	0	0	1	0	1	R <sub>2</sub>	R <sub>1</sub>	R <sub>0</sub>	8	2	

**Instruction Set (cont)**

Mnemonic	Operand	Operation	Operation Code																Skip Condition		
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0		State(I)	Bytes
<b>8-Bit Arithmetic [Register] (cont)</b>																					
ADDNC	$A_r (A) \leftarrow (A) + (r)$ $r_r (r) \leftarrow (r) + (A)$		0	1	1	0	0	0	0	0	0	0	1	0	1	0	0	R <sub>2</sub> R <sub>1</sub> R <sub>0</sub>	8	2	No carry
SUB	$A_r (A) \leftarrow (A) - (r)$		0	1	1	0	0	0	0	0	0	1	1	1	0	0	0	R <sub>2</sub> R <sub>1</sub> R <sub>0</sub>	8	2	No carry
SBB	$r_r (r) \leftarrow (r) - (A)$ $A_r (A) \leftarrow (A) - (r) - (CY)$		0	1	1	0	0	0	0	0	0	1	1	1	1	0	0	R <sub>2</sub> R <sub>1</sub> R <sub>0</sub>	8	2	
SUBNB	$r_r (r) \leftarrow (r) - (A) - (CY)$ $A_r (A) \leftarrow (A) - (r)$		0	1	1	0	0	0	0	0	0	1	1	1	1	0	0	R <sub>2</sub> R <sub>1</sub> R <sub>0</sub>	8	2	No borrow
ANA	$r_r (r) \leftarrow (r) \wedge (r)$		0	1	1	0	0	0	0	0	1	0	0	0	1	0	0	R <sub>2</sub> R <sub>1</sub> R <sub>0</sub>	8	2	
ORA	$r_r (r) \leftarrow (r) \vee (r)$		0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	R <sub>2</sub> R <sub>1</sub> R <sub>0</sub>	8	2	
XRA	$r_r (r) \leftarrow (r) \oplus (r)$		0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	R <sub>2</sub> R <sub>1</sub> R <sub>0</sub>	8	2	
GTA	$r_r (r) \leftarrow (r) \wedge (A)$ $r_r (r) \leftarrow (r) - 1$		0	1	1	0	0	0	0	0	0	1	0	1	0	1	0	R <sub>2</sub> R <sub>1</sub> R <sub>0</sub>	8	2	No borrow
LTA	$r_r (r) \leftarrow (r) - 1$ $r_r (r) \leftarrow (r)$		0	1	1	0	0	0	0	0	0	0	0	1	0	1	1	R <sub>2</sub> R <sub>1</sub> R <sub>0</sub>	8	2	No borrow
NEA	$r_r (r) \leftarrow (r)$		0	1	1	0	0	0	0	0	0	0	0	1	1	0	1	R <sub>2</sub> R <sub>1</sub> R <sub>0</sub>	8	2	Borrow
EQA	$r_r (r) \leftarrow (r)$		0	1	1	0	0	0	0	0	0	0	1	1	0	1	0	R <sub>2</sub> R <sub>1</sub> R <sub>0</sub>	8	2	Zero
ONA	$r_r (r) \leftarrow (r) \wedge (r)$		0	1	1	0	0	0	0	0	0	1	1	0	1	1	0	R <sub>2</sub> R <sub>1</sub> R <sub>0</sub>	8	2	Zero
OFFA	$r_r (r) \leftarrow (r) \wedge (r)$		0	1	1	0	0	0	0	0	1	1	0	1	1	0	1	R <sub>2</sub> R <sub>1</sub> R <sub>0</sub>	8	2	Zero
<b>8-Bit Arithmetic (Memory)</b>																					
ADDCX	$rpa (A) \leftarrow (A) + (rpa)$		0	1	1	1	0	0	0	0	0	1	1	0	0	0	A <sub>2</sub> A <sub>1</sub> A <sub>0</sub>	11	2		
ADCCX	$rpa (A) \leftarrow (A) + (rpa) + (CY)$		0	1	1	1	0	0	0	0	1	1	0	1	0	1	A <sub>2</sub> A <sub>1</sub> A <sub>0</sub>	11	2		
ADDNCX	$rpa (A) \leftarrow (A) + (rpa)$		0	1	1	1	0	0	0	0	1	0	1	0	0	0	A <sub>2</sub> A <sub>1</sub> A <sub>0</sub>	11	2	No carry	
SUBCX	$rpa (A) \leftarrow (A) - (rpa)$		0	1	1	1	0	0	0	0	1	1	1	0	0	0	A <sub>2</sub> A <sub>1</sub> A <sub>0</sub>	11	2		
