STANDARD MICROSYSTEMS CORPORATION

MPU 800
MPU 800-1
MPU 800-4

PRELIMINARY

High-Performance Low-Power Microprocessor

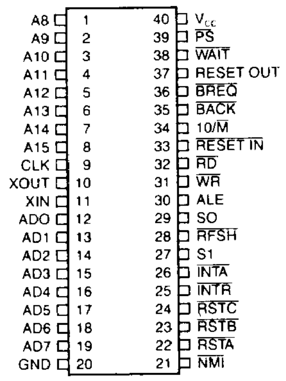
FEATURES

- Variable Power Supply: 2.4V - 6.0V
- Fully Compatible Wth Z80° Instruction Set
- Pin-Compatible With NSC800
- Powerful Set of 158 Instructions
- 10 Addressing Modes
- 22 Internal Registers
- Low Power: 50 mW at 5 V Vcc
- Multiplexed Bus Structure
- On Chip Bus Controller and Clock Generator
- On-Chip 8 bit Dynamic RAM Refresh Circuitry
- Three Speed Versions:

MPU800-4	4 MHz
MPU800	2.5 MHz
MPU800-1	1 MHz

- Capable of addressing 64 k bytes of memory, and 256 I/O devices
- Five interrupt request lines on-chip
- Schmitt trigger input on reset
- Power-Save Feature

PIN CONFIGURATION



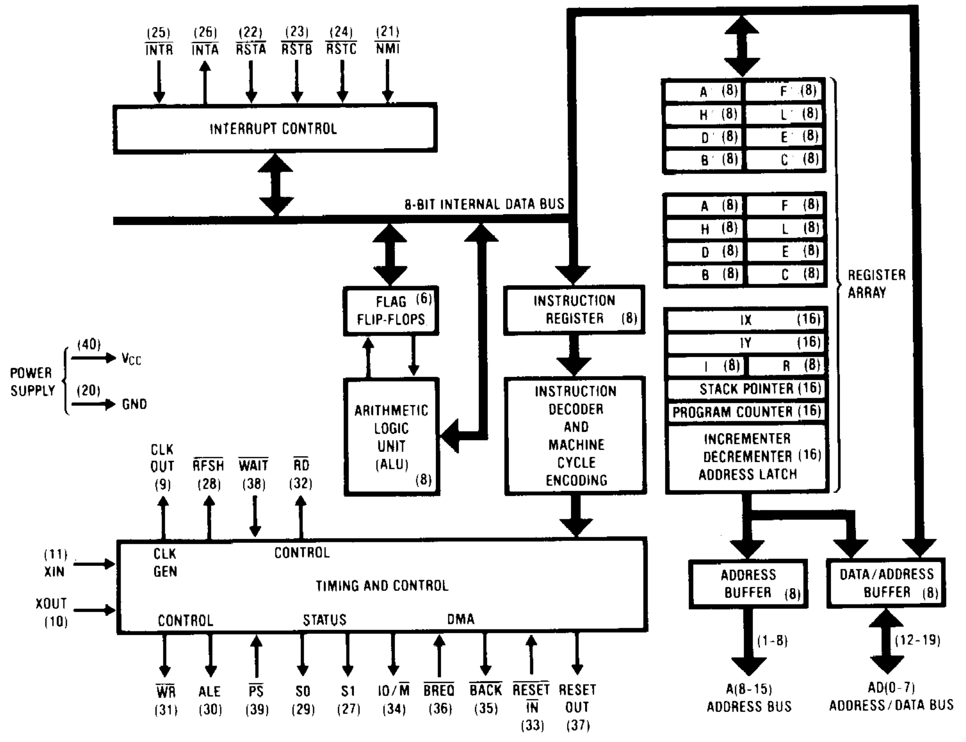
GENERAL DESCRIPTION

The MPU800 is an 8 bit microprocessor that functions as the central processing unit (CPU) in Standard Microsystems MPU800 microcomputer family. The device is fabricated in double-poly CMOS to combine high performance with the low-power of CMOS.

Many system functions are incorporated on the device including: vectored priority interrupts, refresh control, power save, and interrupt acknowledge.

Dedicated peripherals (MPU810 Ram I/O Timer, MPU830 ROM I/O Timer, and (MPU831 I/O Timer) have on-chip logic for direct interface to the MPU800.

Z80 is a registered trademark of Zilog Corporation.



CPU Functional Block Diagram

DESCRIPTION OF PIN FUNCTIONS

PIN NO	NAME	SYMBOL	FUNCTION
40	+5 Volt	Vcc	+5 volt supply
20	Ground	GND	Ground
10	Crystal Out	XOUT	Crystal connection
11	Crystal In	XIN	Crystal connection; XIN may be used as an external clock input
Input/Output Signals			
12-19	Address/Data	AD0-AD7	Multiplexed Address/Data. Active high At $\overline{\text{RD}}$ Time: Input data to CPU. At $\overline{\text{WR}}$ Time: Output data from CPU. At Falling Edge of ALE Time: Least significant byte of address during memory reference cycle. 8-bit port address during I/O reference cycle. During $\overline{\text{BREQ}}$/$\overline{\text{BACK}}$ Cycle: High impedance.
Input Signals			
33	Reset In	$\overline{\text{RESET IN}}$	Active low. Sets A (8-15) and AD (0-7) to TRI-STATE® (high impedance). Clears the contents of PC, I and R registers, disables interrupts, and causes a reset output to be activated.
36	Bus Request	$\overline{\text{BREQ}}$	Active low. Used when another device is requesting the system bus. $\overline{\text{BREQ}}$ is recognized at the end of the current machine cycle, then A(8-15), AD(0-7), IO/$\overline{\text{M}}$, $\overline{\text{RD}}$, and $\overline{\text{WR}}$ are set to the high impedance mode and the request is acknowledged via the $\overline{\text{BACK}}$ output signal.
21	Non-Maskable Interrupt	$\overline{\text{NMI}}$	Active low. The non-maskable interrupt, generated by the peripheral device(s), is the highest priority interrupt request line. The interrupt is edge sensitive and only a pulse is required to set an internal flip-flop which generates the internal interrupt request. Since the $\overline{\text{NMI}}$ flip-flop is monitored on the same clock edge as the other interrupts, it must also meet the minimum set-up time spec for the interrupt to be accepted in the current machine instruction. Once the interrupt is accepted the flip-flop is reset automatically, Its execution is independent of the interrupt enable flip-flop. $\overline{\text{NMI}}$ execution involves saving the PC on the stack and automatic branching to restart address X'0066 in memory.
22-4	Restart Interrupt A,B,C	$\overline{\text{RSTA}}$, $\overline{\text{RSTB}}$, $\overline{\text{RSTC}}$	Active low level sensitive. Restarts generated by the peripherals are recognized at the end of the current instruction if their respective interrupt enable bits and master enable bit are set. Execution is identical to $\overline{\text{NMI}}$ except interrupts are enabled for the following restart addresses: **Name** — **Restart Address (X')** $\overline{\text{NMI}}$ — 0066 $\overline{\text{RSTA}}$ — 003C $\overline{\text{RSTB}}$ — 0034 $\overline{\text{RSTC}}$ — 002C $\overline{\text{INTR}}$ (Mode 1) — 0038 The order of priority is fixed (highest first) as follows: 1) $\overline{\text{NMI}}$ 2) $\overline{\text{RSTA}}$ 3) $\overline{\text{RSTB}}$ 4) $\overline{\text{RSTC}}$ 5) $\overline{\text{INTR}}$
25	Interrupt Request	$\overline{\text{INTR}}$	Active low level sensitive. An interrupt request input generated by a peripheral device is recognized at the end of the current instruction provided that the interrupt enable and master interrupt enable bits are set. INTR is the lowest priority interrupt request input. Under program control, INTR can be executed in three distinct modes in conjunction with the INTA output.
38	Wait	$\overline{\text{WAIT}}$	Active low. When set low during $\overline{\text{RD}}$, $\overline{\text{WR}}$ or $\overline{\text{INTA}}$ machine cycles, the CPU extends its machine cycle in increments of t (wait) states. The wait machine cycle continues until the WAIT input returns high. The wait strobe input will be accepted only during machine cycles that have $\overline{\text{RD}}$, $\overline{\text{WR}}$ or $\overline{\text{INTA}}$ strobes and during the machine cycle immediately after an interrupt has been accepted by the CPU. The later cycle has its RD strobe suppressed but it will still accept the wait.
39	Power Save	$\overline{\text{PS}}$	Active low. $\overline{\text{PS}}$ is sampled at the end of the current instruction cycle. When PS is low, the CPU stops executing at the end of current instruction and keeps itself in the low-power mode. Normal operation resumes when PS is returned high.

TRI-STATE® is a registered trademark of National Semiconductor Corporation.

PIN NO	NAME	SYMBOL	FUNCTION
Output Signals			
35	Bus Acknowledge	$\overline{\text{BACK}}$	Active low. $\overline{\text{BACK}}$ indicates to the bus requesting device that the CPU bus and its control signals are in the TRI-STATE mode. The requesting device may then take control of the bus and its control signals.
1-8	Address Bits 8-15	A8-A15	Active high. These are the most significant 8 bits of the memory address during a memory instruction. During an I/O instruction, the port address on the lower 8 bits of address get duplicated onto these 8 bits. During a $\overline{\text{BREQ}}/\overline{\text{BACK}}$ cycle, the A (8-15) bus is in the TRI-STATE mode.
37	Reset Out	RESET OUT	Active high. When RESET OUT is high, it indicates the CPU is being reset. The signal is normally used to reset the peripheral devices.
34	Input/Output/Memory	IO/$\overline{\text{M}}$	An active high on the IO/$\overline{\text{M}}$ output signifies that the current machine cycle is relative to an input/output device. An active low on the IO/$\overline{\text{M}}$ output signifies that the current machine cycle is relative to memory. It is TRI-STATE during $\overline{\text{BREQ}}/\overline{\text{BACK}}$ cycles.
28	Refresh	$\overline{\text{RFSH}}$	Active low. The refresh output indicates that the dynamic RAM refresh cycle is in progress. $\overline{\text{RFSH}}$ goes low during T3 and T4 states of all M1 cycles. During the refresh cycle, AD(0-7) has the refresh address and A(8-15) indicates the interrupt vector register I.
30	Address Latch Enable	ALE	ALE is active only during the T1 state of any M cycle and also T3 state of M1 cycles. The high to low transition of ALE indicates that a valid memory/I-0/refresh address is available on the AD(0-7) lines.
32	Read Strobe	$\overline{\text{RD}}$	Active low. On the trailing edge of the $\overline{\text{RD}}$ strobe, data is input to the CPU via the AD(0-7) lines. The $\overline{\text{RD}}$ line is in the TRI-STATE mode during $\overline{\text{BREQ}}/\overline{\text{BACK}}$ cycles.
31	Write Strobe	$\overline{\text{WR}}$	While the $\overline{\text{WR}}$ line is low, valid data is output by the CPU on the AD(0-7) lines. The $\overline{\text{WR}}$ line is in the TRI-STATE mode during $\overline{\text{BREQ}}/\overline{\text{BACK}}$ cycles.
9	Clock	$\overline{\text{CLK}}$	$\overline{\text{CLK}}$ is an output provided for use as a system clock. The $\overline{\text{CLK}}$ output is a square wave at one half the input frequency.
26	Interrupt Acknowledge	$\overline{\text{INTA}}$	Active low. The interrupt acknowledge output is activated in the M1 cycle (S) immediately following the t state in which the $\overline{\text{INTR}}$ input is recognized. [Output is normally used to gate the interrupt response vector from the peripheral controller onto the AD(0-7) lines.] It is used in two of the three interrupt modes. In mode 0, an instruction is gated onto the AD (0-7) line during $\overline{\text{INTA}}$. There will be from 1 to 4 $\overline{\text{INTA}}$ strobes issued for each mode 0 interrupt. The amount of $\overline{\text{INTA}}$ strobes issued is instruction dependent. In mode 2, a single interrupt response vector is gated onto the data bus. In mode 1, $\overline{\text{INTA}}$ is not used. In this mode, $\overline{\text{INTR}}$ functions like the restart interrupts.
29, 27	Status	SO, S1	Bus status outputs indicate encoded information regarding the ensuing M cycle as follows:

Machine Cycle	Status			Control	
	$\overline{\text{SO}}$	$\overline{\text{S1}}$	IO/$\overline{\text{M}}$	$\overline{\text{RD}}$	$\overline{\text{WR}}$
Opcode Fetch	1	1	0	0	1
Memory Read	0	1	0	0	1
Memory Write	1	0	0	1	0
I/O Read	0	1	1	0	1
I/0 Write	1	0	1	1	0
Halt*	0	0	0	0	1
Internal Operation*	0	1	0	1	1
Acknowledge of Int**	1	1	0	1	1

*ALE is not suppressed in this cycle.

