NEC

NEC Electronics Inc.

μPD7832x Advanced 8/16-Bit, Real Time Control Microcomputer With A/D Converter

Description

The μPD7832x (78320, 78322) is a single-chip microcomputer designed for process control. It features a 16-bit CPU, an 8-bit external data bus, and a powerful set of on-chip peripherals including counters and timers, an A/D converter, two serial ports, and a maximum of 55 input/output lines.

An advanced interrupt handling facility includes a three-level program-controlled hardware priority interrupt controller and three separate methods of handling interrupt requests. It is manufactured of 1.2μ CMOS process, operates from a single 5 V power supply, and has a maximum oscillator frequency of 16 MHz.

The μPD7832x has 16K bytes of on-chip mask-programmed ROM, and the μPD78320 is a ROM-less version. Both chips have 640 bytes of on-chip RAM and are supplied in a 68-pin PLCC or 74-pin plastic QFP package.

The μPD7832x has an interface for a special dedicated memory chip, the μPD71P301. The μPD71P301 includes memory, interface circuitry, and an instruction prefetch pointer. This makes it possible to fetch instructions from external memory at the same high speed at which they can be fetched from on-chip ROM.

The primary applications of the μPD7832x include automotive engine control, antilock braking control, and control of computer disks and tapes. Its speed and powerful on-chip peripherals, however, make it suitable for all of the more demanding types of process control.

Features

- □ Complete single-chip microcomputer
  - —16-bit ALU
  - —16K bytes of ROM (μPD78322 only)
  - —640 bytes RAM
- □ Powerful instruction set
  - —16-bit multiply and divide
  - —1-bit and 8-bit logic instructions
  - —String instructions
- □ Minimum instruction time
  - —250 ns @ 16-MHz input
- □ 3-byte instruction prefetch queue
- □ Memory expansion
  - —8085 bus compatible
  - —64K-byte address space
  - —High-speed fetch from external memory
- □ Large I/O capacity
  - —Up to 55 I/O port lines
- □ Special interface for turbo access manager (TAM) μPD71P301
- □ Memory-mapped on-chip peripherals (special function registers)
- □ Multipurpose pulse input/output unit
  - —16-/18-bit free-running timer
  - —16-bit timer/event counter
  - —Six 16-bit compare registers
  - —Four 18-bit capture registers
  - —Two 18-bit capture/compare registers
  - —Six external interrupt/capture lines
  - —One external event counter/interrupt line
  - —Six timer-controlled output lines
- □ 10-bit, 8-channel analog to digital converter
  - —On-chip sample and hold amplifier
- □ Two-channel serial communication interface
  - —Asynchronous serial interface (UART)
  - —Serial bus interface
  - —Dedicated baud rate generator
- □ Programmable priority interrupt controller (3 levels)
- □ Three methods of interrupt service
  - —Vectored interrupts
  - —Context switching with hardware save of all general registers
  - —Nine macroservice functions
- □ Watchdog timer with dedicated output
- □ STOP and HALT standby functions
- □ Single 5-volt power supply

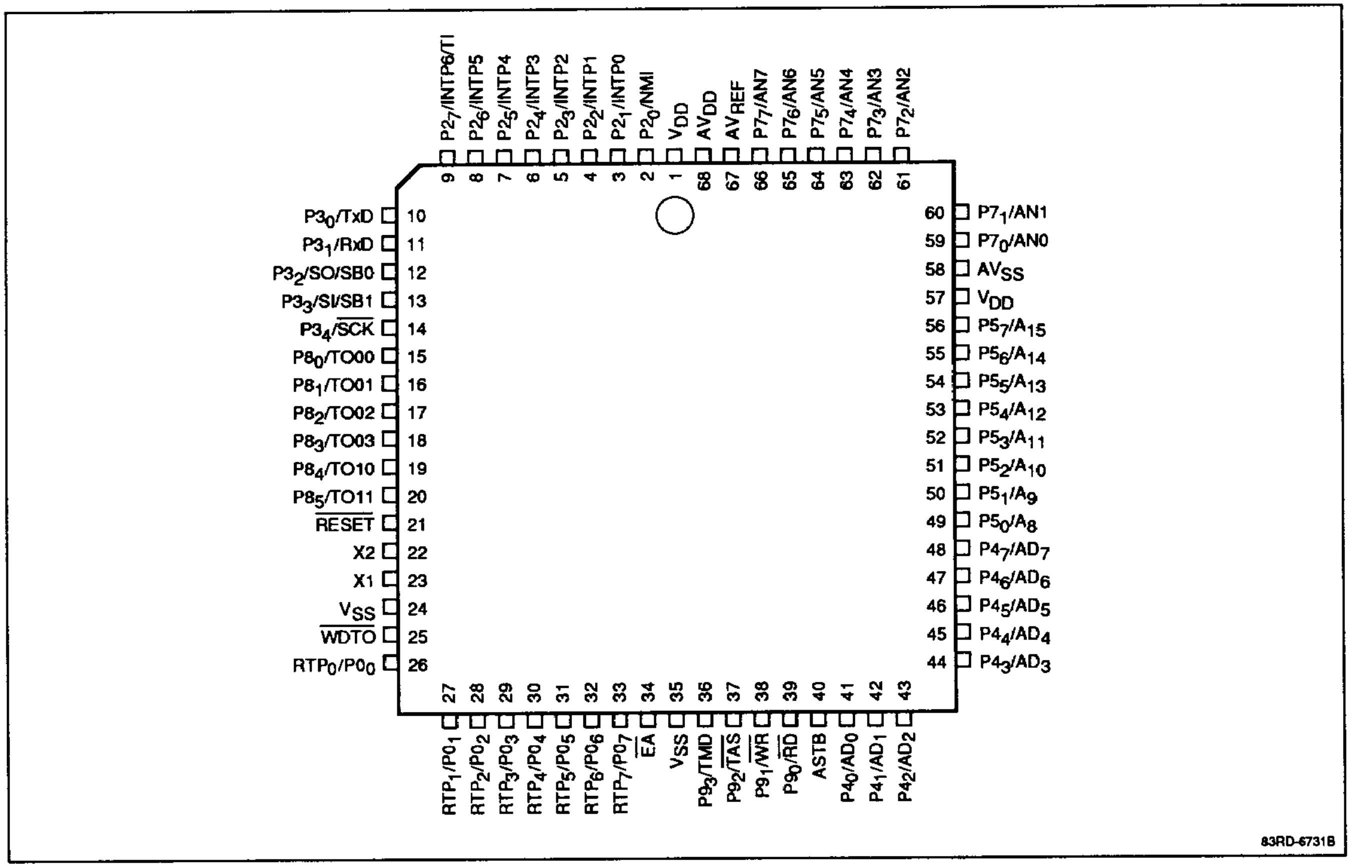
Ordering Information

Part Number	On-Chip ROM	Package Type
μPD78320L	No	68-pin PLCC
μPD78320GJ-5BJ	No	74-pin plastic QFP
μPD78322L-xxx	Yes	68-pin PLCC
μPD78322GJ-xxx-5BJ	Yes	74-pin plastic QFP

xxx is the mask code number

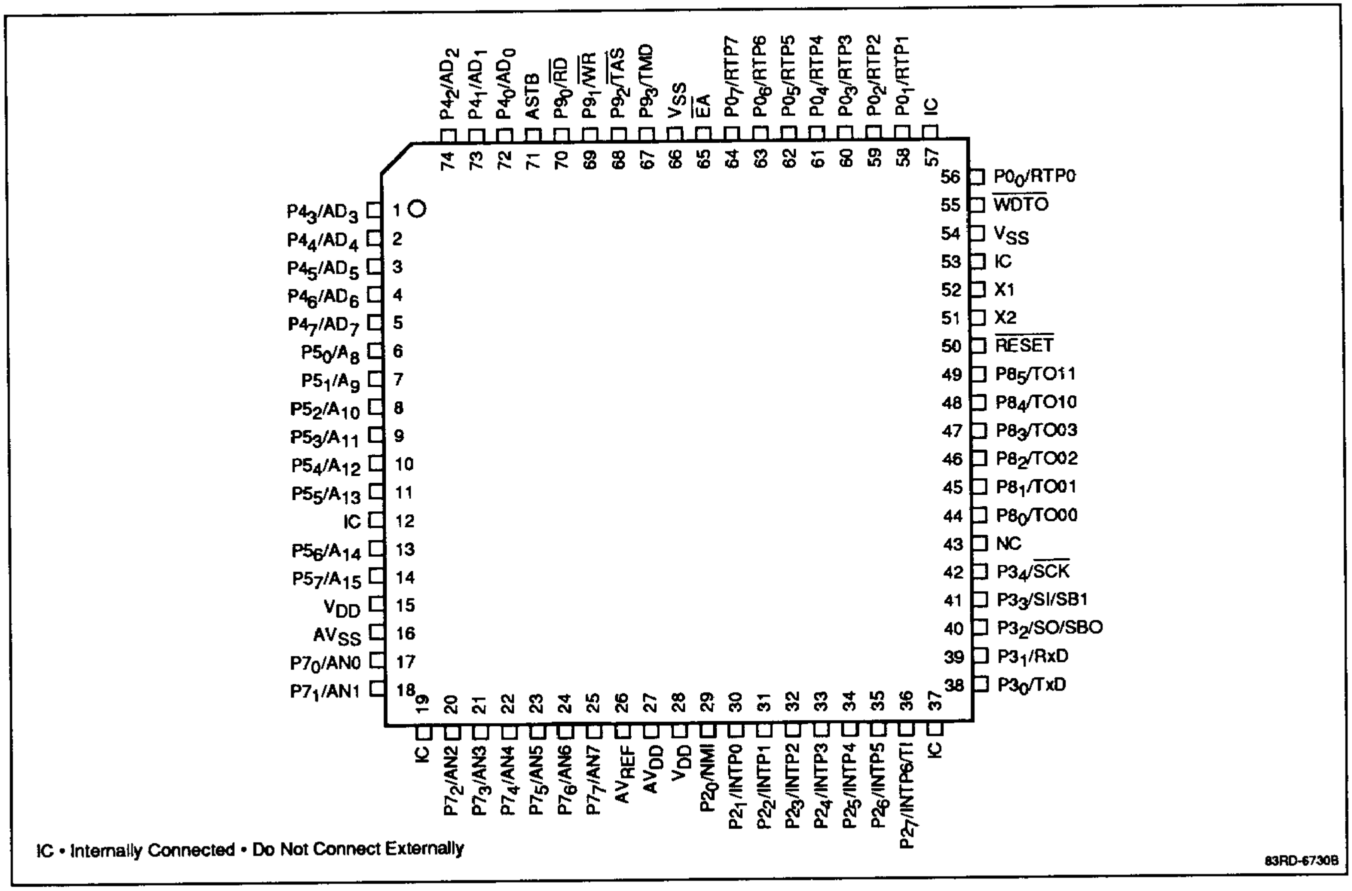
Pin Configurations

68-Pin PLCC



Pin Configuration (cont)

74-Pin Plastic QFP

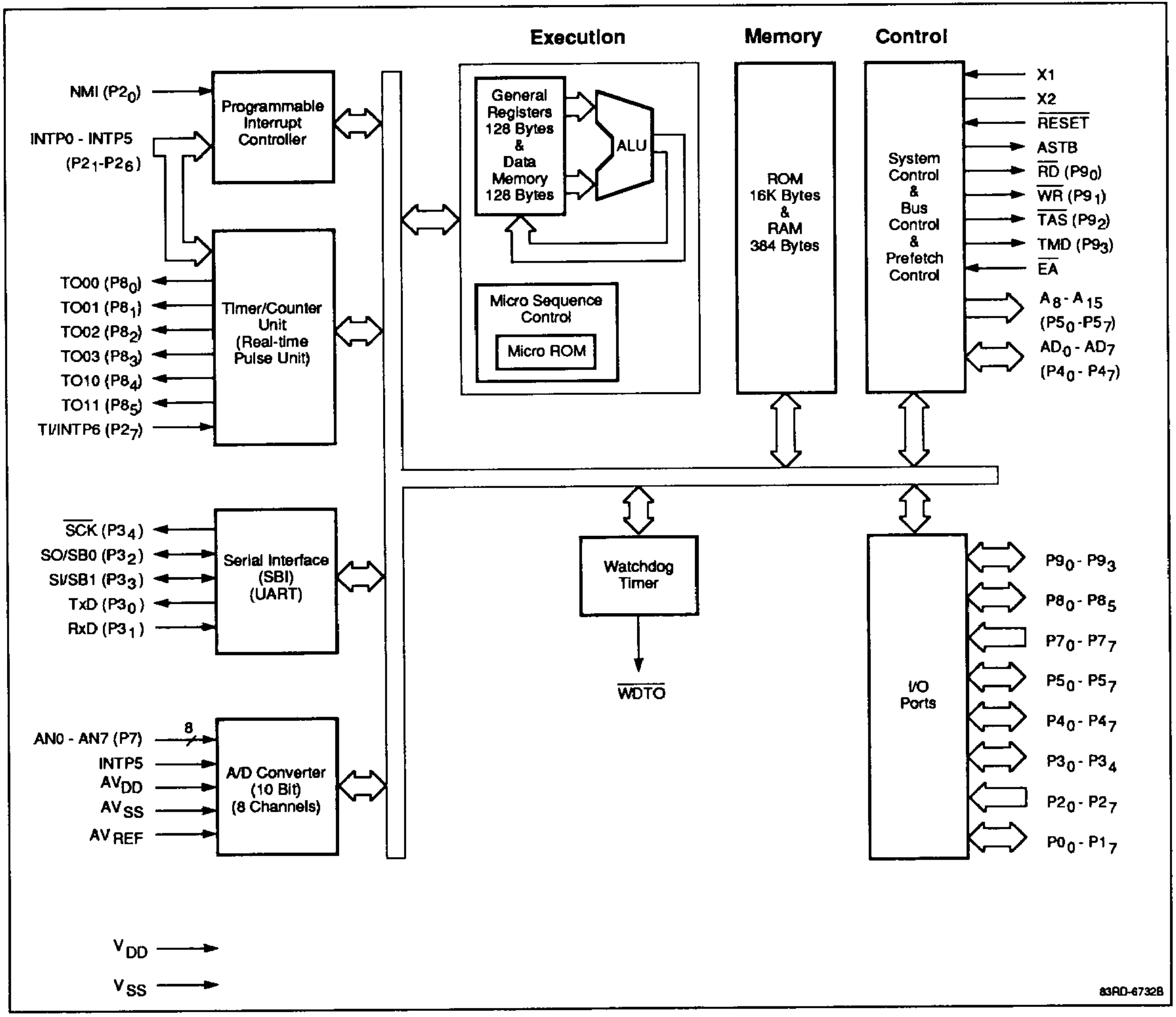


IC • Internally Connected • Do Not Connect Externally

Pin Function

Symbol	First Function	Symbol	Second Function
P0_0-P0_7	Port 0; 8-bit, bit selectable I/O port	RTP$_0$-RTP$_7$	Bit selectable, timer-controlled, real-time output port
P2_0	Port 2; 8-bit input port	NMI	External nonmaskable interrupt
P2_1		INTP0	Maskable external interrupts; edge-selectable
P2_2		INTP1	
P2_3		INTP2	
P2_4		INTP3	
P2_5		INTP4	
P2_6		INTP5	
P2_7		INTP6/T1	External interrupt or timer input
P3_0	Port 3; 5-bit, bit selectable I/O port	TxD	Asynchronous serial transmit
P3_1		RxD	Asysnchronous serial receive
P3_2		SO/SB0	Synchronous serial line
P3_3		SI/SB1	Synchronous serial line
P3_4		$\overline{SCK}$	Serial clock input or output
P4_0-P4_7	Port 4; 8-bit, byte selectable I/O port	AD$_0$-AD$_7$	Low-order byte of external address/data bus
P5_0-P5_7	Port 5; 8-bit, bit selectable I/O port	A$_8$-A$_{15}$	High-order byte of external address bus
P$_7$-P7_7	Port 7; 8-bit input port	AN0-AN7	Inputs for A/D converter
P8_0	Port 8; 6-bit, bit selectable I/O port	TO00	Timer (RPU) output lines
P8_1		TO01	
P8_2		TO02	
P8_3		TO03	
P8_4		TO10	
P8_5		TO11	
P9_0	Port 9; 4-bit, bit-selectable I/O port	$\overline{RD}$	External read strobe
P9_1		$\overline{WR}$	External write strobe
P9_2		$\overline{TAS}$	TAM strobe
P9_3		TMD	TAM control
ASTB	External address latch strobe		
$\overline{EA}$	External access control; a high level enables access to on-chip ROM; a low level is applied if all program memory is external. Must be tied low for the μPD78320.		
$\overline{RESET}$	External system reset input		
$\overline{WDTO}$	Watchdog timer output		
X1, X2	For frequency control of the internal clock oscillator, a crystal is connected to X1 and X2. If the clock is supplied by an external source, the clock signal is connected to X1 and the inverted clock signal is connected to X2.		
AV$_{REF}$	A/D converter reference voltage input		
AV$_{DD}$	A/D converter +5-volt power input		
AV$_{SS}$	A/D converter ground		
V$_{DD}$	+5-volt power input		
V$_{SS}$	Ground		

μPD7832x Block Diagram



FUNCTIONAL DESCRIPTION

Central Processing Unit

The Central Processing Unit (CPU) of the μPD7832x features 16-bit arithmetic including 16-by-16 bit multiply, both signed and unsigned, and 32-by-16 bit divide (producing a 32-bit quotient and 16-bit remainder). String instructions and both 8-bit and 1-bit logic instructions are included.

Instructions range in length from one to five bytes, depending on the instruction and addressing mode. A 1-byte call instruction can access up to 32 addresses specified in the CALLT vector table in lower memory. A 2-byte call instruction can access any routine beginning in a specific CALLF area. A single instruction can test individual bits both in a portion of on-chip RAM and in the special function registers.

A 3-byte instruction prefetch queue makes it possible to fetch instruction bytes on a separate bus during execution cycles. Instructions are fetched from on-chip ROM at a rate of one byte per cycle. An interface is provided for the μPD71P301 memory chip, called the Turbo Access Manager (TAM). TAM makes possible similar fetch rates from external memory.

The CPU clock is generated by dividing the oscillator frequency by two. Therefore, when the oscillator frequency is 16 MHz, the clock is 8 MHz. Some instructions execute in two cycles, and the minimum instruction time is 250 ns.

Addressing

The μPD7832x features 1-byte addressing of both the special function registers and a portion of the on-chip RAM. The 1-byte sfr addressing accesses the entire SFR area, while the 1-byte saddr addressing accesses 32 bytes of the SFR area and 224 bytes of the on-chip RAM. Nine modes for addressing main memory include indexing, double indexing, autoincrement, and autodecrement. Main memory addressing can be used to access the entire 64K address space including the SFR area and RAM. There are also both 8-bit and 16-bit immediate operands.

External Memory

The external memory bus is 8 bits wide, and external memory can be used to fill up the 64K-bit address space. Either ROM or RAM (or both) can be used as required. The low order 8 bits of the address/data bus are multiplexed, and are supplied by I/O port 4. High-order address bits are taken from port 5 as required. Address latch, read, and write strobes are provided. Two special control lines provide access to the TAM. The memory mode register controls the size of the external memory and the number of additional wait states. The high-order address uses 0, 4, 6, or 8 bits from port 5, depending on the amount of external memory required. Any remaining port 5 bits can be used for I/O. Figure 1 shows the memory map of the μPD7832x.

General Registers

Sixteen 8-bit general registers can be used in pairs to function as 16-bit registers. A complete set of 16 registers is mapped into each of eight program selectable register banks stored in RAM. Three bits in the PSW (figure 2) specify which of the register banks is active at any time. Registers have both functional names (A, AX, C, DE, etc.) and absolute names (R1, RP0, R2, RP6, etc.). Each instruction determines whether a register is referred to by functional or absolute name and whether it is 8 or 16 bits.

Two possible relationships may exist between the absolute and functional names of the first four register pairs. The RSS bit in the PSW determines which of these is active at any time. The effect is that the accumulator and counter registers can be saved, and a new set can be specified by toggling the RSS bit. Figure 3 illustrates the general register configuration.