SBBX	$rpa (A) \leftarrow (A) - (rpa) - (CY)$		0	1	1	1	0	0	0	0	1	1	1	1	0	0	A <sub>2</sub> A <sub>1</sub> A <sub>0</sub>	11	2		
SUBNBX	$rpa (A) \leftarrow (A) - (rpa)$		0	1	1	1	0	0	0	0	1	0	1	1	0	1	A <sub>2</sub> A <sub>1</sub> A <sub>0</sub>	11	2	No borrow	
ANAX	$rpa (A) \leftarrow (A) \wedge (rpa)$		0	1	1	1	0	0	0	0	1	0	0	0	1	0	0	A <sub>2</sub> A <sub>1</sub> A <sub>0</sub>	11	2	
ORAX	$rpa (A) \leftarrow (A) \vee (rpa)$		0	1	1	1	0	0	0	0	1	0	0	1	0	1	1	A <sub>2</sub> A <sub>1</sub> A <sub>0</sub>	11	2	

## Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code																Bytes	Skip Condition	
			B1	B2	B3	B4	7	6	5	4	3	2	1	0	State(1)						
<b>8-Bit Arithmetic (Memory) (cont)</b>																					
XRAX	rpa (A) ← (A) ∨ ((rpa))		0	1	1	1	0	0	0	0	1	0	0	1	0	1	0	A <sub>2</sub> A <sub>1</sub> A <sub>0</sub>	11	2	No borrow
GTAX	rpa (A) ← ((rpa)) - 1		0	1	1	1	0	0	0	0	1	0	1	0	1	0	1	A <sub>2</sub> A <sub>1</sub> A <sub>0</sub>	11	2	Borrow
LTAX	rpa (A) ← ((rpa))		0	1	1	1	0	0	0	0	1	0	1	1	1	0	1	A <sub>2</sub> A <sub>1</sub> A <sub>0</sub>	11	2	No zero
NEAX	rpa (A) ← ((rpa))		0	1	1	1	0	0	0	0	1	1	1	0	1	0	1	A <sub>2</sub> A <sub>1</sub> A <sub>0</sub>	11	2	Zero
EOAX	rpa (A) ← ((rpa))		0	1	1	1	0	0	0	0	1	1	1	1	1	0	1	A <sub>2</sub> A <sub>1</sub> A <sub>0</sub>	11	2	No zero
ONAX	rpa (A) ∧ ((rpa))		0	1	1	1	0	0	0	0	1	1	0	0	1	0	1	A <sub>2</sub> A <sub>1</sub> A <sub>0</sub>	11	2	Zero
OFFAX	rpa (A) ∧ ((rpa))		0	1	1	1	0	0	0	0	1	1	0	1	1	0	1	A <sub>2</sub> A <sub>1</sub> A <sub>0</sub>	11	2	Zero
<b>Immediate Data</b>																					
ADI	*A.byte (A) ← (A) + byte		0	1	0	0	0	1	1	0	Data						7	2			
	r.byte (r) ← (r) + byte		0	1	1	1	0	1	0	0	Data						11	3			
	sr2.byte (sr2) ← (sr2) + byte		0	1	1	0	0	1	0	0	Data						20	3			
ACI	*A.byte (A) ← (A) + byte + (CY)		0	1	0	1	0	1	1	0	Data						7	2			
	r.byte (r) ← (r) + byte + (CY)		0	1	1	1	0	1	0	0	Data						11	3			
	sr2.byte (sr2) ← (sr2) + byte + (CY)		0	1	1	0	0	1	0	0	Data						20	3			
ADINC	*A.byte (A) ← (A) + byte		0	0	1	0	0	1	1	0	Data						7	2	No carry		
	r.byte (r) ← (r) + byte		0	1	1	1	0	1	0	0	Data						11	3	No carry		
	sr2.byte (sr2) ← (sr2) + byte		0	1	1	0	0	1	0	0	Data						20	3	No carry		
SUI	*A.byte (A) ← (A) - byte		0	1	1	0	0	1	1	0	Data						7	2			
	r.byte (r) ← (r) - byte		0	1	1	1	0	1	0	0	Data						11	3			
	sr2.byte (sr2) ← (sr2) - byte		0	1	1	0	0	1	0	0	Data						20	3			
SBI	*A.byte (A) ← (A) - byte - (CY)		0	1	1	1	0	1	1	0	Data						7	2			
	r.byte (r) ← (r) - byte - (CY)		0	1	1	1	0	1	0	0	Data						11	3			
	sr2.byte (sr2) ← (sr2) - byte - (CY)		0	1	1	0	0	1	0	0	Data						20	3			

