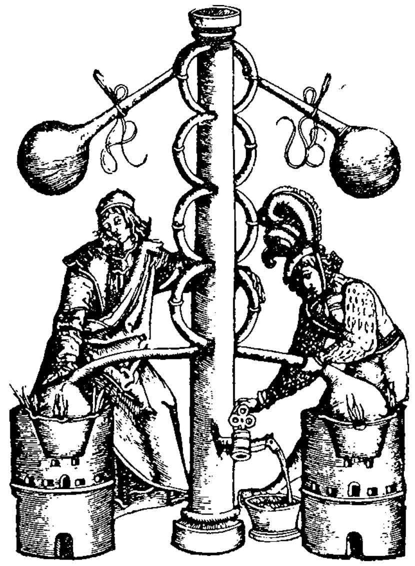

Microcomputer interfacing

What is a logical instruction?

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

MOST MICROCOMPUTERS manipulate information eight bits at a time. For example, the 8080A chip can move the eight bits from internal register to internal register, from internal register to memory, and between the accumulator and an external I/O device. It can also perform arithmetic and logical operations, the former including add, subtract, and compare, and the latter including AND, OR, Exclusive-OR, and complement. In this column, we shall be concerned with the logical operations.

The basic rules governing one-bit logic operations are truth tables. A truth table can be defined as a tabulation that shows the relation of all output logic levels of a digital circuit to all possible combinations of input logic levels in such a way as to characterize the circuit functions completely.₁ The truth tables for the AND, OR, Exclusive-OR, and complement operations are given in *Table* 1. We call these truth tables *one-bit tables* because the data words, A and B, each contain only a single bit.

When we discuss logic instructions, it is useful to employ *Boolean symbols.* Such symbols originate from the subject of *Boolean algebra,* which is the mathematics of logic systems. Alphabetic symbols such as A, B, C, . . . , Q are used to represent logical variables and I and 0 to represent logic states. This particular form of mathematics was originated in England by George Boole in 1847. It did not become widely used until 1938, when Claude Shannon adapted it to analyze multicontact networks for telephone networks.

What you should learn about Boolean algebra are the basic Boolean symbols that are employed in Boolean algebra computations, and thus all digital logic. These symbols include the following:

+	which means *logical addition* and is given the name OR
	which means *logical multiplication* and is given the name AND
⊕	which is given the name *Exclusive-on* or XOR
$\underline{A}$	which means *negation* and is given the name NOT

The negation symbol is a solid bar over a logical variable such as A, B, Q. Thus, the Boolean statement for a two-input AND gate is Q = A • B, or simply Q = AB, where the equality symbol means that the variables or groups of variables on either side of the = symbol are the same, i.e., both are in the same logical state. The symbol operations for the three gates under consideration are summarized in *Table2.*

Table 1

One-bit tables

AND			OR			Exclusive·OR			Complement	
B	A	Q	B	A	Q	B	A	Q	A	Q
0	0	0	0	0	0	0	0	0	0	1
0	1	0	0	1	1	0	1	1	1	0
1	0	0	1	0	1	1	0	1		
1	1	1	1	1	1	1	1	0		

Table 2

Symbol operations

AND	OR	Exclusive·OR	Complement
O· 0 = 0	O+O=O	0 ⊕ 0 = 0	$\overline{0}$ = 1
0.1 = 0	O+1 = 1	0 ⊕ 1 = 1	$\overline{1}$ = 0
1.O = 0	1 + 0 = 1	1 ⊕ 0 = 1	
1 • 1 = 1	1 + 1 = 1	1 ⊕ 1 = 0	

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

Multibit logic operations are treated as many one-bit logic operations. No new principles of logic are involved. The corresponding bits of one binary word logically operate on the corresponding bits of the second binary word to produce an overall multibit logic result. The length of the binary words can be any number of bits: two bits, eight bits, thirty-two bits, etc. Since the 8080A microprocessor performs multibit logic operations on eight-bit words, all of our examples will involve full bytes.

Consider the eight-bit logic variable A. The individual bits in this variable can be labeled as A7, A6, A5, A4, A3, A2, AI, and AO, with AO being the least significant bit (the 2° bit) and A7 being the most significant bit (the 2' bit). Also consider the eight-bit logic variable B which has individual bits that can be labeled as B7, B6, B5, 'B4, B3, B2, BI, and BO. The logic operation A- B = Q means the following eight one-bit logic operations:

AO- BO = QO	A4 - B4 = Q4
AI-BI = QI	AS-BS = *QS*
A2 - B2 = Q2	A6 - B6 = Q6
A3-B3=Q3	A7-B7=Q7

The result of the logic operation is the logic variable Q which has a least significant bit of QO and a most significant bit of Q7. **In** other words, *multibit logic operations are performed bit by bit via a series of one-bit logic operations.* It is easier to perform multibit logic operations if the multibit binary words are placed one under the other. Thus, if A = 11011111, and B = 00100011" then A- B is

```
  1101111 I,
- 00100011,
  00000011,
```

or Q = 00000011,. We have performed a logical AND, and have used the relationships 0 - I = 0 and 1 - 1 = 1 in deriving the final result.

One of the more important uses for multibit logic operations is in situations in which the on-off state of external devices must be monitored. Consider the system of eight devices listed in *Table* 3. We can call the group of eight bits the *status byte* for our system of eight devices. At any instant of time, the status byte will have a specific value. For example, the status byte 11100010, signifies that the pressure is at or below the setpoint, the temperature is above the setpoint, the velocity is at or below the setpoint, etc.