Figure 1. μPD7832x Memory Map

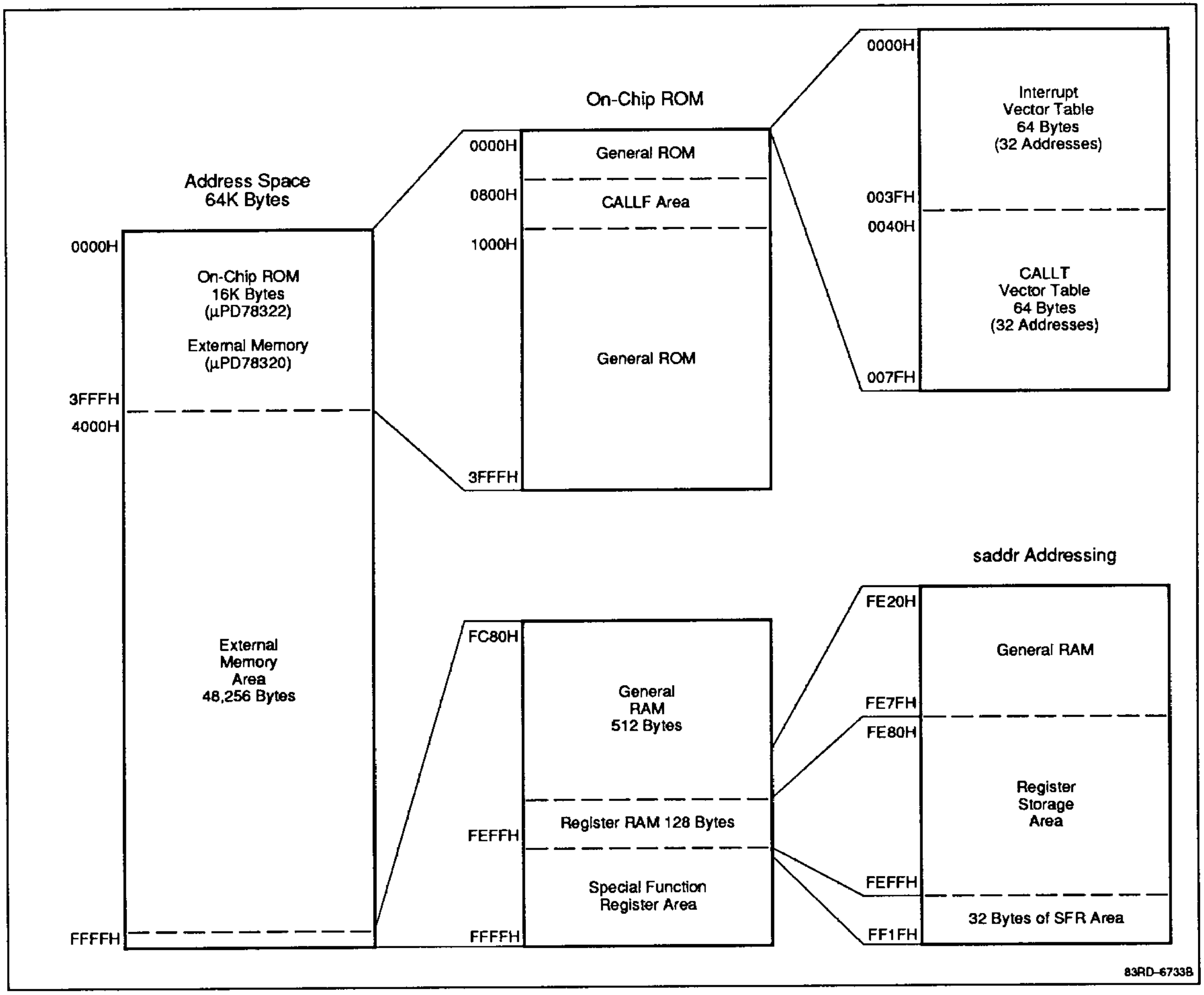


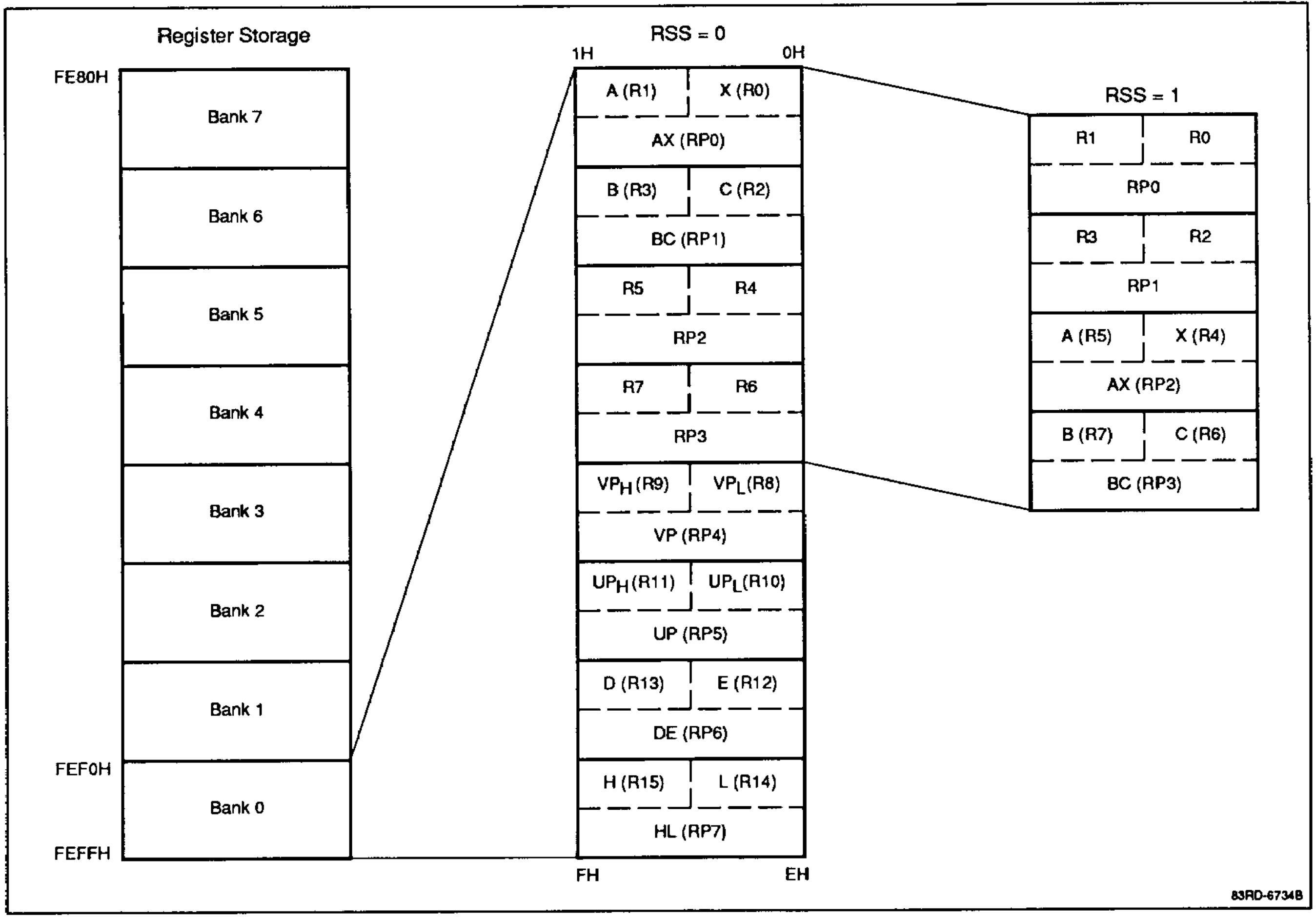
Figure 2. Program Status Word

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UF	RBS2	RBS1	RBS0	0	0	0	0	S	Z	RSS	AC	IE	P/V	LT	CY

UF	User flag
RBS2 - RBS0	Active register bank selector
S	Sign flag
Z	Zero flag
RSS	Register set selection flag
AC	Auxilliary carry flag
IE	Interrupt Enable flag
P/V	Parity or Overflow flag
LT	In-service priority level transition flag
CY	Carry bit

83RD-6735B

Figure 3. Register Configuration and Storage



Input/Output

Eight I/O ports range in size from 4 to 8 bits, providing a total of 55 I/O lines. All I/O lines have alternate control functions which can be specified under program control. All except ports 2, 4, and 7 can be specified for input or output on an individual bit basis. Ports 2 and 7 are input (or control input) only. Port 4 is byte selectable for input, output, or control.

Real-Time Output Port

Port 0 can function on a bit-selectable basis as a real-time output port. Real-time port bits can be directly written under program control, or they can be set or cleared under control of timing signals generated by the real-time pulse unit. This provides output timing that is independent of interrupt latency.

External Interrupts

One nonmaskable and 7 maskable external interrupts share pins with port 2. The maskable interrupts can also be used to trigger capture events in the real-time pulse unit. Any masked interrupt automatically becomes an input line. INTP6 is also used as the counter input for timer TM1 when TM1 is used as an external event counter.

Serial Ports

The μPD7832x has two serial ports. The first is a standard asynchronous serial port that shares pins with $P3_0$ (TxD) and $P3_1$ (RxD). It generates three interrupts INTST (transmit complete), INTSR (receive buffer full), and INTSER (receive error).

The second serial port can be used in one of two modes. The first mode is a 3-wire I/O interface mode with send, receive, and clock lines. Data are sent and received most significant bit first, and the clock line can be driven either internally or externally. The second mode is the 2-wire NEC serial bus interface (SBI) mode. SBI features wake-up signals and distinction between commands, addresses, and data, all decoded by hardware.

The synchronous serial port shares I/O pins with port 3 bits 2-4 and generates a single interrupt, INTCSI. A dedicated baud rate generator is included so that all of the commonly used baud rates can be generated when the oscillator frequency is correctly chosen.

Analog to Digital Converter

An 8-channel 10-bit A/D converter provides a relative accuracy of 0.2% full scale. An on-chip sample-and-hold amplifier is included, and the eight input channels share pins with port 7. The A/D converter can be operated in either the scan mode (where either channels 0-3 or 4-7 are repeatedly scanned) or the select mode (where a specific channel is selected and converted repeatedly). The conversion can be started either by software or by an external signal on INTP5.

Real-Time Pulse Unit

The real-time pulse unit (RPU, figure 4) consists of an 18-bit free-running timer, TM0, 16-bit timer/counter, TM1, six 16-bit compare registers, four 18-bit capture registers, two 18-bit registers which can be used for either capture or compare, and six timed output latches. TM0 always counts the system clock (divided by either 4 or 8) and can be reset by external RESET only. TM1 can count either the system clock (divided by either 8 or 16) or external events. TM1 can be reset by either a compare event (a match between a timer and an associated compare register) or by an external signal in INTP0.

Capture events can be triggered by external maskable interrupts INTP0-INTP5, and compare events can be used to generate interrupts, control timed output pins, or both. In addition, two of them, INTCM03 and INTCCX0, can be used to control the real-time output port. The timed output latches share pins with port 8. Four of them can be toggled or set and reset by compare events, and the remaining two can be toggled. These latches, with the macroservice facility, can be used to generate up to four pulse-width modulated outputs.

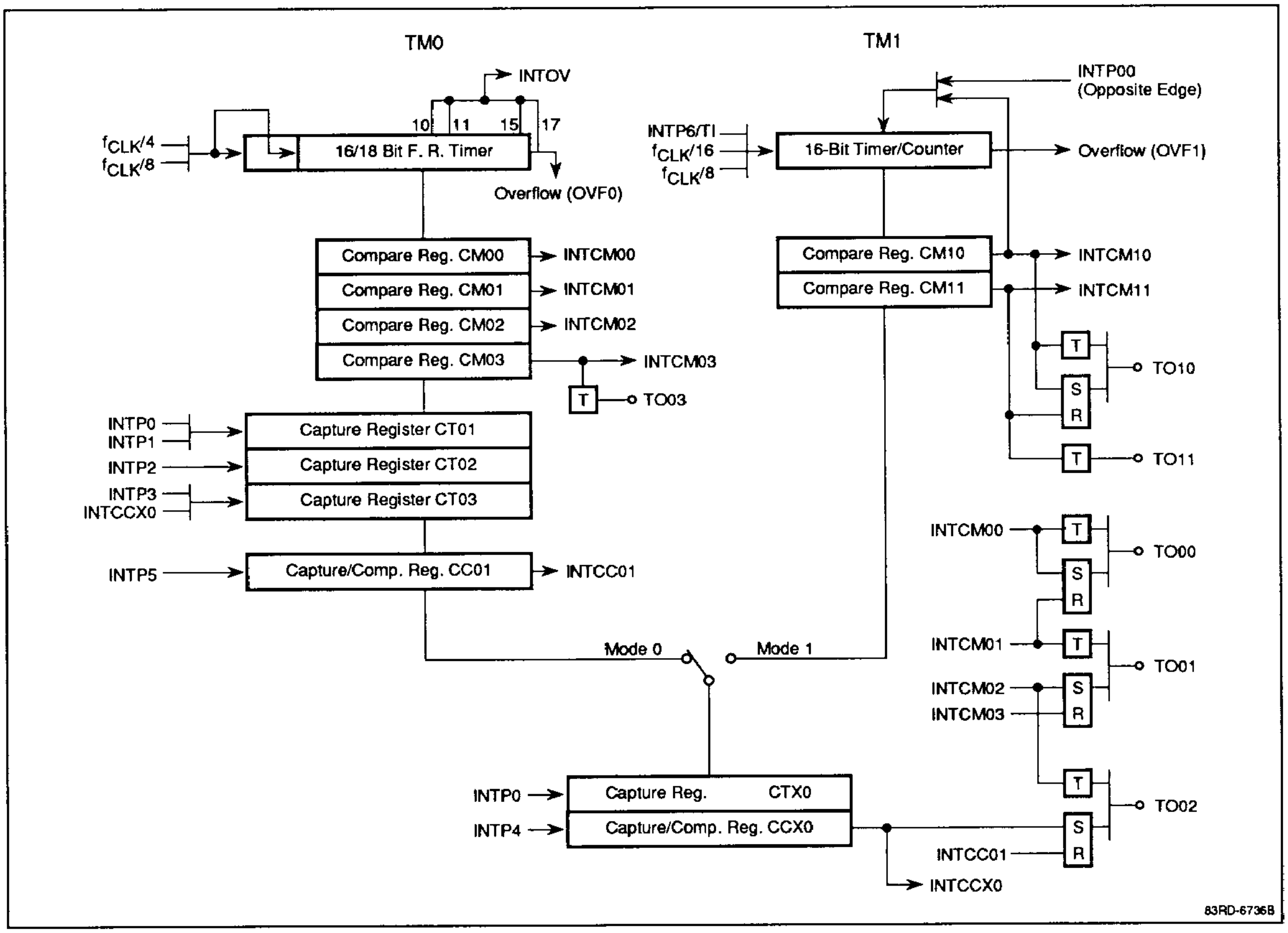
Standby Modes

HALT and STOP modes conserve power when CPU action is not required. In HALT mode, the CPU is stopped and the clock continues to run. Any unmasked interrupt can then restart the CPU. In STOP mode, the CPU and clock are both stopped. Either an external RESET pulse or an external nonmaskable interrupt is required to restart them. The standby control register (STBC) is a protected location and can be written to only by a special instruction.

Watchdog Timer

The watchdog timer protects against inadvertent program loops. A nonmaskable interrupt occurs if the timer is not reset before it overflows. Three program selectable intervals are available: 8.19, 32.7, and 131.0 msec for a system clock frequency of 8 MHz. An output line is provided, which can be connected to the RESET pin or used to control external circuitry. Once started, the timer can be stopped by external RESET only. In addition, the watchdog timer mode register, WDM, is a protected location and can be written to only by a special instruction.

Figure 4. Real-Time Pulse Unit



Interrupt Handling

The μPD7832x has three different methods of handling maskable interrupt requests, standard vectoring, context switching, and macroservice. The programmer can choose the mode that is most advantageous in any given situation. The μPD7832x has 19 maskable hardware interrupt sources: 7 external and 12 internal. In addition, there are two nonmaskable interrupts, two software interrupts, and a RESET. See table 1.

Interrupt Priority

The two nonmaskable interrupts, NMI and INTWDT, take priority over all others. Their priority relative to each other is under program control.

Three hardware controlled priority levels are available for the maskable interrupts. Any one of the three levels can be assigned by software to each of the maskable interrupt lines. Interrupt requests of a priority equal to or higher than the processor's current priority level are accepted. Requests of lower priority are pending until the processor's priority state is lowered by a return instruction from the current service routine. Interrupt requests programmed to be handled by macroservice have priority over all software interrupt service regardless of the assigned priority level. See figure 5.

Software interrupts, the BRK and BRKCS instruction, and operation code trap, are executed regardless of the processor's priority level and do not alter the priority level.

Figure 5. Interrupt Service Sequence

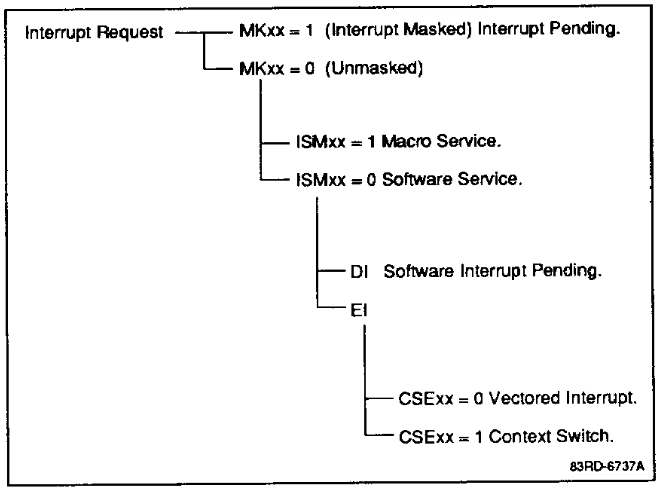

Interrupt Request
MKxx = 1 (Interrupt Masked) Interrupt Pending.
MKxx = 0 (Unmasked)
ISMxx = 1 Macro Service.
ISMxx = 0 Software Service.
DI Software Interrupt Pending.
EI
CSExx = 0 Vectored Interrupt.
CSExx = 1 Context Switch.
83RD-6737A

Table 1. Interrupt Sources

Request	Default Priority	Mnemonic	Source	Vector Address	Macroservice	Control Word*
Software	—	BRK	Break instruction	003EH	N	—
Software	—	TRAP	Opcode trap	003CH	N	—
Nonmaskable	—	NMI	External NMI	0002H	N	—
Nonmaskable	—	INTWDT	Watchdog timer	0004H	N	—
Maskable	0	INTOV	RPU	0006H	Y	FE06H
Maskable	1	INTP0	RPU/External	0008H	Y	FE08H
Maskable	2	INTP1	RPU/External	000AH	Y	FE0AH
Maskable	3	INTP2	RPU/External	000CH	Y	FE0CH
Maskable	4	INTP3	RPU/External	000EH	Y	FE0EH
Maskable	5	INTP4/INTCCX0	RPU/External	0010H	Y	FE10H
Maskable	6	INTP5/INTCC01	RPU/External	0012H	Y	FE12H
Maskable	7	INTP6	External	0014H	Y	FE14H
Maskable	8	INTCM00	RPU	0016H	Y	FE16H
Maskable	9	INTCM01	RPU	0018H	Y	FE18H
Maskable	10	INTCM02	RPU	001AH	Y	FE1AH
Maskable	11	INTCM03	RPU	001CH	Y	FE1CH
Maskable	12	INTCM10	RPU	001EH	Y	FE1EH
Maskable	13	INTCM11	RPU	0020H	Y	FE20H
Maskable	14	INTSER	UART	0022H	N	—
Maskable	15	INTSR	UART	0024H	Y	FE24H
Maskable	16	INTST	UART	0026H	Y	FE26H
Maskable	17	INTCSI	Clocked serial interface	0028H	Y	FE28H
Maskable	18	INTAD	A/D Converter	002AH	Y	FE2AH
RESET	—	RESET	External reset	0000H	N	—

*Address of macroservice control word in on-chip RAM.

Vectored Interrupt

When vectored interrupt is specified for a given interrupt request, the program status word and the program counter are saved on the stack, the processor's priority is raised to that specified for the interrupt, and the routine whose address is in the interrupt vector table is entered. At the completion of the service routine, the RETI instruction (or RETB instruction for software interrupts) reverses the process.

Context Switch

When context switching (figure 6) is specified for a given interrupt, the active register bank is changed to the register bank specified by the three low-order bits of the word in the interrupt vector table. The program counter is loaded from RP2 of the new register bank, and the program counter and program status word are saved in RP2 and RP3 of the new register bank. At the completion of the service routine, the RETCS instruction for routines entered from hardware requests, or the RETCSB instruction for routines entered from the BRKCS instruction, reverses the process. These instructions have a 16-bit immediate operand which must be set to the entry address of the service routine.

Macroservice

When macroservice is specified for a given interrupt, the macroservice hardware performs any one of nine functions during cycles "stolen" from the executing program. Control is then returned to the executing program, and the operation is therefore completely transparent. Macroservice significantly improves response time and makes it unnecessary to save any registers.

For each request on the interrupt line, one operation is performed, and a counter is decremented. When the counter reaches zero (or when some other completion condition is met), a software service routine is entered. Either vectored interrupt or context switch can be specified for entry to the completion routine, and the routine is entered according to the specified priority.

