1.4.1 The MCS6501

1.4.1.1 Introduction

The members of the MCS650X microprocessor family contain very similar internal architectures. A block diagram of this architecture is shown in Figure 1.11. This section begins with an analysis of this block diagram, discussing the function of the various registers, data paths, etc. A detailed discussion of the operation of the various pins on the chip follows.

The internal organization of the processor can be split into two sections. In general, the instructions obtained from program memory are executed by implementing a series of data transfers in one section of the chip (register section). The control lines which actually cause the data transfers to take place are generated in the other section (control section). Instructions enter the processor on the data bus, are latched into the instruction register, and are then decoded along with timing signals to generate the register control signals.

The timing control unit keeps track of the specific cycle being executed. This unit is set to "T0" for each instruction fetch cycle and is advanced at the beginning of each Phase One clock pulse. Each instruction starts in T0 and goes to T1, T2, T3, etc. for as many cycles as are required to complete execution of the instruction. Each data transfer, etc., which takes place in the register section is caused by decoding the contents of both the instruction register and the timing counter.

Additional control lines which affect the execution of the instructions are derived from the Interrupt logic and from the Processor Status register. The Interrupt logic controls the processor interface to the interrupt inputs to assure proper timing, enabling, sequencing, etc. which the processor recognizes and services.

The Processor Status register contains a set of latches which serve to control certain aspects of the processor operation, to indicate the results of processor arithmetic and logic operations, and to indicate the status of data either generated by the processor or transferred into the processor from outside.

Since the real work of the processor is carried on in the register section of the chip, a detailed study will be made of this section. The components are:

* Data Bus Buffers
* Input Data Latch (DL)
* Program Counter (PCL, PCH)
* Accumulator (A)
* Arithmetic Logic Unit (ALU)
* Stack Pointer (S)
* Index Registers (X, Y)
* Address Bus Latches (ABL, ABH)
* Processor Status Register (P)

At 1 MHz, the data which comes into the processor from the program memory, the data memory, or from peripheral devices, appears on the data bus during the last 100 nanoseconds of Phase Two. No attempt is made to actually operate on the data during this short period. Instead, it is simply transferred into the input data latch for use during the next cycle. The data latch serves to trap the data on the data bus during each Phase Two pulse. It can then be transferred onto one of the internal busses and from there into one of the internal registers. For example, data being transferred from memory into the accumulator (A) will be placed on the internal data bus and will then be transferred from the internal data bus into the accumulator. If an arithmetic or logic operation is to be performed using the data from memory and the contents of the accumulator, data in the input data latch will be transferred onto the internal data bus as before. From there it will be transferred into the ALU. At the same time the contents of the accumulator will be transferred onto a bus in the register section and from there into the second input to the ALU. The results of the arithmetic or logic operation will be transferred back to the accumulator on the next cycle by transferring first onto the bus and then into the accumulator. All of these data transfers take place during the Phase One clock pulse.

The program counter (PCL, PCH) provides the addresses which step the processor through sequential instructions in the program. Each time the processor fetches an instruction from program memory, the contents of PCL is placed on the low order eight bits of the address bus and the contents of PCH is placed on the high order eight bits. This counter is incremented each time an instruction or data is fetched from program memory.

The accumulator is a general purpose 8-bit register which stores the results of most arithmetic and logic operations. In addition, the accumulator usually contains one of the two data words used in these operations.

All logic and arithmetic operations take place in the ALU. This includes incrementing and decrementing of internal registers (except PCL and PCH). However, the ALU cannot store data for more than one cycle. If data is placed on the inputs to the ALU at the beginning of one cycle, the result is always gated into one of the storage registers or to external memory during the next cycle. Each bit of the ALU has two inputs. These inputs can be tied to various internal busses or to a logic zero; the ALU then generates the SUM, AND, OR, etc. function using the data on the two inputs.

The stack pointer (S) and the two index registers (X and Y) each consist of 8 simple latches. These registers store data which is to be used in calculating addresses in data memory. The specific operation of each of these is discussed in detail in the Programming Manual.

The address bus buffers (ABL, ABH) consist of a set of latches and TTL compatible drivers. These latches store the addresses which are used in accessing the peripheral devices (ROM, RAM, and I/O).

