



MOTOROLA

MC6802 MC6808 MC6802NS

MICROPROCESSOR WITH CLOCK AND OPTIONAL RAM

The MC6802 is a monolithic 8-bit microprocessor that contains all the registers and accumulators of the present MC6800 plus an internal clock oscillator and driver on the same chip. In addition, the MC6802 has 128 bytes of on-board RAM located at hex addresses \$0000 to \$007F. The first 32 bytes of RAM, at hex addresses \$0000 to \$001F, may be retained in a low power mode by utilizing V_{CC} standby; thus, facilitating memory retention during a power-down situation.

The MC6802 is completely software compatible with the MC6800 as well as the entire M6800 family of parts. Hence, the MC6802 is expandable to 64K words.

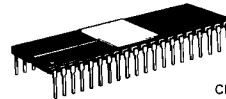
The MC6802NS is identical to the MC6802 without standby RAM feature. The MC6808 is identical to the MC6802 without on-board RAM.

- On-Chip Clock Circuit
- 128 × 8 Bit On-Chip RAM
- 32 Bytes of RAM are Retainable
- Software-Compatible with the MC6800
- Expandable to 64K Words
- Standard TTL-Compatible Inputs and Outputs
- 8-Bit Word Size
- 16-Bit Memory Addressing
- Interrupt Capability

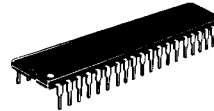
MOS

(N-CHANNEL, SILICON-GATE,
DEPLETION LOAD)

MICROPROCESSOR WITH CLOCK AND OPTIONAL RAM



L SUFFIX
CERAMIC PACKAGE
CASE 715

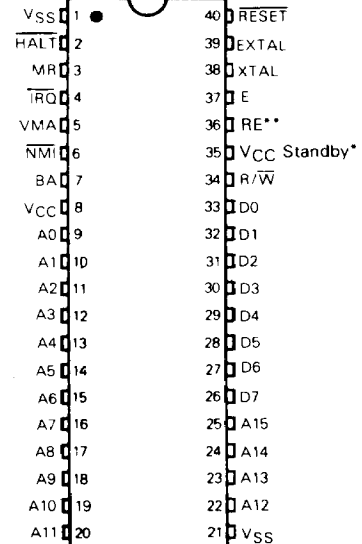


P SUFFIX
PLASTIC PACKAGE
CASE 711

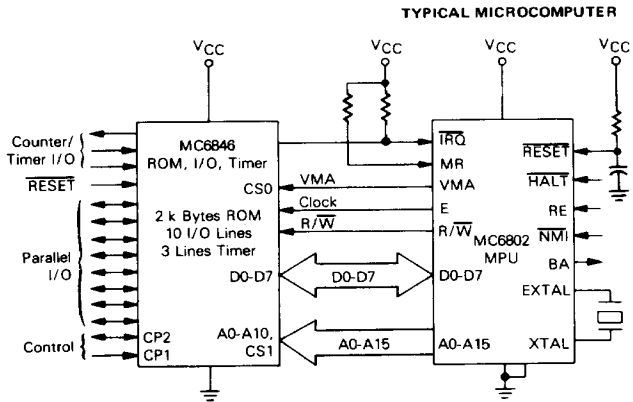
ORDERING INFORMATION

Package Type	Frequency (MHz)	Temperature	Order Number
Ceramic L Suffix	1.0	0°C to 70°C	MC6802L
	1.0	-40°C to 85°C	MC6802CL
	1.0	0°C to 70°C	MC6802NSL
	1.0	0°C to 70°C	MC6808L
	1.5	0°C to 70°C	MC68A02L
	1.5	-40°C to 85°C	MC68A02CL
	1.5	0°C to 70°C	MC68A08L
	2.0	0°C to 70°C	MC68B02L
Plastic P Suffix	2.0	0°C to 70°C	MC68B08L
	1.0	0°C to 70°C	MC6802P
	1.0	-40°C to 85°C	MC6802CP
	1.0	0°C to 70°C	MC6802NSP
	1.0	0°C to 70°C	MC6808P
	1.5	0°C to 70°C	MC68A02P
	1.5	-40°C to 85°C	MC68A02CP
	1.5	0°C to 70°C	MC68A08P
	2.0	0°C to 70°C	MC68B02P
	2.0	0°C to 70°C	MC68B08P

PIN ASSIGNMENT



* Pin 35 must be tied to 5 V on the MC6802NS
 ** Pin 36 must be tied to ground for the MC6808



This block diagram shows a typical cost effective microcomputer. The MPU is the center of the microcomputer system and is shown in a minimum system interfacing with a ROM combination chip. It is not intended that this system be limited to this function but that it be expandable with other parts in the M6800 Microcomputer family.

MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Supply Voltage	V _{CC}	-0.3 to +7.0	V
Input Voltage	V _{in}	-0.3 to +7.0	V
Operating Temperature Range			
MC6802, MC680A02, MC680B02	T _A	0 to +70	°C
MC6802C, MC680A02C		-40 to +85	
MC6802NS		0 to +70	
MC6808, MC68A08, MC68B08		0 to +70	
Storage Temperature Range	T _{stg}	-55 to +150	°C

This input contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage level (e.g., either V_{SS} or V_{CC}).

THERMAL CHARACTERISTICS

Characteristic	Symbol	Value	Unit
Average Thermal Resistance (Junction to Ambient)			
Plastic	θ _{JA}	100	°C/W
Ceramic		50	

POWER CONSIDERATIONS

The average chip-junction temperature, T_J, in °C can be obtained from:

$$T_J = T_A + (P_D \cdot \theta_{JA}) \tag{1}$$

Where:

- T_A = Ambient Temperature, °C
- θ_{JA} = Package Thermal Resistance, Junction-to-Ambient, °C/W
- P_D = P_{INT} + P_{PORT}
- P_{INT} = I_{CC} × V_{CC}, Watts — Chip Internal Power
- P_{PORT} = Port Power Dissipation, Watts — User Determined

For most applications P_{PORT} ≪ P_{INT} and can be neglected. P_{PORT} may become significant if the device is configured to drive Darlington bases or sink LED loads.

An approximate relationship between P_D and T_J (if P_{PORT} is neglected) is:

$$P_D = K + (T_J + 273^\circ\text{C}) \tag{2}$$

Solving equations 1 and 2 for K gives:

$$K = P_D \cdot (T_A + 273^\circ\text{C}) + \theta_{JA} \cdot P_D^2 \tag{3}$$

Where K is a constant pertaining to the particular part. K can be determined from equation 3 by measuring P_D (at equilibrium) for a known T_A. Using this value of K the values of P_D and T_J can be obtained by solving equations (1) and (2) iteratively for any value of T_A.

DC ELECTRICAL CHARACTERISTICS ($V_{CC} = 5.0 \text{ Vdc} \pm 5\%$, $V_{SS} = 0$, $T_A = 0 \text{ to } 70^\circ\text{C}$, unless otherwise noted)

Characteristic	Symbol	Min	Typ	Max	Unit
Input High Voltage Logic, EXTAL, RESET	V_{IH}	$V_{SS} + 2.0$ $V_{SS} + 4.0$	—	V_{CC} V_{CC}	V
Input Low Voltage Logic, EXTAL, RESET	V_{IL}	$V_{SS} - 0.3$	—	$V_{SS} + 0.8$	V
Input Leakage Current ($V_{in} = 0 \text{ to } 5.25 \text{ V}$, $V_{CC} = \text{max}$)	I_{in}	—	1.0	2.5	μA
Output High Voltage ($I_{Load} = -206 \mu\text{A}$, $V_{CC} = \text{min}$) ($I_{Load} = -145 \mu\text{A}$, $V_{CC} = \text{min}$) ($I_{Load} = -100 \mu\text{A}$, $V_{CC} = \text{min}$)	V_{OH}	$V_{SS} + 2.4$ $V_{SS} + 2.4$ $V_{SS} + 2.4$	— — —	— — —	V
Output Low Voltage ($I_{Load} = 1.6 \text{ mA}$, $V_{CC} = \text{min}$)	V_{OL}	—	—	$V_{SS} + 0.4$	V
Internal Power Dissipation (Measured at $T_A = 0^\circ\text{C}$)	P_{INT}	—	0.750	1.0	W
V_{CC} Standby	V_{SB8}	4.0	—	5.25	V
Standby Current	I_{SB8}	—	—	8.0	mA
Capacitance # ($V_{in} = 0$, $T_A = 25^\circ\text{C}$, $f = 1.0 \text{ MHz}$)	C_{in}	—	10 6.5	12.5 10	pF
	C_{out}	—	—	12	pF

*In power-down mode, maximum power dissipation is less than 42 mW.

#Capacitances are periodically sampled rather than 100% tested.

