Form 108.1

OPTOMUX and LOCAL CONTROLLER

PROGRAMMING NOTES

OPTO 22

Table of Contents

Listed below is a series of programming notes that you might find useful. Our engineers have chosen some typical OPTOMUX applications and written programs assuming the OPTO 22 LC2 or LC4 (Local Controller) will be controlling OPTOMUX. The programs are written in IBM PC compatible BASIC or FORTH 83, both of which are resident languages in the LC2/LC4. I believe you will find the code contained in these programs to be useful in many control applications.

PROGRAMMING NOTE 1

Application

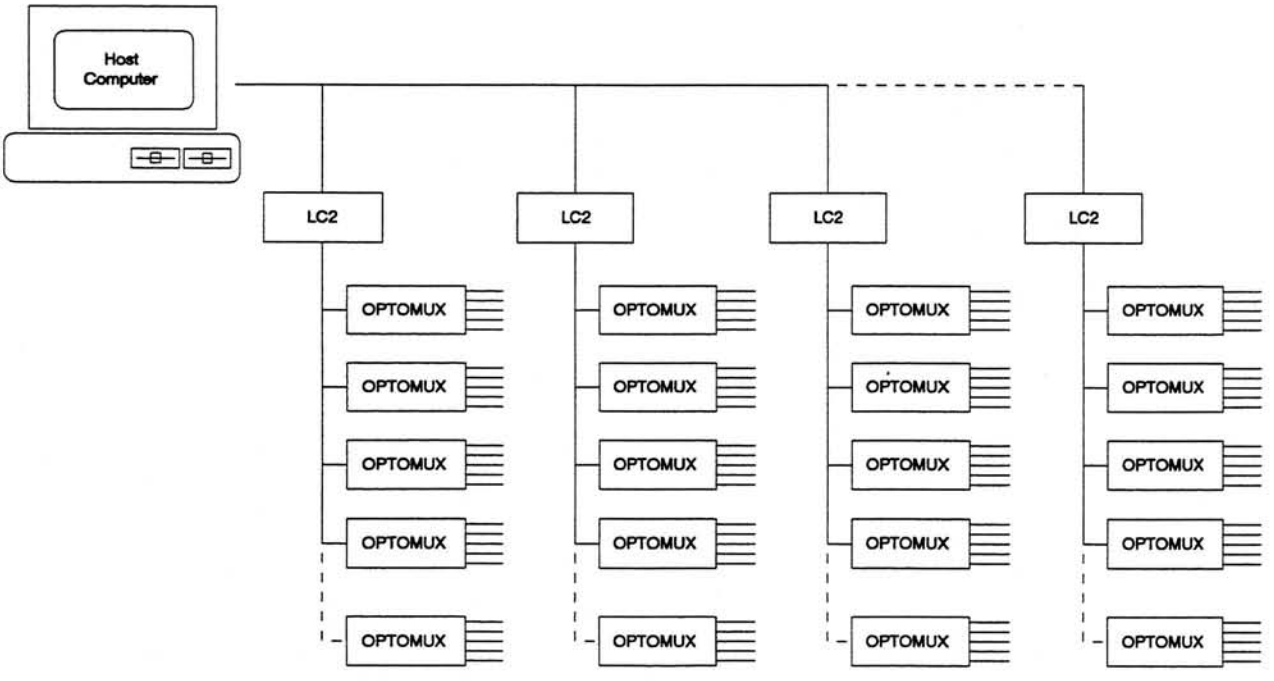
A distributed control/monitoring system with a central host computer having access to the status of the entire system.

Problem

A wide variety of control and monitoring tasks must be performed at different locations throughout a factory. For simplicity and reliablity, it is desirable to have each of these handled locally but a central host computer must be alerted when error conditions occur. Because of the distances involved, it is impractical to run separate communication lines from the host computer to each of the stations. For this reason, all of the local controllers must communicate with the host computer over a single serial communications link.

Solution

The system consists of several LC2/LC4 units multi-dropped off of a single IBM PC serial communications link. Each of the LC2/LC4s performs a local control/monitoring function via a group of OPTOMUX analog and digital controllers. Each of the LC2/LC4s is capable of interrupting the host computer when it has information to relay. Upon receipt of an interrupt message the host computer will poll each of the LC2/LC4s on the link. The LC2/LC4s will respond with a quick message indicating everything is operating normally, or a message indicating the exact nature of the problem that has been detected. The host computer need not wait for an interrupt message to request the status of a LC2/LC4. It can poll any of the LC2/LC4s on the link at any time. Sample programs for both the host computer and LC2/LC4 are given for this application. The samples include all the necessary logic to support communications between the host computer and the LC2/LC4s.