1.4.1.2 The MCS6501 Pinouts

Figure 1.12 shows a diagram of the MCS6501 microprocessor with the various pins designated. These pins and their use in microcomputer systems are discussed separately below.

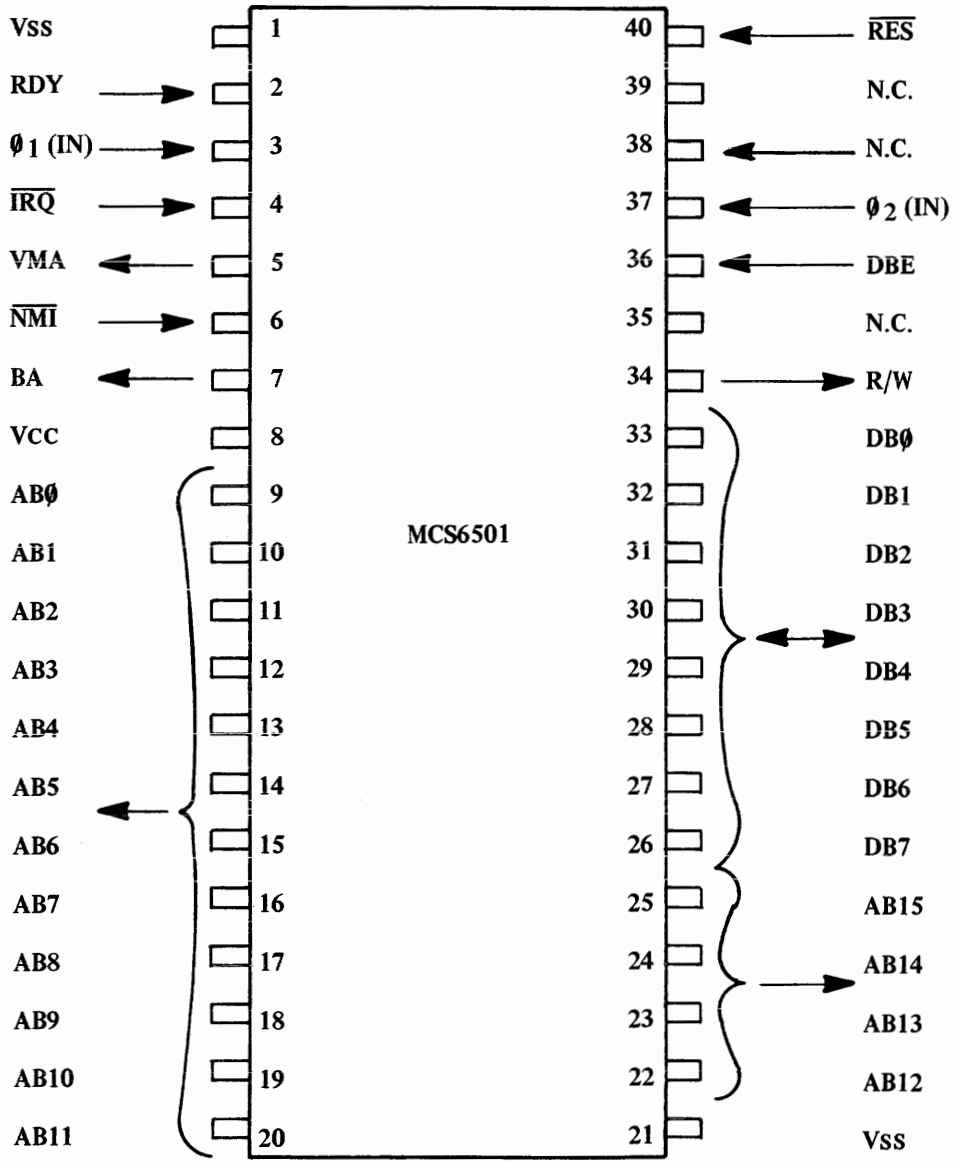
1.4.1.2.1 Vcc, Vss--Supply Lines

The Vcc and Vss pins are the only power supply connections to the chip. The supply voltage on pin 8 is +5.0 V DC ± 5%. The absolute limit on the Vcc input is +7.0 V DC.