**This is the cycle that occurs immediately after the CPU accepts an interrupt ($\overline{\text{RSTA}}$, $\overline{\text{RSTB}}$, $\overline{\text{RSTC}}$, $\overline{\text{INTR}}$, $\overline{\text{NMI}}$).

Note 1: During halt, CPU continues to do dummy opcode fetch from location following the halt instruction with a halt status. This is so CPU can continue to do its dynamic RAM refresh.

Note 2: No early status is provided for interrupt or hardware restarts.

TIMING CONTROL

All necessary timing signals are provided by a single state inverter oscillator contained on the MPU800 chip. The chip operation frequency is equal to one half of the frequency of this oscillator. The oscillator frequency can be controlled by one of the following methods:

1. Leaving the XOUT pin unterminated and driving the XIN pin with an externally generated clock as shown in *Figure 1a*. When driving XIN with a square wave, the minimum duty cycle is 30%-70%, either high or low.

2. Connecting a crystal with the proper biasing network between XIN and XOUT as shown in *Figure 1b*. Recommended crystal is a parallel resonance AT cut crystal.

Resistor capacitor feedback network described in earlier data sheets will not oscillate due to gain of internal inverter circuit. A modification of this circuit by adding two inverters in series between the RC network and XIN will work.

The CPU has a minimum clock frequency input (@ XIN) of 32 kHz, which results in 16 kHz system clock speed. All registers internal to the chip are static, however there is dynamic logic which limits the minimum clock speed. The input clock can be stopped without fear of losing any data or damaging the part. You stop it in the phase of the clock that has XIN low and CLK OUT high. When restarting the CPU, precautions must be taken so that the input clock meets minimum specification. Once started, the CPU will continue operation from the same location at which it was stopped. During DC operation of the CPU, typical current drain will be 2mA. This current drain can be reduced by placing the CPU in a wait state during an opcode fetch cycle then stopping the clock.

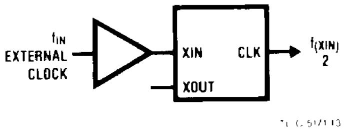
FUNCTIONAL DESCRIPTION

The MPU800 is an 8-bit general purpose microprocessor designed for stand-alone and DMA (direct memory access) applications. A minimum system can be constructed with an MPU800, and MPU810 (RAMI/O Timer) and an 27C16 (EPROM).

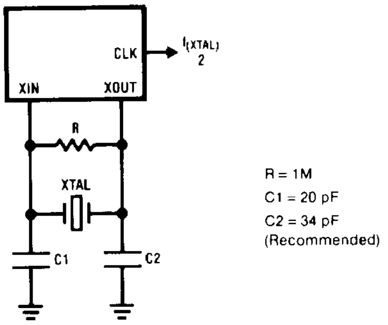
MPU800 uses a multiplexed bus for data and addresses. The 16-bit address bus is divided into a high-order 8-bit address bus that handles bits 8-15 of the address, and a low-order 8-bit mulitplexed address/data bus that handles bits 0-7 of the address and bits 0-7 of the data. Strobe outputs from the MPU800 (ALE, $\overline{RD}$ and $\overline{WR}$) indicate when a valid address or data is present on the bus. IO/$\overline{M}$ indicates whether the ensuing cycle accesses memory or I/O.

During an input or output instruction, the CPU duplicates the lower half of the address [AD(0-7)] onto the upper half [A(8-15)]. The eight bits of address will stay on A(8-15) for the entire machine cycle.

Figure 2 illustrates the timing relationship for opcode fetch cycles with and without a wait state. *Figure 3* illustrates the timing relationship for memory read and write cycles with and without a wait state. Input/output cycles with and without wait state are shown in *Figure 4*. One wait state is automatically inserted into each I/O instruction.



a.



b.