A Distributed Control Network in a Multidrop Arrangement

```
10 '*************************************************************************
20 '*
30 '*
40 '*       EXAMPLE PROGRAM FOR HOST COMPUTER COMMUNICATING WITH MORE
50 '*       THAN ONE LC2/LC4
60 '*
70 '*************************************************************************
80 '
90 '
100 KEY OFF: CLS
110 ON ERROR GOTO 810              'Set Up Error Trapping For Communications Errors
120'
130'       Open The Serial Port
140'
150 OPEN "COM2:19200,N,8,1,CS,CD,RS,DS" AS #1
160'
170'       Define Timeout Counters For Communicating With LC2/LC4
180'
190 TIMEOUT% = 20
200 CLRCOUNT% = 1
210'
220'       Define Number Of LC2/LC4s In The System
230'
240 NUMB.LC% = 5
250'
260'       Set Up Serial Port Interrupt
270'
280 ON COM(2) GOSUB 390
290 COM(2) ON
300'
310'       Wait For Interrupts From LC2/LC4
320'
330 PRINT "OPERATING NORMALLY"
335 GOTO 330
340'*************************************************************************
350'*
360'*       ROUTINE TO HANDLE INTERRUPTS FROM LC2/LC4
370'*
380'*************************************************************************
385'
390 COM(2) OFF                          'Turn Off Communications Interrupts
400 GOSUB 720                           'Clear The Interrupt Message From
405 BEEP                          'Wake Up The Operator
410                                     'The Communications Buffer
420'
430'       Poll To Find Out Who Interrupted Us
440'
450 FOR ADDRESS% = 1 TO NUMB.LC%  'Poll All Of The LC2/LC4s
460 PRINT #1,"?";ADDRESS%               'Send Out Status Request
470 GOSUB 570                           'Get Response
471 PRINT ADDRESS%;" STATUS IS ";
480 PRINT LCMES$                        'Display Status Message
490 NEXT                          'Do The Next LC2/LC4
```

```
500 PRINT #1,CHR$(17);                  'Free The Communications Link
510 COM(2) ON                           'Turn The Interrupts Back On
520 RETURN
530'
540'      Routine To Get A String From LC2/LC4 Without Waiting Forever
550'      Terminates On A Carriage Return Line Feed Or A Timeout
560'
570 LCMES$ = ""                         'Initialize The Input String
580 CHAR$ = ""                          'Initialize Input Character
590 STALL% = 0                          'Initialize Timeout Counter
600'
610'      Get Characters Until A Line Feed Is Received Or Time Out Occurs
620'
630 WHILE (RIGHT$(LCMES$,1) <> CHR$(10)) AND (STALL% < TIMEOUT%)
640 IF LOC(1) = 0 THEN GOTO 660
650 LCMES$ = LCMES$+INPUT$(LOC(1),1)
660 STALL% = STALL%+1                   'Increment Timeout Counter
670 WEND
680 RETURN
690'
700'      Routine To Clear The Communications Buffer
710'
720 FOR I% = 1 TO CLRCOUNT%
730 IF LOC(1) <> 0 THEN JUNK$ = INPUT$(LOC(1),1)
740 NEXT
750 RETURN
760'
770'      It is possible for more than one LC2/LC4 to interrupt us at the same
780'      time which may cause a communications error.  If this occurs ignore
790'      the error and poll the LC2/LC4s to find out what the problem is.
800'
810 RESUME
```

```
10 '*************************************************************************
20 '*
30 '*                  SAMPLE LC2/LC4 PROGRAM
40 '*
50 '*       This program demonstrates a method of allowing multiple LC2/LC4s on the
60 '*       same serial link to interrupt a host computer when a fault condition
70 '*       is detected.  It also provide a mechanism for the host computer to
75 '*       obtain the current status of each LC2/LC4 without being interrupted.
80 '*
90 '*************************************************************************
100'
110 GOSUB 1420                          'Go Initialize Variables
120 GOSUB 390                               'Go Set Up Host Interrupts
130 GOSUB 230                               'Go Do Your Application
140 IF FAULT% = 1 THEN GOSUB 940 'If Any Errors Conditions Alert Host
150 GOTO 130                            'Continue With Your Application
160'
170'*************************************************************************
180'*
190'*             YOUR APPLICATION
200'*
210'*************************************************************************
230'
250'       The code to handle your monitioring or control application goes here.
260'       This routine should set FAULT% to 1 if an error condition is encountered.
270'       The string variable ERROR.MES$ should contain a message describing
280'       the nature of the problem to the host computer.
290'
300'       * * * * * * * * Your Application * * * * * * * * *
300'
310 RETURN
320'*************************************************************************
330'*
340'*        SET UP INTERRUPTS FOR COMMUNICATIONS WITH HOST
350'*
360'*************************************************************************
370'
380'
390 ON KEY(CHR$(17)) GOSUB 840          'Ctrl/Q Means The COM Link Is Free
400 KEY (CHR$(17)) ON                       'Enable The Ctrl/Q Interrupt