1.4.1.2.2 AB00-AB15--Address Bus

The address bus buffers on the MCS650X family of microprocessors are push/pull type drivers capable of driving at least 130 pf and 1 standard TTL load.

The address bus will always contain known data as detailed in Appendix A. The addressing technique involves putting an address on the address bus which is known to be either in program sequence, on the same



N.C. = NO CONNECTION

* VMA IS CONNECTED INTERNALLY TO Vcc. THE VMA SIGNAL IS NOT REQUIRED ON THE MCS6501 AS ON THE MC6800, SINCE THE MCS6501 ALWAYS PUTS OUT KNOWN ADDRESSES ON THE ADDRESS BUS.

MCS6501 Pinout Designations

FIGURE 1.12

page in program memory or at a known point in RAM. A brief study of Appendix A will acquaint the designer with the detailed operation of this bus.

The various processors differ somewhat in the number of address lines provided. In particular, the MCS6504 provides thirteen address lines (AB00 - AB12) and the MCS6503 and MCS6505 provide twelve (AB00 - AB11). As a result, the MCS6504 can address 8,192 bytes of memory and the MCS6503 and MCS6505 can address 4,096 bytes. This total address space should prove to be more than sufficient for the small, cost-sensitive systems where these devices should find their greatest application.

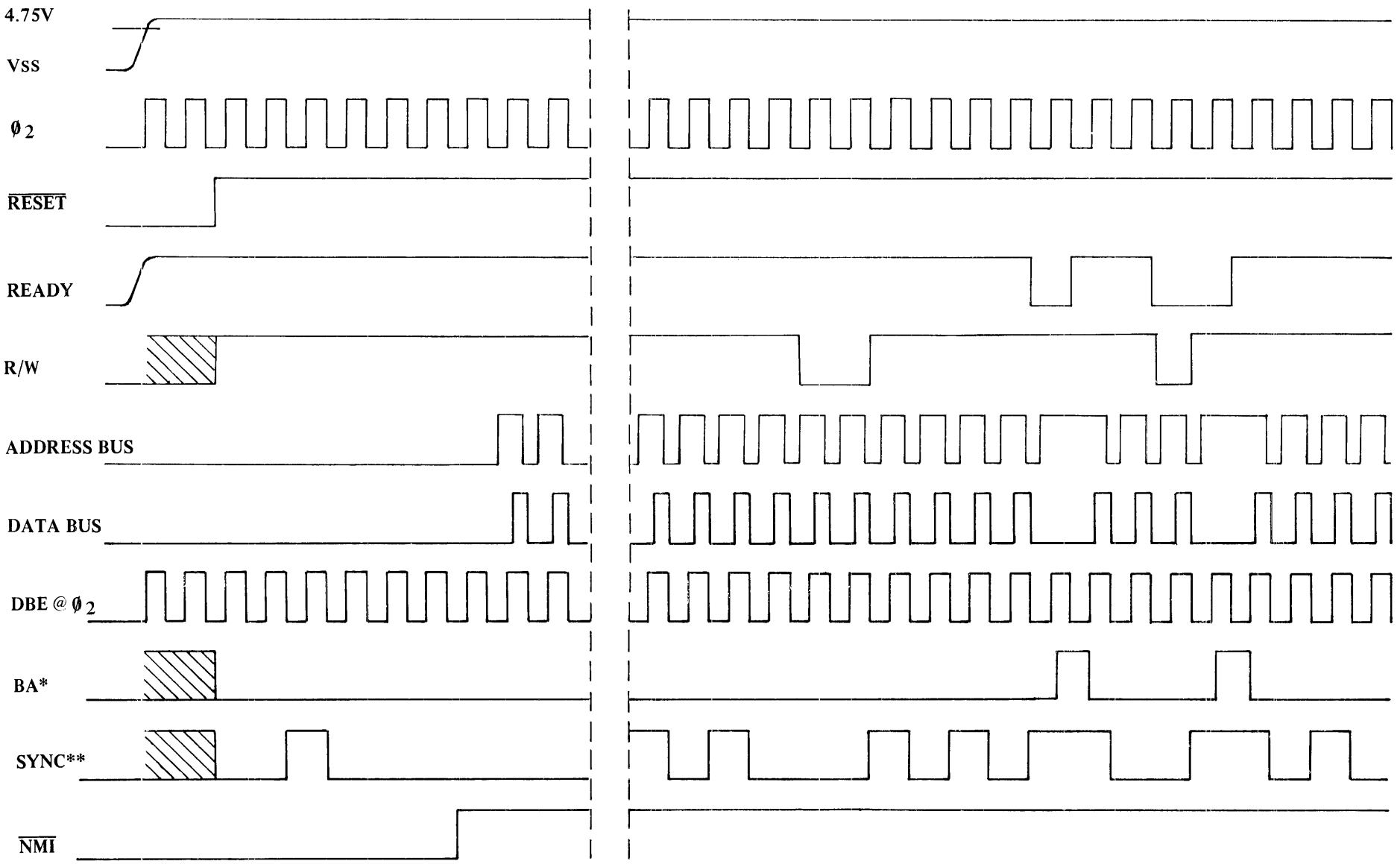
The specific timing of the address bus is exactly the same for all the processors. The address is valid 300 ns (at 1 MHz clock rate) into the Ø1 clock pulse and stays stable until the next Ø1 pulse. This specification will only change for processors which are specified to operate at a higher clock rate. Figure 1.13 details the relation of address bus to other critical signals.