FIGURE 1. Timing Control Configurations

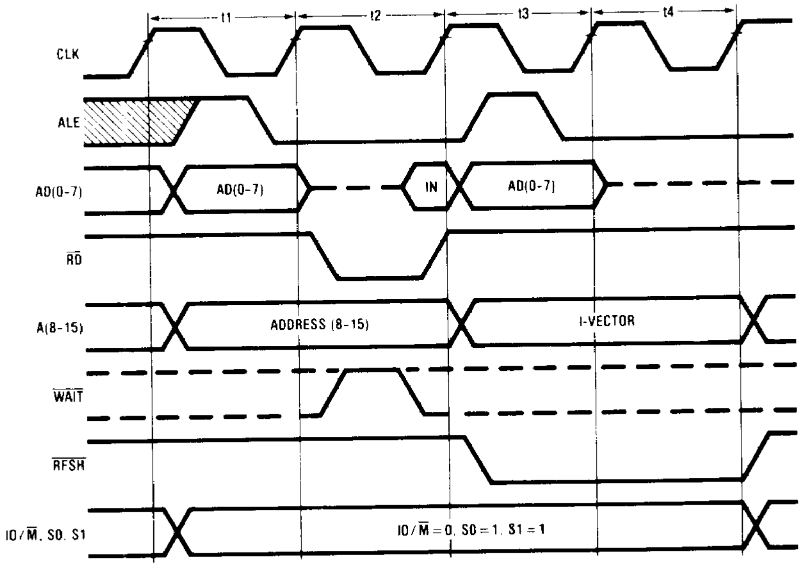


FIGURE 2a. Opcode Fetch Cycles without $\overline{WAIT}$ States

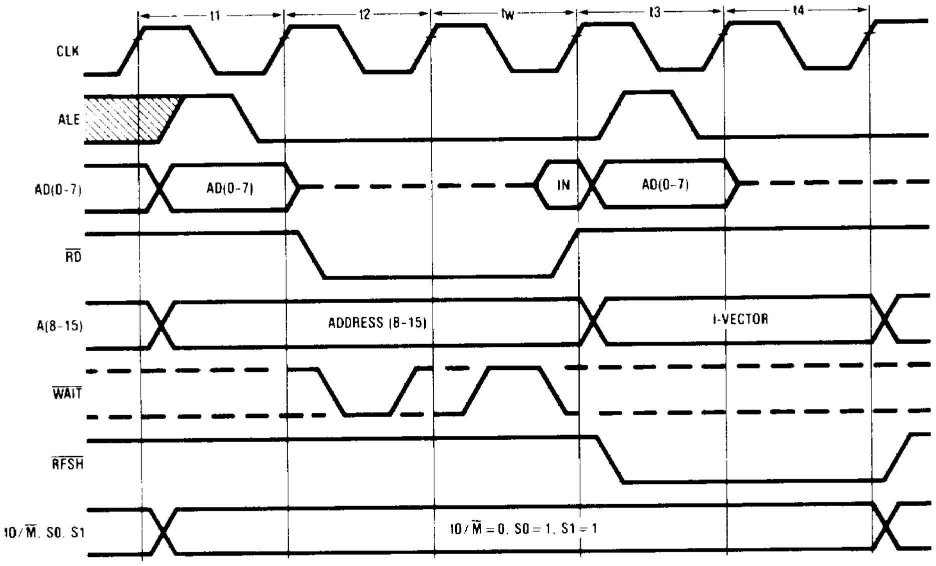


FIGURE 2b. Opcode Fetch Cycles with $\overline{WAIT}$ States

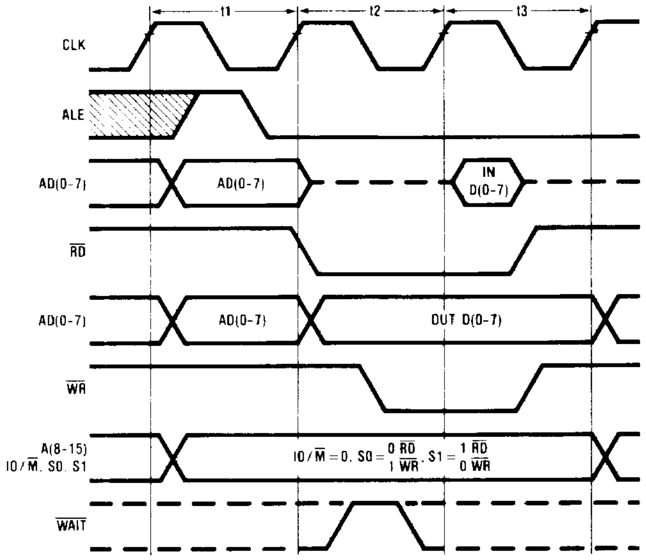


FIGURE 3a. Memory Read/Write Cycles without $\overline{\text{WAIT}}$ States

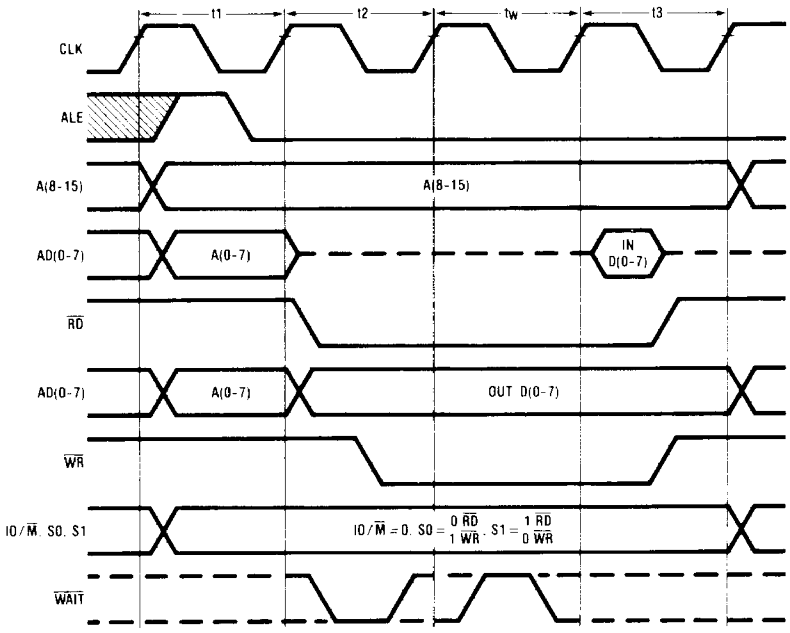


FIGURE 3b. Memory Read and Write with $\overline{\text{WAIT}}$ States

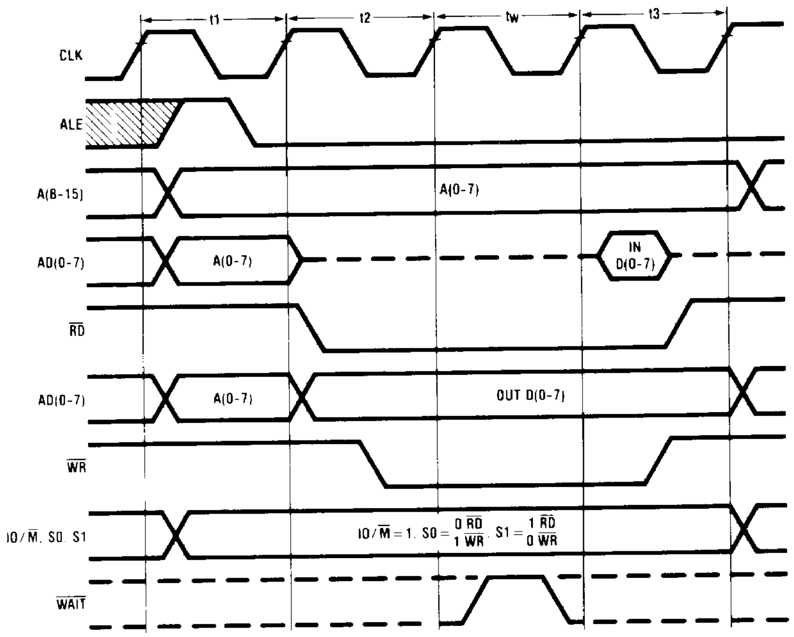
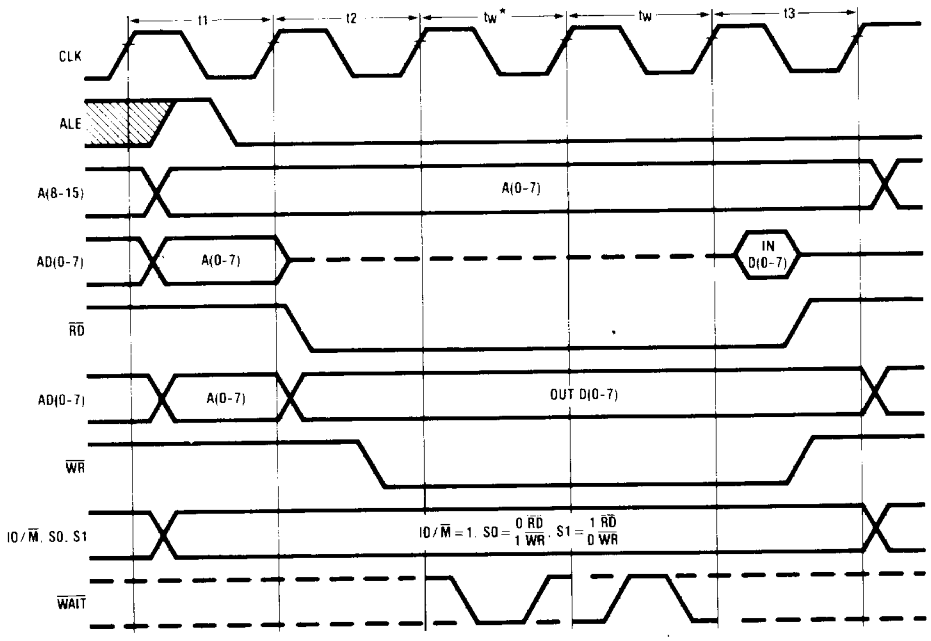


FIGURE 4a. Input and Output Cycles without $\overline{\text{WAIT}}$ States



* $\overline{\text{WAIT}}$ state automatically inserted during IO operation.

FIGURE 4b. Input and Output Cycles with $\overline{\text{WAIT}}$ States

INITIALIZATION

The MPU800 and its peripheral components are initialized by $\overline{\text{RESET IN}}$ and RESET OUT. $\overline{\text{RESET IN}}$ input is associated with an on-chip Schmitt trigger that facilitates using an R-C network power-on reset scheme *(Figure 5).*

To ensure proper power-up conditions for the NSC800, the following power-up and initialization procedure is recommended:

1. Apply power (V_{CC} and GND) and set $\overline{\text{RESET IN}}$ active (low). Allow sufficient time (approximately 100 ms if crystal used) for the oscillator and internal clocks to stabilize. $\overline{\text{RESET IN}}$ must remain low for at least 3t state (CLK) times. RESET OUT, following the clock stabilization period, responds by going high, indicating to the system that the MPU800 is being reset. RESET OUT signal becomes available to reset the peripherals.

2. Set $\overline{\text{RESET IN}}$ high, following which the RESET OUT goes low and the CPU initiates the first opcode fetch cycle.

NOTE: The MPU800 initialization includes: Clear PC to X'0000 (the first opcode fetch, therefore, is from memory location X'0000). Clear registers I (Interrupt Vector Base) and R (Refresh Counter) to X'00. Clear interrupt control register bits IEA, IEB and IEC. The interrupt control bit IEI is set to 1 to maintain INS8080A/Z80A compatibility (see INTERRUPTS for more details). Maskable interrupts are disabled and the CPU enters Interrupt Mode 0. While $\overline{\text{RESET IN}}$ is active (low), the A(8-15) and AD(0-7) lines go to high impedance (TRI-STATE) and all CPU strobes go to the inactive state.

BUS ACCESS CONTROL

Figure 6 illustrates bus access control in the MPU800. The external device controller produces an active $\overline{\text{BREQ}}$ signal that requests the bus. When the CPU responds with BACK then the bus and related control strobes go to high impedance (TRI-STATE). It should be noted that (1) $\overline{\text{BREQ}}$ is sampled at the last t state of any M machine cycle only. (2) the MPU800 will not acknowledge any interrupt/restart requests, and will not perform any dynamic RAM refresh functions until after $\overline{\text{BREQ}}$ input signal is inactive high. (3) $\overline{\text{BREQ}}$ signal has priority over all interrupt request signals, should $\overline{\text{BREQ}}$ and interrupt request become active simultaneously.

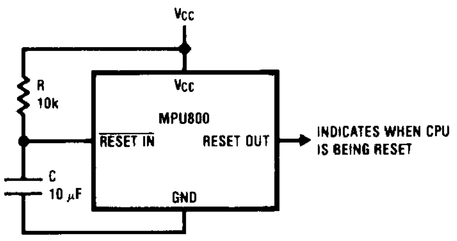
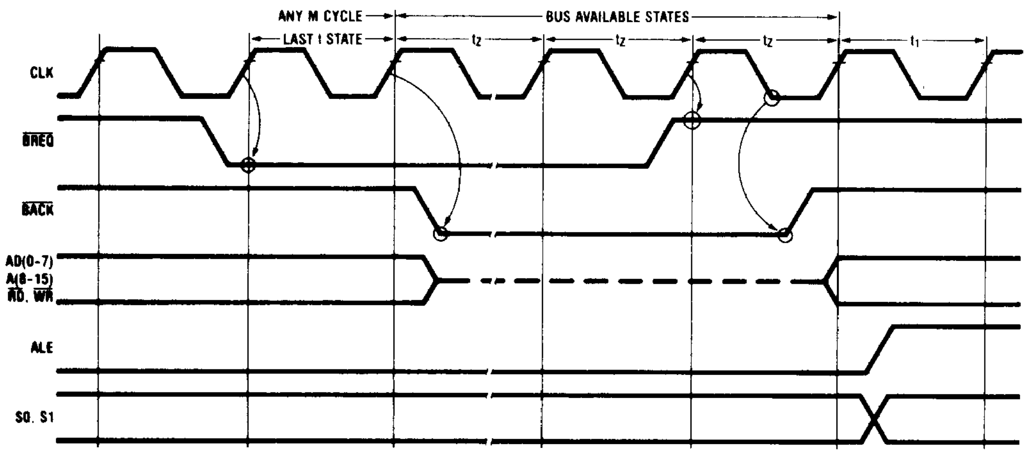


FIGURE 5. Power-On Reset



*S0, S1 during $\overline{\text{BREQ}}$ will indicate same machine cycle as during cycle when $\overline{\text{BREQ}}$ was accepted.
t_Z = time states bus and control signals are in high impedance mode.

FIGURE 6. Bus Acknowledge Cycle

REGISTER CONFIGURATION

The MPU800 contains 22 programmable registers as shown in *Figure 7*. The CPU working registers are arranged in two 8-register configurations, each of which includes an 8-bit accumulator, a flag register, and six general purpose 8-bit registers. Only one 8-bit register set may be active at any given moment. However, simple instructions exist that allow the programmer to exchange the active and alternate register sets.

It should also be noted that the six 8-bit general purpose registers (B, C, D, E, H, and L) can be accessed as 16-bit registers (BC, DE, and HL). The functions of these become apparent in the instruction set description.

CPU Main Working Register Set

Accumulator	(8)	Flags F	(8)
Register B	(8)	Register C	(8)
Register D	(8)	Register E	(8)
Register H	(8)	Register L	(8)

CPU Alternate Working Register Set

Accumulator A′	(8)	Flags F′	(8)
Register B′	(8)	Register C′	(8)
Register D′	(8)	Register E′	(8)
Register H′	(8)	Register L′	(8)

CPU Dedicated Registers

Index Register IX	(16)
Index Register IY	(16)
Interrupt Vector Register I	(8)
Memory Refresh Register R	(8)
Stack Pointer SP	(16)
Program Counter PC	(16)

FIGURE 7. Register Configuration

DEDICATED REGISTERS:

Program Counter (PC): The program counter contains the 16-bit address of the current instruction being fetched from memory. The PC is incremented after its contents have been transferred to the address lines. When a program jump occurs, the new address is placed in the PC, overriding the incrementer.

Stack Pointer (SP): The stack pointer contains the 16-bit address of the current top of a stack located in external system RAM memory. The external stack memory is organized as a last-in, first-out (LIFO) file. The stack allows simple implementation of multiple level interrupts, virtually unlimited subroutine nesting and simplification of many types of data manipulation.

Index Registers (IX and IY): The two 16-bit index registers hold a 16-bit base address used in indexed addressing modes. In this mode, an index register is used as a base to point to a region in memory from which data is to be stored or retrieved. An additional byte is included in indexed instructions to specify a displacement from this base. This displacement is specified as a two's complement signed integer.

