# Microcomputer interrupts

By Jonathan A. Titus, David G. Larsen, and Peter R. Rony

THIS month's column is the first of several that will focus on the concept of an interrupt. When used in the context of a computer, an *interrupt* can be defined as the suspension of normal program execution in order to handle a sudden request for *service*, i.e., assistance, by the computer. At the completion of interrupt service, the computer resumes the interrupted program from the point where it was interrupted.[1] This specific use of interrupt is consistent with the general meaning of the term: to stop a process in such a way that it can be resumed.

A given computer will typically communicate with a variety of external I/O "devices." If it is a minicomputer, it may communicate with a Teletype or alphanumeric keyboard, a CRT display, a printer, a floppy disk, and perhaps one or more laboratory instruments. If it is a microcomputer, it may communicate with smaller devices—motors, solid-state relays, push-button switches, display lights, etc.—within a larger machine or instrument. When used as a replacement for discrete logic devices in a complex digital circuit, a microcomputer may communicate with other TTL-integrated circuit chips such as latches, flip-flops, and three-state buffers.

When communicating with external I/O devices,[2] microcomputers can operate in two general modes, *polled* and *interrupt*. Polling is the periodic interrogation of each I/O device sharing a communications link to a microcomputer in order to determine if it requires servicing. A microcomputer sends a poll that has the effect of asking the selected device, "Do you have anything to transmit?," "Are you ready to receive data?," and similar questions. When a microcomputer services a polled device, it simply exchanges digital information with the device in a manner that is prescribed by software in a subprogram or subroutine called a *software driver*.

In polled operation, the microcomputer sequences through the devices tied to the microcomputer looking for individual devices that need servicing. When it finds a device that requires service, it stops sequencing, calls a software driver, and services the device. Once it is finished, the microcomputer continues checking the devices. Polled operation is most useful with relatively slow devices that do not require frequent service, do not require attention from the microcomputer for excessive periods of time, and can wait to be serviced. Advantage is taken of the difference in speed of operations in the microcomputer and operations in the I/O device. Most common I/O devices are much slower than microcomputers. For example, in 100 msec (teletypewriter response time) an 8080A-based microcomputer can execute approximately 20,000 instructions when operated at a clock rate of 2 MHz. Although a microcomputer may give one the impression that it is doing several things simultaneously, this is only an illusion since it can manipulate data much faster than most I/O devices can respond to changes in data. *A single computer can perform only one task at a time.*

In interrupt operation, the microcomputer juggles the demands of the external I/O devices. There is a distinction between slow devices that require infrequent servicing and high speed devices that demand the attention of the microcomputer for most of the time. The most appropriate description for interrupt-operated systems is that they are *asynchronous*, i.e., they lack a common synchronizing signal and therefore give rise to generally unexpected or unpredictable program execution within the microcomputer. An *asynchronous device* is a device in which the speed of operation is not related to any frequency in the system to which it is connected.[3] The use of asynchronous devices is the rule rather than the exception.

There can exist *priority* in interrupt operation; all I/O devices can be ordered in importance so that some devices take precedence over others. In contrast, there is usually no priority in polled operation; once a device is serviced, it waits its turn until all other devices are sequenced and, if necessary, also serviced. The time between the interrupt request by a device and the first instruction byte of the software that services it is known as the *interrupt response*

*Mr. Titus is President, Tychon, Inc., Dr. Rony, Department of Chemical Engineering, and Mr. Larsen, Department of Chemistry, are with the Virginia Polytechnic Institute and State University.*

*time*. For a high speed device that has high priority, the response time can be very short, less than a millisecond. For a low speed device that has low priority, the response time is variable, since it depends upon the demands placed upon the microcomputer by all higher priority devices.

Three commonly used microcomputer interrupt techniques are the *single-line interrupt*, the *multilevel interrupt*, and the *vectored interrupt* (*Figure 1*). In the single-line interrupt technique, multiple devices must be OR connected to a single interrupt line to the microcomputer. Once an interrupt signal is received, all of the interrupt devices are polled to determine which one caused the interrupt. It is possible to assign software priorities to the various interrupting devices, so that the first device polled that needs service is the one that receives the attention of the microcomputer. A common term used for that part of a program that polls interrupt devices is *flag checking routine*. We shall discuss the concept of a flag in a subsequent column. At the moment, consider a flag to be a single-bit memory that indicates when an operation has been completed or when a condition has been attained.

