

MOTOROLA SEMICONDUCTOR APPLICATION NOTE

AN1012

A Discussion of Interrupts for the MC68000

DESCRIPTION

Commercial and industrial microprocessor systems typically consist of a processor interfaced with some type of peripheral, which usually requires service from the processor. When a peripheral requires service, it flags the processor with an interrupt request.

MC68000 INTERRUPT STRUCTURE

Interrupt requests are input to the MC68000 through three pins, which are needed to represent seven levels of interrupt priority and a quiescent state (no interrupt). The 16-bit status register of the MC68000 contains a three-bit mask that only enables interrupts of a higher priority than the level represented in the three-bit mask (see Figure 1).

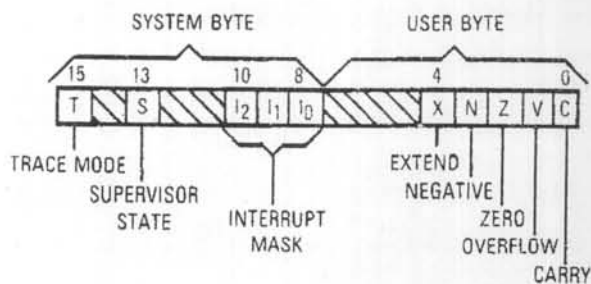


Figure 1. Status Register

If the mask is not set, then the processor has an interrupt service timing requirement, which, along with the data rate and interrupt frequency, can be used to determine the relative priority of each of the peripherals in the system. If two or more peripherals attempt to request service simultaneously, the relative priority of each peripheral determines which peripheral receives service first. To minimize the risk of violating timing requirements of lower priority peripherals, the processor must quickly identify and service the current highest priority interrupt. The address of the service routine is contained in a vector; every interrupt source should have a unique vector

for its service routine. The MC68000 architecture stores vectors in the first 1024 bytes of memory.

Interrupt request level zero (IPL2-IPL0 all high) indicates that no interrupt service is requested. When an interrupt level from one through six is requested via IPL2-IPL0, the processor compares the interrupt request level to the interrupt mask to determine whether the interrupt should be processed. Interrupt requests are ignored for all interrupt request levels that are less than or equal to the current processor priority level as determined by the interrupt mask bits. Level-seven interrupts are nonmaskable and are discussed separately in **LEVEL-SEVEN INTERRUPTS**. Table 1 shows the relationship between the actual requested interrupt level and the state of the interrupt control lines (IPL2-IPL0) as well as the interrupt mask levels required for recognition of the requested level.

Table 1. Interrupt Control Line Status

| Requested Interrupt Level | Control Line Status | | | Interrupt Mask Level Required for Recognition |
|---------------------------|---------------------|------|------|---|
| | IPL2 | IPL1 | IPL0 | |
| 0* | High | High | High | N/A* |
| 1 | High | High | Low | 0 |
| 2 | High | Low | High | 0-1 |
| 3 | High | Low | Low | 0-2 |
| 4 | Low | High | High | 0-3 |
| 5 | Low | High | Low | 0-4 |
| 6 | Low | Low | High | 0-5 |
| 7 | Low | Low | Low | 0-7 |

*Indicates no interrupt requested.

Interrupt requests are considered by the MC68000 to be pending until the completion of the current instruction execution. At that time, if the priority of the pending interrupt is less than or equal to the current processor priority represented by the three-bit interrupt mask, then the next instruction is executed. If the priority of the pending



interrupt is greater than the processor priority, then interrupt exception processing begins. During interrupt exception processing, the MC68000 places the level of interrupt priority on address lines A1, A2, and A3. These lines can be used to quickly determine which group of peripherals might have generated the interrupt request. Simultaneously, the function code outputs (FC2-FC0) are set to indicate an interrupt acknowledge (IACK), which flags the user hardware that exception processing has begun.

Figure 2 is a flowchart of the MC68000 interrupt process.

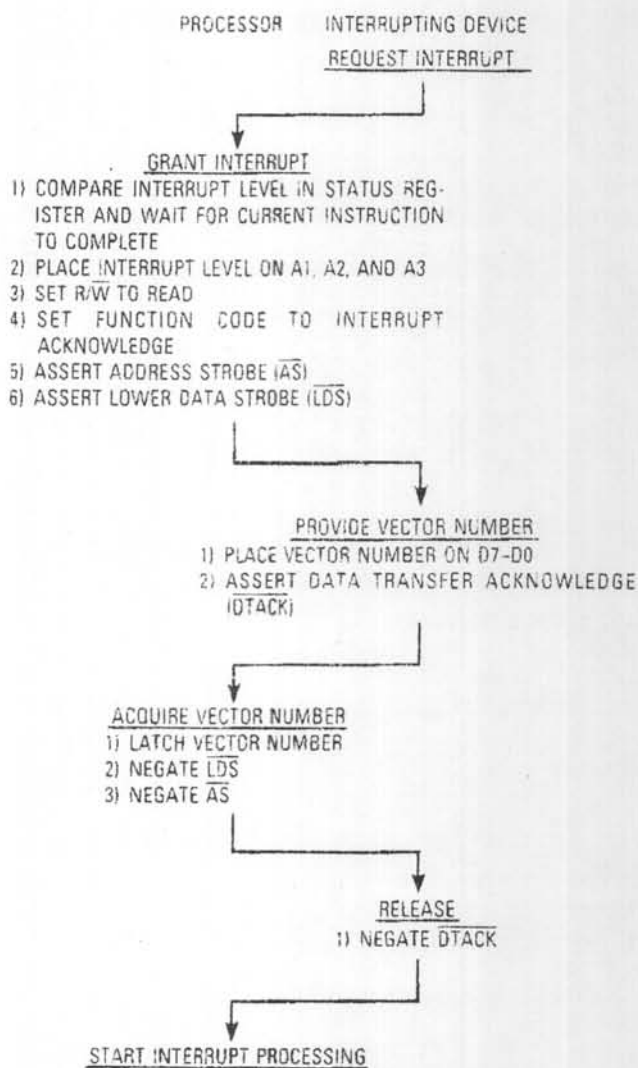


Figure 2. Interrupt Process Flowchart

RECOGNITION OF INTERRUPTS

To ensure that an interrupt will be recognized, the following interrupting level rules should be considered:

1. The incoming interrupt request level must be at a higher priority level than the mask level set in the interrupt mask bits (except for level seven, the nonmaskable interrupt).