Interrupt Page Address Register (I): The MPU800 CPU can indirectly call any memory location in response to a mode 2 interrupt. The I register is used to store the high-order 8 bits of the address. The low-order 8 bits are supplied by the interrupting peripheral. This feature allows interrupt routines to be dynamically located anywhere in memory with minimal access time to the routine.

Memory Refresh Register (R): The MPU800 CPU contains a memory refresh counter to enable dynamic memories to be used with the same ease as static memories. This 8-bit register is automatically incremented after each instruction fetch. The data in the refresh counter is sent out on the lower portion of the address bus along with a refresh control signal while the CPU is decoding and executing the fetched instruction. This mode of refresh is totally transparent to the programmer and does not slow down CPU operation. The programmer can load the R register for testing purposes, but this register is normally not used by the programmer.

ACCUMULATORS AND FLAG REGISTERS

The CPU includes two 8-bit accumulators and two associated 8-bit flag registers. The accumulator holds the results of 8-bit arithmetic or logical operation. The flag register indicates specific conditions for 8-bit or 16-bit operations.

FLAG REGISTERS (F, F')

The two MPU800 flag registers each contain six status bits that are set or reset (cleared) by various CPU operations *(Figure 8)*. Four of these bits (carry, zero, sign, and parity/overflow flags) can be tested by the programmer. The descriptions of the flags follow.

Carry Flag (C): This flag is set by the carry from the highest order bit of the accumulator during an add instruction or a borrow generated during a subtraction instruction. Specific shift and rotate instructions also affect this bit.

Zero Flag (Z): This flag is set when a zero is loaded into the accumulator as a result of an operation. Otherwise it remains clear.

Sign Flag (S): This flag stores the state of bit 7 (the sign bit) in the accumulator after an arithmetic operation. This flag is intended to be used with signed numbers.

Parity/Overflow Flag (P/V): During logical operations this flag is set when the parity of the result is even and reset when it is odd. It represents overflow when signed two's complement arithmetic operations are performed. An overflow occurs when the resultant of a two's complement operation (in the accumulator) is out of range.

The two non-testable flag register bits used for BCD arithmetic are:

Half Carry (H): The flag indicates a BCD carry or borrow result from the least significant four bits of an operation; when using the DAA (Decimal Adjust Accumulator Instruction), it is used to correct the result of a previously packed decimal add or subtract.

Add/Subtract Flag (N): Since the algorithm for correcting BCD operations is different for addition or subtraction, this flag specifies what type of instruction was executed last in order that the DAA operation will be correct for either operation.

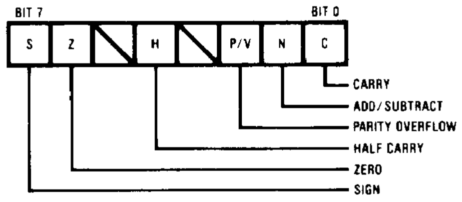


FIGURE 8. Flag Register

INTERRUPTS

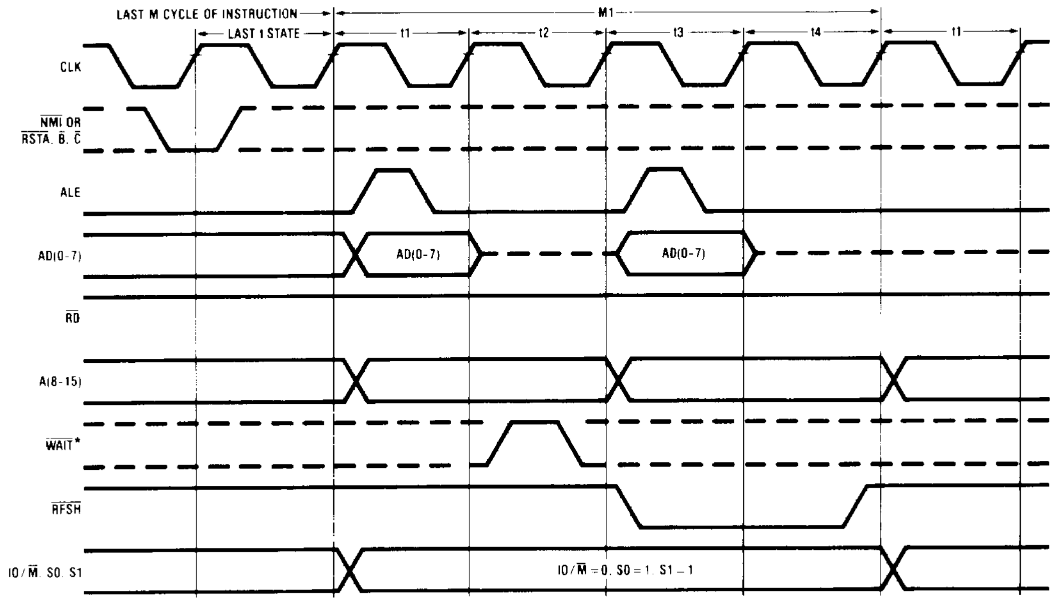
The MPU800 has five interrupt/restart inputs, four are maskable ($\overline{RSTA}$, $\overline{RSTB}$, $\overline{RSTC}$, and $\overline{INTR}$) and one is non-maskable ($\overline{NMI}$). $\overline{NMI}$, having the highest priority of all interrupts, is always serviced and cannot be disabled by the user. After recognizing an active input on $\overline{NMI}$, the CPU stops before the next instruction, pushes the PC onto the stack, and jumps to address X'0066, where the user's interrupt service routine is located (i.e., restart to memory location X'0066). $\overline{NMI}$ is intended for interrupts requiring immediate attention, such as power-down, control panel, etc. $\overline{RSTA}$, $\overline{RSTB}$ and $\overline{RSTC}$ are restart inputs, which, if enabled, execute a restart to memory location X'003C, X'0034, and X'002C, respectively. Note that the CPU response to the $\overline{NMI}$ and $\overline{RST}$ ($\overline{A}$, $\overline{B}$, $\overline{C}$) request input is basically identical. Unlike $\overline{NMI}$, however, restart request inputs must be enabled.

Figure 9 illustrates $\overline{NMI}$ and $\overline{RST}$ interrupt machine cycles. M1 cycle will be a dummy opcode fetch cycle followed by M2 and M3 which are stack push operations. The following instruction will then start from the interrupts restart location.

The MPU800 also provides one more general purpose interrupt request input, $\overline{INTR}$. When enabled, the CPU responds to $\overline{INTR}$ in one of the three modes defined by instruction IM0, IM1, and IM2 for modes 0, 1 and 2, respectively. Following reset, the CPU automatically sets itself in mode).

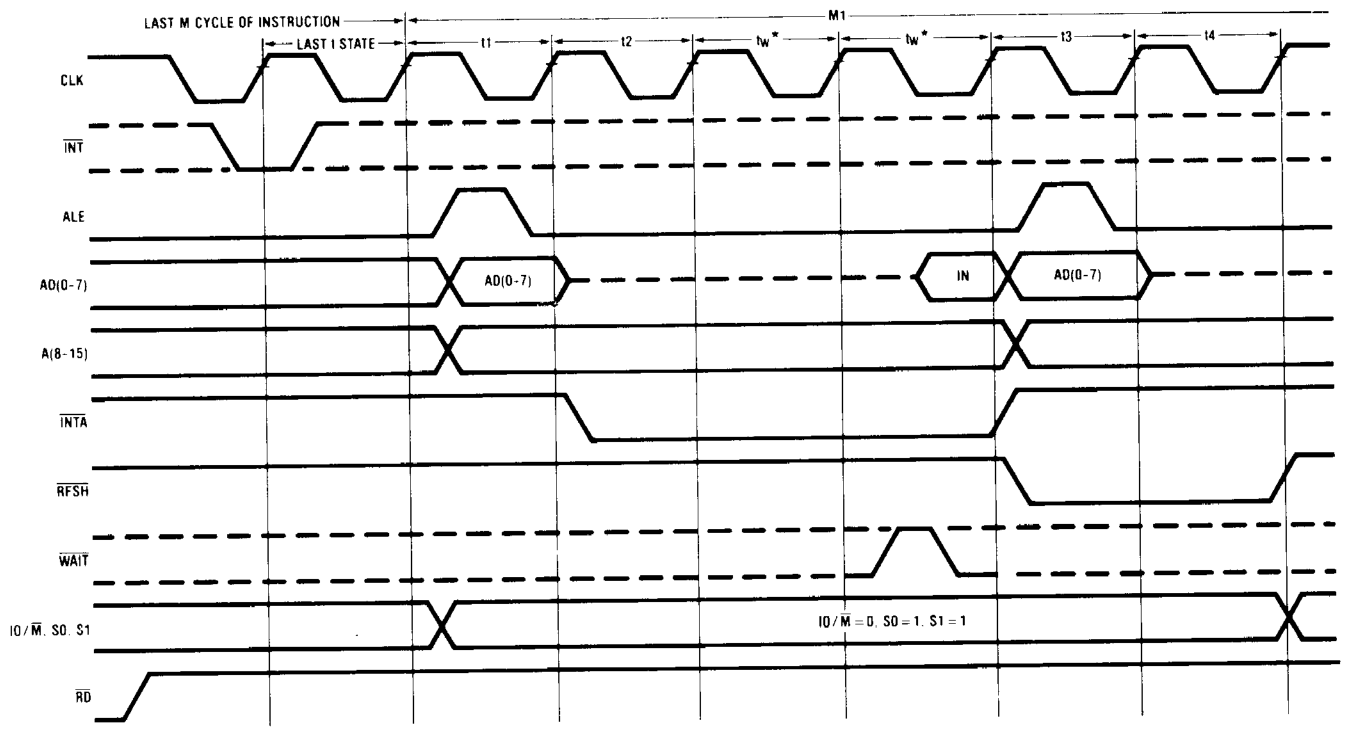
Interrupt (INTR) Mode 0; Similar to INS8080A mode. The CPU responds to an interrupt request by providing an INTA (interrupt acknowledge) strobe, which can be used to gate an instruction from a peripheral onto the data bus. Two wait states are automatically inserted by the CPU during the first $\overline{INTA}$ cycle to allow the interrupting device (or its controller) ample time to gate the instruction and determine external priorities. *(Figure 10).* This can be any instruction from one to four bytes. The most popular instruction would be a one-byte call (restart instruction) or a three-byte call (CALL NN instruction). If it is a three-byte call, the CPU issues a total of three $\overline{INTA}$ strobes. The last two read NN (which do not include wait states).

Interrupt ($\overline{INTR}$) Mode 1: Similar to the restart interrupts except the restart location is X'0038 *(Figure 9).*



*This is the only machine cycle that does not have an $\overline{RD}$, $\overline{WR}$, or $\overline{INTA}$ strobe but will accept a wait strobe.

FIGURE 9. Non-Maskable and Restart Interrupt Machine Guide



*t_W is the CPU generated WAIT state in response to an interrupt request.

FIGURE 10. Interrupt Acknowledge Machine Cycle

Interrupt ($\overline{\text{INTR}}$) Mode 2: With this mode, the programmer maintains a table that contains the 16-bit starting address of every interrupt service routine. This table may be located anywhere in memory. When the mode 2 interrupt is accepted *(Figure 11)*, a 16-bit pointer must be formed to obtain the desired interrupt service routine starting address from the table. The upper 8 bits of this pointer are from the contents of the I register, which has been previously loaded with the desired value by the programmer. The lower 8 bits of the pointer are supplied by the interrupting device with the low-order bit forced to zero. The pointer is used to get two adjacent bytes from the interrupt service routine starting address table to complete the 16-bit service routine starting address. The first byte of each entry in the table is the least significant (low-order) portion of the address. The programmer must obviously fill this table with the desired addresses before any interrupts are to be accepted.