410 ON KEY("?") GOSUB 530           '? Means The Host Is Requesting Status
420 KEY ("?") ON                        'Enable ? Interrupt
430 RETURN
440'
450'*************************************************************************
460'*
470'*        ? HAS BEEN RECEIVED SO CHECK TO SEE IF HOST IS TALKING TO US
480'*
490'*************************************************************************
500'
510'       This Routine Is Executed When A ? Has Been Received At The Host Port
520'
530 BUSY.TIMER = 0                          'Reset The Communications Busy Timer
540 BUSY.FLAG% = 1                      'Set The COM Link To Busy
```

```
550'
560'
570 GOSUB 1210                          'Get The Rest Of The Host Message
580'
590'      Test To See If Our Address Follows The ? Character
600'      If The Address Does Not Match Return
610'
620 IF VAL(HOST.MES$) <> ADDRESS@ THEN RETURN
630'
640'      The Host Is Asking Us To Return Current Status
650'
660 IF FAULT% = 0 THEN GOTO 700         'If No Errors Return OK Message
670 PRINT ERROR.MES$                    'Return The Error Message
680 FAULT% = 0                              'Clear The Error Flag Now That We Have
690 RETURN                                  'Reported The Error To The Host
700 PRINT "OK"                              'Return A Message Indicating Everything
710 RETURN                                  'Is Operating Normally
720'
730'*************************************************************************
740'*
750'*       Ctrl/Q HAS BEEN RECEIVED SO CLEAR THE COMMUNICATIONS BUSY FLAG
760'*
770'*************************************************************************
780'
790'      This routine is executed when a Ctrl/Q has been received at the
800'      the host communications port.  This means that the host has completed
810'      all communications, and that we are free to interrupt if we detect
820'      an error condition.
830'
840 BUSY.FLAG% = 0                      'Set Communications Busy Flag To False
850 RETURN
860'
870'*************************************************************************
880'*
890'*       WE HAVE A FAULT CONDITION SO TRY TO ALERT THE HOST
900'*
910'*************************************************************************
920'
930'
940 IF BUSY.FLAG% = 0 THEN GOTO 1090    'If Link Is Not Busy Go Alert The Host
950'
960'      Increment the busy timer to keep track of how long we have waited
970'      to alert the host. If the host does not reqest status from us before
980'      the timer reaches the time out limit, we will assume that something
990'      has gone wrong and that the host needs to be interrupted again.
1000      '
1010      '
1020      BUSY.TIMER = BUSY.TIMER + 1
1030      IF BUSY.TIMER < TIME.OUT% THEN RETURN  'If No Time Out Return
1040      '
1050      ' Send an ! followed by our address to the host to alert him that we have
1060      ' information to report.  Set the communications link to busy so we will
1070      ' wait to be polled before interrupting again.
```

```
1080    '
1090    PRINT "I";ADDRESS@
1100    BUSY.FLAG% = 1                          'Set Communications Link To Busy
1110    BUSY.TIMER = 0                          'Reset Timer After Each Attempt To
1120    RETURN                                  'Interrupt The Host
1130    '
1140    '*************************************************************************
1150    '*
1160    '*     GET A STRING FROM THE HOST COMMUNICATIONS PORT.  RETURN IF NO
1170    '*     CARRIAGE RETURN LINE FEED IS RECEIVED WITHIN SET PERIOD OF TIME.
1180    '*
1190    '*************************************************************************
1200    '
1210    HOST.MES$ = ""                          'Initialize The Input String
1220    CHAR$ = ""                              'Initialize Input Character
1230    WAIT% = 0                               'Initialize Timeout Counter
1240    '
1270    WHILE CHAR$ <> CHR$(10) AND WAIT% < GET.TIME%
1280    CHAR$ = INKEY$                          'Get Char If Any At Host Port
1290    IF CHAR$ < CHR$(14) THEN GOTO 1310      'Skip Line Feeds And Carriage Returns
1300    HOST.MES$ = HOST.MES$+CHAR$             'Append Character To Input
1310    WAIT% = WAIT% + 1                       'Decrement Timeout Counter
1320    WEND
1330    RETURN
1340    '
1350    '*************************************************************************
1360    '*
1370    '*          INITIALIZE VARIABLES
1380    '*
1390    '*************************************************************************
1400    '
1410    '
1420    TIME.OUT% = 100                         'Limit For Counter That Keeps Track
1430                                            'Of How Long The COM Link Has Been Busy
1440    GET.TIME% = 100                         'Limit For Counter That Keeps Track
1450                                            'Of How Long We Have Waited For A
1460                                            'Message From The Host
1470    RETURN
```