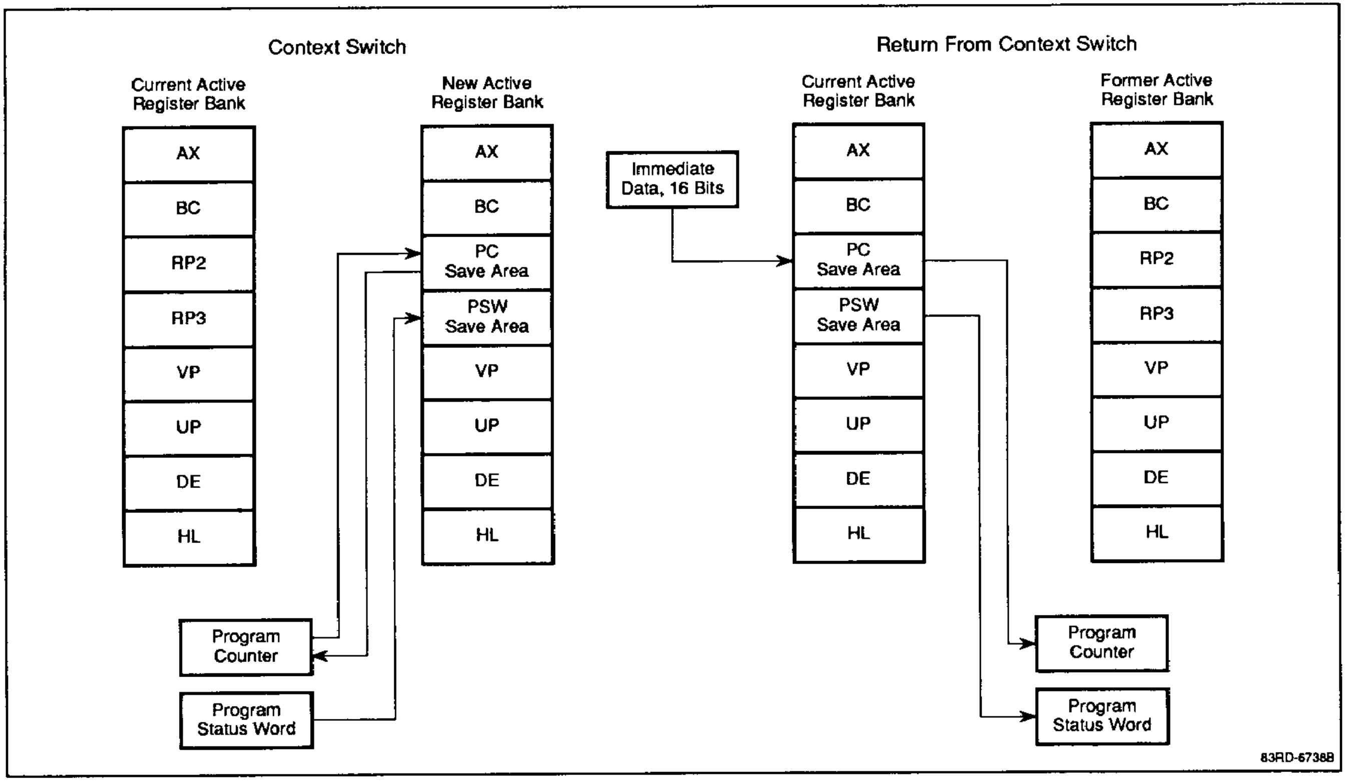
Macroservice is provided for all but one of the maskable interrupt requests, and each has a specific macroservice control word stored in on-chip RAM. The function to be performed is specified in the control word.

The nine macroservice functions are as follows:

Function	Description
EVTCNT	Event counter
DTACMP	Data compare
BITSHT	Bit shift
BITLOG	Bit logic
ADCBUF	A/D converter buffering
BLKTRS	Block transfer
DTADIF	Data difference
DTADIF-P	Data difference-pointer
DTADD	Data addition

The BLKTRS function moves either a byte or word of data in either direction between a specified special function register and a specified memory location. It therefore has an effect similar to that of a DMA channel.

Figure 6. Context Switching and Return



Special-Function Registers

The special-function registers (table 2) include the I/O ports, the counters and timers, all registers associated with peripherals, and all of the control and mode registers. They are memory mapped in the top 256 memory addresses and can be addressed either by main memory addressing or by the special one byte sfr addressing. Most can be either read or written, and individual bits within them can be modified or tested with a single instruction.

Table 2. Special-Function Registers

				Access Unit (Bits)			
Address	**Register**	**Symbol**	**R/W**	**1**	**8**	**16**	**State after RESET**
FF00H	Port 0	P0	R/W	X	X	—	Undefined
FF02H	Port 2	P2	R	—	X	—	Undefined
FF03H	Port 3	P3	R/W	X	X	—	Undefined
FF04H	Port 4	P4	R/W	X	X	—	Undefined
FF05H	Port 5	P5	R/W	X	X	—	Undefined
FF07H	Port 7	P7	R	—	X	—	Undefined
FF08H	Port 8	P8	R/W	X	X	—	Undefined
FF09H	Port 9	P9	R/W	X	X	—	Undefined
FF0AH-FF0BH	Free-running counter (lower 16 bits)*	TM0LW	R	—	—	X	0000H
FF10H-FF11H	Capture register X0 (lower 16 bits)*	CTX0LW	R	—	—	X	Undefined
FF12H-FF13H	Capture register 01 (lower 16 bits)*	CT01LW	R	—	—	X	Undefined
FF14H-FF15H	Capture register 02 (lower 16 bits)*	CT02LW	R	—	—	X	Undefined

Table 2. Special-Function Registers (cont)

				Access Unit (Bits)			
Address	Register	Symbol	R/W	1	8	16	State after RESET
FF16H-FF17H	Capture register 03 (lower 16 bits)*	CT03LW	R	—	—	X	Undefined
FF18H-FF19H	Capture/compare register X0 (lower 16 bits)*	CCX0LW	R/W	—	—	X	Undefined
FF1AH-FF1BH	Capture/compare register 01 (lower 16 bits)*	CC01LW	R/W	—	—	X	Undefined
FF20H	Port 0 mode register	PM0	W	—	X	—	FFH
FF23H	Port 3 mode register	PM3	W	—	X	—	xxx1 1111B
FF25H	Port 5 mode register	PM5	W	—	X	—	FFH
FF28H	Port 8 mode register	PM8	W	—	X	—	xx11 1111B
FF29H	Port 9 mode register	PM9	W	—	X	—	xxxx 1111B
FF2AH-FF2BH	Free running counter (high 16 bits)*	TM0UW	R	—	—	X	0000H
FF2CH-FF2DH	Timer register 1 (lower 16 bits)*	TM1	R	—	—	X	0000H
FF30H-FF31H	Capture register X0 (High 16 bits)*	CTX0UW	R	—	—	X	Undefined
FF32H-FF33H	Capture register 01 (High 16 bits)*	CT01UW	R	—	—	X	Undefined
FF34H-FF35H	Capture register 02 (High 16 bits)*	CT02UW	R	—	—	X	Undefined
FF36H-FF37H	Capture register 03 (High 16 bits)*	CT03UW	R	—	—	X	Undefined
FF38H-FF39H	Capture/compare register X0 (high 16 bits)*	CCX0UW	R/W	—	—	X	Undefined
FF3AH-FF3BH	Capture/compare register 01 (high 16 bits)*	CC01UW	R/W	—	—	X	Undefined
FF40H	Port 0 mode control register	PMC0	W	—	X	—	00H
FF41H	Real-time output port set register	RTPS	R/W	X	X	—	00H
FF43H	Port 3 mode control register	PMC3	W	—	X	—	xxx0 0000B
FF48H	Port 8 mode control register	PMC8	W	—	X	—	xx00 0000B
FF4CH-FF4DH	Baud rate generator	BRG	R/W	—	—	X	Undefined
FF60H	Real-time output port register	RTP	R/W	X	X	—	Undefined
FF61H	Real-time output port reset register	RTPR	R/W	X	X	—	00H
FF62H	Port read control register	PRDC	R/W	X	X	—	00H
FF68H	A/D converter mode register	ADM	R/W	X	X	—	00H
FF6AH	A/D converter result register (16-bit access)	ADCR	R	—	—	X	Undefined
FF6BH	A/D converter result register (high 8 bits)	ADCRH	R	—	X	—	Undefined
FF70H-FF71H	Compare register 00	CM00	R/W	—	—	X	Undefined
FF72H-FF73H	Compare register 01	CM01	R/W	—	—	X	Undefined
FF74H-FF75H	Compare register 02	CM02	R/W	—	—	X	Undefined
FF76H-FF77H	Compare register 03	CM03	R/W	—	—	X	Undefined
FF7CH-FF7DH	Compare register 10	CM10	R/W	—	—	X	Undefined
FF7EH-FF7FH	Compare register 11	CM11	R/W	—	—	X	Undefined
FF80H	Clock synchronized serial interface mode register	CSIM	R/W	X	X	—	00H
FF82H	Serial bus interface control register	SBIC	R/W	X	X	—	00H

Table 2. Special-Function Registers (cont)

				Access Unit (Bits)			
Address	**Register**	**Symbol**	**R/W**	**1**	**8**	**16**	**State after RESET**
FF86H	Serial I/O shift register	SIO	R/W	X	X	—	Undefined
FF88H	Asynchronous serial interface mode register	ASIM	R/W	X	X	—	80H
FF8AH	Asynchronous serial interface status register	ASIS	R	—	X	—	00H
FF8CH	Serial receive buffer: UART	RXB	R	—	X	—	Undefined
FF8EH	Serial transmit shift register: UART	TXS	W	—	X	—	Undefined
FFB0H	Timer control register	TMC	R/W	X	X	—	00H
FFB1H	Baud rate generator mode register	BRGM	R/W	X	X	—	00H
FFB2H	Prescalar mode register	PRM	R/W	X	X	—	00H
FFB8H	Timer output control register 0	TOC0	R/W	X	X	—	00H
FFB9H	Timer output control register 1	TOC1	R/W	X	X	—	00H
FFBFH	Real-time pulse unit mode register	RPUM	R/W	X	X	—	00H
FFC0H	Standby control register	STBC	R/W**	X	X	—	0000 X000B
FFC1H	CPU control word	CCW	R/W	X	X	—	00H
FFC2H	Watchdog timer mode register	WDM	R/W**	X	X	—	00H
FFC4H	Memory extension mode register	MM	R/W	X	X	—	00H
FFC6H	Programmable wait control register	PWC	R/W	X	X	—	22H
FFC9H	Fetch cycle control register	FCC	R/W	X	X	—	00H
FFD0H-FFDFH	External access area		R/W	X	X	—	Undefined
FFE0H	Interrupt request flag register 0L	IF0L/IF0	R/W	X	X	X	00H
FFE1H	Interrupt request flag register 0H	IF0H	R/W	X	X	—	00H
FFE2H	Interrupt request flag register 1L	IF1L/ IF1	R/W	X	X	X	00H
FFE4H	Interrupt mask flag register 0L	MK0L/ MK0	R/W	X	X	X	FFH
FFE5H	Interrupt mask flag register 0H	MK0H	R/W	X	X	—	FFH
FFE6H	Interrupt mask flag register 1L	MK1L/ MK1	R/W	X	X	X	xxxx x111B
FFE8H	Priority selection buffer register 0L	PB0L/ PB0	R/W	X	X	X	00H
FFE9H	Priority selection buffer register 0H	PB0H	R/W	X	X	—	00H
FFEAH	Priority selection buffer register 1L	PB1L/ PB1	R/W	X	X	X	00H
FFECH	Interrupt service mode selection register 0L	ISM0L/ ISM0	R/W	X	X	X	00H
FFEDH	Interrupt service mode selection register 0H	ISM0H	R/W	X	X	—	00H
FFEEH	Interrupt service mode selection register 1L	ISM1L/ ISM1	R/W	X	X	X	00H
FFF0H	Context switch enable register 0L	CSE0L/ CSE0	R/W	X	X	X	00H
FFF1H	Context switch enable register 0H	CSE0H	R/W	X	X	—	00H

Table 2. Special-Function Registers (cont)

				Access Unit (Bits)			
Address	**Register**	**Symbol**	**R/W**	**1**	**8**	**16**	**State after RESET**
FFF2H	Context switch enable register 1L	CSE1L/ CSE1	R/W	X	X	X	00H
FFF4H	External interrupt mode register 0	INTM0	R/W	X	X	—	00H
FFF5H	External interrupt mode register 1	INTM1	R/W	X	X	—	00H
FFF8H	In-service priority register	ISPR	R	—	X	—	00H
FFF9H	Priority selection register	PRSL	R/W	X	X	—	00H

* Lower or upper 16 bits of an 18-bit register.

** Protected location: special instruction required for write.

ELECTRICAL SPECIFICATIONS

Absolute Maximum Ratings

$T_A = 25°C$

Supply voltage, V_{DD}	–0.5 to +7.0 V
Supply voltage, AV_{DD}	–0.5 to V_{DD} +0.5 V
Supply voltage, AV_{SS}	–0.5 to +0.5 V
Input voltage, V_I	-0.5 to V_{DD}+0.5 V
Output voltage, V_O	–0.5 to V_{DD}+0.5 V
Reference input voltage, AV_{REF} $f_{XX} \leq$ 16MHz	–0.5 to AV_{DD} +0.3 V
Output current, low; I_{OL} Each output pin	4.0 mA
Total	90 mA
Output current, high; I_{OH} Each output pin	–1.0 mA
Total	–20 mA
Operating temperature, T_{OPT}	–10 to +70°C
Storage temperature, T_{STG}	–65 to +150°C

Exposure to Absolute Maximum Ratings for extended periods may affect device reliability; exceeding the ratings could cause permanent damage.

Operating Conditions

Oscillator Frequency	T_A	V_{DD}
8 MHz $\leq f_{XX} \leq$ 16 MHz	–10 to +70°C	+5.0 V ± 10%

Capacitance

T_A = 25°C; $V_{DD} = V_{SS}$ = 0 V

Parameter	Symbol	Max	Unit	Conditions
Input pin capacitance	C_I	10	pF	f = 1 MHz; unmeasured pins returned to 0 V
Output pin capacitance	C_O	20	pF	
I/O pin capacitance	C_{IO}	20	pF	

DC Characteristics

T_A = −10 to +70°C; V_{DD} = +5.0 V ±10%; V_{SS} = 0 V

Parameter	Symbol	Min	Typ	Max	Unit	Conditions
Input voltage, low	V_{IL}	0		0.8	V	
Input voltage, high	V_{IH1}	2.2			V	Note 1
	V_{IH2}	$0.8V_{DD}$			V	Note 2
Output voltage, low	V_{OL}			0.45	V	I_{OL} = 2.0 mA
Output voltage, high	V_{OH}	$V_{DD}-1.0$			V	I_{OH} = −400 µA
Input leakage current	I_{LI}			±10	µA	$0 V \le V_I \le V_{DD}$
Output leakage current	I_{LO}			±10	µA	$0 V \le V_O \le V_{DD}$
V_{DD} supply current	I_{DD1}		40	65	mA	Operating mode
	I_{DD2}		20	35	mA	HALT mode
Data retention voltage	V_{DDDR}	2.5			V	STOP mode
Data retention current	I_{DDDR}		2	10	µA	STOP mode V_{DDDR} = 2.5 V
			10	50	µA	V_{DDDR} = 5.0 V ±10%

Notes:

(1) All except $\overline{\text{RESET}}$, X1, X2, $P2_0$/NMI, $P2_1$/INTP0, $P2_2$/INTP1, $P2_3$/INTP2, $P2_4$/INTP3, $P2_5$/$\overline{\text{INTP4}}$, $P2_6$/INTP5, $P2_7$/INTP6/TI, $P3_2$/SB0/SO, $P3_3$/SB1/SI, $P3_4$/$\overline{\text{SCK}}$.

(2) $\overline{\text{RESET}}$, X1, X2, $P2_0$/NMI, $P2_1$/INTP0, $P2_2$/INTP1, $P2_3$/INTP2, $P2_4$/INTP3, $P2_5$/$\overline{\text{INTP4}}$, $P2_6$/INTP5, $P2_7$/INTP6/TI, $P3_2$/SB0/SO, $P3_3$/SB1/SI, $P3_4$/$\overline{\text{SCK}}$.

AC Characteristics

T_A = −10 to +70°C; V_{DD} = +5.0 V ±10%; V_{SS} = 0 V

Parameter	Symbol	Min	Max	Unit	Conditions
Normal External Memory Read/Write Operation ***Turbo Access Manager Data Read/Write Operation*** ***Turbo Access Manager Branch Operation (Fetch Pointer ← address)***					
System clock cycle time	t_{CYK}	125	250	ns	Twice the crystal or external clock input period
Address setup time to ASTB ↓	t_{SAST}	32		ns	t_{CYK} = 125 ns
Address hold after ASTB ↓	t_{HSTA}	32		ns	
Address to $\overline{\text{RD}}$ ↓ delay time	t_{DAR}	85		ns	
$\overline{\text{RD}}$ ↓ to address floating	t_{FRA}		0	ns	
Address to data input	t_{DAID}		222	ns	
$\overline{\text{RD}}$ ↓ to data input	t_{DRID1}		112	ns	
ASTB ↓ to $\overline{\text{RD}}$ ↓ delay time	t_{DSTR}	42		ns	
Data hold time from $\overline{\text{RD}}$ ↑	t_{HRID}	0		ns	
$\overline{\text{RD}}$ ↑ to address active	t_{DRA}	37		ns	
$\overline{\text{RD}}$ width low	t_{WRL}	157		ns	
ASTB width, high	t_{WSTH}	37		ns	
Address to $\overline{\text{WR}}$ ↓ delay	t_{DAW}	85		ns	
ASTB ↓ to data output	t_{DSTOD}		102	ns	

AC Characteristics (cont)

Parameter	Symbol	Min	Max	Unit	Conditions
Normal External Memory Read/Write Operation Turbo Access Manager Data Read/Write Operation Turbo Access Manager Branch Operation (Fetch Pointer ← address) (cont)					
$\overline{WR}$ to data output	t_{DWOD}		40	ns	t_{CYK} = 125 ns
ASTB ↓ to $\overline{WR}$ ↓ delay	t_{DSTW}	42		ns	
Data setup time to $\overline{WR}$ ↑	t_{SODW}	147		ns	
Data hold time after $\overline{WR}$ ↑	t_{HWOD}	32		ns	
$\overline{WR}$ ↑ to ASTB ↑ delay time	t_{DWST}	42		ns	
$\overline{WR}$ width, low	t_{WWL}	157		ns	
Opcode Fetch with Turbo Access Manager: Branch and Continuous Fetch					
$\overline{TAS}$ width, low	t_{WTAL}	37		ns	
$\overline{TAS}$ width, high	t_{WTAH}	42		ns	
$\overline{TAS}$ ↑ to data input	t_{DTAID}		55	ns	
TMD ↑ to $\overline{TAS}$ ↑	t_{DTMRTA}	157		ns	
$\overline{RD}$ ↓ to data input	t_{DRID2}		65	ns	
$\overline{TAS}$ setup to ASTB ↓	t_{STAST}	32		ns	
TMD setup to ASTB ↓	t_{STMST}	42		ns	
TMD ↓ to $\overline{TAS}$ ↑ delay time	t_{DTMFTA}	95		ns	
ASTB ↓ to TMD ↓ delay time	t_{DSTTM}	85		ns	
Data hold after $\overline{TAS}$ ↑	t_{HTMID}	0		ns	

Serial Port Operation

T_A = −10 to +70°C; V_{DD} = +5.0 V ±10%; V_{SS} = 0 V

Parameter	Symbol	Min	Max	Unit	Conditions
$\overline{SCK}$ cycle time	t_{CYSK}	1		μs	$\overline{SCK}$ output from internal clock
		1		μs	$\overline{SCK}$ input from external clock
$\overline{SCK}$ with low	t_{WSKL}	420		ns	$\overline{SCK}$ output from internal clock
		420		ns	$\overline{SCK}$ input from external clock
$\overline{SCK}$ width high	t_{WSKH}	420		ns	$\overline{SCK}$ output from internal clock
		420		ns	$\overline{SCK}$ input from external clock
SI setup time to $\overline{SCK}$ ↑	t_{SRXSK}	80		ns	
SI hold time after $\overline{SCK}$ ↑	t_{HSKRX}	80		ns	
$\overline{SCK}$ ↓ to SO delay time	t_{DSKTX}		210	ns	

Timing Dependent on t_{CYK}

Symbol	Calculation Formula	Min/Max	Unit
t_{SAST}	0.5T - 30	Min	ns
t_{HSTA}	0.5T - 30	Min	ns
t_{DAR}	T - 40	Min	ns
t_{DAID}	(2.5+n)T - 90	Max	ns
t_{DRID1}	(1.5+n)T - 75	Max	ns
t_{DSTR}	0.5T - 20	Min	ns
t_{DRA}	0.5T - 25	Min	ns
t_{WRL}	(1.5+n)T - 30	Min	ns
t_{WSTH}	0.5T - 25	Min	ns
t_{DAW}	T - 40	Min	ns
t_{DSTOD}	0.5T + 40	Max	ns
t_{DSTW}	0.5T - 20	Min	ns
t_{SODW}	1.5T - 40	Min	ns
t_{HWOD}	0.5T - 30	Min	ns
t_{DWST}	0.5T - 20	Min	ns
t_{WWL}	(1.5+n)T - 30	Min	ns
t_{WTAL}	0.5T - 25	Min	ns
t_{WTAH}	0.5T - 20	Min	ns
t_{DTAID}	T - 45	Min	ns
t_{DTMRTA}	1.5T - 30	Min	ns
t_{DRID2}	T - 60	Max	ns
t_{STAST}	0.5T - 30	Min	ns
t_{STMST}	0.5T - 20	Min	ns
t_{DTMFTA}	T - 30	Min	ns
t_{DSTTM}	T - 40	Min	ns

Notes:

(1) n is the number of additional wait cycle specified by the PWC register.