Note that this table can be changed at any time to allow peripherals to be serviced by different service routines. Once the interrupting device supplies the lower portion of the pointer, the CPU automatically pushes the program counter onto the stack, obtains the starting address from the table and does a jump to this address.

The interrupts have fixed priorities built into the MPU800 as:

$\overline{\text{NMI}}$	(Highest Priority)
$\overline{\text{RSTA}}$	
$\overline{\text{RSTB}}$	
$\overline{\text{RSTC}}$	
INTR	(Lowest Priority)

ENABLING INTERRUPTS

NMI, being a non-maskable interrupt request, is executed as it occurs and can never be disabled.

The maskable interrupt inputs ($\overline{\text{RSTA}}$, $\overline{\text{RSTB}}$, $\overline{\text{RSTC}}$, and INTR) are enabled under program control through the use of the interrupt control register and enable/disable interrupt instruction.

The appropriate interrupt control bits in 4-bit control register (IEA, IEB, IEC, and IEI) must be enabled in conjunction with IFF1 and IFF2, before the maskable $\overline{\text{INTR}}$ and $\overline{\text{RST}}$ A, B, C can be accepted by the CPU.

The interrupt control register is an on-chip write only output port located at port address X'BB. It can only be written to by either the OUT (C), r or OUT (N), A instructions (for example OUTI instruction will not affect Interrupt Control Register). Its contents are:

Bit	Name	Function	
0	IEI	Interrupt Enable for	INTR
1	IEC	" " "	$\overline{\text{RSTC}}$
2	IEB	" " "	$\overline{\text{RSTB}}$
3	IEA	" " "	$\overline{\text{RSTA}}$

For example: In order to enable $\overline{\text{RSTB}}$, CPU interrupts must be enabled and IEB must be set.

At reset, IEI bit is set and other mask bits, IEA, IEB, IEC are cleared. This maintains the software compatibility between MPU800 and INS8080A (or Z80A).

Execution of an IO block move instruction will not affect the state of the interrupt control bits. The only two instructions that will modify this write only register are OUT (C), r and OUT (N), A.

POWER-SAVE FEATURE

The MPU800 provides a unique power-save mode by the means of the $\overline{\text{PS}}$ pin. $\overline{\text{PS}}$ input is sampled at the last t state of the last M cycle of an instruction. After recognizing an active (low) level on $\overline{\text{PS}}$, the **MPU800** stops its internal clocks, thereby reducing its power dissipation to one half of operating power, yet maintaining all register values and internal control status. The MPU800 keeps its oscillator running, and makes the CLK signal available to the system. When in power-save the ALE strobe will be stopped high and the address lines [AD(0-7), A(8-15)] will indicate the next machine address. When $\overline{\text{PS}}$ is returned high, the opcode fetch (or M1 cycle) of the CPU begins in a normal manner. Note this M1 cycle could also be an interrupt acknowledge cycle if the MPU800 was interrupted simultaneously with $\overline{\text{PS}}$. *Figure 12* illustrates the power-save feature.

In the event $\overline{\text{BREQ}}$ is asserted (low) at the end of an instruction cycle and $\overline{\text{PS}}$ is active simultaneously, the following occurs:

1. The MPU800 will go into $\overline{\text{BACK}}$ cycle
2. Upon completion of $\overline{\text{BACK}}$ cycle if $\overline{\text{PS}}$ is still active the CPU will go into power-save mode.

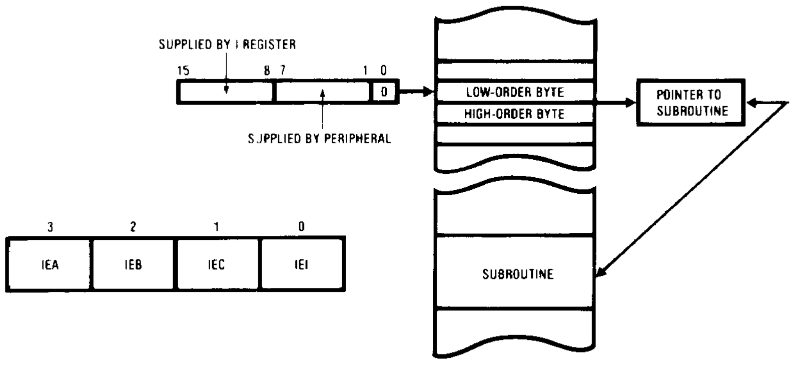


FIGURE 11. Interrupt Mode 2

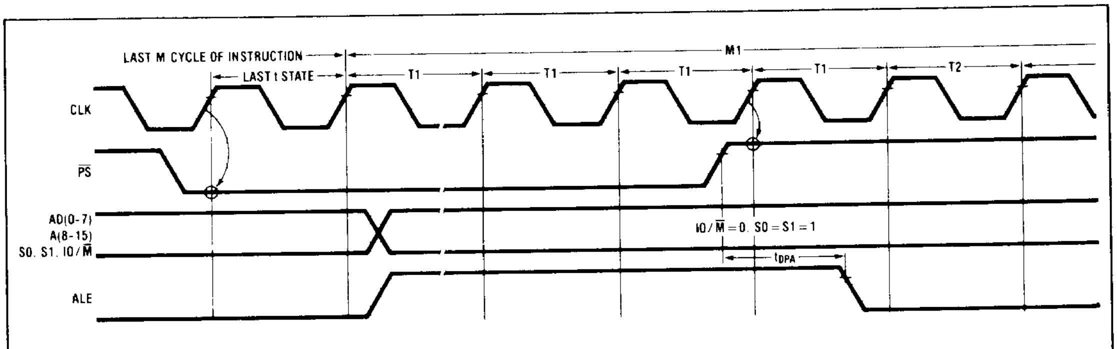


FIGURE 12. MPU800 Power-Save

INSTRUCTION SET

In the following instruction set listing, the notation used is shown below.

b: Used in instructions employing bit mode addressing to designate one bit in a register or memory location.

cc: Designates condition codes used in conditional Jumps, Calls, and Return instructions; may be
NZ = Non Zero (Z Flag = 0)
Z = Zero (Z Flag = 1)
NC = Non Carry (C Flag = 0)
C = Carry (C Flag = 1)

PO = Parity Odd or No Overflow (P/V = 0)
PE = Parity Even or Overflow (P/V = 1)
P = Positive (S = 0)
M = Negative (S = 1)

d: Used in instructions employing relative or indexed modes of addressing to designate 8-bit signed 2's complement displacement.

kk: Subset of cc condition codes used in conjunction with conditional relative jumps; may be NZ, Z, NC or C.

m1: Used in instructions employing register indirect or indexed modes of addressing; may be (HL), (IX + d) or (IY + d).

m2: Used in instructions employing register indirect or direct modes of addressing; may be (BC), (DE), or (nn).

n: Any 8-bit binary number.

nn: Any 16-bit binary number.

pp: Used in 16-bit arithmetic instructions employing register modes of addressing; may be BC, DE, SP, or register designated as destination operand.

qq: Used in instructions employing register modes of addressing; may be BC, DE, HL, AF, IX, or IY.

r: Used in instructions employing register mode of addressing; may be A, B, C, D, E, H, or L.

rr: Used in instructions employing register mode of addressing; may be BC, DE, HL, SP, IX, or IY.

ss: Used in instructions employing register mode of addressing; may be HL, IX, or IY.

T: Used to restart instructions employing modified page zero addressing mode; may take on hex values of 0, 8, 10, 18, 20, 28, 30, or 38.

X_L: Subscript L indicates the high order byte of a 16-bit register.

X_H: Subscript H indicates the high-order byte of a 16-bit register.

( ): Parentheses indicate the contents are considered a pointer to a memory or I/O location.

8-BIT LOADS

REGISTER TO REGISTER

Mnemonic	Description	Operation
LD r_d, r_s	Load register r_d with r_s	$r_d \leftarrow r_s$
LD A,I	Load ACC with register I	A←I
LD I, A	Load register I with ACC	I←A
LD A, r	Load ACC with register R	A←r
LD r, A	Load register R with ACC	r←A
LD r, n	Load register r with immediate data n	r←n

REGISTER TO MEMORY

Mnemonic	Description	Operation
LD m1, r	Load memory from register r	m1←r
LD, m2, A	Load memory from ACC	m2←A
LD m1, n	Load memory with immediate data n	m1←n

MEMORY TO REGISTER

Mnemonic	Description	Operation
LD r, m1	Load register r from memory	r←m1
LD A, m2	Load ACC from memory	A←m2

16-BIT LOADS

REGISTER TO REGISTER

Mnemonic	Description	Operation
LD rr, nn	Load register rr with immediate data nn	rr←nn
LD SP, ss	Load SP register with register ss	SP←ss

REGISTER TO MEMORY

Mnemonic	Description	Operation
LD (nn), rr	Load memory location nn with 16 bit register rr	(nn)←rr_L (nn + 1)←rr_H
PUSH qq	Push contents of 16-bit register qq onto memory stack	(SP−1)←qq_H (SP−2)←qq_L SP←SP−2

MEMORY TO REGISTER

Mnemonic	Description	Operation
LD rr, (nn)	Load 16-bit register rr from memory location nn	rr_L←(nn) rr_H←(nn + 1)
POP qq	Pop contents of stack to register qq	qq_L←(SP) qq_H←(SP + 1) SP←SP + 2

8-BIT ARITHMETIC

REGISTER ADDRESSED ARITHMETIC

Mnemonic	Description	Operation
ADD A, r	Add contents of register r to ACC	A←A + r
ADC, A, r	Add with carry contents of register r to ACC	A←A + r + CY
SUB r	Subtract contents of register r from ACC	A←A−r−r
SBC A, r	Subtract contents of contents of register r from ACC	A←A−r−CY
AND r	Logically AND contents of register r with ACC	A←A ∧ r
OR r	Logically AND contents of register r with ACC	A←A ∨ r
XOR r	Exclusive OR contents of register r with ACC	A←A⊻r
CP r	Compare contents of register r to ACC	A:r Z flag←1 if A = r else Z Flagfi0
INC r	Increment contents of register r	r←r + 1
DEC r	Decrement contents of register r	r←r−1
DAA	Decimal adjust ACC	(ACC adjust for BCD)
CPL	Complement ACC (1's complement)	A←A
NEG	Negate ACC (2's complement)	A←0−A
CCF	Complement carry flag	CY←CY
SCF	Set carry flag	CY←1

IMMEDIATE ADDRESSING MODE ARITHMETIC

Mnemonic	Description	Operation
ADD A, n	Add number n to ACC	A←A + n
ADC A, n	Add with carry number n to ACC	A←A + n + CY
SUB n	Subtract number n from ACC	A←A−n
SBC A, n	Subtract with carry number n from ACC	A←A−n−CY
AND n	AND number n with ACC	A←A ∧ n
OR n	OR number n with ACC	A←A ∨ n
XOR n	Exclusive OR number n with ACC	A←A⊻n1
CP n1	Compare number n to ACC	A: n1 Z flag←1 if A = n else Z Flag←0