2. The $\overline{\text{IPL2}}\text{--}\overline{\text{IPL0}}$ interrupt control lines must be held at the interrupt request level until the MC68000 acknowledges the interrupt by initiating an IACK bus cycle. The processor indicates that it is executing an IACK bus cycle by placing the interrupt acknowledge code (all high) on the three processor status function code pins (FC2-FC0) and also asserting AS.

These rules guarantee that the interrupt will be processed; however, the interrupt could also be processed if the request level is taken away before the IACK bus cycle.

The MC68000 samples the $\overline{\text{IPL2}}\text{--}\overline{\text{IPL0}}$ interrupt control pins and compares the level of these three inputs to the interrupt mask level once during the execution of every instruction. The exact step in the execution of an instruction that samples the interrupt control pins is instruction dependent. It is possible that an interrupt held for as short a time as two clock periods of the system clock could be recognized. For example, assume that 1) the interrupt mask is set at level two and a level-three interrupt request is present on the interrupt control pins for two system clock periods, and 2) a level-six interrupt request is then requested and remains applied to the interrupt control pins. Which interrupt request will be recognized?

The level-three interrupt request may be recognized, but it is not guaranteed since it is not held until the IACK bus cycle is initiated. The level-six interrupt request will be recognized, assuming that it remains applied to the interrupt control pins until its IACK bus cycle begins. Again, to ensure that an interrupt request will be recognized, the level must be held until the three function code outputs are high; AS has fallen (start of IACK bus cycle), and A1, A2, and A3 reflect the interrupt level requested.

The maximum time from interrupt request to the execution of the interrupt handling routine (interrupt latency period) occurs when the processor is executing the following instruction sequence: MOVEM.L (An) + ,D7-D0/A7-A0; and DIVS. At the beginning of the MOVEM.L (An) + ,D7-D0/A7-A0 instruction, the interrupt control pins are sampled; whereas, they are sampled at the end of the DIVS instruction. Thus, if an interrupt is present on the interrupt control pins immediately after the MOVEM instruction samples these lines, it would not be recognized until the end of the DIVS instruction. This maximum interrupt latency period would include a maximum of 146 clock periods plus wait states for the MOVEM instruction, 174 clock periods plus wait states for the DIVS instruction, and 58 clock periods for a worst-case autovector IACK sequence. Therefore, the maximum interrupt latency period for a no-wait-state system is 378 clock periods.

LEVEL-SEVEN INTERRUPTS

Level-seven interrupts are handled differently than interrupt levels one through six. A level-seven interrupt is a nonmaskable interrupt; therefore, a seven in the interrupt mask does not disable a level-seven interrupt. Again, a level-seven interrupt should be maintained on interrupt control pins IPL2-IPL0 until the IACK bus cycle is initiated to guarantee that the interrupt will be recognized.

Level-seven interrupts are edge triggered by a transition from a lower priority request to the level-seven

request, as opposed to interrupt levels one through six, which are level sensitive. Therefore, if a level-seven interrupt persists, the processor will only recognize this level-seven interrupt once since only one transition (from lower priority request to level-seven request) on the interrupt control lines has occurred. For the processor to recognize a level-seven interrupt followed by another level-seven interrupt, one of two following sequences must occur:

1. Interrupt control pins must have a lower priority interrupt request level than level seven: i.e., level zero through six. Then, the interrupt request level on the interrupt control pins changes to level seven and remains at level seven until the IACK bus cycle begins. Later, the interrupt request level returns to a lower interrupt request level and finally back to level seven, causing a second transition on the interrupt control lines.
2. Interrupt control pins have a lower priority interrupt request than level seven: i.e., level zero through six. Then, the interrupt request level on the interrupt control pins changes to level seven and remains at that level. If the interrupt handling routine for the level-seven interrupt lowers the interrupt mask level, a second level-seven interrupt will be recognized even though no transition has occurred on the interrupt control pins, and the interrupt mask will be set back to level seven. The level on the interrupt control pins only had one initial level transition (from lower priority request to level-seven request) and then was held at level seven until the second level-seven IACK bus cycle.

The processor interrupt priority level can be modified 1) directly through the use of instructions that act upon the status register or 2) indirectly through stack manipulation prior to executing an RTE instruction. Using option 2 to reduce the processor interrupt priority from level-seven to some lower level sometimes can cause multiple

level-seven interrupts to be processed. Although the level-seven interrupt is edge sensitive, it can be generated by comparison when the request level remains at seven (after servicing the level-seven request) and the processor priority is set to a lower level by software. For example, during processing of a level-seven interrupt, the stacked SR would be modified to return with a lower interrupt priority level. However, if the IPL pins are still asserted at level-seven during execution of the RTE instruction, then another level-seven interrupt will occur. Under normal circumstances, the interrupt acknowledge would be used to cancel the interrupt request. However, to minimize hardware in low-cost, simple systems, this may not always be the case.

As the other interrupts are level sensitive, multiple interrupts will always occur if the interrupt source is not removed (unless the stacked SR is modified to return configured for a higher priority level).

If stacked mask is level seven and the level-seven request is still asserted at the RTE, it will not cause another interrupt.

INTERRUPT ACKNOWLEDGE SEQUENCE

The purpose of the IACK bus cycle is to indicate to the processor the starting location of a particular interrupt handling routine. The following conditions occur during an IACK bus cycle and are qualified by the assertion of AS: 1) the MC68000 echoes the interrupt level on address lines A1, A2, and A3, which was recognized on the interrupt control lines, and 2) the function code output pins (FC2-FC0) are then driven high. This information is used by external hardware to generate an IACK signal to the interrupting device. Figure 3 shows the timing for the IACK bus cycle.

