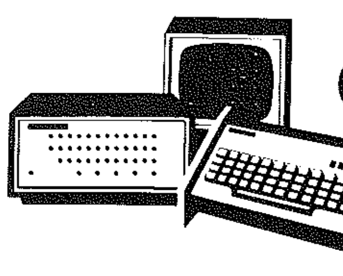

Computer Bits

MICROCOMPUTER INPUT/OUTPUT

By Hal Chamberlin

FROM THE point of view of user convenience, the input and output devices and how they are interfaced to a microcomputer are the most important part of the system. The best processor in the world with a full 64K of memory would still be difficult to use if the only I/O devices were a hexadecimal keyboard and a 6-digit display. Many people would not consider a system complete unless it had a full alphanumeric keyboard, CRT display, printer, and floppy disk although most live with less.

At the actual programming level, the words "input" and "output" simply refer to the hardware and programming technique used to get data from the outside world into the accumulator or memory, and from memory or the accumulator to the outside world respectively. Usually the I/O device—be it a keyboard, printer, or display—makes a conversion between data in TTL logic-level form and data in physical form such as a key depression or print hammer strike. The I/O interface makes a conversion between this data in logic level form and the needs of the microprocessor bus itself. Usually the I/O interface logic connects directly into the computer while the I/O device is physically placed at any convenient location at the end of a cable connected to the interface.

Direct And Memory-Mapped I/O. Any practical microcomputer system can have a number of I/O interfaces. More complex interfaces may have several subsystems, each with its own interface register. Accordingly, a method must be found to address the desired interface register when an I/O operation is performed. A common method used on earlier computers involved specialized input/output instructions. Typically, the instructions were READ, WRITE, CONTROL and TEST. In addition to the operation code, there was an address field of 4 to 8 bits in these instructions. This address field allowed from 16 to 256 different interface registers to be addressed. The READ instruction would read a data word from the addressed interface register into the accumulator of the computer while WRITE would do the opposite. CONTROL specified an operation to be performed with the data and TEST was used to determine if the addressed interface subsystem was still busy completing the

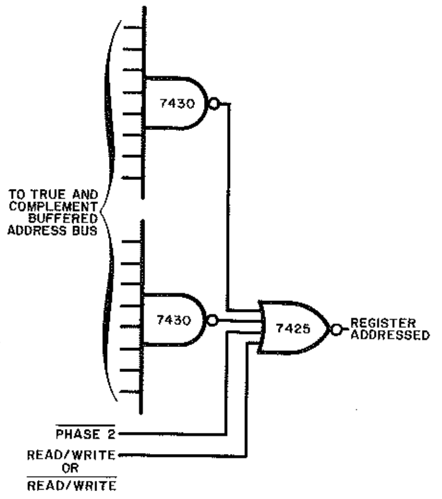

Fig. 1. Address recognizer.

previous operation. TEST usually functioned by skipping the next instruction if the subsystem was free.

Many microcomputers also have specialized I/O instructions although usually only READ and WRITE are provided. CONTROL is accomplished by writing a code into a control register while status testing is performed by reading a status register and looking at the status bits with normal machine instructions. Frequently any I/O register, regardless of its function, is called a *port*.

The 8080 for example has only an IN and an OUT instruction. An 8-bit field in the instruction allows up to 256 interface registers to be addressed. This sounds like a lot until you realize that a complex interface such as a floppy-disk controller might use 8 interface registers. Nevertheless, 256 is almost always ample.

About half of the available microcomputers do not use specialized I/O instructions. Instead, each port register is interfaced to the system bus as if it were a memory location. This is called *memory-mapped* I/O because each I/O interface register corresponds to a memory address. There are several advantages to this method. One is that some operation codes are freed for what may be more useful instructions. Another is that any of the machine's memory reference instructions may be used for manipulating I/O registers, not just load and store. For BASIC language users, the typical PEEK and POKE functions, which are normally used for reading and writing memory, can now be used to operate any I/O device from a BASIC program.

The system bus is somewhat simplified because I/O read and I/O write control signals are no longer needed. Finally, as many I/O addresses as desired may be provided. Typically from 256 to 4096 addresses are set aside from the 65,536 possible memory addresses for I/O functions. Note that processors that normally utilize specialized I/O instructions can also use memory-mapped I/O with its attendant advantages.

Although with memory-mapped I/O each interface register appears as a memory location to the programmer, they often do not act as memory locations. To simplify the circuitry, many of the registers may be *read-only* or *write-only*. An input register from a keyboard, for example, is usually read-only since it does not make much sense to write data to a keyboard. Control and status registers are often write-only and read-only respectively. However if a program needs to know what was last written to a control register, it can save that information somewhere else in regular memory.

I/O Interfacing With Logic. Generally, a bus interface consists of two parts, the address recognizer, and the output latches or input buffers depending on whether it is an output or an input interface.

The typical address recognizer shown in Fig. 1 is basically one multi-input (16) AND gate and can be used on a memory-mapped I/O system such as a KIM 6502. When the proper address pattern is present, and clock phase-2 is high (signifying a valid address), the gate output becomes high. Usually the Read/Write line is also factored in to distinguish between read-only and write-only registers that might be sharing the same address. If several interface registers are on the same board, most of the address bits may be AND'ed together once and factored into several smaller gates, or a decoder, to reduce the number of IC packages required.

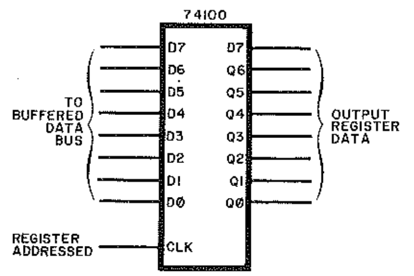

Fig. 2. Simple output port.