**Instruction Set (cont)**

Mnemonic	Operand	Operation	Operation Code																Bytes	Skip Condition
			B1	B2	B3	B4	7	6	5	4	3	2	1	0	State(I)					
<b>Immediate Data (cont)</b>																				
SUIB	*A.byte (A) ← (A) - byte		0	0	1	1	0	1	1	0	Data						7	2	No borrow	
	r.byte (r) ← (r) - byte		0	1	1	0	1	0	0	0	Data						11	3	No borrow	
	sr2.byte (sr2) ← (sr2) - byte		0	1	1	0	1	0	0	S <sub>3</sub>	0	1	1	0	S <sub>2</sub>	S <sub>1</sub>	S <sub>0</sub>	20	3	No borrow
ANI	*A.byte (A) ← (A) ∧ byte		0	0	0	0	1	1	1	Data						7	2			
	r.byte (r) ← (r) ∧ byte		0	1	1	0	1	0	0	Data						11	3			
	sr2.byte (sr2) ← (sr2) ∧ byte		0	1	1	0	1	0	0	S <sub>3</sub>	0	0	0	1	S <sub>2</sub>	S <sub>1</sub>	S <sub>0</sub>	20	3	
ORI	*A.byte (A) ← (A) ∨ byte		0	0	0	1	0	1	1	Data						7	2			
	r.byte (r) ← (r) ∨ byte		0	1	1	0	1	0	0	Data						11	3			
	sr2.byte (sr2) ← (sr2) ∨ byte		0	1	1	0	1	0	0	S <sub>3</sub>	0	0	1	1	S <sub>2</sub>	S <sub>1</sub>	S <sub>0</sub>	20	3	
XRI	*A.byte (A) ← (A) ⊕ byte		0	0	0	1	0	1	1	Data						7	2			
	r.byte (r) ← (r) ⊕ byte		0	1	1	0	1	0	0	Data						11	3			
	sr2.byte (sr2) ← (sr2) ⊕ byte		0	1	1	0	1	0	0	S <sub>3</sub>	0	0	1	1	S <sub>2</sub>	S <sub>1</sub>	S <sub>0</sub>	20	3	
GTI	*A.byte (A) - byte - 1		0	0	1	0	0	1	1	Data						7	2	No borrow		
	r.byte (r) - byte - 1		0	1	1	0	1	0	0	Data						11	3	No borrow		
	sr2.byte (sr2) - byte - 1		0	1	1	0	0	1	0	S <sub>3</sub>	0	1	0	1	S <sub>2</sub>	S <sub>1</sub>	S <sub>0</sub>	14	3	No borrow
LTI	*A.byte (A) - byte		0	0	1	1	0	1	1	Data						7	2	Borrow		
	r.byte (r) - byte		0	1	1	0	1	0	0	Data						11	3	Borrow		
	sr2.byte (sr2) - byte		0	1	1	0	1	0	0	S <sub>3</sub>	0	1	1	1	S <sub>2</sub>	S <sub>1</sub>	S <sub>0</sub>	14	3	Borrow
NEI	*A.byte (A) - byte		0	1	1	0	0	1	1	Data						7	2	No zero		
	r.byte (r) - byte		0	1	1	0	1	0	0	Data						11	3	No zero		

## Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code																Bytes	Skip Condition	
			B1	B2	B3	B4	7	6	5	4	3	2	1	0	State(I)						
Immediate Data (cont)																					
NEI	sr2.byte (sr2) - byte		0	1	1	0	0	1	0	0	S <sub>3</sub>	1	1	0	1	S <sub>2</sub>	S <sub>1</sub>	S <sub>0</sub>	14	3	No zero
Data																					
EI	*A.byte (A) - byte		0	1	1	1	0	1	1	1	Data								7	2	Zero
	r.byte (r) - byte		0	1	1	1	0	1	0	0	0	1	1	1	1	R <sub>2</sub>	R <sub>1</sub>	R <sub>0</sub>	11	3	Zero
Data																					
	sr2.byte (sr2) - byte		0	1	1	0	0	1	0	0	S <sub>3</sub>	1	1	1	1	S <sub>2</sub>	S <sub>1</sub>	S <sub>0</sub>	14	3	Zero
Data																					
ONI	*A.byte (A) ^ byte		0	1	0	0	0	1	1	1	Data								7	2	No zero
	r.byte (r) ^ byte		0	1	1	1	0	1	0	0	0	1	0	0	1	R <sub>2</sub>	R <sub>1</sub>	R <sub>0</sub>	11	3	No zero
Data																					