If the interrupting device has a vector register, it will then place a vector number on data lines D7-D0 and perform a data transfer acknowledge (DTACK) handshake

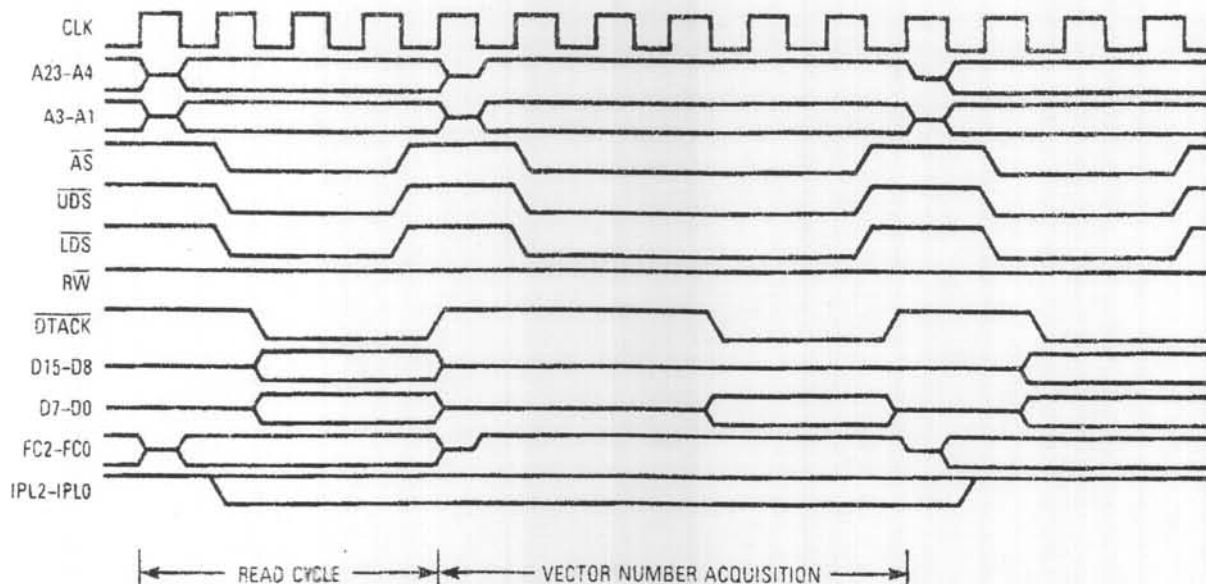


Figure 3. IACK Timing Diagram

to terminate the IACK bus cycle. The MC68000 uses the vector number to determine the starting address of the interrupt handling routine for that particular device.

If the interrupting device does not have a vector register, then external hardware should recognize the IACK signal and assert a valid peripheral address (VPA) to terminate the IACK bus cycle. When VPA is asserted, the MC68000 automatically directs itself to the proper interrupt vector; this is called an autovector interrupt. There are seven autovectors; one autovector corresponds to each of the seven interrupt levels. The MC68000 selects the autovector for the interrupt level that was recognized.

The final way to terminate an (IACK) bus cycle is with the bus error BERR signal. Even though the interrupt control pins are synchronized to enhance noise immunity, it is possible that external system interrupt circuitry may initiate an IACK bus cycle as a result of noise. Since no device is requesting interrupt service, neither DTACK nor VPA will be asserted to signal the end of the nonexistent IACK bus cycle. When there is no response to an IACK bus cycle after a specified period of time, the system "watchdog timer" should assert BERR. This indicates to the processor that it has recognized a spurious interrupt. Since a spurious interrupt is an exception, the MC68000 will go to the spurious interrupt vector to fetch the starting address for this exception handling routine.

VECTORED INTERRUPT SEQUENCE

For a vectored interrupt, the MC68000 executes the following sequence:

1. Make an internal copy of the current status register.
2. Set S bit, clear T bit, and replace the I₀, I₁, I₂ bits of the interrupt mask with the level of the interrupt that was recognized. (Items 1 and 2 take a total of six clock periods.) There is no bus activity during this time.
3. Stack program counter (low word) on system stack (four clock periods with no wait states).
4. Run an IACK bus cycle for vector number acquisition (four clock periods with no wait states, between 10 and 18 clock periods for autovector interrupts).
5. Justify the vector number for vector acquisition (four clock periods and no bus activity during this time).
6. Stack former (internal copy) status register on system stack (four clock periods with no wait states).
7. Stack program counter (high word) on system stack (four clock periods with no wait states).
8. Read exception vector (high word) (four clock periods with no wait states).
9. Read exception vector low word (four clock periods with no wait states).

10. Fetch first word of instruction, of the interrupt handling routine (four clock periods with no wait states).
11. Two nonbus clock periods (dead cycles).
12. Fetch second word of instruction of the interrupt handling routine and check the interrupt control pins for a valid interrupt. If a higher priority interrupt is present, the MC68000 begins an interrupt acknowledge sequence for that higher priority interrupt (four clock periods with no wait states).

Sequence 12 sheds more light on the interrupt example discussed in **RECOGNITION OF INTERRUPTS**. In that example, the interrupt mask is set at level two and a level-three interrupt is present for two clock periods on the interrupt control pins. If the level three is then removed from the interrupt control pins and a level-six interrupt is now pending, which interrupt will be recognized?

The level-three interrupt will be recognized, only if the level-three interrupt is present during the instruction step that samples the interrupt control pins. Assuming that the level-three interrupt is recognized, the interrupt control pins will be sampled again during sequence 12 (previously described) of the level-three IACK sequence. At this time, the processor will recognize the level-six interrupt as a pending interrupt. The MC68000 will fetch the second word of the first instruction of the level-three interrupt handling routine, but this instruction will not be executed; instead, the processor begins the level-six interrupt sequence. At the end of the level-six interrupt handling routine, a return from exception (RTE) instruction should be executed. The processor then fetches the first instruction of the level-three handling routine and samples the interrupt control lines. If there is no higher interrupt present on the interrupt control lines, then the level-three handling routine will be executed.

IACK GENERATION

Peripherals of the M68000 Family with vector number registers have an $\overline{\text{IRQ}}$ output and an $\overline{\text{IACK}}$ input. The $\overline{\text{IACK}}$ signal should be derived from address lines A3, A2, and A1 during the $\overline{\text{IACK}}$ vector number acquisition. A circuit to generate $\overline{\text{IACK1}}$ through $\overline{\text{IACK7}}$ is shown in Figure 4 and a representation of the $\overline{\text{IRQ4}}$ output and $\overline{\text{IACK4}}$ input is shown in Figure 5.