Logical instructions in a digital computer are important because they permit you to determine the following characteristics about the external devices listed in Table 3.

Table 3

Bit position	Device	Logic state information
Bit 0	Pressure sensor	1 = Pressure above setpoint O = Pressure at or below setpolnt
Bit 1	Temperature sensor	1 = Temperature above setpoint O = Temperature at or below setpolnt
Bit 2	Velocity sensor	1 = Velocity above setpoint O = Velocity at or below set point
Bit 3	Flow rate sensor	1 = Flow rate above setpolnt O = Flow rate at or below setpolnt
Bit 4	Concentration sensor	1 = Concentration above sstpolnt O = Concentration at or below setpoint
Bit 5	Valve A	1 = Valve A open O = Valve A closed
Bit 6	Valve B	1 = Valve B open O = Valve B closed
Bit 7	Power	1 = Poweron O= Poweroff

1. Which devices are on, open, or above the setpoint?

2. Which devices are off, closed, or at or below the setpoint?

3. Since the last time we checked, which devices have gone from on to off, open to closed, or above the setpoint to at or below the setpoint?

4. Since the last time we checked, which devices have gone from off to on, closed to open, or at or below the setpoint to above the setpoint?

In other words, using logical instructions you can determine not only the current state of the external devices, but also what changes have occurred since the last time that the devices were interrogated. Since it is not clear how this is done logically, we would like to discuss a specific example based upon these eight devices.

Assume that we have just interrogated all eight devices and have found the current status byte to be 11101010" where the least significant bit, bit 0, is on the far right. One second ago, the status byte was 11101001,. We wish to know what the current state of each device is, which devices have changed state during the last second, and in which direction. The steps that we employ to answer such questions are as follows:

Step 1. *Examine the current status byte. Determine the status of each external device from the logic state of its status bit.*

The current status byte (CSB) is 11101010,. From this value, we conclude that the pressure, velocity, and concentration sensors are all at or below their respective setpoints; the temperature and flow rate sensors are above their respective setpoints; and that valve A, valve B, and power are all on.

Step 2. *Perform an Exclusive-OR operation between the prior status byte (PSB) and the current status byte (CSB). A logic* 1 *in the result will indicate that the logic state of that device has changed.*

The logical operation that we wish to perform is

$$PSB \oplus CSB = QI$$

where PSB = 11101001" CSB

11101010" and Ql is the result of the Exclusive-ox operation. Thus,

```
   11101001   PSB
⊕  11101010   CSB
   00000011   Ql
```

and Ql = 00000011,. We conclude that only the pressure and temperature sensors have changed state.

Step 3. *Perform an* AND *operation between* Q1 *and the prior status byte (PSB). A logic* 1 *in the result indicates a device that has changed state from logic* 1 *to logic* 0.

The logical operation that we wish to perform is

PSB. Ql = Q2

where PSB = 11101001" Ql = 000000 11" and Q2 is the result of the AND operation. We obtain

```
   11101001   PSB
  .00000011   Ql
   00000001   Q2
```

and thus can conclude that the pressure sensor has changed from being above the setpoint to now being at or below the setpoint (logic 1 to logic 0 transition).

Step 4. *Negate (or complement) Q2, and* AND *this complemented result with Q1. A logic* 1 *in the result indicates a device that has changed state from logic 0 to logic 1.*

The logical operation that we now perform is

Q2.Ql = Q3

Since Q2 is 00000001" the complemented value of Q2 must be 1111111O. The result of the AND operation is obtained as follows

```
   11111110   Q2
  .00000011   Ql
   00000010   Q3
```

The result, Q3 = 00000010" permits us to conclude that the temperature sensor has changed from at or below the setpoint to above the setpoint (logic 0 to logic 1 transition).

The reason that we perform such a series of logical operations is to determine what type of corrective actions we must apply to our system if it is not operating properly. If the temperature is below its setpoint, we may have to turn on a heater. If the concentration of a reactant is too high, we may have to turn valve B off and temporarily halt the flow of reactant into the system. If the pressure is above the setpoint, we may have an emergency condition and must shut the power off to the entire system. With a properly interfaced microcomputer, all such decisions can be easily and quickly made under software control and the necessary corrective actions initiated. However, you must be aware of the fact that mechanical and electromechanical devices typically have response times that are much longer than the decision times for a microcomputer. These response times must be taken into account in our software "design," and are important considerations in the field of digital controls. A discussion of response times is beyond the current scope of this column.

Reference

1. GRAF, R.F., *Modern Dictionary of Electronics* (Howard W. Sams & Co., Indianapolis, 1972).

The authors and others will present a special symposium on "Getting Started in Digital Electronics, Computer Interfacing and Microcomputers" as a roundtable discussion at the 1977 Pittsburgh Conference in Cleveland.

Monday, 8:45-12:10, February 28, Little Theater

Charles Wilkins, University of Nebraska; Sam Perone, Purdue University; Stan Crouch, Michigan State University; Gerald Dulaney, Digital Equipment Corp.

Tuesday, 8:45-12:12, March I, Little Theater

Peter Rony, V.P.I. and S.U.; David Larsen, V.P.I. and S.U.; Jonathan Titus, Tychon,

Mr. Larsen and Dr. Rony will conduct a one-day seminar on microcomputer interfacing on either Saturday, February 26 or Sunday, February 27,1977 just prior to the beginning of the Pittsburgh Conference in Cleveland.