MEMORY ADDRESSED ARITHMETIC

Mnemonic	Description	Operation
ADD A, m1	Add memory to ACC	A←A + m1
ADC A, m1	Add with carry memory to ACC	A←A + m1 + CY
SUB m1	Subtract memory from ACC	A←A−m1
SBC A, m1	Subtract with carry memory from ACC	A←A−m1−CY
AND m1	AND memory with ACC	A←A∧m1
OR m1	OR memory with ACC	A←A∨m1
XOR m1	Exclusive OR memory with ACC	A←A⊻m1
CP m1	Compare memory with ACC	A: m1 Z flag←1 if A = r else Z Flag←0
INC m1	Increment memory	m1←m1 + 1
DEC m1	Decrement memory	m1←m1−1

16-BIT ARITHMETIC

REGISTER ADDRESSED ARITHMETIC

Mnemonic	Description	Operation
ADD ss, pp	Add 16-bit register pp to 16-bit register ss	ss←ss + pp
ADC HL, pp	Add with cary 16-bit register pp to HL	HL←HL + pp + CY
SBC HL, pp	Subtract with carry 16-bit register pp from HL	HL←HL −pp−CY
INC rr	Increment 16-bit register rr	rr←rr + 1
DEC rr	Decrement 16-bit register rr	rr←rr−1

BIT SET, RESET, AND TEST

REGISTER

Mnemonic	Description	Operation
SET b, r	Set bit in register r	$r_b \leftarrow 1$
RES b, r	Reset bit in register r	$r_b \leftarrow 0$
BIT b, r	Test bit in register r	$Z \leftarrow r_b$

MEMORY

Mnemonic	Description	Operation
Set b, m1	Set bit in memory location m1	m1b←1
RES b, m1	Reset bit b in memory location m1	m1b←0
BIT b, m1	Test bit b in memory location m1	Z←m1b

EXCHANGES

REGISTER/REGISTER

Mnemonic	Description	Operation
EX DE, HL	Exchange contents of DE and HL register	DE↔HL
EX AF, AF1	Exchange contents of A and F registers with A1 and F1 registers	AF↔AF′
EXX	Exchange contents of BC, DE and HL registers with corresponding alternate registers	BC↔BC′ DE↔DE′ HL↔HL′

REGISTER/MEMORY

Mnemonic	Description	Operation
EX (SP), ss	Exchange top of stack with 16-bit register ss	$(SP) \leftrightarrow ss_L$ $(SP+1) \leftrightarrow ss_H$

MEMORY BLOCK MOVES AND SEARCHES

Block move and search instructions (such as LDIR and INIR) Insert a dummy instruction fetch after each cycle to keep refresh going.

SINGLE OPERATIONS

Mnemonic	Description	Operation
LDI	Move data from memory location (HL) to memory location (DE), increment memory pointers, and decrement byte counter BC.	(DE)←(HL) DE←DE + 1 HL←HL + 1 BC←BC−1
LDD	Move data from memory location (HL) to memory location (HL) to memory location (DE), and decrement memory pointer and byte counter BC.	(DE)←(HL) (DE)←(HL) DE←DE−1 HL←HL−1 BC←HL−1
CPI	Compare data in memory location (HL) to ACC, increment memory pointer and decrement byte counter BC.	A−(HL) HL←HL + 1 BC←BC−1
CPD	Compare data in memory location (HL) to ACC and decrement memory pointer and byte counter BC.	A−(HL) HL←HL−1 BC←BC−1

REPEAT OPERATIONS

Mnemonic	Description	Operation
LDIR	Move data from memory location (HL) to memory location (DE), increment memory pointers, decrement byte counter BC, until BC = 0	(DE)←(HL) DE←DE + 1 HL←HL + 1 BC←BC−1 Repeat until BC = 0
LDDR	Move data from memory location (HL) to memory location (DE), decrement memory pointers and byte counter BC, repeat until BC = 0	(DE)←(HL) DE←DE−1 HL←HL−1 BC←BC−1 Repeat until BC = 0
CPIR	Compare data in memory location (HL) to ACC, increment memory pointer, decrement byte counter BC, repeat until BC = 0 or (HL) = A	A−(HL) HL←HL + 1 BC←BC−1 Repeat until BC = 0 or (HL) = A
CPDR	Compare data in memory location (HL) to ACC, decrement memory pointer and byte counter BC, repeat until BC = 0 or (HL) = A	A−(HL) HL←HL−1 BC←BC−1 Repeat until BC = 0 or (HL) = A

INPUT/OUTPUT

Due to the multiplexed bus structure, the MPU800 handles the address bus differently than the Z80 during input and output instructions. The MPU800 duplicates the port address on the upper and lower halves of the address.

Mnemonic	Description	Operation
IN A, (n)	Input from I/O device at address n to ACC	A←(n)
OUT (n), A	Output to I/O device at address n from ACC	(n)←A
IN r, (C)	Input from I/O device at address (C) to register	r←(C)
OUT (C), r	Output to I/O device at address (C) from register	(C)←r
INI	Input from I/O device at address (C) to memory location (HL), increment pointer, and decrement B counter	(HL)←(C) HL←HL + 1 B←B−1

Mnemonic	Description	Operation
OUTI	Output to I/O at address (C) from memory location (HL), increment pointer, and decrement B counter	(C)←(HL) HL←HL+1 B←B−1
IND	Input from I/O device at address (C) to memory location (HL) and decrement pointer, and B counter	(HL)←(C) HL←HL−1 B←B−1
OUTD	Output to I/O device at address (C) from memory location (HL) and decrement pointer B counter	(C)←(HL) HL←HL−1 B←B−1
INIR	Output to I/O device at at address (C) to memory location (HL), increment pointer, decrement B counter, and repeat until B = 0	(HL)←C HL←HL+1 B←B−1 Repeat until B = 0
OUTIR	Output to I/O device at address (C) from memory location (HL), increment pointer, decrement B counter, and repeat until B = 0	(C)←(HL) HL←HL+1 B←B−1 Repeat until B = 0
INDR	Input from I/0 device at address (C) to memory location (HL), decrement pointer and B counter, and repeat until B = 0	(HL)←(C) HL←HL−1 B←B−1 Repeat until B = 0
OUTDR	Output to I/O device at address (C) from memory location (HL), decrement pointer and B counter, and repeat until B = 0	(C)←(HL) HL←HL−1 B←B−1 Repeat until B = 0

CPU CONTROL

Mnemonic	Description	Operation
NOP	No operation	
HALT*	Halt processor	
DI	Disable Interrupts	
EI	Enable Interrupts	
IM 0	Set Interrupt Mode 0	
IM 1	Set Interrupt Mode 1	
IM 2	Set Interrupt Mode 2	

*Halt instruction locks CPU into an endless cycle of instruction fetches until CPU is reset or interrupted. Therefore dynamic memory refresh continues to run.

PROGRAM CONTROL

JUMPS

Mnemonic	Description	Operation
JP nn	Unconditional jump direct to nn	PC←nn
JP (ss)	Unconditional jump indirect via ss register	PC←ss
JP cc, nn	Conditionally jump direct to nn if cc is true	If cc true, PC←nn, else continue
JR d	Unconditional jump to PC + d	PC←PC + d
JR kk, d	Conditionally jump PC + d if kk is true	if kk true, PC←PC + d
DJNZ, d	Decrement B register and jump to PC + d if B ≠ 0, otherwise continue	B←B−1 if B = 0 PC←PC + d

CALLS

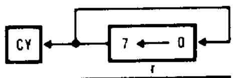
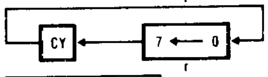
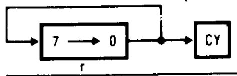
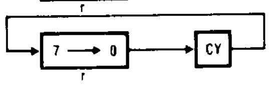
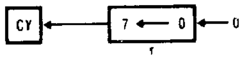
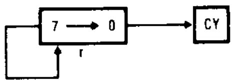
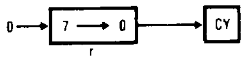
Mnemonic	Description	Operation
CALL nn	Unconditional call to subroutine at location nn	$(SP-1) \leftarrow PC_H$ $(SP-2) \leftarrow PC_L$ PC←nn
CALL cc, nn	Conditional call to subroutine at location nn if cc true	if cc true, $(SP-1) \leftarrow PC_H$ $(SP-2) \leftarrow PC_L$ PC←nn, else continue

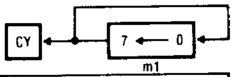
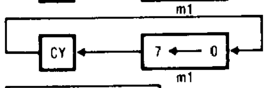
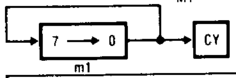
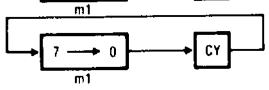
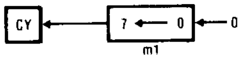
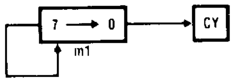
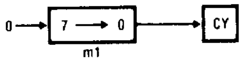
RETURNS

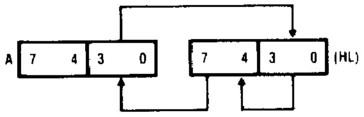
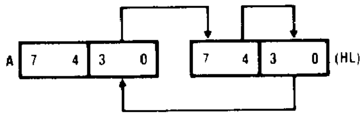
Mnemonic	Description	Operation
RET	Unconditional return from subroutine	$PC_L \leftarrow (SP)$ $PC_H \leftarrow (SP+1)$
RET cc	Conditional return from subroutine	If cc true: $PC_L \leftarrow (SP)$ $PC_H \leftarrow (SP+1)$ else continue
RETI	Return from interrupt	$PC_L \leftarrow (SP)$ $PC_H \leftarrow (SP+1)$
RETN	Return from non-maskable interrupt	$PC_L \leftarrow (SP)$ $PC_H \leftarrow (SP+1)$ Restore interrupt enable status

RESTARTS

Mnemonic	Description	Operation
RST T	Interrupt to location T	$(SP-1) \leftarrow PC_H$ $(SP-2) \leftarrow PC_L$ PC←T

REGISTER MNEMONIC	DESCRIPTION	OPERATION
RLC r	ROTATE REGISTER r LEFT CIRCULAR	
RL r	ROTATE REGISTER r LEFT THROUGH CARRY	
RRC r	ROTATE REGISTER r RIGHT CIRCULAR	
RR r	ROTATE REGISTER r RIGHT THROUGH CARRY	
SLA r	SHIFT REGISTER r LEFT ARITHMETIC	
SRA r	SHIFT REGISTER r RIGHT ARITHMETIC	
SRL r	SHIFT REGISTER r RIGHT LOGICAL	

MEMORY MNEMONIC	DESCRIPTION	OPERATION
RLC m1	ROTATE MEMORY LEFT CIRCULAR	
RL m1	ROTATE MEMORY LEFT THROUGH CARRY	
RRC m1	ROTATE MEMORY RIGHT CIRCULAR	
RR m1	ROTATE MEMORY RIGHT THROUGH CIRCULAR	
SLA m1	SHIFT MEMORY LEFT ARITHMETIC	
SRA m1	SHIFT MEMORY RIGHT ARITHMETIC	
SRL m1	SHIFT MEMORY RIGHT LOGICAL	

REGISTER/MEMORY MNENONIC	DESCRIPTION	OPERATION
RLD	ROTATE DIGIT LEFT AND RIGHT BETWEEN ACC AND MEMORY (HL)	
RRD	ROTATE DIGIT RIGHT AND LEFT BETWEEN ACC AND MEMORY (HL)	

ROTATE AND SHIFT

ABSOLUTE MAXIMUM RATINGS (Note 1)

Storage Temperature	−65°C to +150°C
Voltage on Any Pin with Respect to Ground	−0.3V to V_{CC} + 0.3V
Maximum V_{CC}	7V
Power Dissipation	1W
Lead Temperature (Soldering, 10 seconds)	300°C

DC ELECTRICAL CHARACTERISTICS T_A = 0°C to +70°C, V_{CC} = 5V ± 10%, GND = 0V, unless otherwise specified.