The multilevel interrupt technique has several interrupt lines to the microcomputer, each line being tied to a separate I/O device flag. The microcomputer does not need to poll the devices to determine which one caused the interrupt. This is done internally within the microprocessor chip. Depending upon the nature of the microprocessor chip, this can be a very fast interrupt technique, but it is somewhat difficult to expand.

A vectored interrupt causes a direct branch by the microcomputer to that part of the program that services the interrupt. This interrupt technique requires external integrated circuit chips to supply the memory address of the *interrupt service routine* as well as to set the priority. With the 8080A microprocessor chip, eight different service routine addresses can be readily specified, although one of these addresses coincides with the reset address for the microprocessor, location zero. If you are interested in vectored interrupts, we encourage you to consider the Intel 8259 programmable interrupt controller, which became available commercially in July, 1976.

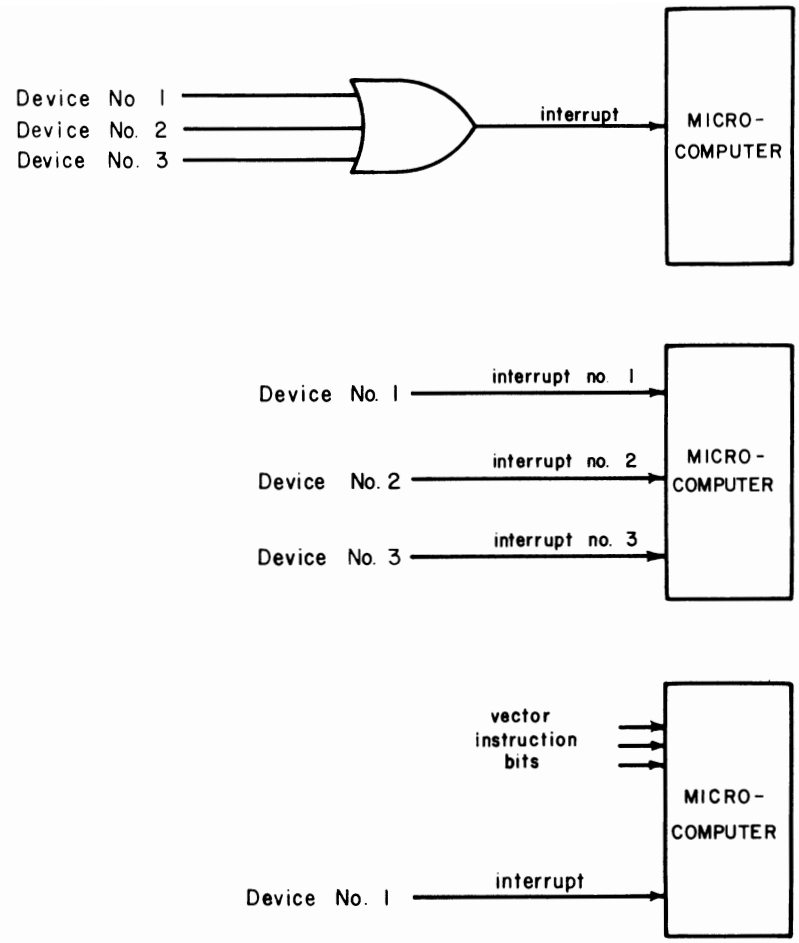


**Figure 1** *Schematic diagram of three different interrupt techniques: the polled interrupt (top), multilevel interrupt (middle), and vectored interrupt (bottom).*

The use of interrupts should be considered very carefully. More complicated software is invariably required. For example, you will generally have to save the status of the microprocessor chip at the time that the interrupt occurred. This means placing the contents of the accumulator, the flags, and the registers into a specified region of memory from which they can be retrieved at a later time, after the interrupting device has been serviced. Pay attention to priorities. Make certain that devices that require high priority and need immediate servicing are given the highest priority. Other devices, such as Teletypes, should be low priority. Also, if you attempt to do too much with an interrupt system, you might find that your microcomputer becomes "interrupt bound," which means that the microcomputer is only working on interrupt tasks and is not working on the main task, which it should be doing while only infrequently servicing interrupt requests.