CONTROL TIMING ($V_{CC} = 5.0 \text{ V} \pm 5\%$, $V_{SS} = 0$, $T_A = T_L \text{ to } T_H$, unless otherwise noted)

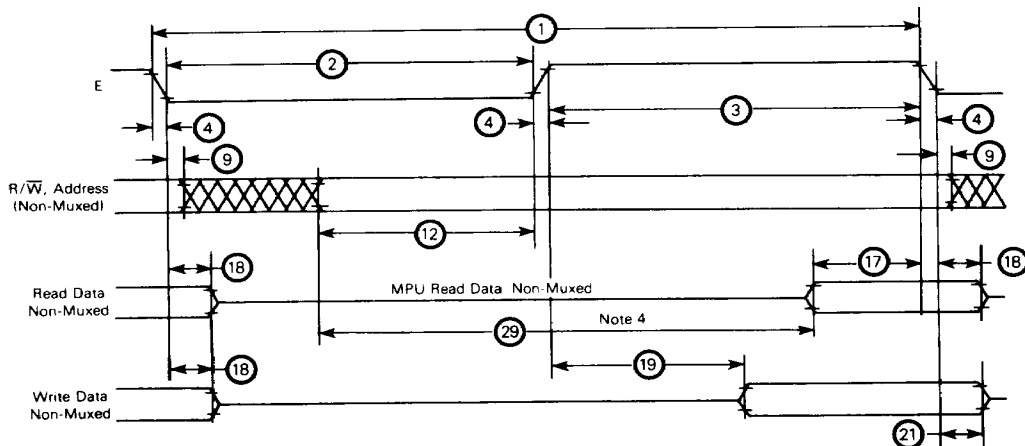
Characteristics	Symbol	MC6802 MC6802NS MC6808		MC68A02 MC68A08		MC68B02 MC68B08		Unit
		Min	Max	Min	Max	Min	Max	
Frequency of Operation	f_o	0.1	1.0	0.1	1.5	0.1	2.0	MHz
Crystal Frequency	f_{XTAL}	1.0	4.0	1.0	6.0	1.0	8.0	MHz
External Oscillator Frequency	$4 \times f_o$	0.4	4.0	0.4	6.0	0.4	8.0	MHz
Crystal Oscillator Start Up Time	t_{rc}	100	—	100	—	100	—	ms
Processor Controls (HALT, MR, RE, RESET, IRQ NMI)								
Processor Control Setup Time	t_{PCS}	200	—	140	—	110	—	ns
Processor Control Rise and Fall Time (Does Not Apply to RESET)	t_{PCr} t_{PCf}	—	100	—	100	—	100	ns

BUS TIMING CHARACTERISTICS

Ident. Number	Characteristic	Symbol	MC6802 MC6802NS MC6808		MC68A02 MC68A08		MC68B02 MC68B08		Unit
			Min	Max	Min	Max	Min	Max	
1	Cycle Time	t_{cyc}	1.0	10	0.667	10	0.5	10	μs
2	Pulse Width, E Low	PW_{EL}	450	5000	280	5000	210	5000	ns
3	Pulse Width, E High	PW_{EH}	450	9500	280	9700	220	9700	ns
4	Clock Rise and Fall Time	$t_{r, f}$	—	25	—	25	—	25	ns
9	Address Hold Time*	t_{AH}	20	—	20	—	20	—	ns
12	Non-Muxed Address Valid Time to E (See Note 5)	t_{AV1} t_{AV2}	160 —	— 270	100 —	— —	50 —	— —	ns
17	Read Data Setup Time	t_{DSR}	100	—	70	—	60	—	ns
18	Read Data Hold Time	t_{DHR}	10	—	10	—	10	—	ns
19	Write Data Delay Time	t_{DDW}	—	225	—	170	—	160	ns
21	Write Data Hold Time*	t_{DHW}	30	—	20	—	20	—	ns
29	Usable Access Time (See Note 4)	t_{ACC}	535	—	335	—	235	—	ns

* Address and data hold times are periodically tested rather than 100% tested.

FIGURE 2 — BUS TIMING



NOTES:

1. Voltage levels shown are $V_L \leq 0.4$ V, $V_H \geq 2.4$ V, unless otherwise specified.
2. Measurement points shown are 0.8 V and 2.0 V, unless otherwise noted.
3. All electricals shown for the MC6802 apply to the MC6802NS and MC6808, unless otherwise noted.
4. Usable access time is computed by: $12 + 3 + 4 - 17$.
5. If programs are not executed from on-board RAM, t_{AV1} applies. If programs are to be stored and executed from on-board RAM, t_{AV2} applies. For normal data storage in the on-board RAM, this extended delay does not apply. Programs cannot be executed from on-board RAM when using A and B parts (MC68A02, MC68A08, MC68B02, MC68B08). On-board RAM can be used for data storage with all parts.
6. All electrical and control characteristics are referenced from: $T_L = 0^\circ\text{C}$ minimum and $T_H = 70^\circ\text{C}$ maximum.

FIGURE 3 — BUS TIMING TEST LOAD

$C = 130 \text{ pF}$ for D0-D7, E
 $= 90 \text{ pF}$ for A0-A15, R/ \overline{W} , and VMA
 $= 30 \text{ pF}$ for BA
 $R = 11.7 \text{ k}\Omega$ for D0-D7, E
 $= 16.5 \text{ k}\Omega$ for A0-A15, R/ \overline{W} , and VMA
 $= 24 \text{ k}\Omega$ for BA

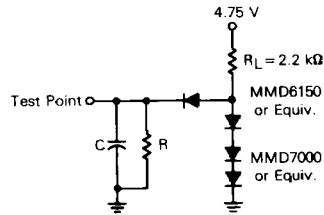


FIGURE 4 — TYPICAL DATA BUS OUTPUT DELAY versus CAPACITIVE LOADING

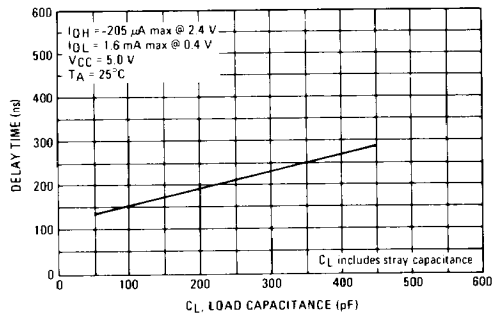


FIGURE 5 — TYPICAL READ/WRITE, VMA AND ADDRESS OUTPUT DELAY versus CAPACITIVE LOADING

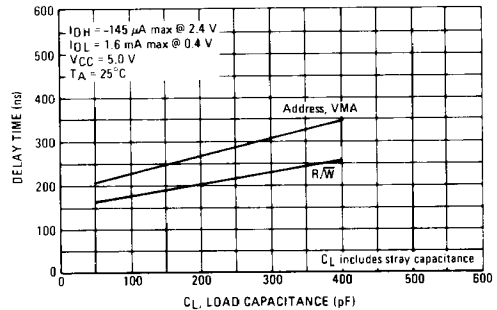
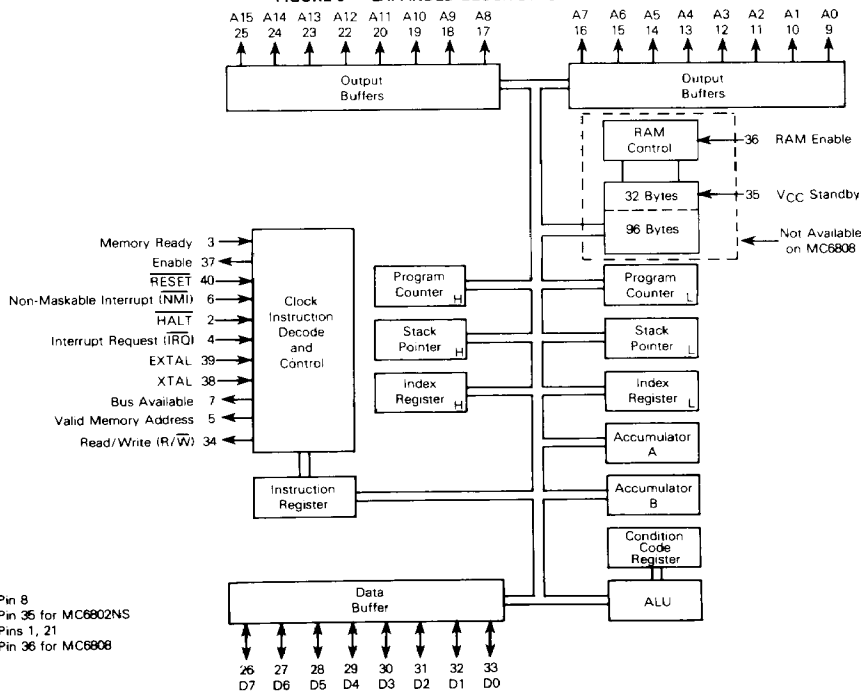


FIGURE 6 — EXPANDED BLOCK DIAGRAM



MPU REGISTERS

A general block diagram of the MC6802 is shown in Figure 6. As shown, the number and configuration of the registers are the same as for the MC6800. The 128 x 8-bit RAM* has been added to the basic MPU. The first 32 bytes can be retained during power-up and power-down conditions via the RE signal.