Because of the reduced number of address lines on the 28-pin processors, it is possible to write a program which attempts to access nonexistent memory address space, i.e., the address bits 13, 14, or 15 set to logic "1." These upper address bits in the program will be ignored and the program will drop into existing address space. This assumes proper memory management when using devices of large addressing capability such that the addressed memory space will fit within the constraints of a device with smaller available memory addressing capability.

1.4.1.2.3 DB0-DB7--Data Bus

The processor data bus is exactly the same for the processors currently available and for the software-compatible processors which will be introduced in the near future. All instructions and data transfers between the processor and memory take place on these lines. The buffers driving the data bus lines have full "three-state" capability. This is necessitated by the fact that the lines are bi-directional.

Each data bus pin is connected to an input and an output buffer, with the output buffer remaining in the "floating" condition except when the processor is transferring data into or out of one of the support chips. All inter-chip data transfers take place during the Phase Two clock pulse. During Phase One the entire data bus is "floating."



*BA IS AVAILABLE ON MCS6501 ONLY
**SYNC IS AVAILABLE ON MCS6502 ONLY

MCS650X System Timing Diagram

FIGURE 1.13

The data bus buffer is a push/pull driver capable of driving 130 pf and 1 standard TTL load at the rated speed. At a 1 MHz clock rate, the data on the data bus must be stable 100 ns before the end of Phase Two. This is true for transfers in either direction. Figure 1.13 details the relationship of the data bus to other signals

1.4.1.2.4 R/W--Read/Write

The Read/Write line allows the processor to control the direction of data transfers between the processor and the support chips. This line is high except when the processor is writing to memory or to a peripheral interface device.

All transitions on this line occur during the Phase One clock pulse (concurrent with the address lines). This allows complete control of the data transition which takes place during the Phase Two clock pulse.

The R/W buffer is similar to the address buffers. They are capable of driving 130 pf and one standard TTL load at the rated speed. Again, Figure 1.13 details the relative timing of the R/W line.

1.4.1.2.5 DBE--Data Bus Enable

On the MCS6501, a data bus enable signal is provided to allow external enabling of the data bus. This line is connected directly to the Phase Two input clock signal for any normally operating system and is detailed in Figure 1.13.

The DBE signal affects only the data bus buffers. It does not affect processor timing and has no effect on the address of the R/W lines.

This input is provided primarily for use in systems which use non-family devices for either the memory or the peripheral interface functions. In particular, it allows the data bus to be enabled for a period longer than the Phase Two clock pulse for systems requiring greater processor hold time on the data bus. This application is covered in greater detail in Chapter 2.

1.4.1.2.6 VMA--Valid Memory Address

As mentioned above, the MCS650X family of microprocessors always puts known addresses on the address bus and, as a result, does not require a VMA signal. However, to remain pin-compatible with the MC6800, the VMA pin

is connected internally to the Vcc power supply. This assures operation in systems in which VMA is part of the chip-select function. This pin is not available on the 28-pin processors.