	sr2.byte (sr2) ^ byte		0	1	1	0	0	1	0	0	S <sub>3</sub>	1	0	0	1	S <sub>2</sub>	S <sub>1</sub>	S <sub>0</sub>	14	3	No zero
Data																					
OFFI	*A.byte (A) ^ byte		0	1	0	1	0	1	1	1	Data								7	2	Zero
	r.byte (r) ^ byte		0	1	1	1	0	1	0	0	0	1	0	1	1	R <sub>2</sub>	R <sub>1</sub>	R <sub>0</sub>	11	3	Zero
Data																					
	sr2.byte (sr2) ^ byte		0	1	1	0	0	1	0	0	S <sub>3</sub>	1	0	1	1	S <sub>2</sub>	S <sub>1</sub>	S <sub>0</sub>	14	3	Zero
Data																					
Working Register																					
ADDW	wa (A) ← (A) + ((V)•(wa))		0	1	1	1	0	1	0	0	1	1	0	0	0	0	0	0	14	3	
Offset																					
ADCW	wa (A) ← (A) + ((V)•(wa)) + (CY)		0	1	1	1	0	1	0	0	1	1	0	1	0	0	0	0	14	3	
Offset																					
ADDNCW	wa (A) ← (A) + ((V)•(wa))		0	1	1	1	0	1	0	0	1	0	1	0	0	0	0	0	14	3	No carry
Offset																					
SUBW	wa (A) ← (A) - ((V)•(wa))		0	1	1	1	0	1	0	0	1	1	1	0	0	0	0	0	14	3	
Offset																					
SBBW	wa (A) ← (A) - ((V)•(wa)) - (CY)		0	1	1	1	0	1	0	0	1	1	1	1	0	0	0	0	14	3	
Offset																					
SUBNBW	wa (A) ← (A) - ((V)•(wa))		0	1	1	1	0	1	0	0	1	0	1	1	0	0	0	0	14	3	No borrow
Offset																					
ANAW	wa (A) ← (A) ^ ((V)•(wa))		0	1	1	1	0	1	0	0	1	0	0	0	1	0	0	0	14	3	
Offset																					

**Instruction Set (cont)**

Mnemonic	Operand	Operation	Operation Code																Skip Condition								
			B1	B2	B3	B4	7	6	5	4	3	2	1	0	7	6	5	4		3	2	1	0	State(I)	Bytes		
<b>Working Register (cont)</b>																											
ORAW	wa	$(A) \leftarrow (A) \vee ((V) \bullet (wa))$	0	1	1	1	0	1	0	0	1	0	0	1	0	0	1	0	0	0	0	14	3				
			Offset																								
XRAW	wa	$(A) \leftarrow (A) \oplus ((V) \bullet (wa))$	0	1	1	1	0	1	0	0	1	0	0	1	0	0	1	0	0	0	0	14	3				
			Offset																								
GTAW	wa	$(A) \leftarrow ((V) \bullet (wa)) - 1$	0	1	1	1	0	1	0	0	1	0	1	0	0	1	0	0	0	0	0	14	3				
			Offset																								
LTAW	wa	$(A) \leftarrow ((V) \bullet (wa))$	0	1	1	1	0	1	0	0	1	0	1	1	0	0	0	0	0	0	0	14	3				
			Offset																								
NEAW	wa	$(A) \leftarrow ((V) \bullet (wa))$	0	1	1	1	0	1	0	0	1	1	0	1	0	0	0	0	0	0	0	14	3				
			Offset																								
EDAW	wa	$(A) \leftarrow ((V) \bullet (wa))$	0	1	1	1	0	1	0	0	1	1	1	1	0	0	0	0	0	0	0	14	3				
			Offset																								
ONAW	wa	$(A) \wedge ((V) \bullet (wa))$	0	1	1	1	0	1	0	0	1	1	0	0	1	0	0	0	0	0	0	14	3				
			Offset																								
OFFAW	wa	$(A) \wedge ((V) \bullet (wa))$	0	1	1	1	0	1	0	0	1	1	0	1	0	0	0	0	0	0	0	14	3				
			Offset																								
ANIW	*wa.byte	$((V) \bullet (wa)) \leftarrow ((V) \bullet (wa)) \wedge \text{byte}$	0	0	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	19	3					