Interrupt request lines $\overline{\text{IRQ7}}-\overline{\text{IRQ1}}$ are encoded and prioritized by U1 before being presented to the MC68000 on the interrupt control lines ($\overline{\text{IPL2}}-\overline{\text{IPL0}}$). When the MC68000 recognizes a valid interrupt, it echoes the interrupt level on address lines A3, A2, and A1. The three-to-eight decoder (U2) decodes the interrupt level when the function code lines (FC2-FC0) are all high and $\overline{\text{AS}}$ low. To maintain compatibility among M68000 Family processors, A23-A4 being driven high (along with FC2-FC0) should be used to enable the E1 input of U. This separates the IACK space of the MC68000 from the

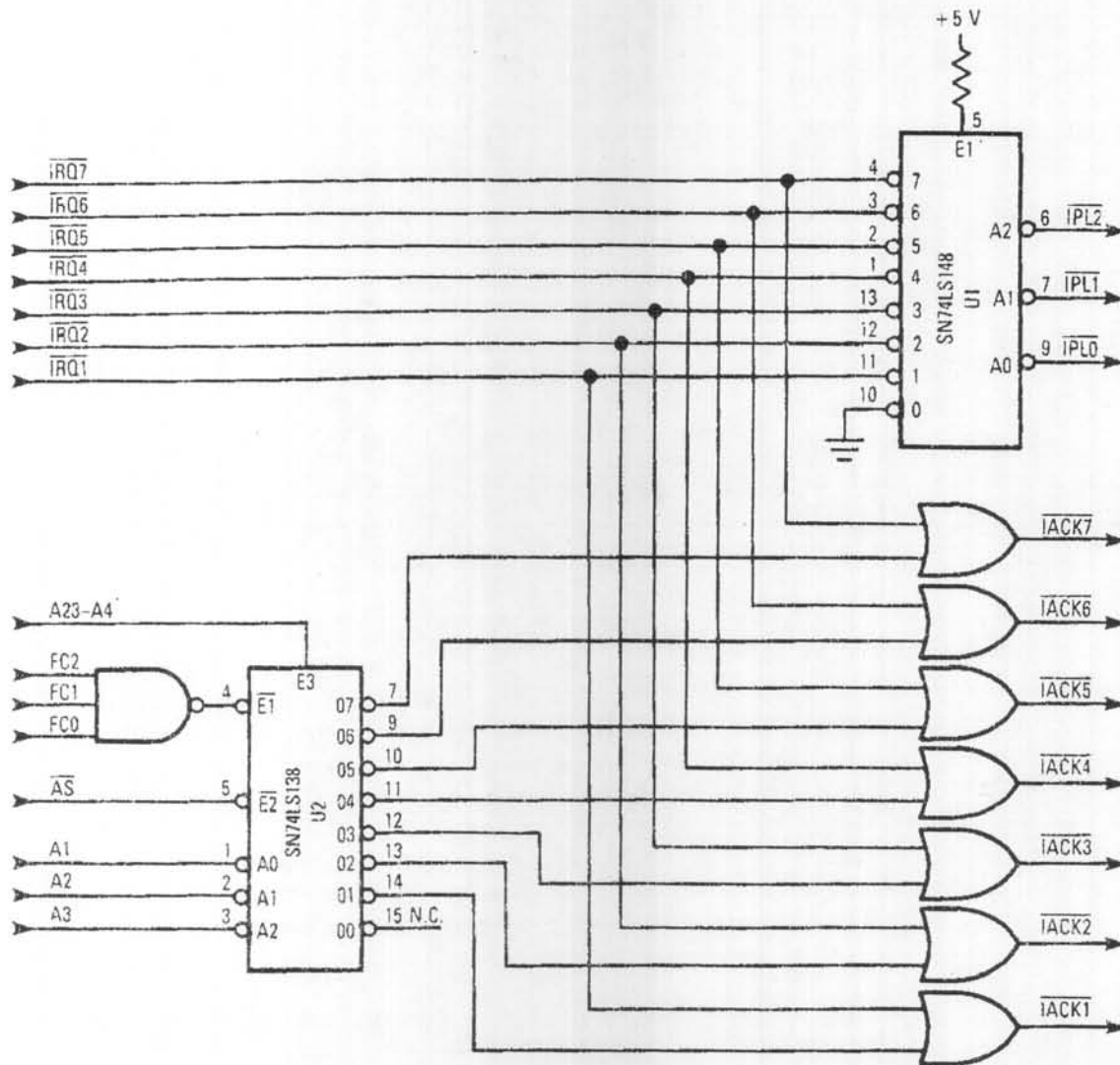


Figure 4. IACK Generation Circuit Diagram

CPU space of the MC68020 and the MC68030. By comparing the decoded interrupt acknowledge level with the $\overline{\text{IRQ}}7$ – $\overline{\text{IRQ}}1$ lines, only the recognized interrupt request is acknowledged instead of all pending interrupt requests.

"DAISY-CHAINING" OF INTERRUPTS

Several interrupting devices may share a common interrupt level. These devices may be prioritized by "daisy-chaining" their interrupt request ($\overline{\text{IRQ}}7$ – $\overline{\text{IRQ}}1$) lines and interrupt acknowledge ($\overline{\text{IACK}}7$ – $\overline{\text{IACK}}1$) pins. Figure 4 shows two interrupting devices, each with an $\overline{\text{IRQ}}4$ input and an $\overline{\text{IACK}}4$ output. Any device is allowed to interrupt the MC68000; however, lower priority devices (device 2) only receive an $\overline{\text{IACK}}$ if higher priority devices (device 1) are not requesting an interrupt. In this manner, devices at the beginning of the "chain" are serviced first.

When a level-four interrupt is recognized, the MC68000 drives the FC1–FC0 function codes high (so that the preset on both U4a and U4b are negated) and asserts $\overline{\text{AS}}$. The interrupt acknowledge circuit of Figure 4 will then assert $\overline{\text{IACK}}4$. This causes U6 (Figure 5) to clock in the current state of the $\overline{\text{IRQ}}$ output line of each device to drive its respective $\overline{\text{IACK}}$ input pin. Clocking is inhibited for an interrupting device if a higher priority device in the chain has its $\overline{\text{IRQ}}$ line asserted. This also ensures that devices at the beginning of a chain are serviced first.

When the interrupting device receives its $\overline{\text{IACK}}$ input, it will place its vector number on the data bus and assert $\overline{\text{DTACK}}$. The MC68000 uses the vector number to acquire the starting address of the interrupt handling routine for that particular device. The $\overline{\text{IACK}}$ pins for all devices are negated when the FC2–FC0 function codes change, signaling a new bus cycle.