(2) T = t_{CYK} = (ns).

(3) Parameters not included in this table are not dependent on t_{CYK}.

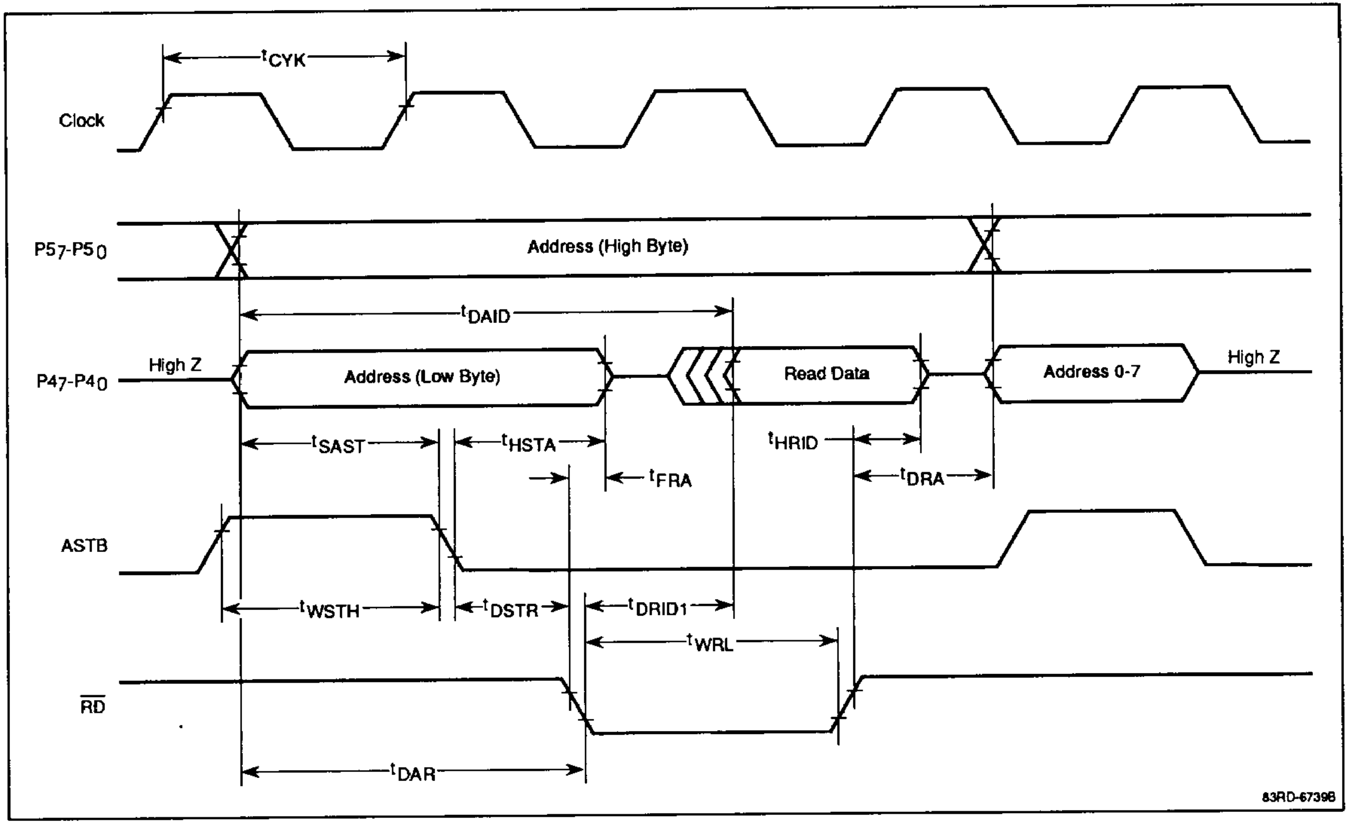
A/D Converter

T_A = −10 to +70°C; V_{DD} = +5.0 V ±10%; AV_{SS} = V_{SS} = 0 V; V_{DD} −0.5 V ≤ AV_{DD} ≤ V_{DD}; 3.4 V ≤ AV_{REF} ≤ V_{DD}

Parameter	Symbol	Min	Typ	Max	Unit
Resolution		10			Bit
Relative accuracy				0.2%	FSR
Quantization error				±1/2	LSB
Conversion time	t_{CONV}	144			t_{CYK}
Sampling time	t_{SAMP}	24			t_{CYK}
Zero offset error			±1.5	LSB	
Full scale error			±1.5	LSB	
Linearity error			±1.5	LSB	
Analog input voltage	V_{IAN}	0		AV_{REF}	V
AV_{REF} current	AI_{REF}		1.0	3.0	mA
AV_{DD} current	AI_{DD}		2.0	6.0	mA

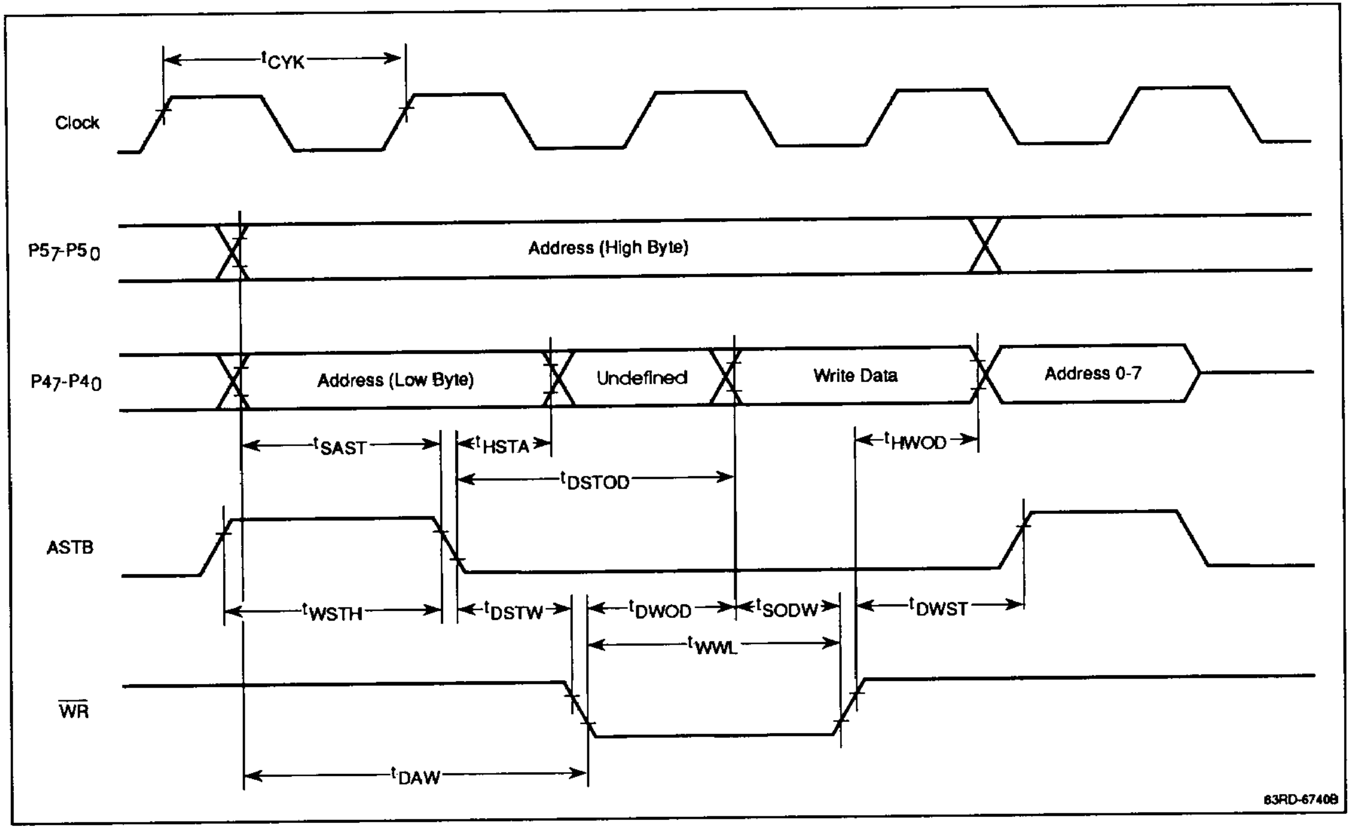
Timing Waveforms

Discontinuous Read Cycle



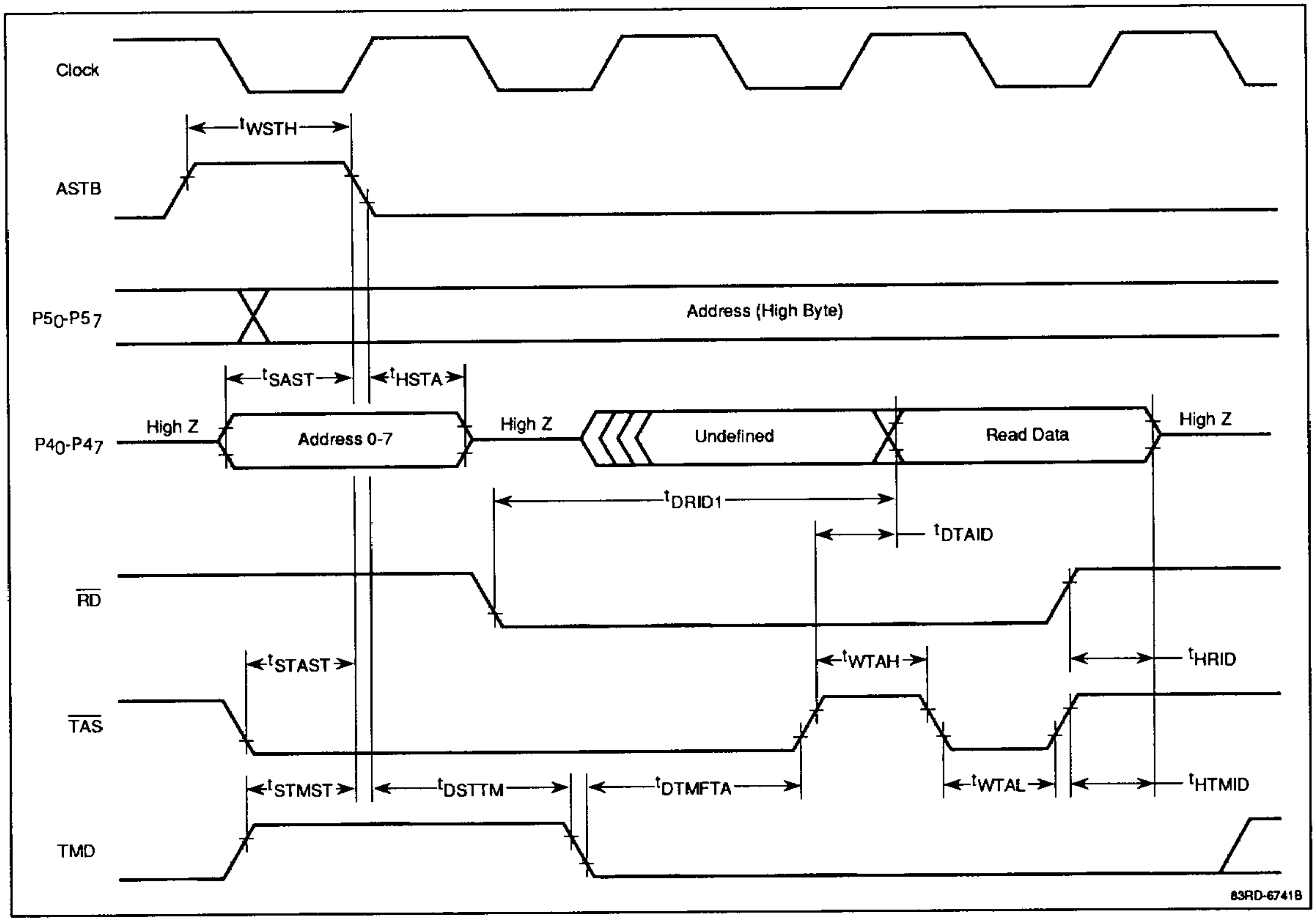
Timing Waveforms (cont)

Discontinuous Write Cycle



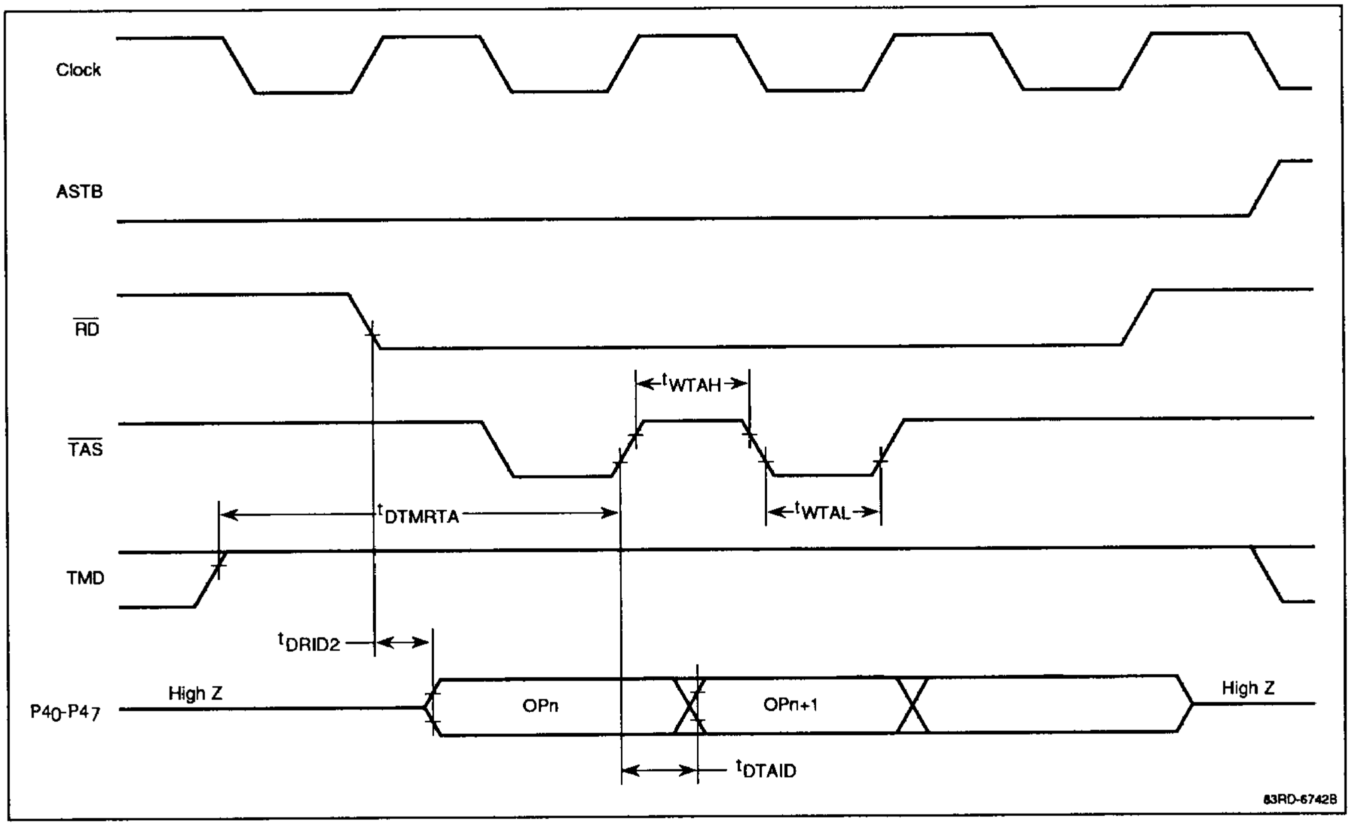
Timing Waveforms (cont)

Branch Cycle, TAM Interface

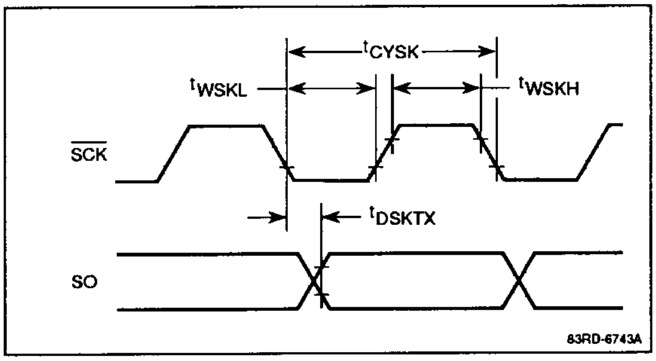


Timing Waveforms (cont)

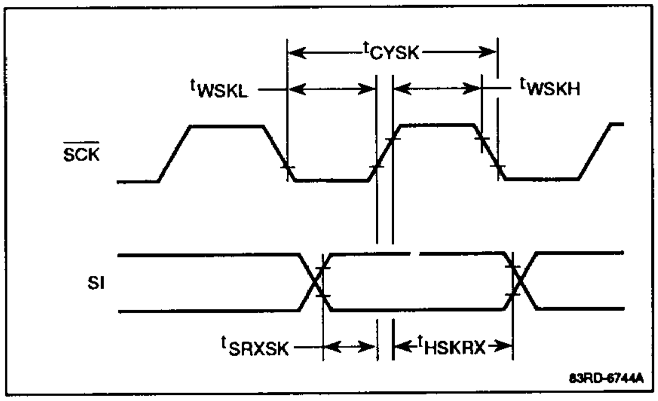
Continuous Instruction Fetch Cycle, TAM Interface



Data Transmit, Serial Port



Data Receive, Serial Port



INSTRUCTION SET

Addressing

On-chip RAM byte location FE20H through FEFFH can be addressed by saddr addressing, in which the machine code specifies the low-order byte only. This addressing mode is also used to address the first 20H special function registers, those with addresses FF00H through FF1FH. Similarly, saddrp addressing is used to specify 16-bit word locations within the same area. The saddrp addresses must be even.

When both source and destination are registers, the destination designation appears in the machine code before the source designation. Similarity, if source and destination are both saddr or saddrp, the destination appears before the source. Both saddr and saddrp addresses are expressed as offsets from either FE00H or FF00H.

Timing

Access to on-chip ROM and to main RAM (FE00H-FEFFH) requires one state per byte. Access to on-chip peripheral RAM (FC80H-FDFFH) and to external memory requires a minimum of three states per byte unless the TAM is used. Instructions can be fetched from the TAM at a rate of one state per byte.

Timing of the PUSH and POP Instructions

The post byte used by the PUSH post, PUSHU post, POP post, and POPU post instructions has a bit set for each register pair to be PUSHed or POPped. Bit 0 specifies RP0, bit 1 RP1,..., bit 7 RP7. The PUSH (and PUSHU) and the POP (and POPU) instructions scan the post byte to determine which register pairs are to be PUSHed or POPped. The PUSH (and PUSHU) instructions begin the scan at the high-order end (bit 7), while the POP (and POPU) instructions begin the scan at the low-order end (bit 0). If the stack is in main RAM (0FE00-0FEFF), the timing formulas are:

PUSH: $t = 3 + 4z + 6n$ states
PUSHU: $t = 4 + 4z + 6n$ states
POP: $t = 6 + 4z + 7n$ states
POPU: $t = 8 + 4z + 7n$ states

where n is the number of register pairs to be PUSHed or POPped, and z is the number of zero bits scanned before all remaining bits are zero.

Example: PUSH RP2, RP3: the post byte is 00001100B.

PUSH: $t = 3 + 4 \times 4 + 6 \times 2 = 31$ states
(4 zeros scanned from high-order end)
POP: $t = 6 + 4 \times 2 + 7 \times 2 = 28$ states
(2 zeros scanned from low-order end)

If the stack is in external RAM or peripheral RAM (0FC80-0FDFF) the formulas become:

PUSH: $t = 3 + 4z + (8 + 2w)n$ states
PUSHU: $t = 4 + 4z + (8 + 2w)n$ states
POP: $t = 6 + 4z + (14 + 2w)n$ states
POPU: $t = 8 + 4z + (14 + 2w)n$ states

where w is the number of additional wait states specified in the PWC register. The timing for the PUSH (and PUSHU) instructions is worst case, and it will improve if the external bus is not busy.