The MC6802NS is identical to the MC6802 except for the standby feature on the first 32 bytes of RAM. The standby feature does not exist on the MC6802NS and thus pin 35 must be tied to 5 V.

The MC6808 is identical to the MC6802 except for on-board RAM. Since the MC6808 does not have on-board RAM pin 36 must be tied to ground allowing the processor to utilize up to 64K bytes of external memory.

The MPU has three 16-bit registers and three 8-bit registers available for use by the programmer (Figure 7).

PROGRAM COUNTER

The program counter is a two byte (16-bit) register that points to the current program address.

STACK POINTER

The stack pointer is a two byte register that contains the address of the next available location in an external push-down/pop-up stack. This stack is normally a random access

read/write memory that may have any location (address) that is convenient. In those applications that require storage of information in the stack when power is lost, the stack must be non-volatile.

INDEX REGISTER

The index register is a two byte register that is used to store data or a 16-bit memory address for the indexed mode of memory addressing.

ACCUMULATORS

The MPU contains two 8-bit accumulators that are used to hold operands and results from an arithmetic logic unit (ALU).

CONDITION CODE REGISTER

The condition code register indicates the results of an Arithmetic Logic Unit operation: Negative (N), Zero (Z), Overflow (V), Carry from bit 7 (C), and Half Carry from bit 3 (H). These bits of the Condition Code Register are used as testable conditions for the conditional branch instructions. Bit 4 is the interrupt mask bit (I). The unused bits of the Condition Code Register (b6 and b7) are ones.

Figure 8 shows the order of saving the microprocessor status within the stack.

*If programs are not executed from on-board RAM, TAV1 applies. If programs are to be stored and executed from on-board RAM, TAV2 applies. For normal data storage in the on-board RAM, this extended delay does not apply. Programs cannot be executed from on-board RAM when using A and B parts (MC68A02, MC68A08, MC68B02, and MC68B08). On-board RAM can be used for data storage with all parts.

FIGURE 7 — PROGRAMMING MODEL OF THE MICROPROCESSING UNIT

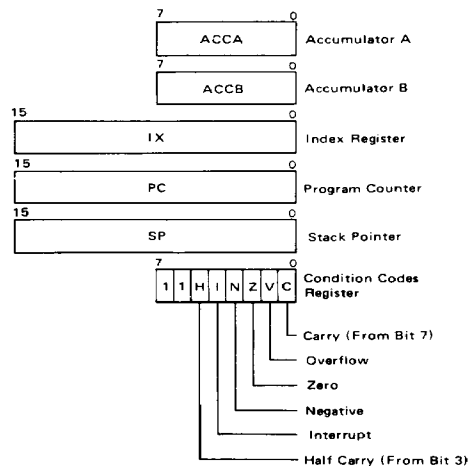
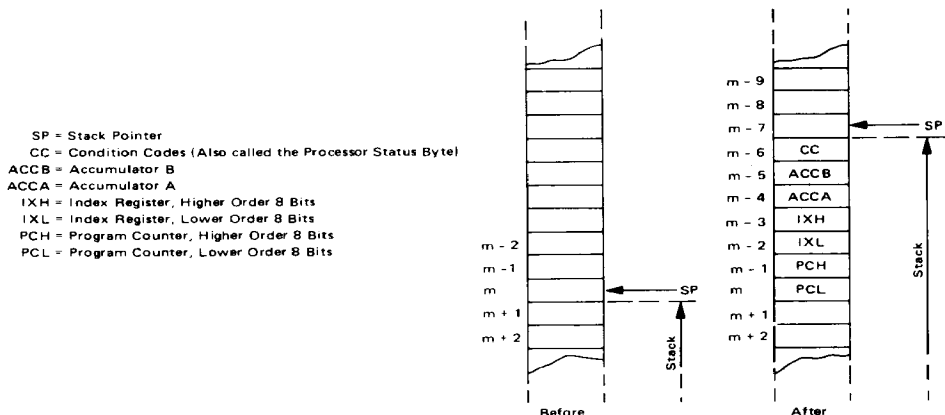


FIGURE 8 — SAVING THE STATUS OF THE MICROPROCESSOR IN THE STACK



MPU SIGNAL DESCRIPTION

Proper operation of the MPU requires that certain control and timing signals be provided to accomplish specific functions and that other signal lines be monitored to determine the state of the processor. These control and timing signals are similar to those of the MC6800 except that TSC, DBE, $\phi 1$, $\phi 2$ input, and two unused pins have been eliminated, and the following signal and timing lines have been added:

RAM Enable (RE)
 Crystal Connections EXTAL and XTAL
 Memory Ready (MR)
 VCC Standby
 Enable $\phi 2$ Output (E)

The following is a summary of the MPU signals:

ADDRESS BUS (A0-A15)

Sixteen pins are used for the address bus. The outputs are capable of driving one standard TTL load and 90 pF. These lines do not have three-state capability.

DATA BUS (D0-D7)

Eight pins are used for the data bus. It is bidirectional, transferring data to and from the memory and peripheral devices. It also has three-state output buffers capable of driving one standard TTL load and 130 pF.

Data bus will be in the output mode when the internal RAM is accessed and RE will be high. This prohibits external data entering the MPU. It should be noted that the internal RAM is fully decoded from \$0000 to \$007F. External RAM at \$0000 to \$007F must be disabled when internal RAM is accessed.

HALT

When this input is in the low state, all activity in the machine will be halted. This input is level sensitive. In the HALT mode, the machine will stop at the end of an instruc-

tion, bus available will be at a high state, valid memory address will be at a low state. The address bus will display the address of the next instruction.

To ensure single instruction operation, transition of the HALT line must occur 1pcS before the falling edge of E and the HALT line must go high for one clock cycle.

HALT should be tied high if not used. This is good engineering design practice in general and necessary to ensure proper operation of the part.

READ/WRITE (R/W)

This TTL-compatible output signals the peripherals and memory devices whether the MPU is in a read (high) or write (low) state. The normal standby state of this signal is read (high). When the processor is halted, it will be in the read state. This output is capable of driving one standard TTL load and 90 pF.

VALID MEMORY ADDRESS (VMA)

This output indicates to peripheral devices that there is a valid address on the address bus. In normal operation, this signal should be utilized for enabling peripheral interfaces such as the PIA and ACIA. This signal is not three-state. One standard TTL load and 90 pF may be directly driven by this active high signal.

BUS AVAILABLE (BA) — The bus available signal will normally be in the low state; when activated, it will go to the high state indicating that the microprocessor has stopped and that the address bus is available (but not in a three-state condition). This will occur if the HALT line is in the low state or the processor is in the WAIT state as a result of the execution of a WAIT instruction. At such time, all three-state output drivers will go to their off-state and other outputs to their normally inactive level. The processor is removed from the

WAIT state by the occurrence of a maskable (mask bit $\bar{I}=0$) or nonmaskable interrupt. This output is capable of driving one standard TTL load and 30 pF.

INTERRUPT REQUEST (\overline{IRQ})

A low level on this input requests that an interrupt sequence be generated within the machine. The processor will wait until it completes the current instruction that is being executed before it recognizes the request. At that time, if the interrupt mask bit in the condition code register is not set, the machine will begin an interrupt sequence. The index register, program counter, accumulators, and condition code register are stored away on the stack. Next the MPU will respond to the interrupt request by setting the interrupt mask bit high so that no further interrupts may occur. At the end of the cycle, a 16-bit vectoring address which is located in memory locations \$FFF8 and \$FFF9 is loaded which causes the MPU to branch to an interrupt routine in memory.

The \overline{HALT} line must be in the high state for interrupts to be serviced. Interrupts will be latched internally while \overline{HALT} is low.

A nominal 3 k Ω pullup resistor to V_{CC} should be used for wire-OR and optimum control of interrupts. \overline{IRQ} may be tied directly to V_{CC} if not used.

RESET

This input is used to reset and start the MPU from a power-down condition, resulting from a power failure or an initial start-up of the processor. When this line is low, the MPU is inactive and the information in the registers will be lost. If a high level is detected on the input, this will signal the MPU to begin the restart sequence. This will start execu-

tion of a routine to initialize the processor from its reset condition. All the higher order address lines will be forced high. For the restart, the last two (\$FFE, \$FFF) locations in memory will be used to load the program that is addressed by the program counter. During the restart routine, the interrupt mask bit is set and must be reset before the MPU can be interrupted by \overline{IRQ} . Power-up and reset timing and power-down sequences are shown in Figures 9 and 10, respectively.

\overline{RESET} , when brought low, must be held low at least three clock cycles. This allows adequate time to respond internally to the reset. This is independent of the t_{rc} power-up reset that is required.

When \overline{RESET} is released it *must* go through the low-to-high threshold without bouncing, oscillating, or otherwise causing an erroneous reset (less than three clock cycles). This may cause improper MPU operation until the next valid reset.