To end this column, we would like to provide one example of an interrupt system. Assume that your

microcomputer is performing mathematical computations on 7-bit ASCII numbers that are entered via a UART chip[4] that is connected to a Teletype operated at 110 Baud, or ten ASCII numbers per second. The exchange of data between the microcomputer and the UART can be performed in 20 to 30 μsec, which leaves 99.97 msec left for the microcomputer to do other things. With the Intel floating point package, for example, each floating point multiplication or division can be performed in 2 to 5 msec with an 8080A-based microcomputer operating at 2 MHz. Sixteen-bit binary multiplications and divisions can be performed even faster. Therefore, it is appropriate for you to consider that the main task of the microcomputer is to perform such computations, and that 0.05% to 0.10% of the time the microcomputer can devote its attention to servicing the interrupting Teletype. The less attractive alternatives are for the microcomputer to either poll the UART or else to wait for a change of state of the UART data ready or transmitter buffer empty flags.

Interrupts are also effective for use with devices that provide data to a microcomputer but which have no buffer of their own to store it. Existing data must be removed from the device and stored in the microcomputer quickly before a new data word can be generated by the device. One example of such a device is an analog-to-digital converter (ADC) in which the conversions are clocked by an external clock at repetitive time intervals.

## References


1. *Microprocessor Buzz Words* (Schweber Electronics Marketing Services, Westbury, N.Y.).
2. LARSEN, D.G., RONY, P.R., and TITUS, J.A., "Microcomputer interfacing: microcomputer I/O devices," *Amer. Lab.* 7 (11), 100 (1975).
3. GRAF, R.F., *Modern Dictionary of Electronics* (Howard W. Sams & Co., Inc., Indianapolis, 1972).
4. LARSEN, D.G. and RONY, P.R., "Computer interfacing: the universal asynchronous receiver/transmitter (UART)," *Amer. Lab.* 7 (2), 113 (1975).

# The vectored interrupt

By David G. Larsen, Peter R. Rony, and Jonathan A. Titus

THIS MONTH, we continue the discussion of computer interrupts, with emphasis upon vector interrupt hardware and software associated with the 8080A microprocessor chip. The three signals used in vector interrupt circuits include INT (input pin 14 on the 8080A chip), INTE (output pin 16), and $\overline{\text{INTA}}$, not available on the 8080A chip but derived externally with additional logic.

A positive clock pulse from an interrupting device supplies a logic 1 state at the INT, or *interrupt request,* input that generates an interrupt request, which the CPU recognizes either at the end of the current instruction being executed or while the CPU is in the halt state. The INTE, or *interrupt enable,* output pin indicates the logic state of the interrupt enable flip-flop present within the 8080A chip. This internal flip-flop can be set (enabled) or cleared (disabled) with the aid of 8080A microcomputer instructions:

363 DI Disable interrupt flip-flop
373 EI Enable interrupt flip-flop

When cleared, the interrupt enable flip-flop inhibits interrupts from being accepted by the CPU. The flip-flop is automatically cleared when an interrupt is accepted; it is also cleared by the RESET input signal applied at pin 12 of the 8080A chip.

The $\overline{\text{INTA}}$, or *interrupt acknowledge,* control signal is generated by applying the INTA and DBIN control signals to a two-input NAND gate (*Figure 1*). A logic 1 at DBIN (output pin 17 on the 8080A chip), or *data bus in,* indicates to external devices that the data bus is in the input mode. The INTA control signal is a positive clock pulse that is generated as a status output with the aid of a status latch connected to the 8080A microprocessor chip[1,2]; we shall talk about the status latch in a subsequent column. The interesting aspect of the $\overline{\text{INTA}}$ control signal is that it permits you to "jam" an interrupt vector instruction byte directly into the instruction register within the 8080A chip. This can only be done during an interrupt, but nevertheless it is a unique and highly interesting operation that is possible with the 8080A microprocessor.