Interrupt Service Timing

Operation	States
Interrupt service by context switch	12
Vector interrupt (stack in main RAM)	17
(stack in any other memory)	31+4n

Macroservice Timing

		States	
Operation		Normal End	Software Interrupt
EVCNT		10	12
DTACMP		15	17
BITSHT		17	19
BITLOG		19	19
ADCBUF		16	26
DTADIF	Byte	22	22
	Word	23	23
DATADIF-P	Byte (1)	24	24
	Byte (2)	26+n	26+n
	Word (1)	25	25
	Word (2)	30+2n	30+2n
DTADD		24	26
BLKTRS mem → sfr	Byte(1)	20	22
	Byte (2)	22+n	24+n
	Word (1)	21	23
	Word (2)	26+2n	28+2n

Macroservice Timing (cont)

		States	
Operation		**Normal End**	**Software Interrupt**
BLKTRS sfr → mem	Byte (1)	19	21
	Byte (2)	19	21
	Word (1)	20	22
	Word (2)	20	22

Notes:

(1) Destination is in main RAM (FE00H-FEFFH).

(2) Destination is anywhere but main RAM.

(3) n = number of additional wait states specified in the PWC register.

In the States column of the Instruction Set, the symbol "n" stands for a number as follows.

Operation	Number "n"
Stack	Register pairs operated on
Shift and rotate	Bits shifted or rotated
String	Characters in the string or the number operated upon before the condition is satisfied

In the States column, a number in parentheses for a conditional branch instruction is the number of states used if the branch is not taken.

Opcodes for Memory Addressing Modes

mod / mem	1 0110	1 0111	0 0110	0 1010
	Register Indirect	Base Index	Base	Index
0 0 0	[DE+] *	[DE+A]	[DE+byte]	word [DE]
0 0 1	[HL+] *	[HL+A]	[SP+byte]	word [A]
0 1 0	[DE−] *	[DE+B]	[HL+byte]	word [HL]
0 1 1	[HL−] *	[HL+B]	[UP+byte]	word [B]
1 0 0	[DE] *	[VP+DE]	[VP+byte]	–
1 0 1	[HL] *	[VP+HL]	–	–
1 1 0	[VP]	–	–	–
1 1 1	[UP]	–	–	–

*One-byte instructions: Defined by special OP Code & mem only.

83RD-6984A

Opcodes for Registers

r

R_3	R_2	R_1	R_0	reg
0	0	0	0	R0
0	0	0	1	R1
0	0	1	0	R2
0	0	1	1	R3
0	1	0	0	R4
0	1	0	1	R5
0	1	1	0	R6
0	1	1	1	R7
1	0	0	0	R8
1	0	0	1	R9
1	0	1	0	R10
1	0	1	1	R11
1	1	0	0	R12
1	1	0	1	R13
1	1	1	0	R14
1	1	1	1	R15

r1

R_2	R_1	R_0	reg
0	0	0	R0
0	0	1	R1
0	1	0	R2
0	1	1	R3
1	0	0	R4
1	0	1	R5
1	1	0	R6
1	1	1	R7

r2

C_0	reg
0	C
1	B

rp

P_2	P_1	P_0	reg-pair
0	0	0	RP0
0	0	1	RP1
0	1	0	RP2
0	1	1	RP3
1	0	0	RP4
1	0	1	RP5
1	1	0	RP6
1	1	1	RP7

rp1

Q_2	Q_1	Q_0	reg-pair
0	0	0	RP0
0	0	1	RP4
0	1	0	RP1
0	1	1	RP5
1	0	0	RP2
1	0	1	RP6
1	1	0	RP3
1	1	1	RP7

rp2

S_1	S_0	reg-pair
0	0	VP
0	1	UP
1	0	DE
1	1	HL

83RD-6985A

Flag Indicators

Symbol	Action
(blank)	No change
0	Set to 0
1	Set to 1
X	Set or cleared according to result
P	P/V indicated parity of result
V	P/V indicates arithmetic overflow
R	Restored from saved PSW

Instruction Set Symbols

Symbol	Definition
r	R0, R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11, R12, R13, R14, R15
r1	R0, R1, R2, R3, R4, R5, R6, R7
r2	C, B
rp	RP0, RP1, RP2, RP3, RP4, RP5, RP6, RP7*
rp1	RP0, RP1, RP2, RP3, RP4, RP5, RP6, RP7*
rp2	DE, HL, VP, UP
sfr	Special function register, 8 bits
sfrp	Special function register, 16 bits
post	RP0, RP1, RP2, RP3, RP4, RP5/PSW, RP6, RP7 Bits set to 1 indicate register pairs to be pushed/ popped to/from stack; RP5 pushed/popped by PUSH/POP, SP is stack pointer; PSW pushed/ popped by PUSHU/POPU, RP5 is stack pointer;
mem	Register indirect: [DE], [HL], [DE+], [HL+], [DE–], [HL–], [VP], [UP] Base Index Mode: [DE+A], [HL+A], [DE+B], [HL+B], [VP+DE], [VP+HL] Base Mode: [DE+byte], [HL+byte], [VP+byte], [UP+byte], [SP+byte] Index Mode: word [A], word [B], word [DE], word [HL]
saddr	FE20-FF1FH: Immediate byte addresses one byte in RAM, or label
saddrp	FE20-FF1FH: Immediate byte (bit 0=0) addresses one word in RAM, or label

Instruction Set Symbols (cont)

Symbol	Definition
word	16 bits of immediate date
byte	8 bits of immediate data
jdisp	8-bit two's complement displacement (immediate data)
f_0-f_{10}	Eleven bits of immediate data corresponding to addr11
t_0-t_4	Five bits of immediate data corresponding to addr5
bit	3 bits of immediate data (bit position in byte), or label
n	3 bits of immediate data
!addr16	16-bit absolute address specified by an immediate address or label
$addr16	Relative branch address [(PC)+jdisp] or label
addr16	16-bit address
!addr11	11-bit immediate address or label
addr11	0800H-0FFFH: 0800H + (11-bit immediate address), or label
addr5	0040H-007EH: 0040H + 2 X (5-bit immediate address), or label
A	A register
X	X register
B	B register
C	C register
D	D register
E	E register
H	H register
L	L register
R0-R15	Register 0 to register 15
AX	Register pair AX (16-bit accumulator)
BC	Register pair BC
DE	Register pair DE
HL	Register pair HL

Instruction Set Symbols (cont)

Symbol	Definition
RP0-RP7	Register pair 0 to register pair 7
PC	Program counter
SP	Stack pointer
UP	User stack pointer (RP5)
PSW	Program status word
CY	Carry flag
AC	Auxiliary carry flag
Z	Zero flag
P/V	Parity/overflow flag
S	Sign flag
SUB	Subtract flag
TPF	Table position flag
RBS	Register bank select flag
RSS	Register set select flag
IE	Interrupt enable flag
STBC	Standby control register
WDM	Watchdog timer mode register

Symbol	Definition
( )	Contents of the location whose address is within parentheses; (+) and (-) indicate that the address is incremented after or decremented after it is used
(( ))	Contents of the memory location defined by the quantity within the sets of parentheses
xxH	Hexadecimal quantity
X_H, X_L	High-order 8 bits and low-order 8 bits of X

* rp and rp1 describe the same registers but generate different machine code.

Instruction Set

Mnemonic	Operand	Operation	Bytes	States	Flags					Operation Code							
					S	Z	AC	P/V	CY	7	6	5	4	3	2	1	0
8-Bit Data Transfer																	
MOV	r1, #byte	r1 ← byte	2	2						1	0	1	1	1	R_2	R_1	R_0
										Data							
	saddr, #byte	(saddr) ← byte	3	3						0	0	1	1	1	0	1	0
										Saddr-offset							
										Data							
	sfr**, #byte	sfr ← byte	3	6						0	0	1	0	1	0	1	1
										Sfr-offset							
										Data							
	r, r1	r ← r1	2	3						0	0	1	0	0	1	0	0
										R_3	R_2	R_1	R_0	0	R_2	R_1	R_0
	A, r1	A ← r1	1	2						1	1	0	1	0	R_2	R_1	R_0
	A, saddr	A ← (saddr)	2	3						0	0	1	0	0	0	0	0
										Saddr-offset							
	saddr, A	(saddr) ← A	2	3						0	0	1	0	0	0	1	0
										Saddr-offset							
	saddr, saddr	(saddr) ← (saddr)	3	4						0	0	1	1	1	0	0	0
										Saddr-offset							
										Saddr-offset							
	A, sfr	A ← sfr	3	4						0	0	0	1	0	0	0	0
										Sfr-offset							
	sfr, A	sfr ← A	2	6						0	0	0	1	0	0	1	0
										Sfr-offset							
	A, mem*	A ← (mem)	1	6						0	1	0	1	1	mem		
	A, mem	A ← (mem)	2-4	8-10						0	0	0	mod				
										0	mem			0	0	0	0
										Low Offset							
										High Offset							
	mem, A*	(mem) ← A	1	4						0	1	0	1	0	mem		
	mem, A	(mem) ← A	2-4	6-8						0	0	0	mod				
										1	mem			0	0	0	0
										Low Offset							
										High Offset							
	A, [saddrp]	A ← ((saddrp))	2	6						0	0	0	1	1	0	0	0
										Saddr-offset							
	[saddrp], A	((saddrp)) ← A	2	4						0	0	0	1	1	0	0	1
										Saddr-offset							
	A, !addr16	A ← (addr16)	4	6						0	0	0	0	1	0	0	1
										1	1	1	1	0	0	0	0
										Low Addr							
										High Addr							

* One byte move instruction when [DE], [HL], [DE+], [DE−], [HL+], or [HL−] is specified for mem.

** A special instruction is used to write to STBC and WDM.

Instruction Set (cont)

					Flags					Operation Code							
Mnemonic	**Operand**	**Operation**	**Bytes**	**States**	**S**	**Z**	**AC**	**P/V**	**CY**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
8-Bit Data Transfer (cont)																	
MOV (cont)	!addr16, A	(addr16) ← A	4	5						0	0	0	0	1	0	0	1
										1	1	1	1	0	0	0	1
										Low Addr							
										High Addr							
	PSWL, #byte	PSW_L ← byte	3	6	X	X	X	X	X	0	0	1	0	1	0	1	1
										1	1	1	1	1	1	1	0
										Data							
	PSWH, #byte	PSW_H ← byte	3	6						0	0	1	0	1	0	1	1
										1	1	1	1	1	1	1	1
										Data							
	PSWL, A	PSW_L ← A	2	6	X	X	X	X	X	0	0	0	1	0	0	1	0
										1	1	1	1	1	1	1	0
	PSWH, A	PSW_H ← A	2	6						0	0	0	1	0	0	1	0
										1	1	1	1	1	1	1	1
	A, PSWL	A ← PSW_L	2	6						0	0	0	1	0	0	0	0
										1	1	1	1	1	1	1	0
	A, PSWH	A ← PSW_H	2	6						0	0	0	1	0	0	0	0
										1	1	1	1	1	1	1	1
XCH	A, r1	A ↔ r1	1	4						1	1	0	1	1	R_2	R_1	R_0
	r, r1	r ↔ r1	2	4						0	0	1	0	0	1	0	1
										R_3	R_2	R_1	R_0	0	R_2	R_1	R_0
	A, mem	A ↔ (mem)	2-4	9-11						0	0	0	mod				
										0	mem			0	1	0	0
										Low Offset							
										High Offset							
	A, saddr	A ↔ (saddr)	2	5						0	0	1	0	0	0	0	1
										Saddr-offset							
	A, sfr	A ↔ sfr	3	13						0	0	0	0	0	0	0	1
										0	0	1	0	0	0	0	1
										Sfr-offset							
	A, [saddrp]	A ↔ ((saddrp))	2	7						0	0	1	0	0	0	1	1
										Saddr-offset							
	saddr, saddr	(saddr) ↔ (saddr)	3	8						0	0	1	1	1	0	0	1
										Saddr-offset							
										Saddr-offset							

Instruction Set (cont)

					Flags					Operation Code							
Mnemonic	Operand	Operation	Bytes	States	S	Z	AC	P/V	CY	7	6	5	4	3	2	1	0
16-Bit Data Transfer																	
MOVW	rp1, #word	rp1 ← word	3	3						0	1	1	0	0	Q_2	Q_1	Q_0
										Low Byte							
										High Byte							
	saddrp, #word	(saddrp) ← word	4	4						0	0	0	0	1	1	0	0
										Saddr-offset							
										Low Byte							
										High Byte							
	sfrp, #word	sfrp ← word	4	7						0	0	0	0	1	0	1	1
										Sfr-offset							
										Low Byte							
										High Byte							
	rp, rp1	rp ← rp1	2	3						0	0	1	0	0	1	0	0
										P_2	P_1	P_0	0	1	Q_2	Q_1	Q_0
	AX, saddrp	AX ← (saddrp)	2	3						0	0	0	1	1	1	0	0
										Saddr-offset							
	saddrp, AX	(saddrp) ← AX	2	3						0	0	0	1	1	0	1	0
										Saddr-offset							
	saddrp, saddrp	(saddrp) ← (saddrp)	3	4						0	0	1	1	1	1	0	0
										Saddr-offset							
										Saddr-offset							
	AX, sfrp	AX ← sfrp	2	6						0	0	0	1	0	0	0	1
										Sfr-offset							
	sfrp, AX	sfrp ← AX	2	6						0	0	0	1	0	0	1	1
										Sfr-offset							
	rp1, !addr16	rp1 ← (addr16)	4	7						0	0	0	0	1	0	0	1
										1	0	0	0	0	Q_2	Q_1	Q_0
										Low Addr							
										High Addr							
	!addr16, rp1	(addr16) ← rp1	4	5						0	0	0	0	1	0	0	1
										1	0	0	1	0	Q_2	Q_1	Q_0
										Low Addr							
										High Addr							
	AX, mem	AX ← (mem)	2-4	6-10						0	0	0	mod				
										0	mem			0	0	0	1
										Low-offset							
										High-offset							
	mem, AX	(mem) ← AX	2-4	4-8						0	0	0	mod				
										1	mem			0	0	0	1
										Low-offset							
										High-offset							

Instruction Set (cont)

					Flags					Operation Code							
Mnemonic	**Operand**	**Operation**	**Bytes**	**States**	**S**	**Z**	**AC**	**P/V**	**CY**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
16-Bit Data Transfer (cont)																	
XCHW	AX, saddrp	AX ↔ (saddrp)	2	5						0	0	0	1	1	0	1	1
										Saddr-offset							
	AX, sfrp	AX ↔ sfrp	3	13						0	0	0	0	0	0	0	1
										0	0	0	1	1	0	1	1
										Sfr-offset							
	saddrp, saddrp	(saddrp) ↔ (saddrp)	3	8						0	0	1	0	1	0	1	0
										Saddr-offset							
										Saddr-offset							
	rp, rp1	rp ↔ rp1	2	4						0	0	1	0	0	1	0	1
										P_2	P_1	P_0	0	1	Q_2	Q_1	Q_0
	AX, mem	AX ↔ (mem)	2-4	9-11						0	0	0	mod				
										0	mem			0	1	0	1
										Low-offset							
										High-offset							
8-Bit Arithmetic																	
ADD	A, #byte	A, CY ← A + byte	2	2	X	X	X	V	X	1	0	1	0	1	0	0	0
										Data							
	saddr, #byte	(saddr), CY ← (saddr) + byte	3	4	X	X	X	V	X	0	1	1	0	1	0	0	0
										Saddr-offset							
										Data							
	sfr, #byte	sfr, CY ← sfr + byte	4	12	X	X	X	V	X	0	0	0	0	0	0	0	1
										0	1	1	0	1	0	0	0
										Sfr-offset							
										Data							
	r, r1	r, CY ← r + r1	2	3	X	X	X	V	X	1	0	0	0	1	0	0	0
										R_3	R_2	R_1	R_0	0	R_2	R_1	R_0
	A, saddr	A, CY ← A + (saddr)	2	4	X	X	X	V	X	1	0	0	1	1	0	0	0
										Saddr-offset							
	A, sfr	A, CY ← A + sfr	3	9	X	X	X	V	X	0	0	0	0	0	0	0	1
										1	0	0	1	1	0	0	0
										Sfr-offset							
	saddr, saddr	(saddr), CY ← (saddr) + (saddr)	3	5	X	X	X	V	X	0	1	1	1	1	0	0	0
										Saddr-offset							
										Saddr-offset							

Instruction Set (cont)

					Flags					Operation Code							
Mnemonic	**Operand**	**Operation**	**Bytes**	**States**	**S**	**Z**	**AC**	**P/V**	**CY**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
8-Bit Arithmetic (cont)																	
ADD (cont)	A, mem	A, CY ← A + (mem)	2-4	8-9	X	X	X	V	X	0	0	0	mod				
										0	mem			1	0	0	0
										Low Offset							
										High Offset							
	mem, A	(mem), CY ← (mem) + A	2-4	8-9	X	X	X	V	X	0	0	0	mod				
										1	mem			1	0	0	0
										Low Offset							
										High Offset							
ADDC	A, #byte	A, CY ← A + byte + CY	2	2	X	X	X	V	X	1	0	1	0	1	0	0	1
										Data							
	saddr, #byte	(saddr), CY ← (saddr) + byte + CY	3	4	X	X	X	V	X	0	1	1	0	1	0	0	1
										Saddr-offset							
										Data							
	sfr, #byte	sfr, CY ← sfr + byte + CY	4	12	X	X	X	V	X	0	0	0	0	0	0	0	1
										0	1	1	0	1	0	0	1
										Sfr-offset							
										Data							
	r, r1	r, CY ← r + r1 + CY	2	3	X	X	X	V	X	1	0	0	0	1	0	0	1
										R_3	R_2	R_1	R_0	0	R_2	R_1	R_0
	A, saddr	A, CY ← A + (saddr) + CY	2	4	X	X	X	V	X	1	0	0	1	1	0	0	1
										Saddr-offset							
	A, sfr	A, CY ← A + sfr + CY	3	9	X	X	X	V	X	0	0	0	0	0	0	0	1
										1	0	0	1	1	0	0	1
										Sfr-offset							
	saddr, saddr	(saddr), CY ← (saddr) + (saddr) + CY	3	5	X	X	X	V	X	0	1	1	1	1	0	0	1
										Saddr-offset							
										Saddr-offset							
	A, mem	A, CY ← A + (mem) + CY	2-4	8-9	X	X	X	V	X	0	0	0	mod				
										0	mem			1	0	0	1
										Low Offset							
										High Offset							
	mem, A	(mem), CY ← (mem) + A + CY	2-4	8-9	X	X	X	V	X	0	0	0	mod				
										1	mem			1	0	0	1
										Low Offset							
										High Offset							

Instruction Set (cont)

					Flags					Operation Code							
Mnemonic	Operand	Operation	Bytes	States	S	Z	AC	P/V	CY	7	6	5	4	3	2	1	0
8-Bit Arithmetic (cont)																	
SUB	A, #byte	A, CY ← A − byte	2	2	X	X	X	V	X	1	0	1	0	1	0	1	0
										Data							
	saddr, #byte	(saddr), CY ← (saddr) − byte	3	4	X	X	X	V	X	0	1	1	0	1	0	1	0
										Saddr-offset							
										Data							
	sfr, #byte	sfr, CY ← sfr − byte	4	12	X	X	X	V	X	0	0	0	0	0	0	0	1
										0	1	1	0	1	0	1	0
										Sfr-offset							
										Data							
	r, r1	r, CY ← r − r1	2	3	X	X	X	V	X	1	0	0	0	1	0	1	0
										R_3	R_2	R_1	R_0	0	R_2	R_1	R_0
	A, saddr	A, CY ← A − (saddr)	2	4	X	X	X	V	X	1	0	0	1	1	0	1	0
										Saddr-offset							
	A, sfr	A, CY ← A − sfr	3	9	X	X	X	V	X	0	0	0	0	0	0	0	1
										1	0	0	1	1	0	1	0
										Sfr-offset							
	saddr, saddr	(saddr), CY ← (saddr) − (saddr)	3	5	X	X	X	V	X	0	1	1	1	1	0	1	0
										Saddr-offset							
										Saddr-offset							
	A, mem	A, CY ← A − (mem)	2-4	8-9	X	X	X	V	X	0	0	0	mod				
										0	mem			1	0	1	0
										Low Offset							
										High Offset							
	mem, A	(mem), CY ← (mem) − A	2-4	8-9	X	X	X	V	X	0	0	0	mod				
										1	mem			1	0	1	0
										Low Offset							
										High Offset							
SUBC	A, #byte	A, CY ← A − byte − CY	2	2	X	X	X	V	X	1	0	1	0	1	0	1	1
										Data							
	saddr, #byte	(saddr), CY ← (saddr) − byte − CY	3	4	X	X	X	V	X	0	1	1	0	1	0	1	1
										Saddr-offset							
										Data							
	sfr, #byte	sfr, CY ← sfr − byte − CY	4	12	X	X	X	V	X	0	0	0	0	0	0	0	1
										0	1	1	0	1	0	1	1
										Sfr-offset							
										Data							

Instruction Set (cont)