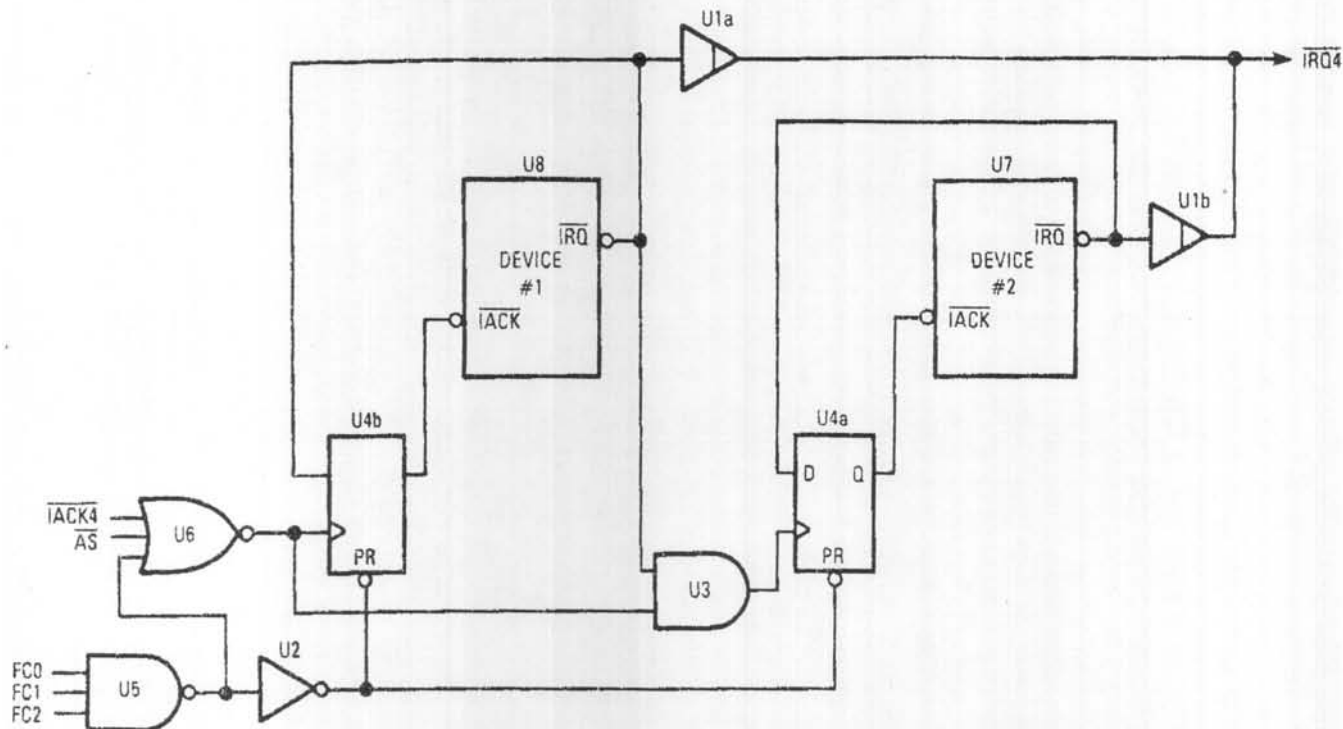


Figure 5. "Daisy-Chain" Interrupt Circuit Diagram

If any peripheral generates an interrupt, it must suppress IACK to all remaining peripherals and then place the vector number on the lower eight bits of the MC68000 data bus and assert \overline{DTACK} . Systems that have a large number of interrupt sources and timing requirements too critical for software polling can also generate the vector number and \overline{DTACK} by the same method. However, since every peripheral must have the capability to generate these signals, redundant hardware is spread throughout the system, making debug, modification, and maintenance difficult. Alternatively, all interrupt lines could be brought to a central location where both \overline{DTACK} and the proper vector number could be supplied. The application describes a system that will provide \overline{DTACK} and the vector number for up to 192 possible interrupt sources.

VECTORS

Any exception to free-running operation, such as an interrupt, has a vector stored at a unique location in the 1024 byte exception memory map. Exceptions other than interrupts include reset, system errors, software traps, and unimplemented instruction emulators, as shown in Table 2. Since each vector, except reset, requires a 32-bit address, four bytes are required to store each vector. Reset is a special case, which requires two 32-bit addresses or eight bytes of memory.

The exception vector map can be divided into 256 unique vectors, which can be represented by an eight-bit vector number. The vector number is not the vector; it is a pointer to one particular vector. During any exception processing, the MC68000 fetches the vector pointed to by a vector number. Each exception has a unique vector number, which, for all exceptions except user interrupts, is generated internally by the MC68000. User interrupts require that the vector number be placed on the lower eight bits of the data bus during interrupt acknowledge. The addition of the vector number during fetch requires the peripheral to supply only the eight-bit vector number instead of the whole 32-bit vector.

The MC68000 allows two methods of interrupt vector number generation, internal or external. To generate the vector number internally, \overline{VPA} is connected to IACK. In this mode, a unique vector number is generated for each interrupt priority level. This mode, called the autovector mode, is ideal for users requiring less than eight levels of interrupts or users with more than seven peripherals whose timing requirements are noncritical.

For users with more than seven peripherals and whose timing requirements demand service faster than possible with a software polling method, the MC68000 provides an additional 192 interrupts that require external vector number generation. In this case, IACK is not connected