			Data																								
ORIW	*wa.byte	$((V) \bullet (wa)) \leftarrow ((V) \bullet (wa)) \vee \text{byte}$	0	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	19	3					
			Data																								
GTIW	*wa.byte	$((V) \bullet (wa)) - \text{byte} - 1$	0	0	1	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	13	3					
			Data																								
LTIW	*wa.byte	$((V) \bullet (wa)) - \text{byte}$	0	0	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	13	3					
			Data																								
NEIW	*wa.byte	$((V) \bullet (wa)) - \text{byte}$	0	1	1	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	13	3					
			Data																								
EQIW	*wa.byte	$((V) \bullet (wa)) - \text{byte}$	0	1	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	13	3					
			Data																								
ONIW	*wa.byte	$((V) \bullet (wa)) \wedge \text{byte}$	0	1	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	13	3					
			Data																								
OFFIW	*wa.byte	$((V) \bullet (wa)) \wedge \text{byte}$	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	13	3					
			Data																								

### Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code																Bytes	State(1)	Skip Condition									
			B1				B2				B3				B4															
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0				7	6	5	4	3	2	1	0	
<b>16-Bit Arithmetic</b>																														
EADD	EA,r2	$(EA) \leftarrow (EA) + (r2)$	0	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	R <sub>1</sub>	R <sub>0</sub>	11	2			
DADD	EA,rp3	$(EA) \leftarrow (EA) + (rp3)$	0	1	1	1	0	1	0	0	0	1	1	0	0	0	1	P <sub>1</sub>	P <sub>0</sub>	11										
DADD	EA,rp3	$(EA) \leftarrow (EA) + (rp3) + (CY)$	0	1	1	1	0	1	0	0	1	1	0	1	0	1	P <sub>1</sub>	P <sub>0</sub>	11											
DADDNC	EA,rp3	$(EA) \leftarrow (EA) + (rp3)$	0	1	1	1	0	1	0	0	1	0	1	0	0	1	P <sub>1</sub>	P <sub>0</sub>	11										No carry	
ESUB	EA,r2	$(EA) \leftarrow (EA) - (r2)$	0	1	1	1	0	0	0	0	0	1	1	0	0	0	R <sub>1</sub>	R <sub>0</sub>	11											
DSUB	EA,rp3	$(EA) \leftarrow (EA) - (rp3)$	0	1	1	1	0	1	0	0	1	1	1	0	0	1	P <sub>1</sub>	P <sub>0</sub>	11											
DSBB	EA,rp3	$(EA) \leftarrow (EA) - (rp3) - (CY)$	0	1	1	1	0	1	0	0	1	1	1	0	1	0	1	P <sub>1</sub>	P <sub>0</sub>	11										
DSUBNB	EA,rp3	$(EA) \leftarrow (EA) - (rp3)$	0	1	1	1	0	1	0	0	1	0	1	0	1	0	1	P <sub>1</sub>	P <sub>0</sub>	11										No borrow
DAN	EA,rp3	$(EA) \leftarrow (EA) \wedge (rp3)$	0	1	1	1	0	1	0	0	1	0	0	1	0	1	P <sub>1</sub>	P <sub>0</sub>	11											
DOR	EA,rp3	$(EA) \leftarrow (EA) \vee (rp3)$	0	1	1	1	0	1	0	0	1	0	0	1	0	1	P <sub>1</sub>	P <sub>0</sub>	11											