A simple circuit that demonstrates how a single-byte instruction can be jammed into the instruction register is provided in *Figure 2.* Assuming that the interrupt enable flip-flop has been enabled previously by the instruction, 373, the interrupting device must supply a logic 1 input at INT in order to generate an interrupt request. The

*Mr. Larsen, Department of Chemistry, and Dr. Rony, Department of Chemical Engineering, are with the Virginia Polytechnic Institute and State University. Mr. Titus is President, Tychon, Inc.*

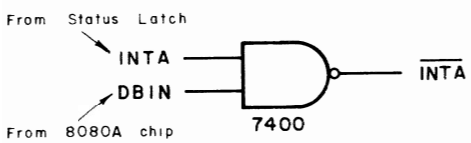


**Figure 1** *The $\overline{\text{INTA}}$ control signal is generated from the DBIN and INTA control signals, only one of which is available on the 8080A chip.*

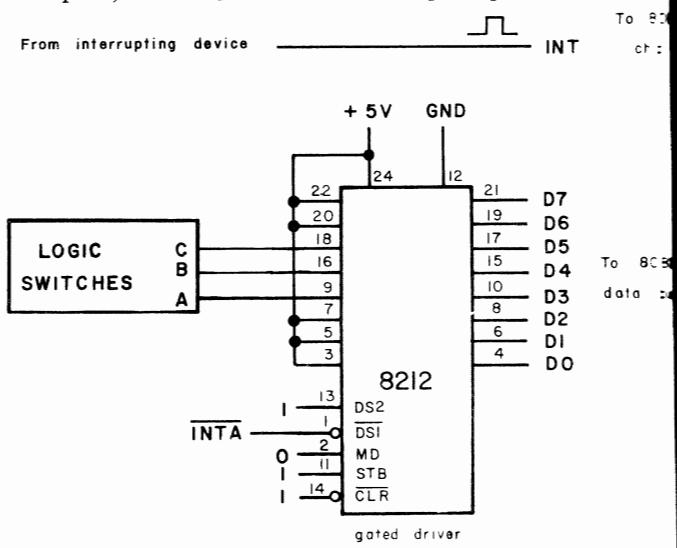


**Figure 2** *Interface circuit for the jamming of a single-byte instruction into the instruction register of an 8080A microprocessor chip.*