					Flags					Operation Code							
Mnemonic	Operand	Operation	Bytes	States	S	Z	AC	P/V	CY	7	6	5	4	3	2	1	0
8-Bit Arithmetic (cont)																	
SUBC (cont)	r, r1	r, CY ← r−r1−CY	2	3	X	X	X	V	X	1	0	0	0	1	0	1	1
										R_3	R_2	R_1	R_0	0	R_2	R_1	R_0
	A, saddr	A, CY ← A−(saddr)−CY	2	4	X	X	X	V	X	1	0	0	1	1	0	1	1
										Saddr-offset							
	A, sfr	A, CY ← A−sfr−CY	3	9	X	X	X	V	X	0	0	0	0	0	0	0	1
										1	0	0	1	1	0	1	1
										Sfr-offset							
	saddr, saddr	(saddr), CY ← (saddr)−(saddr)−CY	3	5	X	X	X	V	X	0	1	1	1	1	0	1	1
										Saddr-offset							
										Saddr-offset							
	A, mem	A, CY ← A−(mem)−CY	2-4	8-9	X	X	X	V	X	0	0	0	mod				
										0	mem			1	0	1	1
										Low Offset							
										High Offset							
	mem, A	(mem), CY ← (mem)−A−CY	2-4	8-9	X	X	X	V	X	0	0	0	mod				
										1	mem			1	0	1	1
										Low Offset							
										High Offset							
8-Bit Logic																	
AND	A, #byte	A ← A AND byte	2	2	X	X		P		1	0	1	0	1	1	0	0
										Data							
	saddr, #byte	(saddr) ← (saddr) AND byte	3	4	X	X		P		0	1	1	0	1	1	0	0
										Saddr-offset							
										Data							
	sfr, #byte	sfr ← sfr AND byte	4	12	X	X		P		0	0	0	0	0	0	0	1
										0	1	1	0	1	1	0	0
										Sfr-offset							
										Data							
	r, r1	r ← r AND r1	2	3	X	X		P		1	0	0	0	1	1	0	0
										R_3	R_2	R_1	R_0	0	R_2	R_1	R_0
	A, saddr	A ← A AND (saddr)	2	4	X	X		P		1	0	0	1	1	1	0	0
										Saddr-offset							
	A, sfr	A ← A AND sfr	3	9	X	X		P		0	0	0	0	0	0	0	1
										1	0	0	1	1	1	0	0
										Sfr-offset							

Instruction Set (cont)

					Flags					Operation Code							
Mnemonic	**Operand**	**Operation**	**Bytes**	**States**	**S**	**Z**	**AC**	**P/V**	**CY**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
8-Bit Logic (cont)																	
AND (cont)	saddr, saddr	(saddr) ← (saddr) AND (saddr)	3	5	X	X		P		0	1	1	1	1	1	0	0
										Saddr-offset							
										Saddr-offset							
	A, mem	A ← A AND (mem)	2-4	8-9	X	X		P		0	0	0	mod				
										0	mem			1	1	0	0
										Low Offset							
										High Offset							
	mem, A	(mem) ← (mem) AND A	2-4	8-9	X	X		P		0	0	0	mod				
										1	mem			1	1	0	0
										Low Offset							
										High Offset							
OR	A, #byte	A ← A OR byte	3	4	X	X		P		1	0	1	0	1	1	1	0
										Data							
	saddr, #byte	(saddr) ← (saddr) OR byte	3	4	X	X		P		0	1	1	0	1	1	1	0
										Saddr-offset							
										Data							
	sfr, #byte	sfr ← sfr OR byte	4	12	X	X		P		0	0	0	0	0	0	0	
										0	1	1	0	1	1	1	0
										Sfr-offset							
										Data							
	r, r1	r ← r OR r1	2	3	X	X		P		1	0	0	0	1	1	1	0
										R_3	R_2	R_1	R_0	0	R_2	R_1	R_0
	A, saddr	A ← A OR (saddr)	2	4	X	X		P		1	0	0	1	1	1	1	0
										Saddr-offset							
	A, sfr	A ← A OR sfr	3	9	X	X		P		0	0	0	0	0	0	0	1
										1	0	0	1	1	1	1	0
										Sfr-offset							
	saddr, saddr	(saddr) ← (saddr) OR (saddr)	3	5	X	X		P		0	1	1	1	1	1	1	0
										Saddr-offset							
										Saddr-offset							
	A, mem	A ← A OR (mem)	2-4	8-9	X	X		P		0	0	0	mod				
										0	mem			1	1	1	0
										Low Offset							
										High Offset							
	mem, A	(mem) ← (mem) OR A	2-4	8-9	X	X		P		0	0	0	mod				
										1	mem			1	1	1	0
										Low Offset							
										High Offset							

Instruction Set (cont)

					Flags					Operation Code							
Mnemonic	Operand	Operation	Bytes	States	S	Z	AC	P/V	CY	7	6	5	4	3	2	1	0
8-Bit Logic (cont)																	
XOR	A, #byte	A ← A XOR byte	2	2	X	X		P		1	0	1	0	1	1	0	1
										Data							
	saddr, #byte	(saddr) ← (saddr) XOR byte	3	4	X	X		P		0	1	1	0	1	1	0	1
										Saddr-offset							
										Data							
	sfr, #byte	sfr ← sfr XOR byte	4	12	X	X		P		0	0	0	0	0	0	0	1
										0	1	1	0	1	1	0	1
										Sfr-offset							
										Data							
	r, r1	r ← r XOR r1	2	3	X	X		P		1	0	0	0	1	1	0	1
										R_3	R_2	R_1	R_0	0	R_2	R_1	R_0
	A, saddr	A ← A XOR (saddr)	2	4	X	X		P		1	0	0	1	1	1	0	1
										Saddr-offset							
	A, sfr	A ← A XOR sfr	3	9	X	X		P		0	0	0	0	0	0	0	1
										1	0	0	1	1	1	0	1
										Sfr-offset							
	saddr, saddr	(saddr) ← (saddr) XOR (saddr)	3	5	X	X		P		0	1	1	1	1	1	0	1
										Saddr-offset							
										Saddr-offset							
	A, mem	A ← A XOR (mem)	2-4	8-9	X	X		P		0	0	0	mod				
										0	mem			1	1	0	1
										Low Offset							
										High Offset							
	mem, A	(mem) ← (mem) XOR A	2-4	8-9	X	X		P		0	0	0	mod				
										1	mem			1	1	0	1
										Low Offset							
										High Offset							
CMP	A, #byte	A − byte	2	2	X	X	X	V	X	1	0	1	0	1	1	1	1
										Data							
	saddr, #byte	(saddr) − byte	3	4	X	X	X	V	X	0	1	1	0	1	1	1	1
										Saddr-offset							
										Data							
	sfr, #byte	sfr − byte	4	12	X	X	X	V	X	0	0	0	0	0	0	0	1
										0	1	1	0	1	1	1	1
										Sfr-offset							
										Data							

Instruction Set (cont)

					Flags					Operation Code							
Mnemonic	Operand	Operation	Bytes	States	S	Z	AC	P/V	CY	7	6	5	4	3	2	1	0
8-Bit Logic (cont)																	
CMP (cont)	r, r1	r − r1	2	3	X	X	X	V	X	1	0	0	0	1	1	1	1
										R_3	R_2	R_1	R_0	0	R_2	R_1	R_0
	A, saddr	A − (saddr)	2	4	X	X	X	V	X	1	0	0	1	1	1	1	1
										Saddr-offset							
	A, sfr	A − sfr	3	9	X	X	X	V	X	0	0	0	0	0	0	0	1
										1	0	0	1	1	1	1	1
										Sfr-offset							
	saddr, saddr	(saddr) − (saddr)	3	5	X	X	X	V	X	0	1	1	1	1	1	1	1
										Saddr-offset							
										Saddr-offset							
	A, mem	A − (mem)	2-4	8-9	X	X	X	V	X	0	0	0	mod				
										0	mem			1	1	1	1
										Low Offset							
										High Offset							
	mem, A	(mem) − A	2-4	8-9	X	X	X	V	X	0	0	0	mod				
										1	mem			1	1	1	1
										Low Offset							
										High Offset							
16-Bit Arithmetic																	
ADDW	AX, #word	AX, CY ← AX + word	3	3	X	X	X	V	X	0	0	1	0	1	1	0	1
										Low Byte							
										High Byte							
	saddrp, #word	(saddrp), CY ← (saddrp) + word	4	5	X	X	X	V	X	0	0	0	0	1	1	0	1
										Saddr-offset							
										Low Byte							
										High Byte							
	sfrp, #word	sfrp, CY ← sfrp + word	5	10	X	X	X	V	X	0	0	0	0	0	0	0	1
										0	0	0	0	1	1	0	1
										Sfr-offset							
										Low Byte							
										High Byte							
	rp, rp1	rp, CY ← rp + rp1	2	3	X	X	X	V	X	1	0	0	0	1	0	0	0
										P_2	P_1	P_0	0	1	Q_2	Q_1	Q_0
	AX, saddrp	AX, CY ← AX + (saddrp)	2	4	X	X	X	V	X	0	0	0	1	1	1	0	1
										Saddr-offset							

Instruction Set (cont)

					Flags					Operation Code							
Mnemonic	Operand	Operation	Bytes	States	S	Z	AC	P/V	CY	7	6	5	4	3	2	1	0
16-Bit Arithmetic (cont)																	
ADDW (cont)	AX, sfrp	AX, CY ← AX + sfrp	3	9	X	X	X	V	X	0	0	0	0	0	0	0	1
										0	0	0	1	1	1	0	1
										Sfr-offset							
	saddrp, saddrp	(saddrp), CY ← (saddrp) + (saddrp)	3	5	X	X	X	V	X	0	0	1	1	1	1	0	1
										Saddr-offset							
										Saddr-offset							
SUBW	AX, #word	AX, CY ← AX − word	3	3	X	X	X	V	X	0	0	1	0	1	1	1	0
										Low Byte							
										High Byte							
	saddrp, #word	(saddrp), CY ← (saddrp) − word	4	5	X	X	X	V	X	0	0	0	0	1	1	1	0
										Saddr-offset							
										Low Byte							
										High Byte							
	sfrp, #word	sfrp, CY ← sfrp − word	5	10	X	X	X	V	X	0	0	0	0	0	0	0	1
										0	0	0	0	1	1	1	0
										Sfr-offset							
										Low Byte							
										High Byte							
	rp, rp1	rp, CY ← rp − rp1	2	3	X	X	X	V	X	1	0	0	0	1	0	1	0
										P_2	P_1	P_0	0	1	Q_2	Q_1	Q_0
	AX, saddrp	AX, CY ← AX − (saddrp)	2	4	X	X	X	V	X	0	0	0	1	1	1	1	0
										Saddr-offset							
	AX, sfrp	AX, CY ← AX − sfrp	3	9	X	X	X	V	X	0	0	0	0	0	0	0	1
										0	0	0	1	1	1	1	0
										Sfr-offset							
	saddrp, saddrp	(saddrp), CY ← (saddrp) − (saddrp)	3	5	X	X	X	V	X	0	0	1	1	1	1	1	0
										Saddr-offset							
										Saddr-offset							
CMPW	AX, #word	AX − word	3	3	X	X	X	V	X	0	0	1	0	1	1	1	1
										Low Byte							
										High Byte							
	saddrp, #word	(saddrp) − word	4	5	X	X	X	V	X	0	0	0	0	1	1	1	1
										Saddr-offset							
										Low Byte							
										High Byte							

Instruction Set (cont)

					Flags					Operation Code							
Mnemonic	**Operand**	**Operation**	**Bytes**	**States**	**S**	**Z**	**AC**	**P/V**	**CY**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
16-Bit Arithmetic (cont)																	
CMPW (cont)	sfrp, #word	sfrp − word	5	10	X	X	X	V	X	0	0	0	0	0	0	0	1
										0	0	0	0	1	1	1	1
										Sfr-offset							
										Low Byte							
										High Byte							
	rp, rp1	rp − rp1	2	3	X	X	X	V	X	1	0	0	0	1	1	1	1
										P_2	P_1	P_0	0	1	Q_2	Q_1	Q_0
	AX, saddrp	AX − (saddrp)	2	4	X	X	X	V	X	0	0	0	1	1	1	1	1
										Saddr-offset							
	AX, sfrp	AX − sfrp	3	9	X	X	X	V	X	0	0	0	0	0	0	0	1
										0	0	0	1	1	1	1	1
										Sfr-offset							
	saddrp, saddrp	(saddrp) − (saddrp)	3	5	X	X	X	V	X	0	0	1	1	1	1	1	1
										Saddr-offset							
										Saddr-offset							
Multiplication/Division																	
MULU	r1	AX ← A × r1	2	14						0	0	0	0	0	1	0	1
										0	0	0	0	1	R_2	R_1	R_0
DIVUW	r1	AX (Quotient), r1 (Remainder) ← AX ÷ r1	2	23						0	0	0	0	0	1	0	1
										0	0	0	1	1	R_2	R_1	R_0
MULUW	rp1	AX (High Order 16 Bits), rp1 (Low Order 16 Bits), ← AX × rp1	2	22						0	0	0	0	0	1	0	1
										0	0	1	0	1	Q_2	Q_1	Q_0
DIVUX	rp1	AXDE (Quotient), rp1 (Remainder) ← AXDE ÷ rp1	2	43						0	0	0	0	0	1	0	1
										1	1	1	0	1	Q_2	Q_1	Q_0
MULW*	rp1	AX (High Order 16 Bits), rp1 (Low Order 16 Bits), ← AX × rp1	2	24-28						0	0	0	0	0	1	0	1
										0	0	1	1	1	Q_2	Q_1	Q_0
Increment/Decrement																	
INC	r1	r1 ← r1 + 1	1	2	X	X	X	V		1	1	0	0	0	R_2	R_1	R_0
	saddr	(saddr) ← (saddr) + 1	2	3	X	X	X	V		0	0	1	0	0	1	1	0
										Saddr-offset							
DEC	r1	r1 ← r1 − 1	1	2	X	X	X	V		1	1	0	0	1	R_2	R_1	R_0
	saddr	(saddr) ← (saddr) − 1	2	3	X	X	X	V		0	0	1	0	0	1	1	1
										Saddr-offset							
INCW	rp2	rp2 ← rp2 + 1	1	2						0	1	0	0	0	1	S_1	S_0
	saddrp	(saddrp) ← (saddrp) + 1	3	4						0	0	0	0	0	1	1	1
										1	1	1	0	1	0	0	0
										Saddr-offset							

* 16-bit signed multiply instruction.

Instruction Set (cont)

					Flags					Operation Code							
Mnemonic	Operand	Operation	Bytes	States	S	Z	AC	P/V	CY	7	6	5	4	3	2	1	0
Increment/Decrement (cont)																	
DECW	rp2	$rp2 \leftarrow rp2-1$	1	2						0	1	0	0	1	1	S_1	S_0
	saddrp	$(saddrp) \leftarrow (saddrp)-1$	3	4						0	0	0	0	0	1	1	1
										1	1	1	0	1	0	0	1
										Saddr-offset							
Shift/Rotate																	
ROR	r1, n	$(CY, r1_7 \leftarrow r1_0,$ $r1_{m-1} \leftarrow r1_m) \times n$	2	6+n				P	X	0	0	1	1	0	0	0	0
										0	1	N_2	N_1	N_0	R_2	R_1	R_0
ROL	r1, n	$(CY, r1_0 \leftarrow r1_7,$ $r1_{m+1} \leftarrow r1_m) \times n$	2	6+n				P	X	0	0	1	1	0	0	0	1
										0	1	N_2	N_1	N_0	R_2	R_1	R_0
RORC	r1, n	$(CY \leftarrow r1_0, r1_7 \leftarrow CY,$ $r1_{m-1} \leftarrow r1_m) \times n$	2	6+n				P	X	0	0	1	1	0	0	0	0
										0	0	N_2	N_1	N_0	R_2	R_1	R_0
ROLC	r1, n	$(CY \leftarrow r1_7, r1_0 \leftarrow CY,$ $r1_{m+1} \leftarrow r1_m) \times n$	2	6+n				P	X	0	0	1	1	0	0	0	1
										0	0	N_2	N_1	N_0	R_2	R_1	R_0
SHR	r1, n	$(CY \leftarrow r1_0, r1_7 \leftarrow 0,$ $r1_{m-1} \leftarrow r1_m) \times n$	2	6+n	X	X	0	P	X	0	0	1	1	0	0	0	0
										1	0	N_2	N_1	N_0	R_2	R_1	R_0
SHL	r1, n	$(CY \leftarrow r1_7, r1_0 \leftarrow 0,$ $r1_{m+1} \leftarrow r1_m) \times n$	2	6+n	X	X	0	P	X	0	0	1	1	0	0	0	1
										1	0	N_2	N_1	N_0	R_2	R_1	R_0
SHRW	rp1, n	$(CY \leftarrow rp1_0, rp1_{15} \leftarrow 0,$ $rp1_{m-1} \leftarrow rp1_m) \times n$	2	6+n	X	X	0	P	X	0	0	1	1	0	0	0	0
										1	1	N_2	N_1	N_0	Q_2	Q_1	Q_0
SHLW	rp1, n	$(CY \leftarrow rp1_{15}, rp1_0 \leftarrow 0,$ $rp1_{m+1} \leftarrow rp1_m) \times n$	2	6+n	X	X	0	P	X	0	0	1	1	0	0	0	1
										1	1	N_2	N_1	N_0	Q_2	Q_1	Q_0
ROR4	[rp1]	$A_{3-0} \leftarrow (rp1)_{3-0}, (rp1)_{7-4} \leftarrow A_{3-0},$ $(rp1)_{3-0} \leftarrow (rp1)_{7-4}$	2	8						0	0	0	0	0	1	0	1
										1	0	0	0	1	Q_2	Q_1	Q_0
ROL4	[rp1]	$A_{3-0} \leftarrow (rp1)_{7-4}, (rp1)_{3-0} \leftarrow A_{3-0},$ $(rp1)_{7-4} \leftarrow (rp1)_{3-0}$	2	8						0	0	0	0	0	1	0	1
										1	0	0	1	1	Q_2	Q_1	Q_0
BCD Adjustment																	
ADJBA		Decimal Adjust Accumulator after add	2	5	X	X	X	P	X	0	0	0	0	0	1	0	1
										1	1	1	1	1	1	1	0
ADJBS		Decimal Adjust Accumulator after subtract	2	5	X	X	X	P	X	0	0	0	0	0	1	0	1
										1	1	1	1	1	1	1	1
Data Expansion																	
CVTBW		$X \leftarrow A, A_{6-0} \leftarrow A_7$	1	3						0	0	0	0	0	1	0	0
Bit Manipulation																	
MOV1	CY, saddr.bit	$CY \leftarrow (saddr.bit)$	3	6					X	0	0	0	0	1	0	0	0
										0	0	0	0	0	B_2	B_1	B_0
										Saddr-offset							
	CY, sfr.bit	$CY \leftarrow sfr.bit$	3	9					X	0	0	0	0	1	0	0	0
										0	0	0	0	1	B_2	B_1	B_0
										Sfr-offset							

Instruction Set (cont)

					Flags					Operation Code							
Mnemonic	**Operand**	**Operation**	**Bytes**	**States**	**S**	**Z**	**AC**	**P/V**	**CY**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Bit Manipulation (cont)																	
MOV1 (cont)	CY, A.bit	CY ← A.bit	2	6					X	0	0	0	0	0	0	1	1
										0	0	0	0	1	B_2	B_1	B_0
	CY, X.bit	CY ← X.bit	2	6					X	0	0	0	0	0	0	1	1
										0	0	0	0	0	B_2	B_1	B_0
	CY, PSWH.bit	CY ← PSW$_H$.bit	2	6					X	0	0	0	0	0	0	1	0
										0	0	0	0	1	B_2	B_1	B_0
	CY, PSWL.bit	CY ← PSW$_L$.bit	2	6					X	0	0	0	0	0	0	1	0
										0	0	0	0	0	B_2	B_1	B_0
	saddr.bit, CY	(saddr.bit) ← CY	3	5						0	0	0	0	1	0	0	0
										0	0	0	1	0	B_2	B_1	B_0
										Saddr-offset							
	sfr.bit, CY	sfr.bit ← CY	3	8						0	0	0	0	1	0	0	0
										0	0	0	1	1	B_2	B_1	B_0
										Sfr-offset							
	A.bit, CY	A.bit ← CY	2	7						0	0	0	0	0	0	1	1
										0	0	0	1	1	B_2	B_1	B_0
	X.bit, CY	X.bit ← CY	2	7						0	0	0	0	0	0	1	1
										0	0	0	1	0	B_2	B_1	B_0
	PSWH.bit, CY	PSW$_H$.bit ← CY	2	8						0	0	0	0	0	0	1	0
										0	0	0	1	1	B_2	B_1	B_0
	PSWL.bit, CY	PSW$_L$.bit ← CY	2	8	X	X	X	X		0	0	0	0	0	0	1	0
										0	0	0	1	0	B_2	B_1	B_0
AND1	CY, saddr.bit	CY ← CY AND (saddr.bit)	3	6					X	0	0	0	0	1	0	0	0
										0	0	1	0	0	B_2	B_1	B_0
										Saddr-offset							
	CY, /saddr.bit	CY ← CY AND ($\overline{\text{saddr.bit}}$)	3	6					X	0	0	0	0	1	0	0	0
										0	0	1	1	0	B_2	B_1	B_0
										Saddr-offset							
	CY, sfr.bit	CY ← CY AND sfr.bit	3	9					X	0	0	0	0	1	0	0	0
										0	0	1	0	1	B_2	B_1	B_0
										Sfr-offset							
	CY, /sfr.bit	CY ← CY AND $\overline{\text{sfr.bit}}$	3	9					X	0	0	0	0	1	0	0	0
										0	0	1	1	1	B_2	B_1	B_0
										Sfr-offset							
	CY, A.bit	CY ← CY AND A.bit	2	6					X	0	0	0	0	0	0	1	1
										0	0	1	0	1	B_2	B_1	B_0
	CY, /A.bit	CY ← CY AND $\overline{\text{A.bit}}$	2	6					X	0	0	0	0	0	0	1	1
										0	0	1	1	1	B_2	B_1	B_0
	CY, X.bit	CY ← CY AND X.bit	2	6					X	0	0	0	0	0	0	1	1
										0	0	1	0	0	B_2	B_1	B_0