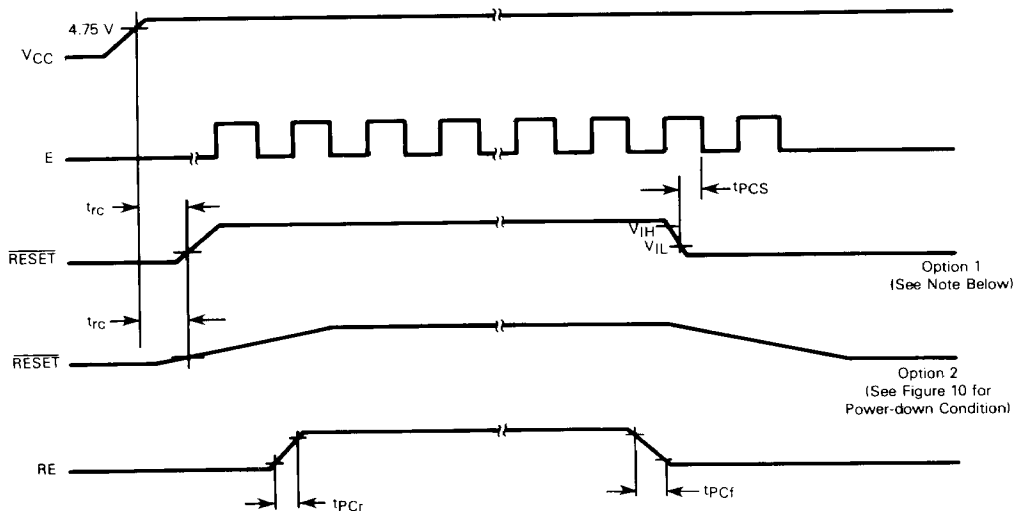
NON-MASKABLE INTERRUPT (\overline{NMI})

A low-going edge on this input requests that a non-maskable interrupt sequence be generated within the processor. As with the interrupt request signal, the processor will complete the current instruction that is being executed before it recognizes the \overline{NMI} signal. The interrupt mask bit in the condition code register has no effect on \overline{NMI} .

The index register, program counter, accumulators, and condition code registers are stored away on the stack. At the end of the cycle, a 16-bit vectoring address which is located in memory locations \$FFFC and \$FFFD is loaded causing the MPU to branch to an interrupt service routine in memory.

A nominal 3 k Ω pullup resistor to V_{CC} should be used for wire-OR and optimum control of interrupts. \overline{NMI} may be tied

FIGURE 9 — POWER-UP AND RESET TIMING



NOTE: If option 1 is chosen, \overline{RESET} and \overline{RE} pins can be tied together.

directly to V_{CC} if not used.

Inputs \overline{IRQ} and \overline{NMI} are hardware interrupt lines that are sampled when E is high and will start the interrupt routine on a low E following the completion of an instruction.

Figure 11 is a flowchart describing the major decision paths and interrupt vectors of the microprocessor. Table 1 gives the memory map for interrupt vectors.

TABLE 1 — MEMORY MAP FOR INTERRUPT VECTORS

Vector		Description
MS	LS	
\$FFFE	\$FFFF	Restart
\$FFFC	\$FFFD	Non-Maskable Interrupt
\$FFFA	\$FFFB	Software Interrupt
\$FFFB	\$FFF9	Interrupt Request