DXR	EA,rp3	$(EA) \leftarrow (EA) \oplus (rp3)$	0	1	1	1	0	1	0	0	1	0	0	1	0	1	P <sub>1</sub>	P <sub>0</sub>	11											
DGT	EA,rp3	$(EA) \leftarrow (rp3) - 1$	0	1	1	1	0	1	0	0	1	0	1	0	0	1	P <sub>1</sub>	P <sub>0</sub>	11										No borrow	
DLT	EA,rp3	$(EA) \leftarrow (rp3)$	0	1	1	1	0	1	0	0	1	0	1	0	1	1	P <sub>1</sub>	P <sub>0</sub>	11										Borrow	
DNE	EA,rp3	$(EA) \leftarrow (rp3)$	0	1	1	1	0	1	0	0	1	1	0	1	0	1	P <sub>1</sub>	P <sub>0</sub>	11										No zero	
DEO	EA,rp3	$(EA) \leftarrow (rp3)$	0	1	1	1	0	1	0	0	1	1	1	0	1	1	P <sub>1</sub>	P <sub>0</sub>	11										Zero	
DON	EA,rp3	$(EA) \wedge (rp3)$	0	1	1	1	0	1	0	0	1	1	0	0	1	0	1	P <sub>1</sub>	P <sub>0</sub>	11									No zero	
DOFF	EA,rp3	$(EA) \wedge (rp3)$	0	1	1	1	0	1	0	0	1	1	0	1	1	0	1	P <sub>1</sub>	P <sub>0</sub>	11									Zero	
<b>Multiply/Divide</b>																														
MUL	r2	$(EA) \leftarrow (A) \times (r2)$	0	1	0	0	1	0	0	0	0	0	0	1	0	1	R <sub>1</sub>	R <sub>0</sub>	32											
DIV	r2	$(EA) \leftarrow (EA) + (r2), (r2) \leftarrow \text{Remainder}$	0	1	0	0	1	0	0	0	0	0	0	1	1	1	R <sub>1</sub>	R <sub>0</sub>	59											
<b>Increment/Decrement</b>																														
INR	r2	$(r2) \leftarrow (r2) + 1$	0	1	0	0	0	0	0	R <sub>1</sub>	R <sub>0</sub>									4									1	Carry
INRW	*wa	$((V) \leftarrow (wa)) \leftarrow ((V) \leftarrow (wa)) + 1$	0	0	1	0	0	0	0	0										16									2	Carry
INX	rp	$(rp) \leftarrow (rp) + 1$	0	0	P <sub>1</sub>	P <sub>0</sub>	0	0	1	0										7										1
	EA	$(EA) \leftarrow (EA) + 1$	1	0	1	0	1	0	0	0										7										1
DCR	r2	$(r2) \leftarrow (r2) - 1$	0	1	0	1	0	0	0	R <sub>1</sub>	R <sub>0</sub>									4									1	Borrow
DCRW	*wa	$((V) \leftarrow (wa)) \leftarrow ((V) \leftarrow (wa)) - 1$	0	0	1	1	0	0	0											16									2	Borrow
DCX	rp	$(rp) \leftarrow (rp) - 1$	0	0	P <sub>1</sub>	P <sub>0</sub>	0	0	1	1										7										1
	EA	$(EA) \leftarrow (EA) - 1$	1	0	1	0	1	0	0	1										7										1
<b>Others</b>																														
DAA		Decimal Adjust Accumulator	0	1	1	0	0	0	0	1										4										1
STC		$(CY) \leftarrow 1$	0	1	0	0	1	0	0	0										8										2
CLC		$(CY) \leftarrow 0$	0	1	0	0	1	0	0	0										8										2



**Instruction Set (cont)**

Mnemonic	Operand	Operation	Operation Code																Bytes	Skip Condition		
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			State(1)	
<b>Others (cont)</b>																						
MEGA	(A) ← (A) + 1		0	1	0	0	1	0	0	0	0	0	1	1	1	0	1	0	8	2		
<b>Rotate and Shift</b>																						
RLD	Rotate left digit		0	1	0	0	1	0	0	0	0	0	1	1	1	0	0	0	17	2		
RHD	Rotate right digit		0	1	0	0	1	0	0	0	0	0	1	1	1	0	0	1	17	2		