Instruction Set (cont)

					Flags					Operation Code							
Mnemonic	**Operand**	**Operation**	**Bytes**	**States**	**S**	**Z**	**AC**	**P/V**	**CY**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Bit Manipulation (cont)																	
AND1 (cont)	CY, /X.bit	CY ← CY AND $\overline{\text{X.bit}}$	2	6					X	0	0	0	0	0	0	1	1
										0	0	1	1	0	B_2	B_1	B_0
	CY, PSWH.bit	CY ← CY AND PSW_H.bit	2	6					X	0	0	0	0	0	0	1	0
										0	0	1	0	1	B_2	B_1	B_0
	CY, /PSWH.bit	CY ← CY AND $\overline{\text{PSW}_H\text{.bit}}$	2	6					X	0	0	0	0	0	0	1	0
										0	0	1	1	1	B_2	B_1	B_0
	CY, PSWL.bit	CY ← CY AND PSW_L.bit	2	6					X	0	0	0	0	0	0	1	0
										0	0	1	0	0	B_2	B_1	B_0
	CY, /PSWL.bit	CY ← CY AND $\overline{\text{PSW}_L\text{.bit}}$	2	6					X	0	0	0	0	0	0	1	0
										0	0	1	1	0	B_2	B_1	B_0
OR1	CY, saddr.bit	CY ← CY OR (saddr.bit)	3	6					X	0	0	0	0	1	0	0	0
										0	1	0	0	0	B_2	B_1	B_0
										Saddr-offset							
	CY, /saddr.bit	CY ← CY OR ($\overline{\text{saddr.bit}}$)	3	6					X	0	0	0	0	1	0	0	0
										0	1	0	1	0	B_2	B_1	B_0
										Saddr-offset							
	CY, sfr.bit	CY ← CY OR sfr.bit	3	9					X	0	0	0	0	1	0	0	0
										0	1	0	0	1	B_2	B_1	B_0
										Sfr-offset							
	CY, /sfr.bit	CY ← CY OR $\overline{\text{sfr.bit}}$	3	9					X	0	0	0	0	1	0	0	0
										0	1	0	1	1	B_2	B_1	B_0
										Sfr-offset							
	CY, A.bit	CY ← CY OR A.bit	2	6					X	0	0	0	0	0	0	1	1
										0	1	0	0	1	B_2	B_1	B_0
	CY, /A.bit	CY ← CY OR $\overline{\text{A.bit}}$	2	6					X	0	0	0	0	0	0	1	1
										0	1	0	1	1	B_2	B_1	B_0
	CY, X.bit	CY ← CY OR X.bit	2	6					X	0	0	0	0	0	0	1	1
										0	1	0	0	0	B_2	B_1	B_0
	CY, /X.bit	CY ← CY OR $\overline{\text{X.bit}}$	2	6					X	0	0	0	0	0	0	1	1
										0	1	0	1	0	B_2	B_1	B_0
	CY, PSWH.bit	CY ← CY OR PSW_H.bit	2	6					X	0	0	0	0	0	0	1	0
										0	1	0	0	1	B_2	B_1	B_0
	CY, /PSWH.bit	CY ← CY OR $\overline{\text{PSW}_H\text{.bit}}$	2	6					X	0	0	0	0	0	0	1	0
										0	1	0	1	1	B_2	B_1	B_0
	CY, PSWL.bit	CY ← CY OR PSW_L.bit	2	6					X	0	0	0	0	0	0	1	0
										0	1	0	0	0	B_2	B_1	B_0
	CY, /PSWL.bit	CY ← CY OR $\overline{\text{PSW}_L\text{.bit}}$	2	6					X	0	0	0	0	0	0	1	0
										0	1	0	1	0	B_2	B_1	B_0

Instruction Set (cont)

					Flags					Operation Code							
Mnemonic	Operand	Operation	Bytes	States	S	Z	AC	P/V	CY	7	6	5	4	3	2	1	0
Bit Manipulation (cont)																	
XOR1	CY, saddr.bit	CY ← CY XOR (saddr.bit)	3	6					X	0	0	0	0	1	0	0	0
										0	1	1	0	0	B_2	B_1	B_0
										Saddr-offset							
	CY, sfr.bit	CY ← CY XOR sfr.bit	3	9					X	0	0	0	0	1	0	0	0
										0	1	1	0	1	B_2	B_1	B_0
										Sfr-offset							
	CY, A.bit	CY ← CY XOR A.bit	2	6					X	0	0	0	0	0	0	1	1
										0	1	1	0	1	B_2	B_1	B_0
	CY, X.bit	CY ← CY XOR X.bit	2	6					X	0	0	0	0	0	0	1	1
										0	1	1	0	0	B_2	B_1	B_0
	CY, PSWH.bit	CY ← CY XOR PSW_H.bit	2	6					X	0	0	0	0	0	0	1	0
										0	1	1	0	1	B_2	B_1	B_0
	CY, PSWL.bit	CY ← CY XOR PSW_L.bit	2	6					X	0	0	0	0	0	0	1	0
										0	1	1	0	0	B_2	B_1	B_0
SET1	saddr.bit	(saddr.bit) ← 1	2	4						1	0	1	1	0	B_2	B_1	B_0
										Saddr-offset							
	sfr.bit	sfr.bit ← 1	3	11						0	0	0	0	1	0	0	0
										1	0	0	0	1	B_2	B_1	B_0
										Sfr-offset							
	A.bit	A.bit ← 1	2	6						0	0	0	0	0	0	1	1
										1	0	0	0	1	B_2	B_1	B_0
	X.bit	X.bit ← 1	2	6						0	0	0	0	0	0	1	1
										1	0	0	0	0	B_2	B_1	B_0
	PSWH.bit	PSW_H.bit ← 1	2	7						0	0	0	0	0	0	1	0
										1	0	0	0	1	B_2	B_1	B_0
	PSWL.bit	PSW_L.bit ← 1	2	7	X	X	X	X	X	0	0	0	0	0	0	1	0
										1	0	0	0	0	B_2	B_1	B_0
CLR1	saddr.bit	(saddr.bit) ← 0	2	4						1	0	1	0	0	B_2	B_1	B_0
										Saddr-offset							
	sfr.bit	sfr.bit ← 0	3	11						0	0	0	0	1	0	0	0
										1	0	0	1	1	B_2	B_1	B_0
										Sfr-offset							
	A.bit	A.bit ← 0	2	6						0	0	0	0	0	0	1	1
										1	0	0	1	1	B_2	B_1	B_0
	X.bit	X.bit ← 0	2	6						0	0	0	0	0	0	1	1
										1	0	0	1	0	B_2	B_1	B_0
	PSWH.bit	PSW_H.bit ← 0	2	7						0	0	0	0	0	0	1	0
										1	0	0	1	1	B_2	B_1	B_0
	PSWL.bit	PSW_L.bit ← 0	2	7	X	X	X	X	X	0	0	0	0	0	0	1	0
										1	0	0	1	0	B_2	B_1	B_0

Instruction Set (cont)

					Flags					Operation Code							
Mnemonic	Operand	Operation	Bytes	States	S	Z	AC	P/V	CY	7	6	5	4	3	2	1	0
Bit Manipulation (cont)																	
NOT1	saddr.bit	(saddr.bit) ← $\overline{(\text{saddr.bit})}$	3	5						0	0	0	0	1	0	0	0
										0	1	1	1	0	B_2	B_1	B_0
										Saddr-offset							
	sfr.bit	sfr.bit ← $\overline{\text{sfr.bit}}$	3	11						0	0	0	0	1	0	0	0
										0	1	1	1	1	B_2	B_1	B_0
										Sfr-offset							
	A.bit	A.bit ← $\overline{\text{A.bit}}$	2	6						0	0	0	0	0	0	1	1
										0	1	1	1	1	B_2	B_1	B_0
	X.bit	X.bit ← $\overline{\text{X.bit}}$	2	6						0	0	0	0	0	0	1	1
										0	1	1	1	0	B_2	B_1	B_0
	PSWH.bit	PSW_H.bit ← $\overline{PSW_H\text{.bit}}$	2	7						0	0	0	0	0	0	1	0
										0	1	1	1	1	B_2	B_1	B_0
	PSWL.bit	PSW_L.bit ← $\overline{PSW_L\text{.bit}}$	2	7	X	X	X	X	X	0	0	0	0	0	0	1	0
										0	1	1	1	0	B_2	B_1	B_0
SET1	CY	CY ← 1	1	2					1	0	1	0	0	0	0	0	1
CLR1	CY	CY ← 0	1	2					0	0	1	0	0	0	0	0	0
NOT1	CY	CY ← $\overline{CY}$	1	2					X	0	1	0	0	0	0	1	0
Subroutine Linkage																	
CALL	!addr16	$(SP-1) \leftarrow (PC+3)_H$, $(SP-2) \leftarrow (PC+3)_L$, PC ← addr16, SP ← SP−2	3	6						0	0	1	0	1	0	0	0
										Low Addr							
										High Addr							
	rp1	$(SP-1) \leftarrow (PC+2)_H$, $(SP-2) \leftarrow (PC+2)_L$, $PC_H \leftarrow rp1_H$, $PC_L \leftarrow rp1_L$, SP ← SP−2	2	7						0	0	0	0	0	1	0	1
										0	1	0	1	1	Q_2	Q_1	Q_0
	[rp1]	$(SP-1) \leftarrow (PC+2)_H$, $(SP-2) \leftarrow (PC+2)_L$, $PC_H \leftarrow (rp1+1)$, $PC_L \leftarrow (rp1)$, SP ← SP−2	2	10						0	0	0	0	0	1	0	1
										0	1	1	1	1	Q_2	Q_1	Q_0
CALLF	!addr11	$(SP-1) \leftarrow (PC+2)_H$, $(SP-2) \leftarrow (PC+2)_L$, $PC_{15\text{-}11} \leftarrow 00001$, $PC_{10\text{-}0} \leftarrow$ addr11, SP ← SP−2	2	6						1	0	0	1	0	f_{10}	f_9	f_8
										f_7	f_6	f_5	f_4	f_3	f_2	f_1	f_0
CALLT	[addr5]	$(SP-1) \leftarrow (PC+1)_H$, $(SP-2) \leftarrow (PC+1)_L$, $PC_H \leftarrow$ (TPFx8000H + 2 × addr5 + 41H), $PC_L \leftarrow$ (TPFx8000H + 2 × addr5 + 40H), SP ← SP−2	1	9						1	1	1	t_4	t_3	t_2	t_1	t_0
BRK		$(SP-1) \leftarrow PSW_H$, $(SP-2) \leftarrow PSW_L$, $(SP-3) \leftarrow (PC+1)_H$, $(SP-4) \leftarrow (PC+1)_L$, $PC_L \leftarrow$ (003EH), $PC_H \leftarrow$ (003FH), SP ← SP−4, IE ← 0	1	12						0	1	0	1	1	1	1	0

Instruction Set (cont)

					Flags					Operation Code							
Mnemonic	**Operand**	**Operation**	**Bytes**	**States**	**S**	**Z**	**AC**	**P/V**	**CY**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Subroutine Linkage (cont)																	
RET		$PC_L \leftarrow (SP), PC_H \leftarrow (SP+1),$ $SP \leftarrow SP+2$	1	6						0	1	0	1	0	1	1	0
RETB		$PC_L \leftarrow (SP), PC_H \leftarrow (SP+1),$ $PSW_L \leftarrow (SP+2), PSW_H \leftarrow (SP+3),$ $SP \leftarrow SP+4$	1	10	R	R	R	R	R	0	1	0	1	1	1	1	1
RETI		$PC_L \leftarrow (SP), PC_H \leftarrow (SP+1),$ $PSW_L \leftarrow (SP+2), PSW_H \leftarrow (SP+3),$ $SP \leftarrow SP+4$	1	10	R	R	R	R	R	0	1	0	1	0	1	1	1
Stack Manipulation																	
PUSH	sfrp	$(SP-1) \leftarrow sfr_H, (SP-2) \leftarrow sfr_L,$ $SP \leftarrow SP-2$	3	9						0	0	0	0	0	1	1	1
										1	1	0	1	1	0	0	1
										Sfr-offset							
	post	$\{(SP-1) \leftarrow rpp_H, (SP-2) \leftarrow rpp_L,$ $SP \leftarrow SP-2\} \times n^*$	2	9-51**						0	0	1	1	0	1	0	1
										Post Byte							
	PSW	$(PS-1) \leftarrow PSW_H, (SP-2) \leftarrow PSW_L,$ $SP \leftarrow SP-2$	1	3						0	1	0	0	1	0	0	1
PUSHU	post	$\{(UP-1) \leftarrow rpp_H, (UP-2) \leftarrow rpp_L,$ $UP \leftarrow UP-2\} \times n^*$	2	10-52**						0	0	1	1	0	1	1	1
										Post Byte							
POP	sfrp	$sfr_L \leftarrow (SP), sfr_H \leftarrow (SP+1),$ $SP \leftarrow SP+2$	3	10						0	0	0	0	0	1	1	1
										1	1	0	1	1	0	0	0
										Sfr-offset							
	post	$\{rpp_L \leftarrow (SP), rpp_H \leftarrow (SP+1)$ $SP \leftarrow SP+2\} \times n^*$	2	13-62**						0	0	1	1	0	1	0	0
										Post Byte							
	PSW	$PSW_L \leftarrow (SP), PSW_H \leftarrow (SP+1)$ $SP \leftarrow SP+2$	1	5	R	R	R	R	R	0	1	0	0	1	0	0	0
POPU	post	$\{rpp_L \leftarrow (UP), rpp_H \leftarrow (UP+1),$ $UP \leftarrow UP+2\} \times n^*$	2	15-64**						0	0	1	1	0	1	1	0
										Post Byte							
MOVW	SP, #word	$SP \leftarrow word$	4	7						0	0	0	0	1	0	1	1
										1	1	1	1	1	1	0	0
										Low Byte							
										High Byte							
	SP, AX	$SP \leftarrow AX$	2	6						0	0	0	1	0	0	1	1
										1	1	1	1	1	1	0	0
	AX, SP	$AX \leftarrow SP$	2	6						0	0	0	1	0	0	0	1
										1	1	1	1	1	1	0	0
INCW	SP	$SP \leftarrow SP+1$	2	3						0	0	0	0	0	1	0	1
										1	1	0	0	1	0	0	0
DECW	SP	$SP \leftarrow SP-1$	2	3						0	0	0	0	0	1	0	1
										1	1	0	0	1	0	0	1

* rpp refers to register pairs specified in post byte. n is the number of register pairs specified in post byte.

** The details of the timing are described under "Timing of the PUSH and POP Instructions."

Instruction Set (cont)

					Flags					Operation Code							
Mnemonic	**Operand**	**Operation**	**Bytes**	**States**	**S**	**Z**	**AC**	**P/V**	**CY**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Pin Level Test																	
CHKL	sfr	(Pin level) XOR (internal signal level)	3	12	X	X		P		0	0	0	0	0	1	1	1
										1	1	0	0	1	0	0	0
										Sfr-offset							
CHKLA	sfr	A ← (Pin level) XOR (internal signal level)	3	12	X	X		P		0	0	0	0	0	1	1	1
										1	1	0	0	1	0	0	1
										Sfr-offset							
Unconditional Branch																	
BR	!addr16	PC ← addr16	3	4						0	0	1	0	1	1	0	0
										Low Addr							
										High Addr							
	rp1	PC_H ← $rp1_H$, PC_L ← $rp1_L$	2	4						0	0	0	0	0	1	0	1
										0	1	0	0	1	Q_2	Q_1	Q_0
	[rp1]	PC_H ← (rp1 + 1), PC_L ← (rp1)	2	8						0	0	0	0	0	1	0	1
										0	1	1	0	1	Q_2	Q_1	Q_0
	$addr16	PC ← addr16	2	4						0	0	0	1	0	1	0	0
										jdisp							
Conditional Branch																	
BC, BL	$addr16	PC ← addr16 if CY = 1	2	4						1	0	0	0	0	0	1	1
										jdisp							
BNC, BNL	$addr16	PC ← addr16 if CY = 0	2	4						1	0	0	0	0	0	1	0
										jdisp							
BZ, BE	$addr16	PC ← addr16 if Z = 1	2	4						1	0	0	0	0	0	0	1
										jdisp							
BNZ, BNE	$addr16	PC ← addr16 if Z = 0	2	4						1	0	0	0	0	0	0	0
										jdisp							
BV, BPE	$addr16	PC ← addr16 if P/V = 1	2	4						1	0	0	0	0	1	0	1
										jdisp							
BNV, BPO	$addr16	PC ← addr16 if P/V = 0	2	4						1	0	0	0	0	1	0	0
										jdisp							
BN	$addr16	PC ← addr16 if S = 1	2	4						1	0	0	0	0	1	1	1
										jdisp							
BP	$addr16	PC ← addr16 if S = 0	2	4						1	0	0	0	0	1	1	0
										jdisp							
BGT	$addr16	PC ← addr16 if (P/V XOR S) OR Z = 0	3	5						0	0	0	0	0	1	1	1
										1	1	1	1	1	0	1	1
										jdisp							
BGE	$addr16	PC ← addr16 if P/V XOR S = 0	3	5						0	0	0	0	0	1	1	1
										1	1	1	1	1	0	0	1
										jdisp							

Instruction Set (cont)

					Flags					Operation Code							
Mnemonic	Operand	Operation	Bytes	States	S	Z	AC	P/V	CY	7	6	5	4	3	2	1	0
Conditional Branch (cont)																	
BLT	\$addr16	PC ← addr16 if P/V XOR S = 1	3	5						0	0	0	0	0	1	1	1
										1	1	1	1	1	0	0	0
										jdisp							
BLE	\$addr16	PC ← addr16 if (P/V XOR S) OR Z = 1	3	5						0	0	0	0	0	1	1	1
										1	1	1	1	1	0	1	0
										jdisp							
BH	\$addr16	PC ← addr16 if Z OR CY = 0	3	5						0	0	0	0	0	1	1	1
										1	1	1	1	1	1	0	1
										jdisp							
BNH	\$addr16	PC ← addr16 if Z OR CY = 1	3	5						0	0	0	0	0	1	1	1
										1	1	1	1	1	1	0	0
										jdisp							
BT	saddr.bit, \$addr16	PC ← addr16 if (saddr.bit) = 1	3	7						0	1	1	1	0	B_2	B_1	B_0
										Saddr-offset							
										jdisp							
	sfr.bit, \$addr16	PC ← addr16 if sfr.bit = 1	4	8						0	0	0	0	1	0	0	0
										1	0	1	1	1	B_2	B_1	B_0
										Sfr-offset							
										jdisp							
	A.bit, \$addr16	PC ← addr16 if A.bit = 1	3	8						0	0	0	0	0	0	1	1
										1	0	1	1	1	B_2	B_1	B_0
										jdisp							
	X.bit, \$addr16	PC ← addr16 if X.bit = 1	3	8						0	0	0	0	0	0	1	1
										1	0	1	1	0	B_2	B_1	B_0
										jdisp							
	PSWH.bit, \$addr16	PC ← addr16 if PSW_H.bit = 1	3	8						0	0	0	0	0	0	1	0
										1	0	1	1	1	B_2	B_1	B_0
										jdisp							
	PSWL.bit, \$addr16	PC ← addr16 if PSW_L.bit = 1	3	8						0	0	0	0	0	0	1	0
										1	0	1	1	0	B_2	B_1	B_0
										jdisp							

Instruction Set (cont)