FIGURE 10 — POWER-DOWN SEQUENCE

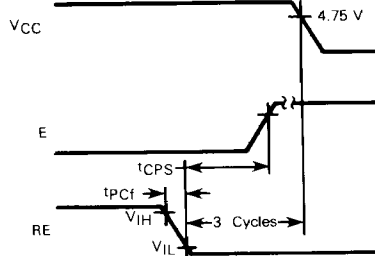


FIGURE 11 — MPU FLOWCHART

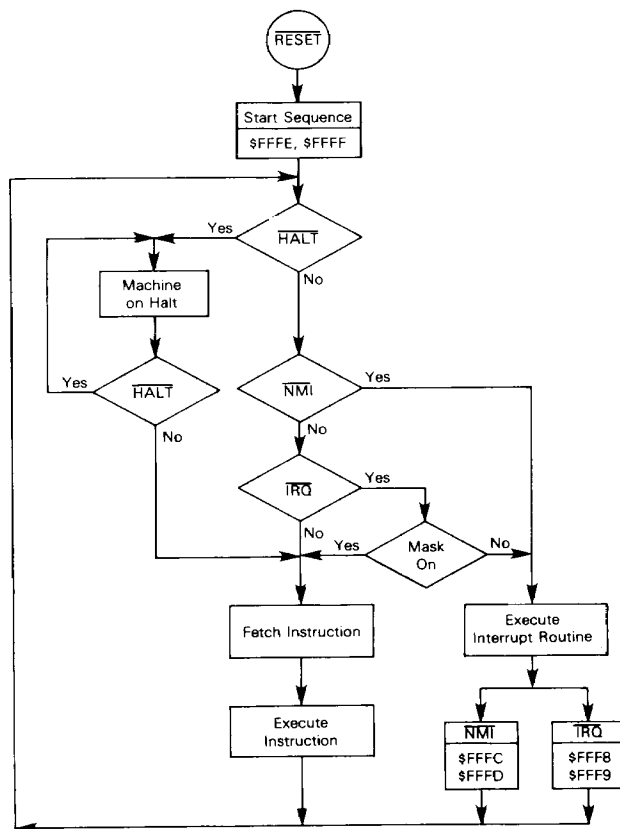


FIGURE 23 — PROGRAM FLOW FOR INTERRUPTS

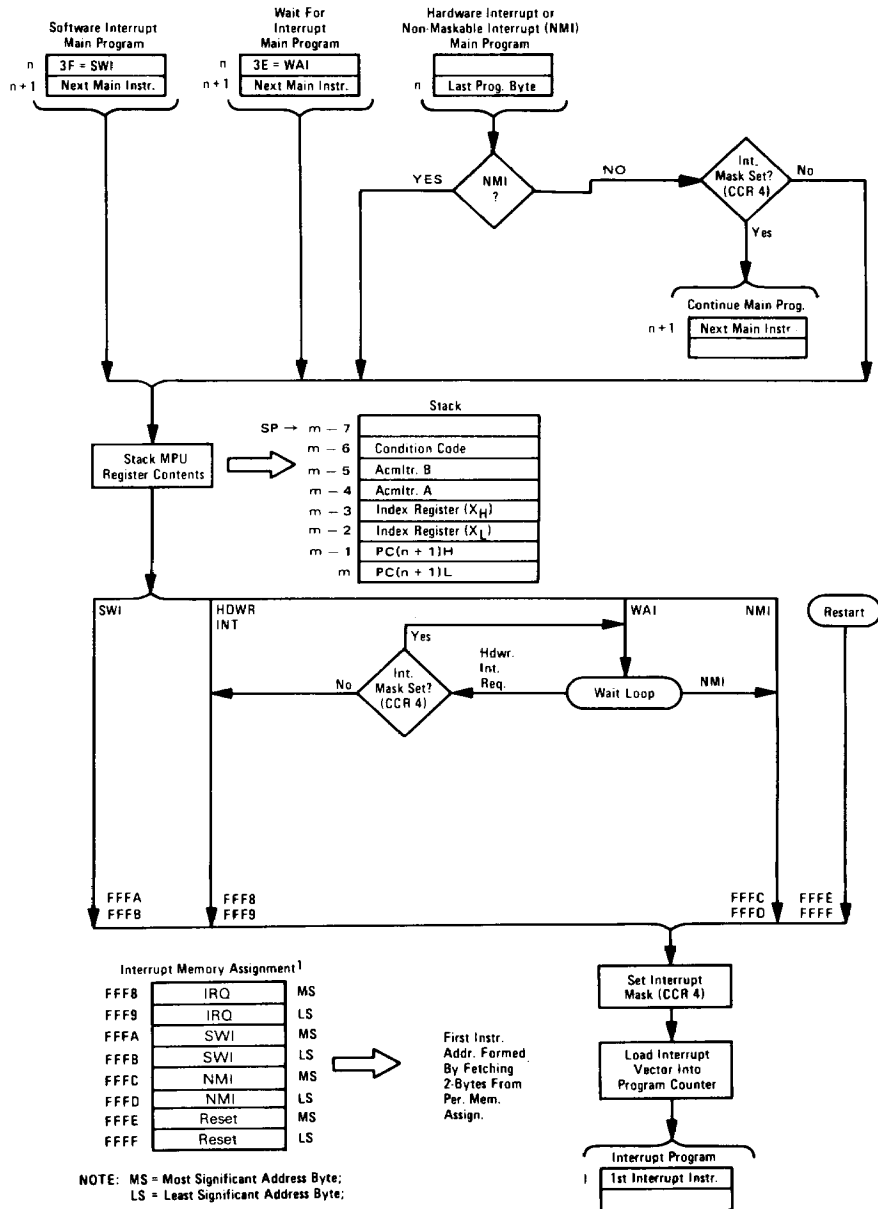


FIGURE 14 — MEMORY READY SYNCHRONIZATION

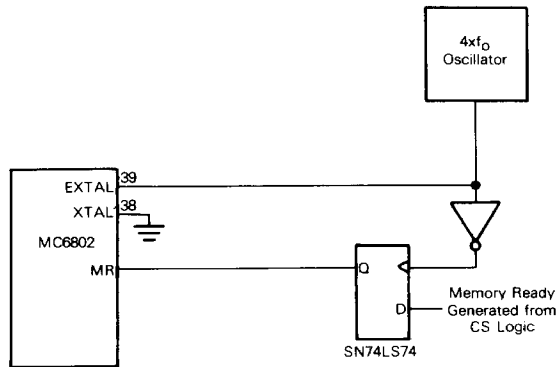
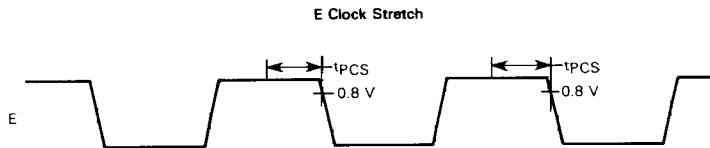
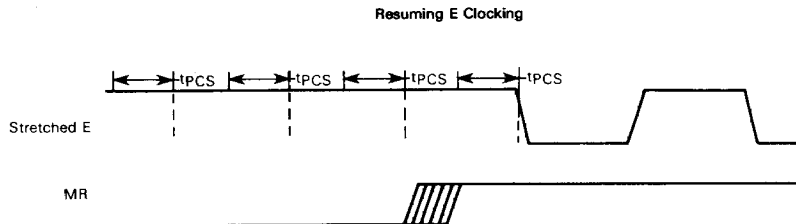


FIGURE 15 — MR NEGATIVE SETUP TIME REQUIREMENT



The E clock will be stretched at end of E high of the cycle during which MR negative meets the t_{PCS} setup time. The t_{PCS} setup time is referenced to the fall of E. If the t_{PCS} setup time is not met, E will be stretched at the end of the next E-high $\frac{1}{2}$ cycle. E will be stretched in integral multiples of $\frac{1}{2}$ cycles.



The E clock will resume normal operation at the end of the $\frac{1}{2}$ cycle during which MR assertion meets the t_{PCS} setup time. The t_{PCS} setup time is referenced to transitions of E were it not stretched. If t_{PCS} setup time is not met, E will fall at the second possible transition time after MR is asserted. There is no direct means of determining when the t_{PCS} references occur, unless the synchronizing circuit of Figure 14 is used.

RAM ENABLE (RE — MC6802 + MC6802NS ONLY)

A TTL-compatible RAM enable input controls the on-chip RAM of the MC6802. When placed in the high state, the on-chip memory is enabled to respond to the MPU controls. In the low state, RAM is disabled. This pin may also be utilized to disable reading and writing the on-chip RAM during a power-down situation. RAM Enable must be low three cycles before V_{CC} goes below 4.75 V during power-down. RAM enable must be tied low on the MC6808. RE should be tied to the correct high or low state if not used.

EXTAL AND XTAL

These inputs are used for the internal oscillator that may be crystal controlled. These connections are for a parallel resonant fundamental crystal (see Figure 12). (AT-cut.) A divide-by-four circuit has been added so a 4 MHz crystal may be used in lieu of a 1 MHz crystal for a more cost-effective system. An example of the crystal circuit layout is shown in Figure 13. Pin 39 may be driven externally by a TTL input signal four times the required E clock frequency. Pin 38 is to be grounded.

An RC network is not directly usable as a frequency source on pins 38 and 39. An RC network type TTL or CMOS oscillator will work well as long as the TTL or CMOS output drives the on-chip oscillator.

LC networks are not recommended to be used in place of the crystal.

If an external clock is used, it may not be halted for more than tp_{WOL} . The MC6802, MC6808 and MC6802NS are dynamic parts except for the internal RAM, and require the external clock to retain information.

MEMORY READY (MR)

MR is a TTL-compatible input signal controlling the stretching of E. Use of MR requires synchronization with the $4xf_0$ signal, as shown in Figure 14. When MR is high, E will be in normal operation. When MR is low, E will be stretched integral numbers of half periods, thus allowing interface to slow memories. Memory Ready timing is shown in Figure 15.

MR should be tied high (connected directly to V_{CC}) if not used. This is necessary to ensure proper operation of the part. A maximum stretch is t_{cyc} .

ENABLE (E)

This pin supplies the clock for the MPU and the rest of the system. This is a single-phase, TTL-compatible clock. This clock may be conditioned by a memory read signal. This is equivalent to ϕ_2 on the MC6800. This output is capable of driving one standard TTL load and 130 pF.

VCC STANDBY (MC6802 ONLY)

This pin supplies the dc voltage to the first 32 bytes of RAM as well as the RAM Enable (RE) control logic. Thus, retention of data in this portion of the RAM on a power-up, power-down, or standby condition is guaranteed. Maximum current drain at V_{SB} maximum is I_{SBB} . For the MC6802NS this pin must be connected to V_{CC} .

MPU INSTRUCTION SET

The instruction set has 72 different instructions. Included are binary and decimal arithmetic, logical, shift, rotate, load, store, conditional or unconditional branch, interrupt and stack manipulation instructions (Tables 2 through 6). The instruction set is the same as that for the MC6800.

MPU ADDRESSING MODES

There are seven address modes that can be used by a programmer, with the addressing mode a function of both the type of instruction and the coding within the instruction. A summary of the addressing modes for a particular instruction can be found in Table 7 along with the associated instruction execution time that is given in machine cycles. With a bus frequency of 1 MHz, these times would be microseconds.

ACCUMULATOR (ACCX) ADDRESSING

In accumulator only addressing, either accumulator A or accumulator B is specified. These are one-byte instructions.

IMMEDIATE ADDRESSING

In immediate addressing, the operand is contained in the second byte of the instruction except LDS and LDX which have the operand in the second and third bytes of the instruction. The MPU addresses this location when it fetches the immediate instruction for execution. These are two- or three-byte instructions.

DIRECT ADDRESSING

In direct addressing, the address of the operand is contained in the second byte of the instruction. Direct addressing allows the user to directly address the lowest 256 bytes in the machine, i.e., locations zero through 255. Enhanced execution times are achieved by storing data in these locations. In most configurations, it should be a random-access memory. These are two-byte instructions.

EXTENDED ADDRESSING

In extended addressing, the address contained in the second byte of the instruction is used as the higher eight bits of the address of the operand. The third byte of the instruction is used as the lower eight bits of the address for the operand. This is an absolute address in memory. These are three-byte instructions.

INDEXED ADDRESSING

In indexed addressing, the address contained in the second byte of the instruction is added to the index register's lowest eight bits in the MPU. The carry is then added to the higher order eight bits of the index register. This result is then used to address memory. The modified address is held in a temporary address register so there is no change to the index register. These are two-byte instructions.

IMPLIED ADDRESSING

In the implied addressing mode, the instruction gives the address (i.e., stack pointer, index register, etc.). These are one-byte instructions.

RELATIVE ADDRESSING

In relative addressing, the address contained in the second

byte of the instruction is added to the program counter's lowest eight bits plus two. The carry or borrow is then added to the high eight bits. This allows the user to address data within a range of -125 to $+129$ bytes of the present instruction. These are two-byte instructions.

TABLE 2 — MICROPROCESSOR INSTRUCTION SET — ALPHABETIC SEQUENCE

ABA	Add Accumulators	CLR	Clear	PUL	Pull Data
ADC	Add with Carry	CLV	Clear Overflow	ROL	Rotate Left
ADD	Add	CMP	Compare	ROR	Rotate Right
AND	Logical And	COM	Complement	RTI	Return from Interrupt
ASL	Arithmetic Shift Left	CPX	Compare Index Register	RTS	Return from Subroutine
ASR	Arithmetic Shift Right	DAA	Decimal Adjust	SBA	Subtract Accumulators
BCC	Branch if Carry Clear	DEC	Decrement	SBC	Subtract with Carry
BCS	Branch if Carry Set	DES	Decrement Stack Pointer	SEC	Set Carry
BEQ	Branch if Equal to Zero	DEX	Decrement Index Register	SEI	Set Interrupt Mask
BGE	Branch if Greater or Equal Zero	EOR	Exclusive OR	SEV	Set Overflow
BGT	Branch if Greater than Zero	INC	Increment	STA	Store Accumulator
BHI	Branch if Higher	INS	Increment Stack Pointer	STS	Store Stack Register
BIT	Bit Test	INX	Increment Index Register	STX	Store Index Register
BLE	Branch if Less or Equal	JMP	Jump	SUB	Subtract
BLS	Branch if Lower or Same	JSR	Jump to Subroutine	SWI	Software Interrupt
BLT	Branch if Less than Zero	LDA	Load Accumulator	TAB	Transfer Accumulators
BMI	Branch if Minus	LDS	Load Stack Pointer	TAP	Transfer Accumulators to Condition Code Reg.
BNE	Branch if Not Equal to Zero	LDX	Load Index Register	TBA	Transfer Accumulators
BPL	Branch if Plus	LSR	Logical Shift Right	TPA	Transfer Condition Code Reg. to Accumulator
BRA	Branch Always	NEG	Negate	TST	Test
BSR	Branch to Subroutine	NOP	No Operation	TSX	Transfer Stack Pointer to Index Register
BVC	Branch if Overflow Clear	ORA	Inclusive OR Accumulator	TXS	Transfer Index Register to Stack Pointer
BVS	Branch if Overflow Set	PSH	Push Data	WAI	Wait for Interrupt
CBA	Compare Accumulators				
CLC	Clear Carry				
CLI	Clear Interrupt Mask				

TABLE 3 — ACCUMULATOR AND MEMORY INSTRUCTIONS

ADDRESSING MODES										BOOLEAN/ARITHMETIC OPERATION										COND. CODE REG.																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
OPERATIONS	MNEMONIC	IMMED		DIRECT		INDEX		EXTND		IMPLIED		(All register labels refer to contents)										5	4	3	2	1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																								
		OP	~	OP	~	OP	~	OP	~	OP	~																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																								

LEGEND:

OP Operation Code (Hexadecimal).