Table 2. Exception Vector Assignment

| Vector Number(s) | Address | | | Assignment |
|------------------|---------|-----|-------|--------------------------------|
| | Dec | Hex | Space | |
| 0 | 0 | 000 | SP | Reset: Initial SSP |
| — | 4 | 004 | SP | Reset: Initial PC |
| 2 | 8 | 008 | SD | Bus Error |
| 3 | 12 | 00C | SD | Address Error |
| 4 | 16 | 010 | SD | Illegal Instruction |
| 5 | 20 | 014 | SD | Zero Divide |
| 6 | 24 | 018 | SD | CHK Instruction |
| 7 | 28 | 01C | SD | TRAPV Instruction |
| 8 | 32 | 020 | SD | Privilege Violation |
| 9 | 36 | 024 | SD | Trace |
| 10 | 40 | 028 | SD | Line 1010 Emulator |
| 11 | 44 | 02C | SD | Line 1111 Emulator |
| 12* | 48 | 030 | SD | (Unassigned, Reserved) |
| 13* | 52 | 034 | SD | (Unassigned, Reserved) |
| 14* | 56 | 038 | SD | (Unassigned, Reserved) |
| 15 | 60 | 03C | SD | Uninitialized Interrupt Vector |
| 16-23* | 64 | 04C | SD | (Unassigned, Reserved) |
| — | 95 | 05F | — | — |
| 24 | 96 | 060 | SD | Spurious Interrupt |
| 25 | 100 | 064 | SD | Level 1 Interrupt Autovector |
| 26 | 104 | 068 | SD | Level 2 Interrupt Autovector |
| 27 | 108 | 06C | SD | Level 3 Interrupt Autovector |
| 28 | 112 | 070 | SD | Level 4 Interrupt Autovector |
| 29 | 116 | 074 | SD | Level 5 Interrupt Autovector |
| 30 | 120 | 078 | SD | Level 6 Interrupt Autovector |
| 31 | 124 | 07C | SD | Level 7 Interrupt Autovector |
| 32-47 | 128 | 080 | SD | TRAP Instruction Vectors |
| — | 191 | 0BF | — | — |
| 48-63* | 192 | 0C0 | SD | (Unassigned, Reserved) |
| — | 255 | 0FF | — | — |
| 64-255 | 256 | 100 | SD | User Interrupt Vectors |
| — | 1023 | 3FF | — | — |

*Vector numbers 12, 13, 14, 16-23 and 48-63 are reserved for future enhancements by Motorola. No user peripheral devices should be assigned these numbers.

to \overline{VPA} ; instead, it is used by external hardware to determine that a vector number is needed by the MC68000 and to provide the proper vector number and \overline{DTACK} .

VECTOR NUMBER GENERATION

The 192 user interrupt vectors are referenced by sequential vector numbers 64 through 255. Therefore, if

fewer than 193 interrupts are required, each interrupt can be assigned a unique vector number that can also be interpreted as a priority. Vector number 64 can be assumed to have the lowest priority, and vector number 255, the highest. The 192 levels of externally generated priority can be represented by eight bits (10111111-00000000). The vector number is the externally generated priority offset by 64; the vector number

can be generated by encoding the interrupt to its priority and then adding 64. This is essentially the same format that is used in the autovectors.

Figure 6 is a block diagram of such a system. All circuitry, except the processor, can be located in one area rather than spread throughout the system. Two sets of latches have been inserted to guarantee that no interrupts are lost and that the vector number placed on the data bus is the result of only one interrupt. Otherwise, if an interrupt request is generated during interrupt acknowledge, it could cause the vector number to be in a state of transition when the MC68000 is attempting to latch the vector number from the data bus. Latch number one prohibits any new interrupts from being accepted until the vector number has been latched by latch number two. Latch number two isolates the vector number from the data bus until IACK is asserted. After a delay sufficient to allow the vector number to propagate to latch number two, latch number one is released to allow new interrupts to be accepted.

The circuitry shown in Figure 7 performs all the tasks necessary to provide vector numbers for up to 192 possible interrupt sources. The 192 interrupt request lines are divided into 24 groups of eight, which are input through an SN74LS373 octal latch to an SN74LS148 eight-to-three encoder. The SN74LS148 encoders are daisy-chained so that each stage can disable all succeeding stages, which effectively prioritizes the interrupts by group. Within each group, interrupts are prioritized into eight levels, which the encoder represents with a three-bit encoded number on lines A0, A1, and A2.

The A0 line from each of the 24 groups is NANDed to form the A0 of the vector number. The A1 and A2 lines are handled in an identical manner. Bits 3 and 7 of the encoded interrupt are made by NANDing selected GS outputs of the encoders. Bits 6 and 7 of the vector number

differ from bits 6 and 7 of the encoded interrupt because of the offset of 64.

After the vector number has had sufficient time to propagate, a second SN74LS373 octal latch is used to capture the number and allow the latches in the 24 interrupt groups to be released to accept new interrupt requests. The delay, imposed by a SN74LS95 four-bit shift register to latch the vector number, assumes that all 24 groups are implemented and the MC68000 is running at 8 MHz. Timing requirements can be derived from Figure 8.

A final SN74LS148 is used to encode the three-bit interrupt requests to the MC68000. The inputs to this encoder are the GS outputs of the SN74LS148 encoders from any group of interrupts. The group providing the GS output and all preceding groups can activate the level to which the GS output is input. However, GS outputs of preceding groups, which are input to a higher level in the final SN74LS148, will effectively disable the lower level interrupt request.

If R9M or T6E mask types are used in the systems, then the seven levels of interrupts must be latched before encoding on the rising edge of the processor clock. The latch is necessary to synchronize the interrupt request lines.

The following variations to the system given in the application can be considered in light of specific system requirements:

1. Flip-flops could be inserted on each group to latch an edge-type interrupt input.
2. The level of interrupt that initiated exception processing could be decoded from address lines A1, A2, and A3 of the MC68000.
3. If vector numbers reserved for other functions (e.g., TRAPS, autovectors) are unused, then they could be used for user interrupts. However, observe caution when using any reserved vector numbers.