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{IH}	Logical 1 Input Voltage		0.7 V_{CC}		V_{CC}	V
V_{IL}	Logical 0 Input Voltage		0		0.2 V_{CC}	V
V_{HY}	Hysteresis at RESET IN input	V_{CC} = 5V	0.25	0.5		V
V_{OH1}	Logical 1 Output Voltage	I_{OUT} = −1.0 mA	2.4			V
V_{OH2}	Logical 1 Output Voltage	I_{OUT} = −10 µA	V_{CC}−.05			V
V_{OL1}	Logical 0 Output Voltage	I_{OL} = 2 mA	0		0.4	V
V_{OL2}	Logical 0 Output Voltage	I_{OUT} = 10 µA	0		0.1	V
I_{IL}	Input Leakage Current	$0 \leq V_{IN} \leq V_{CC}$	−10.0		10.0	µA
I_{OL}	Output Leakage Current	$0 \leq V_{IN} \leq V_{CC}$	−10.0		10.0	µA
I_{CCA}	Active Supply Current	I_{OUT} = 0, $f_{(XIN)}$ = 5 MHz		10	15	mA
I_{CCA}	Active Supply Current	I_{OUT} = 0, $f_{(XIN)}$ = 8 MHz		15	21	mA
I_{CCQ}	Quiescent Current	$f_{(XIN)}$ = 0 MHz		2	4	mA
I_{CPS}	Power-Save Current	$f_{(XIN)}$ = 5.0 MHz		5		mA
C_{IN}	Input Capacitance			6	10	pF
C_{OUT}	Output Capacitance			8	12	pF
V_{CC}	Power Supply Voltage	Note 2	2.4	5	6	V

Note 1: Absolute Maximum Ratings indicate limits beyond which permanent damage may occur. Continuous operation at these limits is not intended and should be limited to those conditions specified under DC Electrical Characteristics.

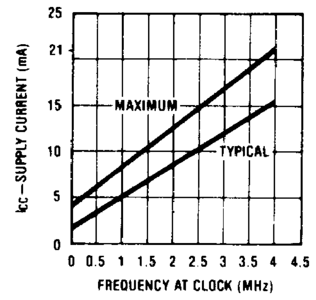
Note 2: CPU operation at lower voltages will reduce the maximum operating speed. DC and AC electrical characteristics at voltages other than 5V ± 10% are forthcoming.

Preliminary (not tested)

Max CPU Speed*	MPU800-1	MPU800	MPU800-4	Units
@2.4V		500	500	kHz
@3.0V		1	1	MHz

*Speed of CPU is expressed in clock speed, not crystal speed.

I_{CC} vs System Speed



PRELIMINARY
Notice: This is not a final specification. Some parametric limits are subject to change.

AC ELECTRICAL CHARACTERISTICS V_{CC} = 5V ± 10%, GND = OV

Symbol	Parameter	MPU800-1		MPU800		MPU800-4		Units
		Min	Max	Min	Max	Min	Max	
t_x	Period at XIN and XOUT Pins	500	31250	200	31250	125	31250	ns
T	Period at Clock Output (= 2 t_X)	1000	62500	400	62500	250	62500	ns

AC ELECTRICAL CHARACTERISTICS (Continued) $V_{CC} = 5V \pm 10\%$, GND = OV

Symbol	Parameter	MPU800-1		MPU800		MPU800-4		Units	Notes
		Min	Max	Min	Max	Min	Max		
t_x	Period at XIN and XOUT Pins	500	31250	200	31250	125	31250	ns	
T	Period at Clock Output ($= 2 t_X$)	1000	62500	400	62500	250	62500	ns	
t_R	Clock Rise Time		110		110		75	ns	Measured from 10%-90% of signal
t_F	Clock Fall Time		60		60		40	ns	Measured from 10%-90% of signal
t_L	Clock Low Time	490		190		95		ns	50% duty cycle, square wave input on XIN
t_H	Clock High Time	450		150		80		ns	50% duty cycle, square wave input on XIN
$t_{ACC(RD)}$	ALE to Valid Data		1375		500		300	ns	Add t for each $\overline{\text{WAIT}}$ STATE Add t/2 for memory read cycles
t_{AFR}	AD(0-7) Float after $\overline{\text{RD}}$ Falling		0		0		0	ns	
t_{BABE}	$\overline{\text{BACK}}$ Rising to Bus Enable		1000		400		250	ns	
t_{BABF}	$\overline{\text{BACK}}$ Falling to Bus Float		50		50		50	ns	
t_{BACL}	$\overline{\text{BACK}}$ Falling to CLK Falling	425		125		55		ns	
t_{BRH}	$\overline{\text{BREQ}}$ Hold Time	0		0		0		ns	
t_{BRS}	$\overline{\text{BREQ}}$ Set-Up Time	100		50		35		ns	
t_{CAF}	Clock Falling to ALE Falling	0	30	0	30	0	35	ns	
t_{CAR}	Clock Rising to ALE Rising	0	100	0	100	0	75	ns	
t_{DAI}	ALE Falling to $\overline{\text{INTA}}$ Falling	530		230		100		ns	
t_{DAR}	ALE Falling to $\overline{\text{RD}}$ Falling	525	575	225	250	125	160	ns	
t_{DAW}	ALE Falling to $\overline{\text{WR}}$ Falling	990	1010	390	410	220	250	ns	
$t_{D(BACK)1}$	ALE Falling to $\overline{\text{BACK}}$ Falling	2500		1000		600		ns	Add t for each $\overline{\text{WAIT}}$ state Add t for opcode fetch cycles
$t_{D(BACK)2}$	$\overline{\text{BREQ}}$ Rising to $\overline{\text{BACK}}$ Rising	500	1600	200	700	125	475	ns	
$t_{D(I)}$	ALE Falling to $\overline{\text{INTR}}$, NMI, RSTA-C, $\overline{\text{PS}}$, $\overline{\text{BREQ}}$ Inputs Valid		1375		475		250	ns	Add t for each $\overline{\text{WAIT}}$ state Add t for opcode fetch cycles
t_{DPA}	Rising $\overline{\text{PS}}$ to Falling ALE	500	1550	200	650	125	475	ns	See *Figure 12* also
$t_{D(RFSH)}$	Falling ALE to Falling $\overline{\text{RFSH}}$	1500		600		325		ns	Add t for each $\overline{\text{WAIT}}$ state
$t_{D(WAIT)}$	ALE Falling to $\overline{\text{WAIT}}$ input Valid		550		250		125	ns	
$t_{H(ADH)1}$	A(8-15) Holt Time During Opcode Fetch	0		0		0		ns	
$t_{H(ADH)2}$	A(8-15) Hold Time During Memory or IO, $\overline{\text{RD}}$ and $\overline{\text{WR}}$	400		100		60		ns	
$t_{H(ADL)}$	AD(0-7) Hold Time	400		100		50		ns	
$t_{H(WD)}$	Write Data Hold Time	400		100		50		ns	
$_{INH}$	Interrupt Hold Time	0		0		0		ns	
t_{INS}	Interrupt Set-Up Time	100		50		35		ns	
t_{NMI}	Width of NMI Input	50		30		20		ns	
t_{RDH}	Data Hold after Read	0		0		0		ns	
t_{RFL}	$\overline{\text{RFSH}}$ Rising to ALE Rising		−100		−100		−70	ns	Negative number means ALE occurs first

AC ELECTRICAL CHARACTERISTICS (Continued) V_{CC} = 5V ± 10%, GND = OV

Symbol	Parameter	MPU800-1		MPU800		MPU800-4		Units	Notes
		Min	Max	Min	Max	Min	Max		
$t_{RL(MR)}$	$\overline{RD}$ Rising to ALE Rising (Memory Read)	450		150		85		ns	
$t_{RL(OP)}$	$\overline{RD}$ Rising to ALE Rising Opcode		-75		-65		-55	ns	
$T_{S(AD)}$	AD(0-7) Set-Up Time	300		80		40		ns	
$t_{S(ALE)}$	A98-15), SO, SI, IO/$\overline{M}$ Set-Up Time	350		100		50		ns	
$t_{S(WD)}$	Write Data Set-Up Time	385		85		50		ns	
$t_{W(ALE)}$	ALE Width	430		130		75		ns	
t_{WH}	$\overline{WAIT}$ Hold Time	0		0		0		ns	
$t_{W(I)}$	Width of $\overline{INTR}$, $\overline{RSTA-C}$, $\overline{PS}$, $\overline{BREQ}$	500		200		125		ns	
$t_{W(INTA)}$	$\overline{INTA}$ Strobe Width	1000		400		200		ns	Add two t states for first $\overline{INTA}$ of each interrupt response string Add t for each $\overline{WAIT}$ state
t_{WL}	$\overline{WR}$ Rising to ALE Rising	450		150		90		ns	
$t_{W(RD)}$	Read Strobe Width During Opcode Fetch	1000		400		225		ns	Add t for each $\overline{WAIT}$ State Add t/2 for Memory Read Cycles
$t_{W(RFSH)}$	Refresh Strobe Width	1925		725		400		ns	
t_{WS}	$\overline{WAIT}$ Set-Up Time	100		50		35		ns	
$t_{W(WAIT)}$	$\overline{WAIT}$ Input Width	550		250		175		ns	
$t_{W(WR)}$	Write Strobe Width	1000		400		220		ns	Add t for each $\overline{WAIT}$ state
t_{XCF}	XIN to Clock Falling	25	55	25	55	25	55	ns	
t_{XCR}	XIN to Clock Rising	45	75	45	75	45	75	ns	

Note 1: Test conditions: t = 1000 ns for MPU800-1, 400 ns for MPU800, 250 ns for MPU800-4.