~ Number of MPU Cycles.

= Number of Program Bytes.

+ Arithmetic Plus.

- Arithmetic Minus.

- Boolean AND.

Msp Contents of memory location pointed to by Stack Pointer.

+ Boolean Exclusive OR.

⊖ Boolean Exclusive OR.

M Complement of M.

- Transfer into.

0 Bit = Zero.

00 Byte = Zero.

CONDITION CODE SYMBOLS:

H Half carry from bit 3.

I Interrupt mask.

N Negative (sign bit).

Z Zero (byte).

V Overflow, 2's complement.

C Carry from bit 7.

R Reset Always.

S Set Always.

1 Test and set if true, cleared otherwise.

• Not Affected.

TABLE 4 — INDEX REGISTER AND STACK MANIPULATION INSTRUCTIONS

															COND. CODE REG.								
POINTER OPERATIONS	MNEMONIC	IMMED			DIRECT			INDEX			EXTND			IMPLIED			BOOLEAN/ARITHMETIC OPERATION	H	I	N	Z	V	C
		OP	~	=	OP	~	=	OP	~	=	OP	~	=	OP	~	=		•	•	•	•	•	•
Compare Index Reg	CPX	8C	3	3	9C	4	2	AC	6	2	BC	5	3	09	4	1	$X_H - M, X_L - (M + 1)$	•	•	⑦	•	•	⑧
Decrement Index Reg	DEX													04	4	1	$X - 1 - X$	•	•	•	•	•	•
Decrement Stack Ptr	DES													34	4	1	$SP - 1 - SP$	•	•	•	•	•	•
Increment Index Reg	INX													08	4	1	$X + 1 - X$	•	•	•	•	•	•
Increment Stack Ptr	INS													31	4	1	$SP + 1 - SP$	•	•	•	•	•	•
Load Index Reg	LDX	CE	3	3	DE	4	2	EE	6	2	FE	5	3				$M - X_H, (M + 1) - X_L$	•	•	⑨	•	•	R
Load Stack Ptr	LDS	8E	3	3	9E	4	2	AE	6	2	BE	5	3				$M - SP_H, (M + 1) - SP_L$	•	•	⑩	•	•	R
Store Index Reg	STX				0F	5	2	EF	7	2	FF	6	3				$X_H - M, X_L - (M + 1)$	•	•	•	•	•	⑪
Store Stack Ptr	STS				9F	5	2	AF	7	2	BF	6	3				$SP_H - M, SP_L - (M + 1)$	•	•	•	•	•	⑫
Idx Reg - Stack Ptr	TXS													35	4	1	$X - 1 - SP$	•	•	•	•	•	•
Stack Ptr - Idx Reg	TSX													30	4	1	$SP + 1 - X$	•	•	•	•	•	•

TABLE 5 — JUMP AND BRANCH INSTRUCTIONS

OPERATIONS	MNEMONIC													COND. CODE REG.					
		RELATIVE			INDEX			EXTND			IMPLIED			BRANCH TEST					
		OP	~	=	OP	~	=	OP	~	=	OP	~	=	H	I	N	Z	V	C
Branch Always	BRA	20	4	2										None	•	•	•	•	•
Branch If Carry Clear	BCC	24	4	2										$C = 0$	•	•	•	•	•
Branch If Carry Set	BCS	25	4	2										$C = 1$	•	•	•	•	•
Branch If = Zero	BEQ	27	4	2										$Z = 1$	•	•	•	•	•
Branch If ≥ Zero	BGE	2C	4	2										$N \oplus V = 0$	•	•	•	•	•
Branch If > Zero	BGT	2E	4	2										$Z + (N \oplus V) = 0$	•	•	•	•	•
Branch If Higher	BHI	22	4	2										$C + Z = 0$	•	•	•	•	•
Branch If ≤ Zero	BLE	2F	4	2										$Z + (N \oplus V) = 1$	•	•	•	•	•
Branch If Lower Or Same	BLS	23	4	2										$C + Z = 1$	•	•	•	•	•
Branch If < Zero	BLT	2D	4	2										$N \oplus V = 1$	•	•	•	•	•
Branch If Minus	BMI	28	4	2										$N = 1$	•	•	•	•	•
Branch If Not Equal Zero	BNE	26	4	2										$Z = 0$	•	•	•	•	•
Branch If Overflow Clear	BVC	28	4	2										$V = 0$	•	•	•	•	•
Branch If Overflow Set	BVS	29	4	2										$V = 1$	•	•	•	•	•
Branch If Plus	BPL	2A	4	2										$N = 0$	•	•	•	•	•
Branch To Subroutine	BSR	8D	8	2											•	•	•	•	•
Jump	JMP				6E	4	2	7E	3	3				See Special Operations (Figure 16)	•	•	•	•	•
Jump To Subroutine	JSR				AD	8	2	8D	9	3				Advances Prog. Cntr. Only	•	•	•	•	•
No Operation	NOP										01	2	1						
Return From Interrupt	RTI										3B	10	1						
Return From Subroutine	RTS										39	5	1						
Software Interrupt	SWI										3F	12	1						
Wait for Interrupt	WAI										3E	9	1						

FIGURE 16 — SPECIAL OPERATIONS

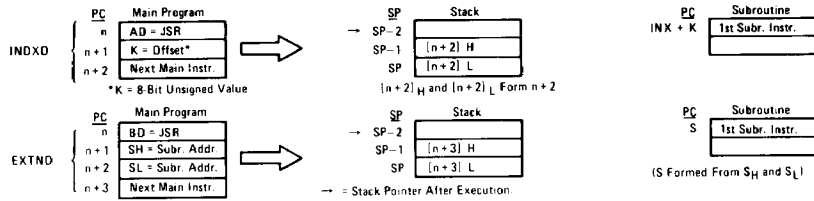
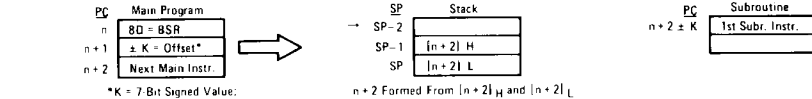
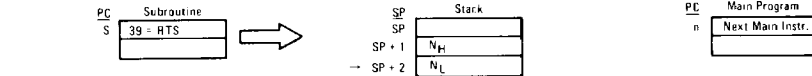
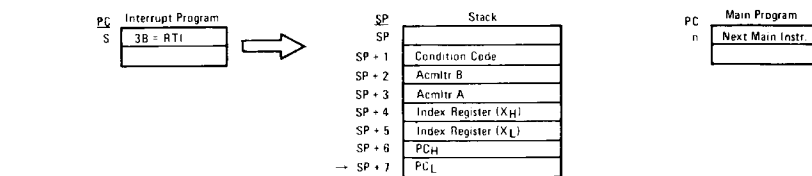
SPECIAL OPERATIONS**JSR, JUMP TO SUBROUTINE:****BSR, BRANCH TO SUBROUTINE:****JMP, JUMP:****RTS, RETURN FROM SUBROUTINE:****RTI, RETURN FROM INTERRUPT:**

TABLE 6 — CONDITION CODE REGISTER MANIPULATION INSTRUCTIONS

OPERATIONS	MNEMONIC	OP	~	=	BOOLEAN OPERATION	COND. CODE REG.							
						5	4	3	2	1	0		
Clear Carry	CLC	0C	2	1	0 ← C	•	•	•	•	•	•	R	
Clear Interrupt Mask	CLI	0E	2	1	0 ← I	•	R	•	•	•	•		
Clear Overflow	CLV	0A	2	1	0 ← V	•	•	•	•	•	R		
Set Carry	SEC	0D	2	1	1 ← C	•	•	•	•	•	•	S	
Set Interrupt Mask	SEI	0F	2	1	1 ← I	•	S	•	•	•	•		
Set Overflow	SEV	0B	2	1	1 ← V	•	•	•	•	•	S		
Accmtr A ← CCR	TAP	06	2	1	A ← CCR	12							
CCR ← Accmtr A	TPA	07	2	1	CCR ← A								