1.4.1.2.7 BA--Bus Available

The bus available signal is provided on the MCS6501 to signal to a DMA controller, etc. that the processor is stopped and that the data and address busses can be used for other than processor program execution.

This operation is similar to that of the MC6800 bus available signal except that much less time is required to stop the MCS6501 since the MC6800 requires completion of the current instruction before stopping. If no write operation takes place during the cycle in which the RDY signal goes low, the BA will go high (> 2.4V) during Phase Two of the same cycle. In general, BA will go high during the first Phase Two pulse during which the R/W line is high. For the current processors, the maximum time is 3-1/2 cycles.

1.4.1.2.8 RDY--Ready

The RDY input delays execution of any cycle during which the RDY line is pulled low. This line should change during the Phase One clock pulse. This change is then recognized during the next Phase Two pulse to enable or disable the execution of the current internal machine cycle. This execution normally occurs during the next Phase One clock; timing is shown in Figure 1.13.

The primary purpose of the RDY line is to delay execution of a program fetch cycle until data is available from memory. This has direct application in prototype systems employing light-erasable PROMs or EAROMs. Both of these devices have relatively slow access times and require implementation of the RDY function if the processor is to operate at full speed. Without the RDY function a reduction in the frequency of the system clock would be necessary.

The RDY function will not stop the processor in a cycle in which a WRITE operation is being performed. If the RDY line goes from high to low during a WRITE cycle the processor will execute that cycle and will then stop in the next READ cycle (R/W = 1).

1.4.1.2.9 $\overline{\text{NMI}}$--Non-Maskable Interrupt

The $\overline{\text{NMI}}$ input, when in the interrupted state, always interrupts the processor after it completes the instruction currently being executed. This interrupt is not "maskable," i.e., there is no way for the processor to prevent recognition of the interrupt.

The $\overline{\text{NMI}}$ input responds to a negative transition. To interrupt the processor, the $\overline{\text{NMI}}$ input must go from high (> +2.4V) to low (< +0.4V). It can then stay low for an indefinite period without affecting the processor operation and without another interrupt. The processor will not detect another interrupt until this line goes high and then back to low. The $\overline{\text{NMI}}$ signal must be low for at least two clock cycles for the interrupt to be recognized, whereupon new program count vectors are fetched.

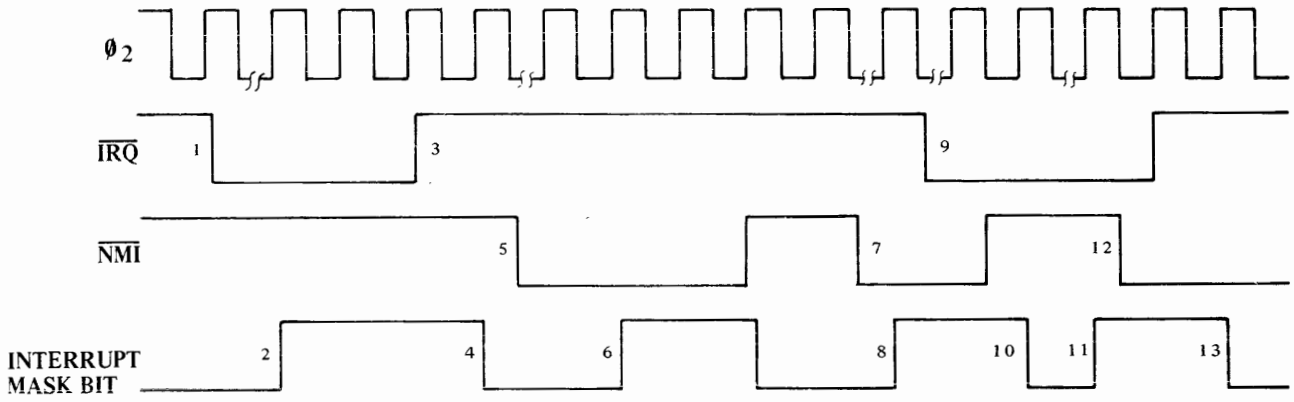
1.4.1.2.10 $\overline{\text{IRQ}}$--Interrupt Request

The interrupt request ($\overline{\text{IRQ}}$) responds in much the same manner as $\overline{\text{NMI}}$. However, this function can be enabled or disabled by the interrupt inhibit bit in the processor status register. As long as the I flag (interrupt inhibit flag) is a logic 1, the signal on the $\overline{\text{IRQ}}$ pin will not affect the processor.