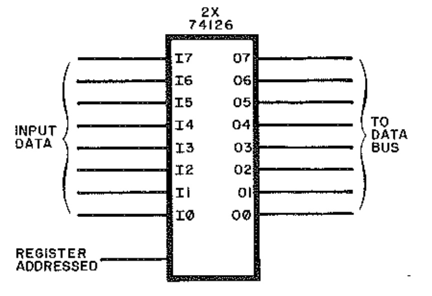

Fig. 3. Simple input port.

An output register is interfaced as shown in Fig. 2. The clock input is triggered at the end of clock phase-2 after the data bus has stabilized.

An input interface is shown in Fig. 3. Data from some source such as a keyboard, or other TTL register, is gated onto the system data bus through the 3-state buffers whenever the interface is addressed during a read cycle. A read/write interface register may be made by connecting an output register to an input interface.

Often in process control and other applications, it is necessary to read and write individual bits that may correspond to separate solenoids, valves, switches, etc. This may be accomplished with ordinary 8-bit input and output ports and the machine's AND, OR, and SHIFT instructions. A simpler way from both software and hardware standpoints however is to use multiplexer and addressable latch elements as shown in Fig. 4A.

Again using the 6502 processor as an example, 16 individually addressable input bits may be interfaced with one IC in addition to the usual address decoder. Note that each bit has its own address and that when read, the bit will appear in position 7 where it may be easily tested. In fact with the 6502, the bit can be tested directly by executing an ASL ADDR where ADDR is the address of the input bit to be tested. This "shift memory" instruction causes the addressed bit to be copied into the carry flag without disturbing the accumulator or other registers. Following this, a conditional branch may be executed. The circuit may be easily expanded to more inputs, each still individually addressable. A large number of inputs may be scanned by using the indexed addressing form of the ASL instruction.

Shown in Fig. 4B, is a one-IC circuit (in addition to the address decoder) for implementing 8 independently controllable output bits. The key to its operation is a relatively inexpensive addressable latch IC—the 9334. The device essentially operates like an 8-bit write-only memory in which the status of each cell is available at a package pin. A particular bit may be controlled by storing the accumulator into the desired bit address with bit-0 of the accumulator set to the desired state. Because of the choice of bit-0 for the data connection, there are easier ways to manipulate the addressed bit. For example, the instruction ASL ADDR will set the addressed bit to zero because bits shifted in on a left shift are zeroes. Similarly, ASR ADDR will set it to one because the data bus will float to ones (a 10,000-ohm resistor from DATA 1 to +5 volts will insure this) during the read prior to shifting. Clearly, memory mapped I/O interfacing is very powerful on a processor like the 6502.

I/O Interface Chips. Another popular method of parallel I/O interfacing, particularly in very small systems, is the use of specialized I/O interface IC's.

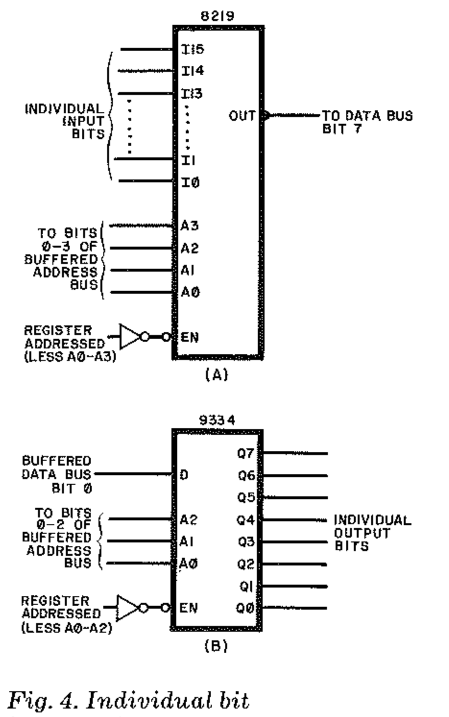

Fig. 4. Individual bit input and output.

Sixteen I/O lines in two groups of eight are provided with the other pins used for connection to the system data bus, addressing, and other functions.

A unique feature of the device is that the function, either input or output, of each of the 16 I/O lines is *programmable*. Two "data direction registers", one for each group of eight I/O lines, may be written into using program instructions to determine whether a line is an output or an input. If a direction bit is a zero, then the corresponding I/O line is an input, otherwise it is an output. Usually these direction bits are set at power-up by the monitor, but some interface designs may make them do double duty by switching between input and output.

A convenient feature of the chip is that *all* registers are read/write. In a memory-mapped I/O system, this allows operations such as incrementing, decrementing, or shifting of output registers directly without loading them into the accumulator. Additional circuitry on the chip handles the generation and acknowledgement of interrupts without the need for external circuitry. Newer versions of parallel I/O chips even have a built-in interval timer. ◇

John Simonton's time-proven design provides two envelope generators VCA, VCO & VCF in a low cost, easy to use package.

Use alone with it's built-in ribbon controller or modify to use with guitar, electronic piano, polytonic keyboards, etc.

The perfect introduction to electronic music and best of all, the Gnome is only $48.95 in easy to assemble kit form. Is it any wonder why we've sold thousands?

( ) Send the GNOME MICRO-SYNTHESIZER Kit ($48.95 plus $2.00 postage)

( ) Send FREE CATALOG

name:________________

address:________________

city:________ state:____ zip:_____

BAC/VISA__ MC__ card no.________

cut out & mail to: (405) 843-9626

PAIA DEPT. 5-P 1020 W. WILSHIRE, OKLAHOMA CITY, OK 73116

CIRCLE NO. 36 ON FREE INFORMATION CARD