CONDITION CODE REGISTER NOTES: (Bit set if test is true and cleared otherwise)

- | | |
|--|---|
| 1 (Bit V) Test: Result = 10000000? | 7 (Bit N) Test: Sign bit of most significant (MS) byte = 1? |
| 2 (Bit C) Test: Result ≠ 00000000? | 8 (Bit V) Test: 2's complement overflow from subtraction of MS bytes? |
| 3 (Bit C) Test: Decimal value of most significant BCD Character greater than nine?
(Not cleared if previously set.) | 9 (Bit N) Test: Result less than zero? (Bit 15 = 1) |
| 4 (Bit VI) Test: Operand = 10000000 prior to execution? | 10 (All) Load Condition Code Register from Stack. (See Special Operations) |
| 5 (Bit VI) Test: Operand = 01111111 prior to execution? | 11 (Bit I) Set when interrupt occurs. If previously set, a Non-Maskable Interrupt is required to exit the wait state. |
| 6 (Bit VI) Test: Set equal to result of NDC after shift has occurred. | 12 (All) Set according to the contents of Accumulator A. |

TABLE 7 — INSTRUCTION ADDRESSING MODES AND ASSOCIATED EXECUTION TIMES
(Times in Machine Cycle)

	(Dual Operand)	ACCX	Immediate	Direct	Extended	Indexed	Implied	Relative		(Dual Operand)	ACCX	Immediate	Direct	Extended	Indexed	Implied
ABA		•	•	•	•	•	•	•	INC	•	•	•	•	•	•	•
ADC	x	•	•	•	•	•	•	•	INS	•	•	•	•	•	•	•
ADD	x	•	•	•	•	•	•	•	INX	•	•	•	•	•	•	•
AND	x	•	•	•	•	•	•	•	JMP	•	•	•	•	•	•	•
ASL	•	•	•	•	•	•	•	•	JSR	•	•	•	•	•	•	•
ASR	•	•	•	•	•	•	•	•	LDA	x	•	•	•	•	•	•
BCC	•	•	•	•	•	•	•	•	LDS	•	•	•	•	•	•	•
BCS	•	•	•	•	•	•	•	•	LDX	•	•	•	•	•	•	•
BEA	•	•	•	•	•	•	•	•	LSR	•	•	•	•	•	•	•
BGE	•	•	•	•	•	•	•	•	NEG	•	•	•	•	•	•	•
BGT	•	•	•	•	•	•	•	•	NOP	•	•	•	•	•	•	•
BHI	•	•	•	•	•	•	•	•	ORA	x	•	•	•	•	•	•
BIT	x	•	•	•	•	•	•	•	PSH	•	•	•	•	•	•	•
BLE	•	•	•	•	•	•	•	•	PUL	•	•	•	•	•	•	•
BLS	•	•	•	•	•	•	•	•	ROL	•	•	•	•	•	•	•
BLT	•	•	•	•	•	•	•	•	ROR	•	•	•	•	•	•	•
BMI	•	•	•	•	•	•	•	•	RTI	•	•	•	•	•	•	•
BNE	•	•	•	•	•	•	•	•	RTS	•	•	•	•	•	•	•
BPL	•	•	•	•	•	•	•	•	SBA	•	•	•	•	•	•	•
BRA	•	•	•	•	•	•	•	•	SBC	x	•	•	•	•	•	•
BSR	•	•	•	•	•	•	•	•	SEC	•	•	•	•	•	•	•
BVC	•	•	•	•	•	•	•	•	SEI	•	•	•	•	•	•	•
BVS	•	•	•	•	•	•	•	•	SEV	•	•	•	•	•	•	•
CBA	•	•	•	•	•	•	•	•	STA	x	•	•	•	•	•	•
CLC	•	•	•	•	•	•	•	•	STS	•	•	•	•	•	•	•
CLI	•	•	•	•	•	•	•	•	STX	•	•	•	•	•	•	•
CLR	•	•	•	•	•	•	•	•	SUB	x	•	•	•	•	•	•
CLV	•	•	•	•	•	•	•	•	SWI	•	•	•	•	•	•	•
CMP	x	•	•	•	•	•	•	•	TAB	•	•	•	•	•	•	•
COM	•	•	•	•	•	•	•	•	TAP	•	•	•	•	•	•	•
CPX	•	•	•	•	•	•	•	•	TBA	•	•	•	•	•	•	•
DAA	•	•	•	•	•	•	•	•	TPA	•	•	•	•	•	•	•
DEC	•	•	•	•	•	•	•	•	TST	•	•	•	•	•	•	•
DES	•	•	•	•	•	•	•	•	TSX	•	•	•	•	•	•	•
DEX	•	•	•	•	•	•	•	•	TSX	•	•	•	•	•	•	•
EOR	x	•	•	•	•	•	•	•	WAI	•	•	•	•	•	•	•

NOTE Interrupt time is 12 cycles from the end of the instruction being executed, except following a WAI instruction. Then it is 4 cycles.

SUMMARY OF CYCLE-BY-CYCLE OPERATION

Table 8 provides a detailed description of the information present on the address bus, data bus, valid memory address line (VMA), and the read/write line (R/W) during each cycle for each instruction.

This information is useful in comparing actual with expected results during debug of both software and hardware

as the control program is executed. The information is categorized in groups according to addressing modes and number of cycles per instruction. (In general, instructions with the same addressing mode and number of cycles execute in the same manner; exceptions are indicated in the table.)

TABLE 8 — OPERATIONS SUMMARY

Address Mode and Instructions	Cycles	Cycle #	VMA Line	Address Bus	R/W Line	Data Bus
IMMEDIATE						
ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	2	1 2	1 1	Op Code Address Op Code Address + 1	1 1	Op Code Operand Data
CPX LDS LDX	3	1 2 3	1 1 1	Op Code Address Op Code Address + 1 Op Code Address + 2	1 1 1	Op Code Operand Data (High Order Byte) Operand Data (Low Order Byte)
DIRECT						
ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	3	1 2 3	1 1 1	Op Code Address Op Code Address + 1 Address of Operand	1 1 1	Op Code Address of Operand Operand Data
CPX LDS LDX	4	1 2 3 4	1 1 1 1	Op Code Address Op Code Address + 1 Address of Operand Operand Address + 1	1 1 1 1	Op Code Address of Operand Operand Data (High Order Byte) Operand Data (Low Order Byte)
STA	4	1 2 3 4	1 1 0 1	Op Code Address Op Code Address + 1 Destination Address Destination Address	1 1 1 0	Op Code Destination Address Irrelevant Data (Note 1) Data from Accumulator
STS STX	5	1 2 3 4 5	1 1 0 1 1	Op Code Address Op Code Address + 1 Address of Operand Address of Operand Address of Operand + 1	1 1 1 0 0	Op Code Address of Operand Irrelevant Data (Note 1) Register Data (High Order Byte) Register Data (Low Order Byte)
INDEXED						
JMP	4	1 2 3 4	1 1 0 0	Op Code Address Op Code Address + 1 Index Register Index Register Plus Offset (w/o Carry)	1 1 1 1	Op Code Offset Irrelevant Data (Note 1) Irrelevant Data (Note 1)
ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	5	1 2 3 4 5	1 1 0 0 1	Op Code Address Op Code Address + 1 Index Register Index Register Plus Offset (w/o Carry) Index Register Plus Offset	1 1 1 1 1	Op Code Offset Irrelevant Data (Note 1) Irrelevant Data (Note 1) Operand Data
CPX LDS LDX	6	1 2 3 4 5 6	1 1 0 0 1 1	Op Code Address Op Code Address + 1 Index Register Index Register Plus Offset (w/o Carry) Index Register Plus Offset Index Register Plus Offset + 1	1 1 1 1 1 1	Op Code Offset Irrelevant Data (Note 1) Irrelevant Data (Note 1) Operand Data (High Order Byte) Operand Data (Low Order Byte)

TABLE 8 — OPERATIONS SUMMARY (CONTINUED)