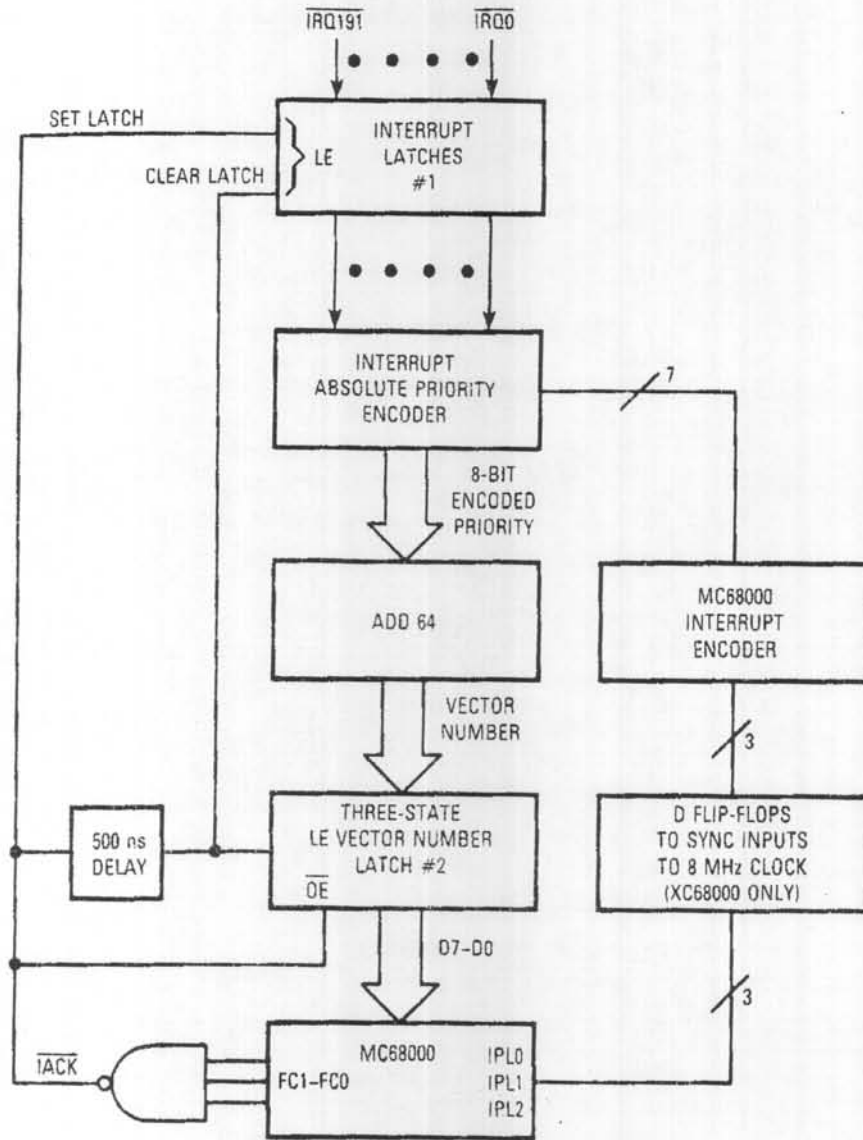


Figure 6. Vector Generation Circuit — Block Diagram

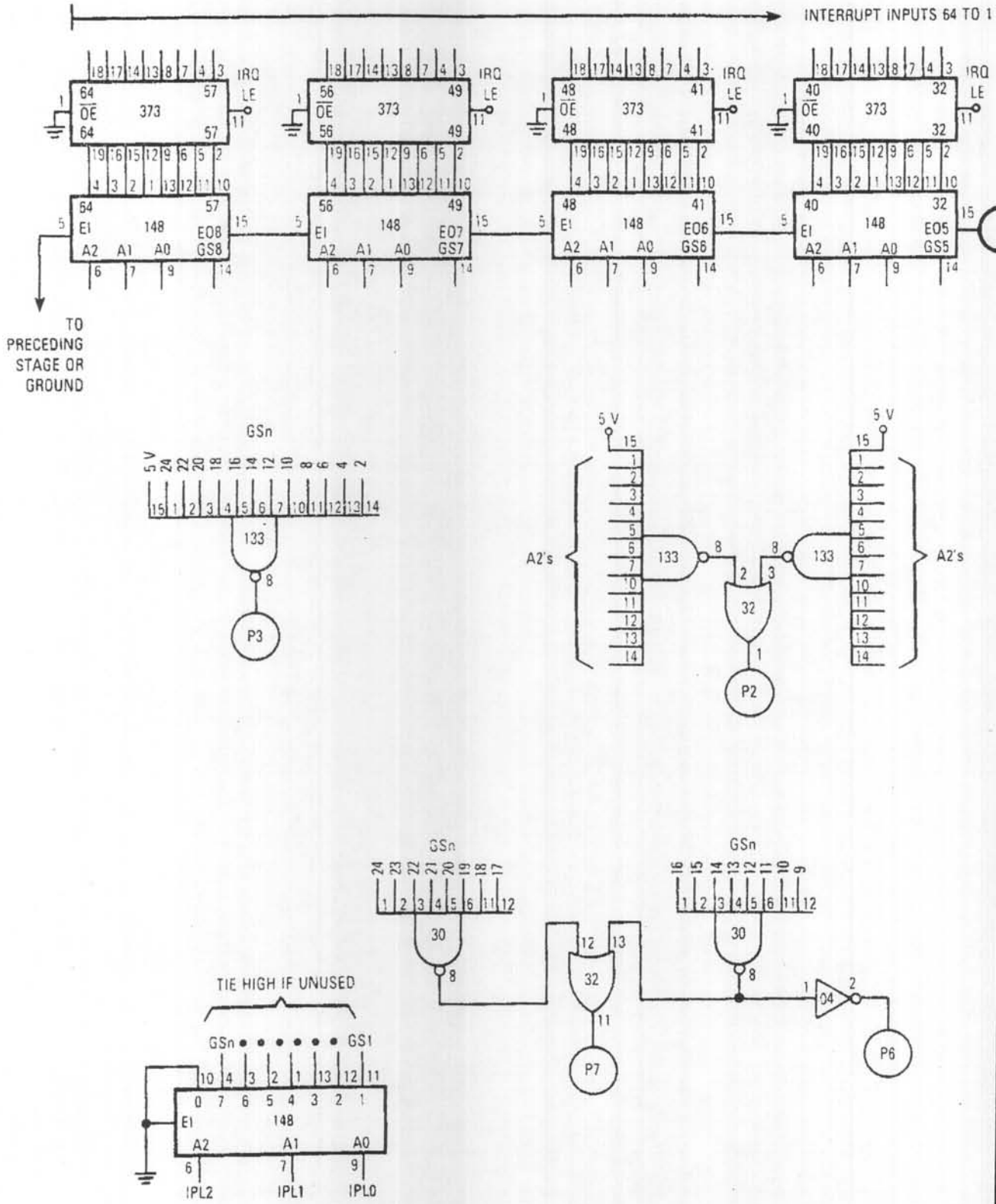


Figure 7. Vector Generation Circuit Schematic (Sheet 1 of 2)