RL	(R <sub>2m</sub> + 1) ← (R <sub>2m</sub> ), (R <sub>20</sub> ) ← (CY), (CY) ← (R <sub>27</sub> )		0	1	0	0	1	0	0	0	0	0	1	1	0	1	R <sub>1</sub> R <sub>0</sub>	8	2			
RLR	(R <sub>2m</sub> - 1) ← (R <sub>2m</sub> ), (R <sub>27</sub> ) ← (CY), (CY) ← (R <sub>20</sub> )		0	1	0	0	1	0	0	0	0	0	1	1	0	0	R <sub>1</sub> R <sub>0</sub>	8	2			
SLL	(R <sub>2m</sub> + 1) ← (R <sub>2m</sub> ), (R <sub>20</sub> ) ← 0, (CY) ← (R <sub>27</sub> )		0	1	0	0	1	0	0	0	0	0	1	0	0	1	R <sub>1</sub> R <sub>0</sub>	8	2			
SLR	(R <sub>2m</sub> - 1) ← (R <sub>2m</sub> ), (R <sub>27</sub> ) ← 0, (CY) ← (R <sub>20</sub> )		0	1	0	0	1	0	0	0	0	0	1	0	0	0	R <sub>1</sub> R <sub>0</sub>	8	2			
SLLC	(R <sub>2m</sub> + 1) ← (R <sub>2m</sub> ), (R <sub>20</sub> ) ← 0, (CY) ← (R <sub>27</sub> )		0	1	0	0	1	0	0	0	0	0	0	0	0	1	R <sub>1</sub> R <sub>0</sub>	8	2			
SLRC	(R <sub>2m</sub> - 1) ← (R <sub>2m</sub> ), (R <sub>27</sub> ) ← 0, (CY) ← (R <sub>20</sub> )		0	1	0	0	1	0	0	0	0	0	0	0	0	0	R <sub>1</sub> R <sub>0</sub>	8	2			
DRLL	(EA <sub>n</sub> + 1) ← (EA <sub>n</sub> ), (EA <sub>0</sub> ) ← (CY), (CY) ← (EA <sub>15</sub> )		0	1	0	0	1	0	0	0	0	0	1	0	1	0	1	0	0	8	2	
DRLR	(EA <sub>n</sub> - 1) ← (EA <sub>n</sub> ), (EA <sub>15</sub> ) ← (CY), (CY) ← (EA <sub>0</sub> )		0	1	0	0	1	0	0	0	1	0	1	1	0	0	0	0	8	2		
DSLL	(EA <sub>n</sub> + 1) ← (EA <sub>n</sub> ), (EA <sub>0</sub> ) ← 0, (CY) ← (EA <sub>15</sub> )		0	1	0	0	1	0	0	0	1	0	1	0	0	1	0	0	8	2		
DSLRL	(EA <sub>n</sub> - 1) ← (EA <sub>n</sub> ), (EA <sub>15</sub> ) ← 0, (CY) ← (EA <sub>0</sub> )		0	1	0	0	1	0	0	0	1	0	1	0	0	0	0	0	8	2		
<b>Jump</b>																						
JMP	*word (PC) ← word		0	1	0	1	0	1	0	0	High addr						Low addr				10	3
<b>Call</b>																						
JB	(PC <sub>H</sub> ) ← (B), (PC <sub>L</sub> ) ← (C)		0	0	1	0	0	0	0	1	High addr						Low addr				4	1
JR	word (PC) ← (PC) + 1 + jdispl		1	1	←	jdispl		→										10	1			
JPE	*word (PC) ← (PC) + 2 + jdispl		0	1	0	0	1	1	1	←	jdispl								10	2		
JEA	(PC) ← (EA)		0	1	0	0	1	0	0	0	0 0 1 0 1 0 0 0						8	2				
<b>Call</b>																						
CALL	*word ((SP) - 1) ← ((PC) + 3) <sub>H</sub> , ((SP) - 2) ← ((PC) + 3) <sub>L</sub> , (PC) ← word, (SP) ← (SP) - 2		0	1	0	0	0	0	0	0	High addr						Low addr				16	3
CALB	((SP) - 1) ← ((PC) + 2) <sub>H</sub> , ((SP) - 2) ← ((PC) + 2) <sub>L</sub> , (PC <sub>H</sub> ) ← (B), (PC <sub>L</sub> ) ← (C), (SP) ← (SP) - 2		0	1	0	0	1	0	0	0	0 0 1 0 1 0 0 0						17	2				
CALF	*word ((SP) - 1) ← ((PC) + 2) <sub>H</sub> , ((SP) - 2) ← ((PC) + 2) <sub>L</sub> , (PC <sub>15:11</sub> ) ← 00001, (PC <sub>10:0</sub> ) ← fa, (SP) ← (SP) - 2		0	1	1	1	1	←	fa						13	2						