Mnemonic	Operand	Operation	Bytes	States	Flags					Operation Code							
					S	Z	AC	P/V	CY	7	6	5	4	3	2	1	0
Conditional Branch (cont)																	
BF	saddr.bit, $addr16	PC ← addr16 if (saddr.bit) = 0	4	7						0	0	0	0	1	0	0	0
										1	0	1	0	0	B_2	B_1	B_0
										Saddr-offset							
										jdisp							
	sfr.bit, $addr16	PC ← addr16 if sfr.bit = 0	4	8						0	0	0	0	1	0	0	0
										1	0	1	0	1	B_2	B_1	B_0
										Sfr-offset							
										jdisp							
	A.bit, $addr16	PC ← addr16 if A.bit = 0	3	8						0	0	0	0	0	0	1	1
										1	0	1	0	1	B_2	B_1	B_0
										jdisp							
	X.bit, $addr16	PC ← addr16 if X.bit = 0	3	8						0	0	0	0	0	0	1	1
										1	0	1	0	0	B_2	B_1	B_0
										jdisp							
	PSWH.bit, $addr16	PC ← addr16 if PSW_H.bit = 0	3	8						0	0	0	0	0	0	1	0
										1	0	1	0	1	B_2	B_1	B_0
										jdisp							
	PSWL.bit, $addr16	PC ← addr16 if PSW_L.bit = 0	3	8						0	0	0	0	0	0	1	0
										1	0	1	0	0	B_2	B_1	B_0
										jdisp							
BTCLR	saddr.bit, $addr16	PC ← addr16 if (saddr.bit) = 1 then reset (saddr.bit)	4	8/10						0	0	0	0	1	0	0	0
										1	1	0	1	0	B_2	B_1	B_0
										Saddr-offset							
										jdisp							
	sfr.bit, $addr16	PC ← addr16 if sfr.bit = 1 then reset sfr.bit	4	8/10						0	0	0	0	1	0	0	0
										1	1	0	1	1	B_2	B_1	B_0
										Sfr-offset							
										jdisp							
	A.bit, $addr16	PC ← addr16 if A.bit = 1 then reset A.bit	3	8/10						0	0	0	0	0	0	1	1
										1	1	0	1	1	B_2	B_1	B_0
										jdisp							
	X.bit, $addr16	PC ← addr16 if X.bit = 1 then reset X.bit	3	8/10						0	0	0	0	0	0	1	1
										1	1	0	1	0	B_2	B_1	B_0
										jdisp							
	PSWH.bit, $addr16	PC ← addr16 if PSW_H.bit = 1 then reset PSW_H.bit	3	8/10						0	0	0	0	0	0	1	0
										1	1	0	1	1	B_2	B_1	B_0
										jdisp							

Instruction Set (cont)

Mnemonic	Operand	Operation	Bytes	States	Flags					Operation Code							
					S	Z	AC	P/V	CY	7	6	5	4	3	2	1	0
Conditional Branch (cont)																	
BTCLR	PSWL.bit,	PC ← addr16 if $\mathrm{PSW_L}$.bit = 1	3	8/10	X	X	X	X	X	0	0	0	0	0	0	1	0
(cont)	$addr16	then reset $\mathrm{PSW_L}$.bit								1	1	0	1	0	B_2	B_1	B_0
										jdisp							
BFSET	saddr.bit,	PC ← addr16 if (saddr.bit) = 0	4	8/10						0	0	0	0	1	0	0	0
	$addr16	then set (saddr.bit)								1	1	0	0	0	B_2	B_1	B_0
										Saddr-offset							
										jdisp							
	sfr.bit, $addr16	PC ← addr16 if sfr.bit = 0	4	8/10						0	0	0	0	1	0	0	0
		then set sfr.bit								1	1	0	0	1	B_2	B_1	B_0
										Sfr-offset							
										jdisp							
	A.bit, $addr16	PC ← addr16 if A.bit = 0	3	8/10						0	0	0	0	0	0	1	1
		then set A.bit								1	1	0	0	1	B_2	B_1	B_0
										jdisp							
	X.bit, $addr16	PC ← addr16 if X.bit = 0	3	8/10						0	0	0	0	0	0	1	1
		then set X.bit								1	1	0	0	0	B_2	B_1	B_0
										jdisp							
	PSWH.bit,	PC ← addr16 if $\mathrm{PSW_H}$.bit = 0	3	8/10						0	0	0	0	0	0	1	0
	$addr16	then set $\mathrm{PSW_H}$.bit								1	1	0	0	1	B_2	B_1	B_0
										jdisp							
	PSWL.bit,	PC ← addr16 if $\mathrm{PSW_L}$.bit = 0	3	8/10	X	X	X	X	X	0	0	0	0	0	0	1	0
	$addr16	then set $\mathrm{PSW_L}$.bit								1	1	0	0	0	B_2	B_1	B_0
										jdisp							
DBNZ	r2, $addr16	r2 ← r2 − 1,	2	5/6						0	0	1	1	0	0	1	C_0
		then PC ← addr16 if r2 ≠ 0								jdisp							
	saddr, $addr16	(saddr) ← (saddr) − 1,	3	6/7						0	0	1	1	1	0	1	1
		then PC ← addr16 if saddr ≠ 0								Saddr-offset							
										jdisp							
Context Switching																	
BRKCS	RBn	$\mathrm{PC_H}$ ↔ R5, $\mathrm{PC_L}$ ↔ R4,	2	7						0	0	0	0	0	1	0	1
		R7 ← $\mathrm{PSW_H}$, R6 ← $\mathrm{PSW_L}$,								1	1	0	1	1	N_2	N_1	N_0
		$\mathrm{RBS_{2\text{-}0}}$ ← n, RSS ← 0, IE ← 0															
RETCS	!addr16	$\mathrm{PC_H}$ ← R5, $\mathrm{PC_L}$ ← R4,	3	5	R	R	R	R	R	0	0	1	0	1	0	0	1
		R5, R4 ← addr16, $\mathrm{PSW_H}$ ← R7								Low Addr							
		$\mathrm{PSW_L}$ ← R6 (priority change)								High Addr							
RETCSB	!addr16	$\mathrm{PC_H}$ ← R5, $\mathrm{PC_L}$ ← R4,	4	5	R	R	R	R	R	0	0	0	0	1	0	0	1
		R5, R4 ← addr16, $\mathrm{PSW_H}$ ← R7								1	1	1	0	0	0	0	0
		$\mathrm{PSW_L}$ ← R6 (no priority change)								Low Addr							
										High Addr							

Instruction Set (cont)

							Flags						Operation Code				
Mnemonic	Operand	Operation	Bytes	States	S	Z	AC	P/V	CY	7	6	5	4	3	2	1	0
String Manipulation																	
MOVM	[DE+], A	(DE+) ← A, C ← C–1	2	3+6n						0	0	0	1	0	1	0	1
		End if C = 0								0	0	0	0	0	0	0	0
	[DE–], A	(DE–) ← A, C ← C–1	2	3+6n						0	0	0	1	0	1	0	1
		End if C = 0								0	0	0	1	0	0	0	0
MOVBK	[DE+], [HL+]	(DE+) ← (HL+), C ← C–1	2	3+9n						0	0	0	1	0	1	0	1
		End if C = 0								0	0	1	0	0	0	0	0
	[DE–], [HL–]	(DE–) ← (HL–), C ← C–1	2	3+9n						0	0	0	1	0	1	0	1
		End if C = 0								0	0	1	1	0	0	0	0
XCHM	[DE+], A	(DE+) ↔ A, C ← C–1	2	3+10n						0	0	0	1	0	1	0	1
		End if C = 0								0	0	0	0	0	0	0	1
	[DE–], A	(DE–) ↔ A, C ← C–1	2	3+10n						0	0	0	1	0	1	0	1
		End if C = 0								0	0	0	1	0	0	0	1
XCHBK	[DE+], [HL+]	(DE+) ↔ (HL+), C ← C–1	2	3+16n						0	0	0	1	0	1	0	1
		End if C = 0								0	0	1	0	0	0	0	1
	[DE–], [HL–]	(DE–) ↔ (HL–), C ← C–1	2	3+16n						0	0	0	1	0	1	0	1
		End if C = 0								0	0	1	1	0	0	0	1
CMPME	[DE+], A	(DE+) – A, C ← C–1	2	3+10n	X	X	X	V	X	0	0	0	1	0	1	0	1
		End if C = 0 or Z = 0								0	0	0	0	0	1	0	0
	[DE–], A	(DE–) – A, C ← C–1	2	3+10n	X	X	X	V	X	0	0	0	1	0	1	0	1
		End if C = 0 or Z = 0								0	0	0	1	0	1	0	0
CMPBKE	[DE+], [HL+}	(DE+) – (HL+), C ← C–1	2	3+13n	X	X	X	V	X	0	0	0	1	0	1	0	1
		End if C = 0 or Z = 0								0	0	1	0	0	1	0	0
	[DE–], [HL–]	(DE–) – (HL–), C ← C–1	2	3+13n	X	X	X	V	X	0	0	0	1	0	1	0	1
		End if C = 0 or Z = 0								0	0	1	1	0	1	0	0
CMPMNE	[DE+], A	(DE+) – A, C ← C–1	2	3+10n	X	X	X	V	X	0	0	0	1	0	1	0	1
		End if C = 0 or Z = 1								0	0	0	0	0	1	0	1
	[DE–], A	(DE–) – A, C ← C–1	2	3+10n	X	X	X	V	X	0	0	0	1	0	1	0	1
		End if C = 0 or Z = 1								0	0	0	1	0	1	0	1
CMPBKNE	[DE+], [HL+]	(DE+) – (HL+), C ← C–1	2	3+13n	X	X	X	V	X	0	0	0	1	0	1	0	1
		End if C = 0 or Z = 1								0	0	1	0	0	1	0	1
	[DE–], [HL–]	(DE–) – (HL–), C ← C–1	2	3+13n	X	X	X	V	X	0	0	0	1	0	1	0	1
		End if C = 0 or Z = 1								0	0	1	1	0	1	0	1
CMPMC	[DE+], A	(DE+) – A, C ← C–1	2	3+10n	X	X	X	V	X	0	0	0	1	0	1	0	1
		End if C = 0 or CY = 0								0	0	0	0	0	1	1	1
	[DE–], A	(DE–) – A, C ← C–1	2	3+10n	X	X	X	V	X	0	0	0	1	0	1	0	1
		End if C = 0 or CY = 0								0	0	0	1	0	1	1	1
CMPBKC	[DE+], [HL+]	(DE+) – (HL+), C ← C–1	2	3+13n	X	X	X	V	X	0	0	0	1	0	1	0	1
		End if C = 0 or CY = 0								0	0	1	0	0	1	1	1
	[DE–], [HL–]	(DE–) – (HL–), C ← C–1	2	3+13n	X	X	X	V	X	0	0	0	1	0	1	0	1
		End if C = 0 or CY = 0								0	0	1	1	0	1	1	1

Instruction Set (cont)

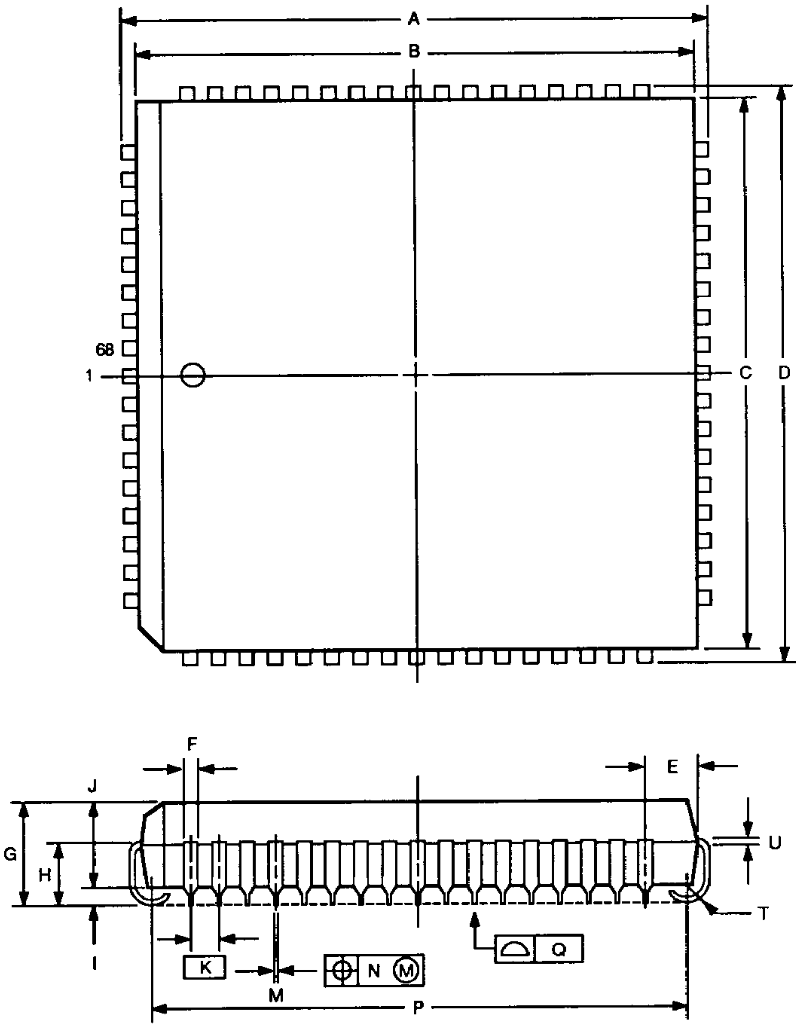
					Flags					Operation Code							
Mnemonic	Operand	Operation	Bytes	States	S	Z	AC	P/V	CY	7	6	5	4	3	2	1	0
String Manipulation (cont)																	
CMPMNC	[DE+], A	(DE+) − A, C ← C − 1	2	3+10n	X	X	X	V	X	0	0	0	1	0	1	0	1
		End if C = 0 or CY = 1								0	0	0	0	0	1	1	0
	[DE−], A	(DE−) − A, C ← C − 1	2	3+10n	X	X	X	V	X	0	0	0	1	0	1	0	1
		End if C = 0 or CY = 1								0	0	0	1	0	1	1	0
CMPBKNC	[DE+], [HL+]	(DE+) − (HL+), C ← C − 1	2	3+13n	X	X	X	V	X	0	0	0	1	0	1	0	1
		End if C = 0 or CY = 1								0	0	1	0	0	1	1	0
	[DE−], [HL−]	(DE−) − (HL−), C ← C − 1	2	3+13n	X	X	X	V	X	0	0	0	1	0	1	0	1
		End if C = 0 or CY = 1								0	0	1	1	0	1	1	0
CPU Control																	
MOV	STBC, #byte	STBC ← byte*	4	11						0	0	0	0	1	0	0	1
										1	1	0	0	0	0	0	0
										$\overline{\text{Data}}$							
										Data							
	WDM, #byte	WDM ← byte*	4	11						0	0	0	0	1	0	0	1
										1	1	0	0	0	0	1	0
										$\overline{\text{Data}}$							
										Data							
SWRS		RSS ← $\overline{\text{RSS}}$	1	2						0	1	0	0	0	0	1	1
SEL	RBn	RSS ← 0, RBS_{2-0} ← n	2	3						0	0	0	0	0	1	0	1
										1	0	1	0	1	N_2	N_1	N_0
	RBn, ALT	RSS ← 1, RBS_{2-0} ← n	2	3						0	0	0	0	0	1	0	1
										1	0	1	1	1	N_2	N_1	N_0
NOP		No Operation	1	2						0	0	0	0	0	0	0	0
EI		IE ← 1 (Enable Interrupt)	1	3						0	1	0	0	1	0	1	1
DI		IE ← 0 (Disable Interrupt)	1	3						0	1	0	0	1	0	1	0

* Trap if data bytes are not ones complement.
If trap, then: (SP−1) ← PSW_H,
(SP−2) ← PSW_L, (SP−3) ← $(PC-4)_H$, (SP−4) ← $(PC-4)_L$,
PC_L ← (003CH), PC_H ← (003DH),
SP ← SP−4, IE ← 0.

Package Drawings

68-Pin PLCC

Item	Millimeters	Inches
A	25.2 ±0.2	.992 ±.008
B	24.20	.953
C	24.20	.953
D	25.2 ±0.2	.992 ±.008
E	1.94 ±0.15	$.076^{+.007}_{-.006}$
F	0.6	.024
G	4.4 ±0.2	$.173^{+.009}_{-.008}$
H	2.8 ±0.2	$.110^{+.009}_{-.008}$
I	0.9 min	.035 min
J	3.4	.134
K	1.27 (TP)	.050 (TP)
M	0.40 ±0.10	$.016^{+.004}_{-.005}$
N	0.12	.005
P	23.12 ±0.20	$.910^{+.009}_{-.008}$
Q	0.15	.006
T	0.8 radius	.031 radius
U	$0.20^{+0.10}_{-0.05}$	$.008^{+.004}_{-.002}$



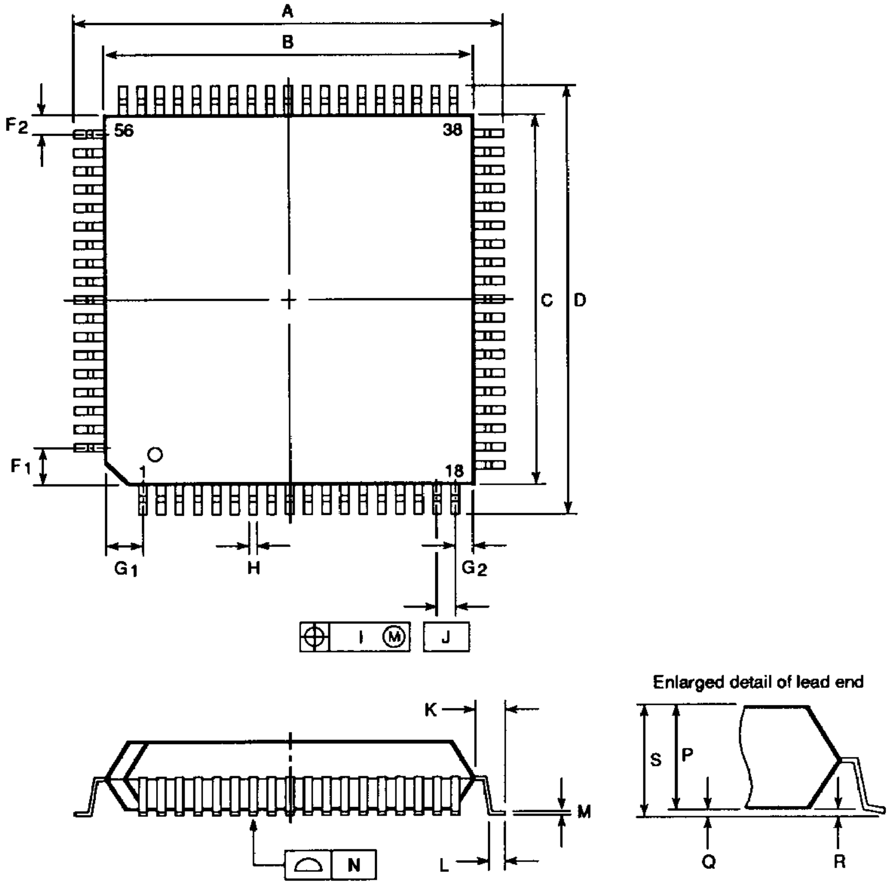
P68L-50A1-1

(2/90)
83YL-5561B

Package Drawings (cont)

74-Pin Plastic QFP

Item	Millimeters	Inches
A	23.2 ±0.4	$.913^{+.017}_{-.016}$
B	20.0 ±0.2	$.787^{+.009}_{-.008}$
C	20.0 ±0.2	$.787^{+.009}_{-.008}$
D	23.2 ±0.4	$.913^{+.017}_{-.016}$
F_1	2.0	.079
F_2	1.0	.039
G_1	2.0	.079
G_2	1.0	.039
H	0.40 ±0.10	$.016^{+.004}_{-.005}$
I	0.20	.008
J	1.0 (TP)	.039 (TP)
K	1.6 ±0.2	.063 ±.002
L	0.8 ±0.2	$.031^{+.009}_{-.008}$
M	$0.15^{+0.10}_{-0.05}$	$.006^{+.004}_{-.005}$
N	0.15	.006
P	3.7	.146
Q	0.1 ±0.1	.004 ±.004
R	0.1 ±0.1	.004 ±.004
S	4.0 max	.158 max



S74GJ-100-5BJ-1

49NR-347B (2/90)

NEC

NEC Electronics Inc.
CORPORATE HEADQUARTERS

401 Ellis Street
P.O. Box 7241
Mountain View, CA 94039
TEL 415-960-6000
TLX 3715792

For literature, call toll-free 8 a.m. to 4 p.m. Pacific time:
1-800-632-3531

No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Electronics Inc. The information in this document is subject to change without notice. Devices sold by NEC Electronics Inc. are covered by the warranty and patent indemnification provisions appearing in NEC Electronics Inc. Terms and Conditions of Sale only. NEC Electronics Inc. makes no warranty, express, statutory, implied, or by description, regarding the information set forth herein or regarding the freedom of the described devices from patent infringement. NEC Electronics Inc. makes no warranty of merchantability or fitness for any purpose. NEC Electronics Inc. assumes no responsibility for any errors that may appear in this document. NEC Electronics Inc. makes no commitment to update or to keep current information contained in this document.