Address Mode and Instructions	Cycles	Cycle #	VMA Line	Address Bus	R/W Line	Data Bus
INDEXED (Continued)						
STA	6	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Offset
		3	0	Index Register	1	Irrelevant Data (Note 1)
		4	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (Note 1)
		5	0	Index Register Plus Offset	1	Irrelevant Data (Note 1)
		6	1	Index Register Plus Offset	0	Operand Data
ASL LSR ASR NEG CLR ROL COM ROR DEC TST INC	7	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Offset
		3	0	Index Register	1	Irrelevant Data (Note 1)
		4	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (Note 1)
		5	1	Index Register Plus Offset	1	Current Operand Data
		6	0	Index Register Plus Offset	1	Irrelevant Data (Note 1)
		7	1/0 (Note 3)	Index Register Plus Offset	0	New Operand Data (Note 3)
STS STX	7	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Offset
		3	0	Index Register	1	Irrelevant Data (Note 1)
		4	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (Note 1)
		5	0	Index Register Plus Offset	1	Irrelevant Data (Note 1)
		6	1	Index Register Plus Offset	0	Operand Data (High Order Byte)
		7	1	Index Register Plus Offset + 1	0	Operand Data (Low Order Byte)
JSR	8	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Offset
		3	0	Index Register	1	Irrelevant Data (Note 1)
		4	1	Stack Pointer	0	Return Address (Low Order Byte)
		5	1	Stack Pointer - 1	0	Return Address (High Order Byte)
		6	0	Stack Pointer - 2	1	Irrelevant Data (Note 1)
		7	0	Index Register	1	Irrelevant Data (Note 1)
		8	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (Note 1)
EXTENDED						
JMP	3	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Jump Address (High Order Byte)
		3	1	Op Code Address + 2	1	Jump Address (Low Order Byte)
ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Operand (High Order Byte)
		3	1	Op Code Address + 2	1	Address of Operand (Low Order Byte)
		4	1	Address of Operand	1	Operand Data
CPX LDS LDX	5	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Operand (High Order Byte)
		3	1	Op Code Address + 2	1	Address of Operand (Low Order Byte)
		4	1	Address of Operand	1	Operand Data (High Order Byte)
		5	1	Address of Operand + 1	1	Operand Data (Low Order Byte)
STA A STA B	5	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Destination Address (High Order Byte)
		3	1	Op Code Address + 2	1	Destination Address (Low Order Byte)
		4	0	Operand Destination Address	1	Irrelevant Data (Note 1)
		5	1	Operand Destination Address	0	Data from Accumulator
ASL LSR ASR NEG CLR ROL COM ROR DEC TST INC	6	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Operand (High Order Byte)
		3	1	Op Code Address + 2	1	Address of Operand (Low Order Byte)
		4	1	Address of Operand	1	Current Operand Data
		5	0	Address of Operand	1	Irrelevant Data (Note 1)
		6	1/0 (Note 3)	Address of Operand	0	New Operand Data (Note 3)

TABLE 8 — OPERATIONS SUMMARY (CONTINUED)

Address Mode and Instructions	Cycles	Cycle #	VMA Line	Address Bus	R/W Line	Data Bus	
EXTENDED (Continued)							
STS STX	6	1	1	Op Code Address	1	Op Code	
		2	1	Op Code Address + 1	1	Address of Operand (High Order Byte)	
		3	1	Op Code Address + 2	1	Address of Operand (Low Order Byte)	
		4	0	Address of Operand	1	Irrelevant Data (Note 1)	
		5	1	Address of Operand	0	Operand Data (High Order Byte)	
		6	1	Address of Operand + 1	0	Operand Data (Low Order Byte)	
JSR	9	1	1	Op Code Address	1	Op Code	
		2	1	Op Code Address + 1	1	Address of Subroutine (High Order Byte)	
		3	1	Op Code Address + 2	1	Address of Subroutine (Low Order Byte)	
		4	1	Subroutine Starting Address	1	Op Code of Next Instruction	
		5	1	Stack Pointer	0	Return Address (Low Order Byte)	
		6	1	Stack Pointer - 1	0	Return Address (High Order Byte)	
		7	0	Stack Pointer - 2	1	Irrelevant Data (Note 1)	
		8	0	Op Code Address + 2	1	Irrelevant Data (Note 1)	
		9	1	Op Code Address + 2	1	Address of Subroutine (Low Order Byte)	
INHERENT							
ABA DAA SEC ASL DEC SEI ASR INC SEV CBA LSR TAB CLC NEG TAP CLI NOP TBA CLR ROL TPA CLV ROR TST COM SBA	2	1	1	Op Code Address	1	Op Code	
		2	1	Op Code Address + 1	1	Op Code of Next Instruction	
	DES DEX INS INX	4	1	1	Op Code Address	1	Op Code
			2	1	Op Code Address + 1	1	Op Code of Next Instruction
			3	0	Previous Register Contents	1	Irrelevant Data (Note 1)
			4	0	New Register Contents	1	Irrelevant Data (Note 1)
	PSH	4	1	1	Op Code Address	1	Op Code
			2	1	Op Code Address + 1	1	Op Code of Next Instruction
			3	1	Stack Pointer	0	Accumulator Data
			4	0	Stack Pointer - 1	1	Accumulator Data
PUL	4	1	1	Op Code Address	1	Op Code	
		2	1	Op Code Address + 1	1	Op Code of Next Instruction	
		3	0	Stack Pointer	1	Irrelevant Data (Note 1)	
		4	1	Stack Pointer + 1	1	Operand Data from Stack	
TSX	4	1	1	Op Code Address	1	Op Code	
		2	1	Op Code Address + 1	1	Op Code of Next Instruction	
		3	0	Stack Pointer	1	Irrelevant Data (Note 1)	
		4	0	New Index Register	1	Irrelevant Data (Note 1)	
TXS	4	1	1	Op Code Address	1	Op Code	
		2	1	Op Code Address + 1	1	Op Code of Next Instruction	
		3	0	Index Register	1	Irrelevant Data	
		4	0	New Stack Pointer	1	Irrelevant Data	
RTS	5	1	1	Op Code Address	1	Op Code	
		2	1	Op Code Address + 1	1	Irrelevant Data (Note 2)	
		3	0	Stack Pointer	1	Irrelevant Data (Note 1)	
		4	1	Stack Pointer + 1	1	Address of Next Instruction (High Order Byte)	
		5	1	Stack Pointer + 2	1	Address of Next Instruction (Low Order Byte)	

TABLE 8 — OPERATIONS SUMMARY (CONCLUDED)

Address Mode and Instructions	Cycles	Cycle #	VMA Line	Address Bus	R/W Line	Data Bus
INHERENT (Continued)						
WAI	9	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Op Code of Next Instruction
		3	1	Stack Pointer	0	Return Address (Low Order Byte)
		4	1	Stack Pointer - 1	0	Return Address (High Order Byte)
		5	1	Stack Pointer - 2	0	Index Register (Low Order Byte)
		6	1	Stack Pointer - 3	0	Index Register (High Order Byte)
		7	1	Stack Pointer - 4	0	Contents of Accumulator A
		8	1	Stack Pointer - 5	0	Contents of Accumulator B
		9	1	Stack Pointer - 6	1	Contents of Cond. Code Register
RTI	10	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Irrelevant Data (Note 2)
		3	0	Stack Pointer	1	Irrelevant Data (Note 1)
		4	1	Stack Pointer + 1	1	Contents of Cond. Code Register from Stack
		5	1	Stack Pointer + 2	1	Contents of Accumulator B from Stack
		6	1	Stack Pointer + 3	1	Contents of Accumulator A from Stack
		7	1	Stack Pointer + 4	1	Index Register from Stack (High Order Byte)
		8	1	Stack Pointer + 5	1	Index Register from Stack (Low Order Byte)
		9	1	Stack Pointer + 6	1	Next Instruction Address from Stack (High Order Byte)
		10	1	Stack Pointer + 7	1	Next Instruction Address from Stack (Low Order Byte)
SWI	12	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Irrelevant Data (Note 1)
		3	1	Stack Pointer	0	Return Address (Low Order Byte)
		4	1	Stack Pointer - 1	0	Return Address (High Order Byte)
		5	1	Stack Pointer - 2	0	Index Register (Low Order Byte)
		6	1	Stack Pointer - 3	0	Index Register (High Order Byte)
		7	1	Stack Pointer - 4	0	Contents of Accumulator A
		8	1	Stack Pointer - 5	0	Contents of Accumulator B
		9	1	Stack Pointer - 6	0	Contents of Cond. Code Register
		10	0	Stack Pointer - 7	1	Irrelevant Data (Note 1)
		11	1	Vector Address FFFA (Hex)	1	Address of Subroutine (High Order Byte)
		12	1	Vector Address FFFB (Hex)	1	Address of Subroutine (Low Order Byte)
RELATIVE						
BCC BHI BNE BCS BLE BPL BEQ BLS BRA BGE BLT BVC BGT BMI BVS	4	1	1	Op Code Address	1	Op Code
2		1	Op Code Address + 1	1	Branch Offset	
3		0	Op Code Address + 2	1	Irrelevant Data (Note 1)	
4		0	Branch Address	1	Irrelevant Data (Note 1)	
BSR	8	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Branch Offset
		3	0	Return Address of Main Program	1	Irrelevant Data (Note 1)
		4	1	Stack Pointer	0	Return Address (Low Order Byte)
		5	1	Stack Pointer - 1	0	Return Address (High Order Byte)
		6	0	Stack Pointer - 2	1	Irrelevant Data (Note 1)
		7	0	Return Address of Main Program	1	Irrelevant Data (Note 1)
		8	0	Subroutine Address (Note 4)	1	Irrelevant Data (Note 1)

NOTES:

1. If device which is addressed during this cycle uses VMA, then the Data Bus will go to the high-impedance three-state condition. Depending on bus capacitance, data from the previous cycle may be retained on the Data Bus.
2. Data is ignored by the MPU.
3. For TST, VMA = 0 and Operand data does not change.
4. MS Byte of Address Bus = MS Byte of Address of BSR instruction and LS Byte of Address Bus = LS Byte of Sub-Routine Address.