**Instruction Set (cont)**

Mnemonic	Operand	Operation	Operation Code																Bytes	Skip Condition					
			B1	B2	B3	B4	7	6	5	4	3	2	1	0	State(1)										
Call (cont)			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	State(1)	Bytes	Skip Condition				
CALL	word	$((SP) - 1) \leftarrow ((PC) + 1)_H$ , $((SP) - 2) \leftarrow ((PC) + 1)_L$ , $(PC) \leftarrow (128 + 21a), (PC) \leftarrow$ $(129 + 21a), (SP) \leftarrow (SP) - 2$	1	0	0	←	1a	→											16	1					
SOFT		$(SP) - 1) \leftarrow (PSW), (SP) - 2) \leftarrow$ $((PC) + 1)_H, ((SP) - 3) \leftarrow ((PC) + 1)_L$ , $(PC) \leftarrow 0060H, (SP) \leftarrow (SP) - 3$	0	1	1	1	0	0	1	0									16	1					
Return																									
RET		$(PC) \leftarrow ((SP), (PC)) \leftarrow ((SP) + 1)$ $(SP) \leftarrow (SP) + 2$	1	0	1	1	0	0	0										10	1					
RETS		$(PC) \leftarrow ((SP), (PC)) \leftarrow ((SP) + 1)$ $(SP) \leftarrow (SP) + 2, (PC) \leftarrow (PC) + n$	1	0	1	1	0	0	1										10	1	Unconditional Skip				
RETI		$(PC) \leftarrow ((SP), (PC)) \leftarrow ((SP) + 1)$ $(PSW) \leftarrow ((SP) + 2), (SP) \leftarrow (SP) + 3$	0	1	1	0	0	1	0										13	1					
Skip																									
Bit	bit, wa		0	1	0	1	1	B <sub>2</sub>	B <sub>1</sub>	B <sub>0</sub>	Offset						10	2	Bit Test						
CPU Control																									
SK	f	Skip if f = 1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1	F <sub>2</sub>	F <sub>1</sub>	F <sub>0</sub>	8	2	f = 1		
SKN	f	Skip if f = 0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	1	F <sub>2</sub>	F <sub>1</sub>	F <sub>0</sub>	8	2	f = 0		
SKIT	irf	Skip if irf = 1, then reset irf	0	1	0	0	1	0	0	0	0	1	0	1	0	1	0	l <sub>4</sub>	l <sub>3</sub>	l <sub>2</sub>	l <sub>1</sub>	l <sub>0</sub>	8	2	irf = 1
SKNIT	irf	Skip if irf = 0 Reset irf if irf = 1	0	1	0	0	1	0	0	0	0	1	1	1	0	1	0	l <sub>4</sub>	l <sub>3</sub>	l <sub>2</sub>	l <sub>1</sub>	l <sub>0</sub>	8	2	irf = 0
NOP		No operation	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						4	1	
EI		Enable interrupt	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0						4	1	
DI		Disable interrupt	1	0	1	1	0	1	0	1	0	0	0	0	0	0	0						4	1	
HLT		Halt	0	1	0	0	1	0	0	0	0	0	0	0	1	1	0	1	0	1	1	0	1	11	2

**Notes:**  
 (1) In the case of skip condition, the idle states are as follows:  
 1-byte instruction: 4 states  
 2-byte instruction (with \*): 7 states  
 3-byte instruction (with \*): 10 states  
 4-byte instruction: 14 states  
 2-byte instruction: 8 states  
 3-byte instruction: 11 states  
 (2) B2 (Data): rpa2 = D + byte, H + byte.  
 (3) Right side of slash (/) in states indicates case rpa2, rpa3 = D + byte, H + A, H + B, H + EA, H + byte.  
 (4) B3 (Data): rpa3 = D + byte, H + byte