Note 2: Output timings are measured with a purely capacitive load of 150 pF. The following correction factor can be used for other loads:
150 pF <C_L≤300 pF: +0.25 ns/pF
50 pF <C_L≤150 pF: −0.15 ns/pF

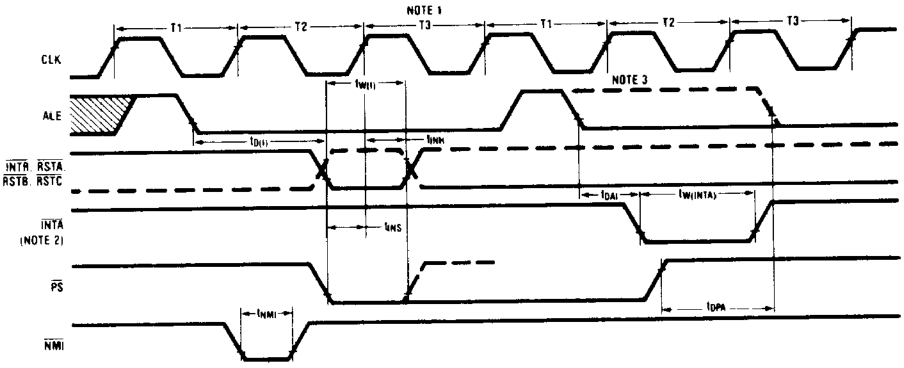
TABLE 1. BUS TIMING AS T DEPENDENT

Symbol	1/T < 2.5 MHz	2.5 MHz < 1/T < 4.0 MHz		Symbol	1/T < 2.5 MHz	2.5 MHz < 1/T < 4.0 MHz	
t_L	(1/2)T−10	(1/2)T−30	min	$T_{D(RFSH)}$	(3/2 + N)T	(3/2 + N)T−50	Min
t_H	(1/2)T−50	(1/2)T−45	Min	$t_{D(WAIT)}$	(1/2)T + 50	(1/2)T	Max
$t_{ACC(RD)}$	(1 + N)T + 100	(1 + N)T + 50	Max	$t_{(ADH)2}$	(1/2)T−100	(1/2)T−65	Min
t_{BABE}	T	T	Max	$t_{H(ADH)2}$	(1/2)T−100	(1/2)T−6	Min
t_{BACL}	(1/2)T−75	(1/2)T−70	Min	$t_{H(ADL)}$	(1/2)T−100	(1/2)T−50	Min
t_{DAI}	(1/2)T + 30	(1/2)T−25	Min	$t_{RL(MR)}$	(1/2)T−50	(1/2)T-40	Min
t_{DAR}	(1/2)T + 25	(1/2)T	Min	$t_{S(AD)}$	(1/2)T−120	(1/2)T−85	Min
t_{DAR}	(1/2)T + 50	(1/2)T + 35	Max	$t_{S(ALE)}$	(1/2)T−100	(1/2)T−75	Min
t_{DAW}	T−10	T−30	Min	$t_{S(WD)}$	(1/2)T−115	(1/2)T−75	Min
t_{DAW}	T + 10	T	Max	$t_{W(ALE)}$	(1/2)T−70	(1/2)T−50	Min
$t_{D(BACK)1}$	(5/2 + N)T	(5/2 + N)T−25	Min	$t_{W(INTA)}$	(1 + N)T	(1 + N)T−50	Min
$t_{D(BACK)2}$	(1/2)T	(1/2)T	Min	t_{WL}	(1/2)T−50	(1/2)T−35	Min
$t_{D(BACK)2}$	(3/2)T + 100	(3/2)T + 100	Max	$t_{W(RD)}$	(1 + N)T	(1 + N)T−25	Min
$t_{D(I)}$	(3/2 + N)T−125	(3/2 + N)T−125	Max	$t_{W(RFSH)}$	2T−75	2T−100	Min
t_{DPA}	(1/2)T	(1/2)T	Min	$t_{W(WR)}$	(1 + N)T	(1 + N)T−30	Min
t_{DPA}	(3/2)T + 50	(3/2)T + 100	Max				

Note: N is equal to number of WAIT states.

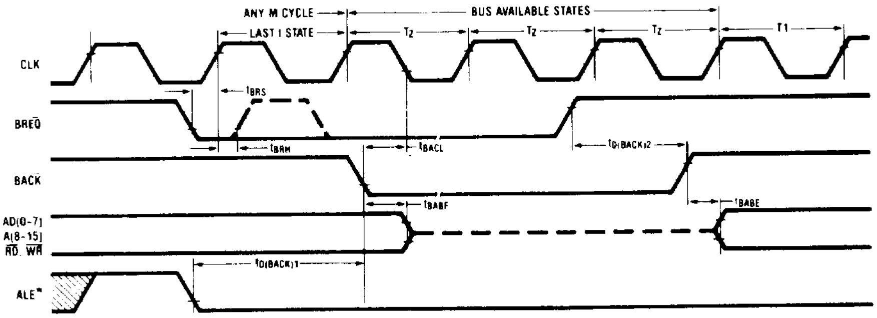
TIMING REFERENCE

Interrupt—Power-Save Cycle



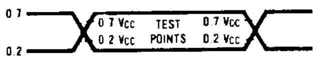
Note 1: This t state is the last t state of the last M cycle of any instruction.
Note 2: Response to INTR input.
Note 3: Response to PS input.

Bus Acknowledge Cycle

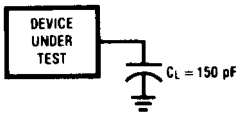


*Waveform not drawn to proportion. Use only for specifying test points.

AC Testing Input/Output Waveform

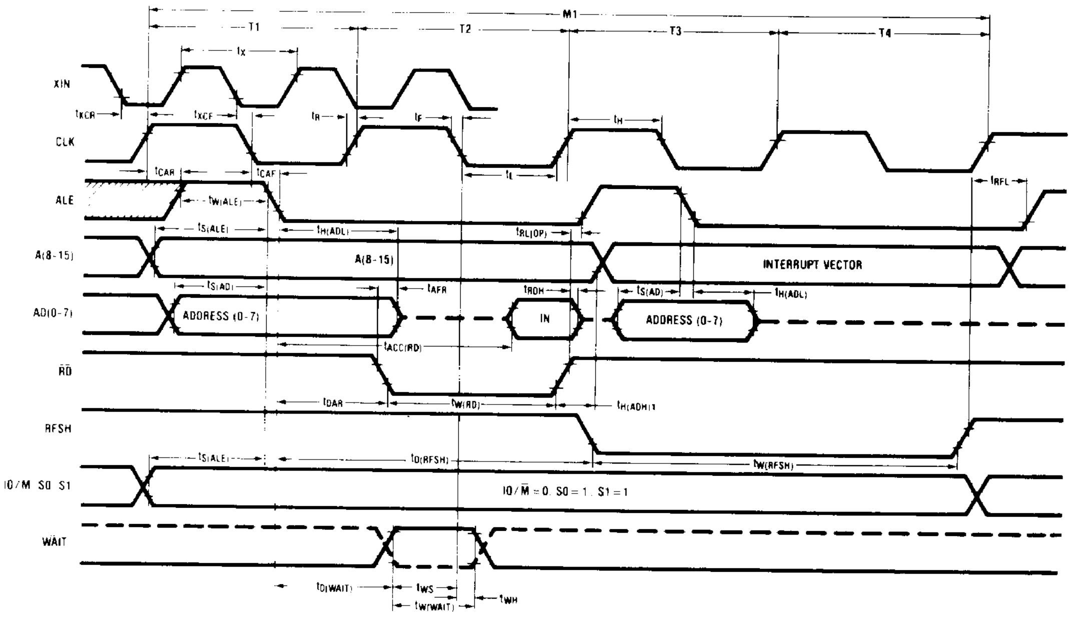


AC Testing Load Circuit

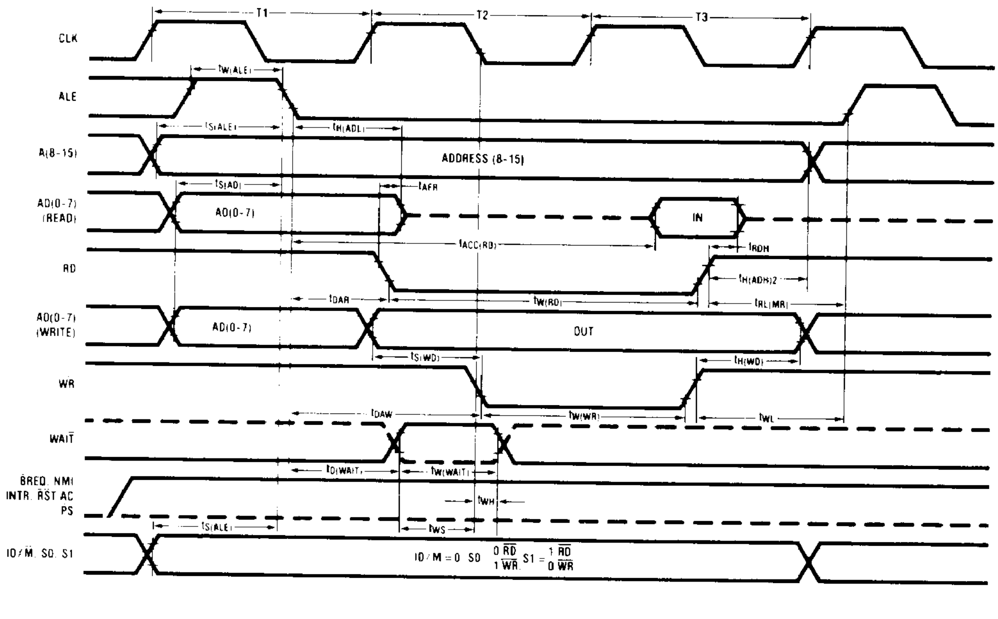


TIMING WAVEFORMS

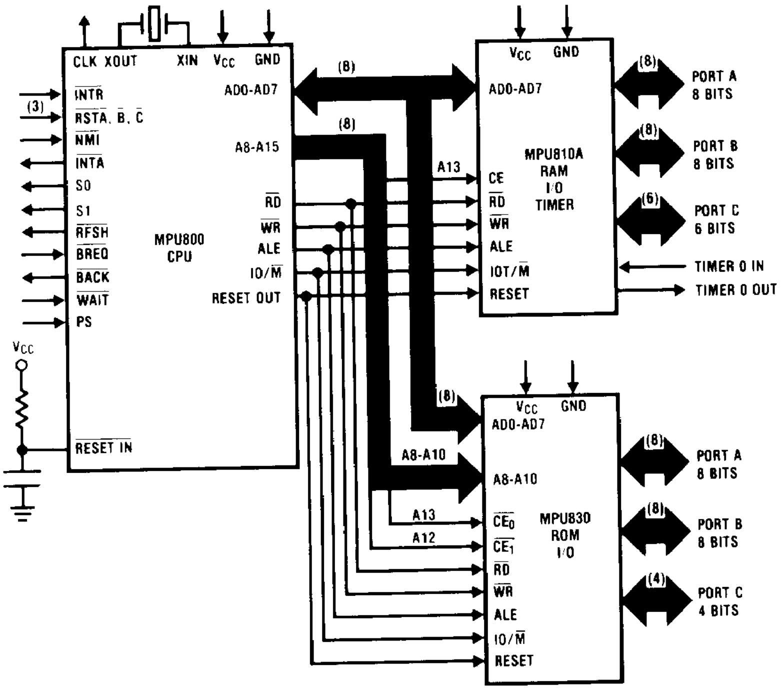
Opcode Fetch Cycle



Memory Read and Write Cycle



MICROCOMPUTER FAMILY BLOCK DIAGRAM



STANDARD MICROSYSTEMS CORPORATION

35 Marcus Blvd. Hauppauge, N.Y. 11788 [illegible]