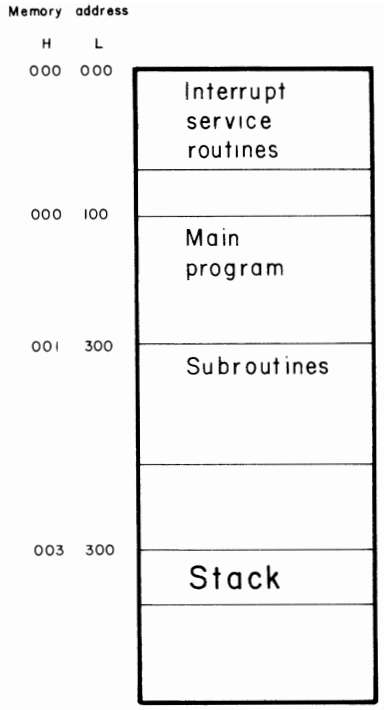


**Figure 3** *Possible memory "map" for 8080A microcomputers.*

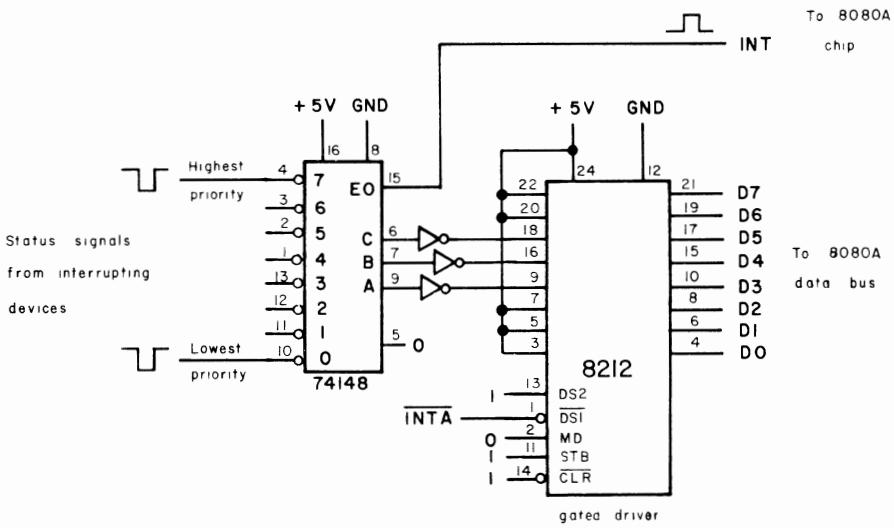


**Figure 4** *Priority interrupt encoding scheme for 8080A microcomputers.*

microcomputer finishes the current instruction, and then generates the interrupt acknowledge signal, $\overline{\text{INTA}}$, which jams the desired vector instruction byte on the data bus and into the instruction register. Although any interruption byte can be jammed into the instruction register during an interrupt, usually the eight instructions listed in *Table 1* are used to produce a useful result. Thus, the first sixty-four memory locations are reserved for *interrupt service routines* or *pointers*, extremely short programs, often consisting of only a single-jump instruction, that tell the 8080 microcomputer what to do or where to go for a specified interrupt condition. Such routines precede the main program and associated subroutines in memory (*Figure 3*). If interrupts or restart instructions are not used, this portion of memory does not have any special significance.

*Figure 4* is probably the simplest priority encoder interrupt circuit that can be used with an 8080 microcomputer. The Intel 8212 chip is used as an 8-bit three-state buffer that inputs the instruction byte into the instruction register. The 74148 8-line-to-3-line priority encoder chip has the truth table shown in *Table 2*.

The purpose of the circuit in Figure 4 is to input the restart instruction, 3Y7, into the microcomputer. Five of the eight inputs to the 8212 chip are tied to a logic 1 state. The remaining three bits supply the encoded vector address of the restart subroutine. By virtue of its truth table, the 74148 priority encoder chip provides eight priority levels. The inputs to this chip should be latched. The chip provides the three-bit binary output that corresponds to the highest valued priority input which is at a logic 0 state. The inverters invert this information to supply the three-bit "Y" component of the restart instruction. If there is a logic 0 at any of the inputs to the 74148 chip, a logic 1 output will be generated at the E0 output (pin 15). This output serves as the input to the interrupt request pin, INT, on the 8080A chip. Upon receiving an interrupt request, the microcomputer responds with an interrupt acknowledge output, $\overline{\text{INTA}}$, that strobes the selected highest priority restart instruction into the instruction register.

## References


1. "Intel 8080 Microcomputer Systems User's Manual" (Intel Corporation, Santa Clara, Sept. 1975).
2. RONY, P.R., LARSEN, D.G., and TITUS, J.A., *Bugbook III. 8080 Microcomputer Interfacing Experiments* (E & L Instruments, Inc., Derby, 1975).


Table 1

| | | |
|---|---|---|
| 307 | RST 0 | Call the subroutine that starts at HI = 000 and LO = 000 |
| 317 | RST 1 | Call the subroutine that starts at HI = 000 and LO = 010 |
| 327 | RST 2 | Call the subroutine that starts at HI = 000 and LO = 020 |
| 337 | RST 3 | Call the subroutine that starts at HI = 000 and LO = 030 |
| 347 | RST 4 | Call the subroutine that starts at HI = 000 and LO = 040 |
| 357 | RST 5 | Call the subroutine that starts at HI = 000 and LO = 050 |
| 367 | RST 6 | Call the subroutine that starts at HI = 000 and LO = 060 |
| 377 | RST 7 | Call the subroutine that starts at HI = 000 and LO = 070 |

Table 2

| Inputs[a] | | | | | | | | Outputs | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | C | B | A | E0 |
| X | X | X | X | X | X | X | 0 | 0 | 0 | 0 | 1 |
| X | X | X | X | X | X | 0 | 1 | 0 | 0 | 1 | 1 |
| X | X | X | X | X | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| X | X | X | X | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| X | X | X | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| X | X | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| X | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

[a]The letter X means that the logic state is irrelevant.