The $\overline{\text{IRQ}}$ pin is not edge-sensitive. Instead, the processor will be interrupted as long as the I flag is a logic"0" and the signal on the $\overline{\text{IRQ}}$ input is at GND. Because of this, the $\overline{\text{IRQ}}$ signal must be held low until it is recognized, i.e., until the processor completes the instruction currently being executed. If I is set when $\overline{\text{IRQ}}$ goes low, the interrupt will not be recognized until I is cleared through software control. To assure that the processor will not recognize the interrupt more than once, the I flag is set automatically during the last cycle before the processor begins executing the interrupt software, beginning with the fetch of program count.

The final requirement is that the interrupt input must be cleared before the I flag is reset. If there is more than one active interrupt driving these two lines (OR'ed together), the recommended procedure is to service and clear both interrupts before clearing the I flag. However, if the interrupts are cleared one at a time and the I flag is reset after each, the processor will simply recognize any interrupts still active and will process them properly but more slowly because of the time required to return from one interrupt before recognizing the next. If the

procedure recommended above is followed, each interrupt will be recognized and processed only once. Figure 1.14 provides several examples of interrupts, microprocessor recognition of each interrupt ($\overline{IRQ}$ and $\overline{NMI}$), and processor selection of interrupts during overlapped requests.



Examples of Interrupt Recognition by MCS650X

FIGURE 1.14

Each major event affecting the microprocessor is numbered in the figure with the corresponding explanations below.

Event Number	System Activity
1.	Processor is executing from main program and $\overline{IRQ}$ goes to low state.
2.	Upon completion of current instruction, the processor recognizes the interrupt, stores the contents of PC and P onto the stack and then sets I during the fetch of the interrupt vector.
3.	After servicing the interrupt, $\overline{IRQ}$ should be reset before resetting the interrupt mask bit to avoid double interrupting.
4.	Before the processor resumes normal main program execution the interrupt mask bit will be reset low.
5.	$\overline{NMI}$ now goes low, signalling a non-maskable interrupt request.

Event Number	System Activity
6.	The $\overline{\text{NMI}}$ interrupt is recognized and serviced in the same manner as $\overline{\text{IRQ}}$.
7.	The processor has resumed normal operation when $\overline{\text{NMI}}$ again goes low requesting an interrupt.
8.	The interrupt mask bit is set high in response to the $\overline{\text{NMI}}$ request.
9.	Here $\overline{\text{IRQ}}$ has gone low to signal an interrupt request. This request is ignored since the $\overline{\text{NMI}}$ interrupt is being serviced and the interrupt mask is set.
10.	The interrupt mask bit is reset after servicing the $\overline{\text{NMI}}$ interrupt.
11.	The processor is now able to recognize the $\overline{\text{IRQ}}$ signal, which is still low, and does so by setting the interrupt mask bit.
12.	During the servicing of $\overline{\text{IRQ}}$, $\overline{\text{NMI}}$ goes from high to low. The processor then completes the current instruction and abandons the $\overline{\text{IRQ}}$ interrupt to service $\overline{\text{NMI}}$. $\overline{\text{NMI}}$ is serviced regardless of the state of the interrupt mask bit.
13.	After completing the $\overline{\text{NMI}}$ interrupt routine, the processor will resume execution of the $\overline{\text{IRQ}}$ routine, even though $\overline{\text{IRQ}}$ has subsequently gone high.

1.4.1.2.11 $\overline{\text{RES}}$--Reset

The $\overline{\text{RES}}$ line is used to initialize the microprocessor from a power-down condition. During the power-up time this line is held low, and writing from the microprocessor is inhibited. When the line goes high, the microprocessor will delay 6 cycles and then fetch the new program count vectors from specific locations in memory (PCL from location FFFC and PCH from location FFFD). This is the start of the user's code. It should be assumed that any time the reset line has been pulled low and then high, the internal states of the machine are unknown and all registers must be re-initialized during the restart sequence. Timing for the reset sequence is shown in Figure 1.13.