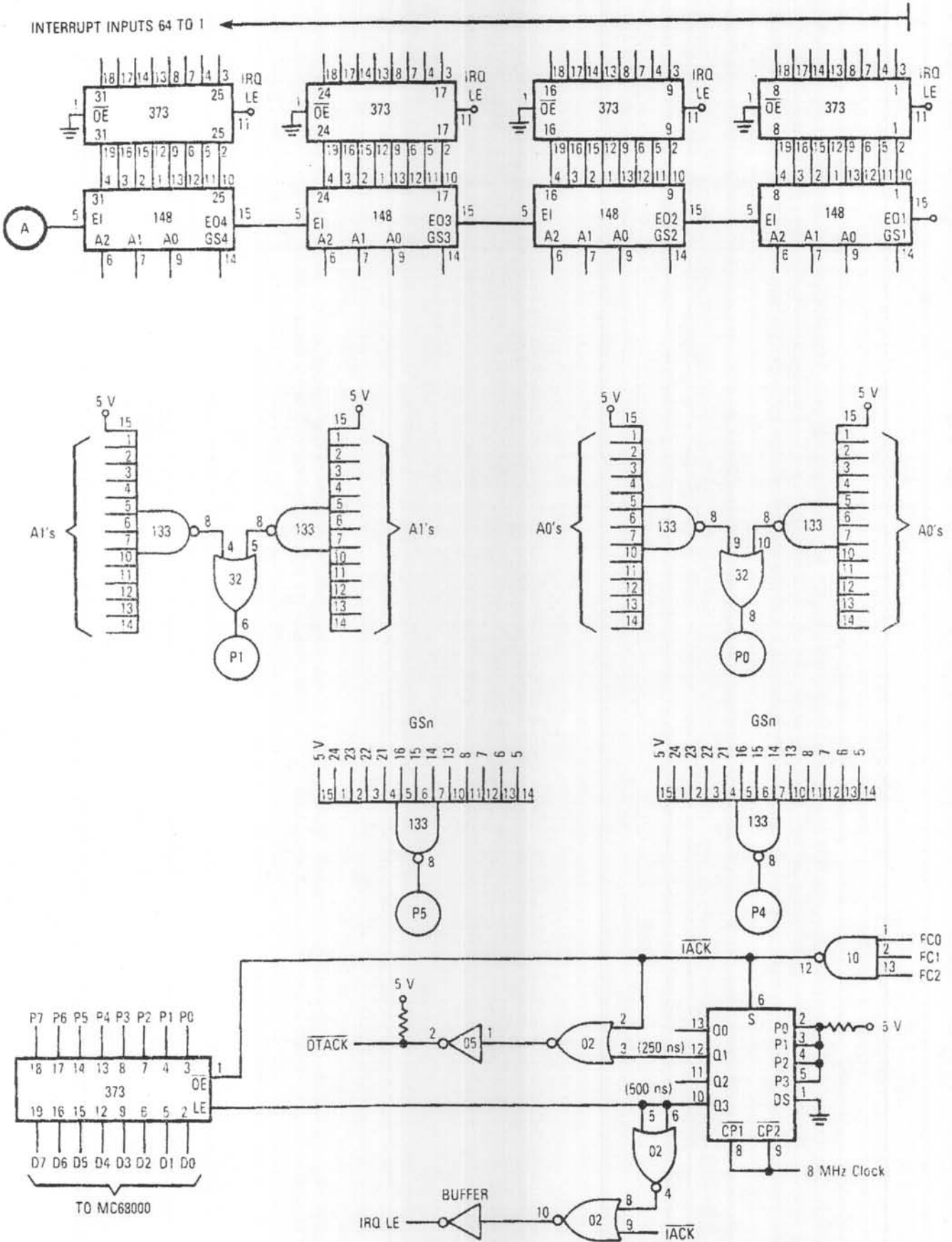
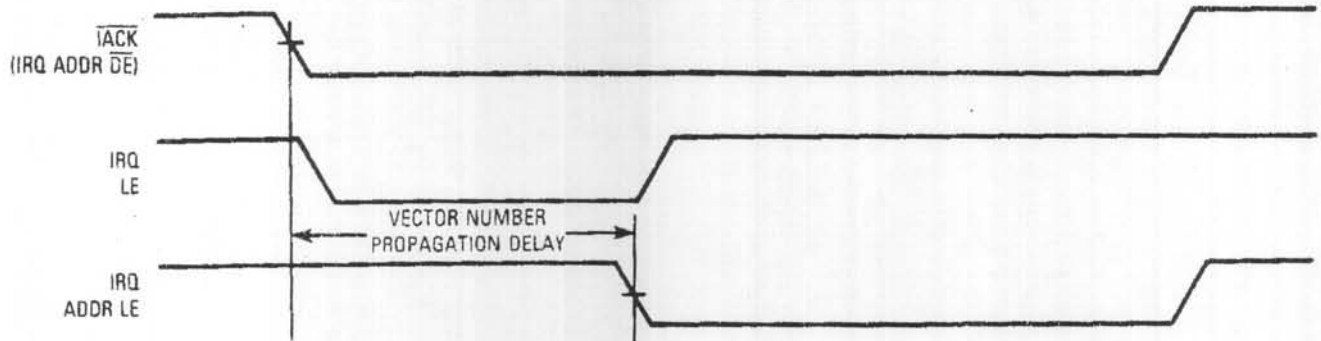



Figure 7. Vector Generation Circuit Schematic (Sheet 2 of 2)



\overline{DTACK} DELAY = VECTOR NUMBER PROPAGATION DELAY -- 235 ns
 (IF $\leq \phi$ THEN NO WAIT STATES ARE NECESSARY)

Figure 8. Vector Generation Circuit — Timing Diagram

Motorola reserves the right to make changes without further notice to any products herein to improve reliability, function or design. Motorola does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others. Motorola products are not authorized for use as components in life support devices or systems intended for surgical implant into the body or intended to support or sustain life. Buyer agrees to notify Motorola of any such intended end use whereupon Motorola shall determine availability and suitability of its product or products for the use intended. Motorola and  are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Employment Opportunity/Affirmative Action Employer.

Literature Distribution Centers:

USA: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036.

EUROPE: Motorola Ltd.; European Literature Center; 88 Tanners Drive, Blakelands Milton Keynes, MK145BP, England.

ASIA PACIFIC: Motorola Semiconductors H.K. Ltd.; P.O. Box 80300; Cheung Sha Wan Post Office; Kowloon Hong Kong.



MOTOROLA

AN1012/D

413435 PRINTED IN USA 1/89 ORIGINAL LITHO 05/91 41,000 19/18