PROGRAMMING NOTE 2

Application

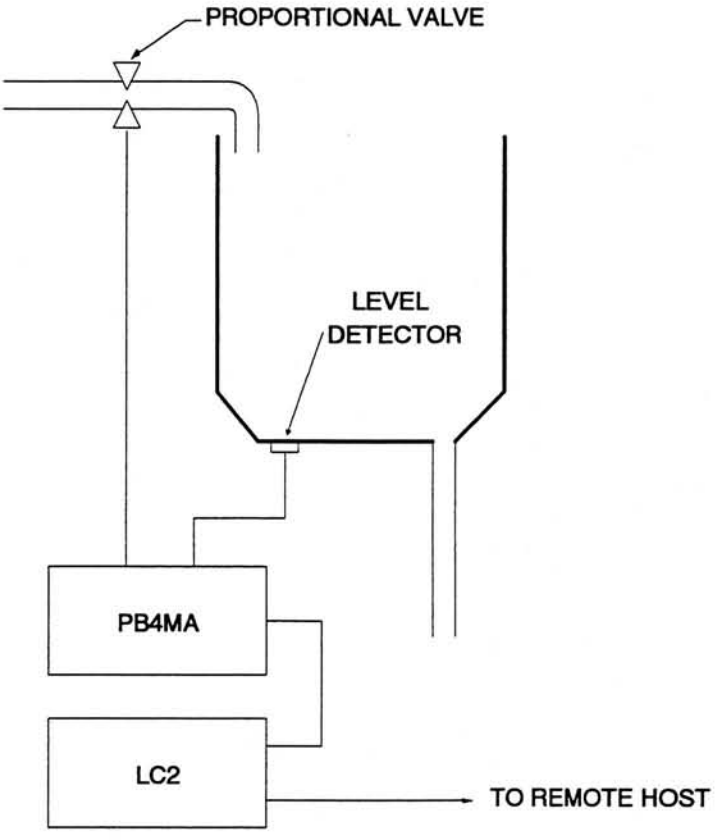
Control water level in a water tower.

Problem

Water towers must have a control system to maintain a safe and constant volume of water in the tank. Because of the remote location of many water towers it is desirable to have a control system located at the tower that can perform the actual control as well as communicate with a centrally located host computer that can monitor the status of several towers. It is also desirable for the host computer to have the ability to change setpoints at each remote tower location.

Solution

The control system will consist of an LC2/LC4 and a four-point analog OPTOMUX unit. The OPTOMUX unit will contain an input analog module to sense tank level and an analog output module to control the fill rate of the tank with a proportional valve. The tank level will be maintained using a PID control loop running on the LC2/LC4. The control program will also allow the host computer to request the current level of the tank and the current valve position, and to change the tank level setpoint.



Water Tower Level Control

```
10 '*************************************************************************
20 '*
30 '*        WATER TOWER LEVEL CONTROL PROGRAM
40 '*
50 '*      This program demonstrates the use of OPTOMUX and LC2/LC4 as Remote
60 '*      Terminal Unit (RTU) for monitoring and control of tank level.
70 '*      Proportional-Intergal-Derivative (PID) control is used to maintain
80 '*      the tank level.
90 '*
100'*************************************************************************
110'
120'
130 GOSUB 1300                          'Initialize Variables
140 GOSUB 1560                          'Initialize OPTOMUX
150 GOSUB 1060                          'Initialize PID Loop Parameters
160 GOSUB 2250                          'Set Up Host Interrupts
170 GOTO 170                            'Wait For Timer Interrupt
180'
190'*************************************************************************
200'*
210'*        PID ROUTINE
220'*
230'*************************************************************************
240'
250'
260 GOSUB 810                               'Sample The Input
270 GOSUB 490                               'Do PID Calculation
280 GOSUB 930                               'Update Analog Output
290 RETURN
300'
310'*************************************************************************
320'*
330'*        PID CALCULATION
340'*
350'*************************************************************************
360'
370'
380'       PID.VALUE = SP + (K * e) + (K * 1 / Ti * S) - (K * Td * M)
390'
400' where: SP = Setpoint
410'              K = Proportional Gain
420'              e = Error
430'              Ti = Integral Time Interval (Reset rate)
440'              S = Integral Of Error
450'              Td = Derivative Time Interval
460'              M = Derivative Of Error
470'
480'
490 OLD.ERROR% = NEW.ERROR%    'Save Last Error
500 NEW.ERROR% = SET.POINT% - INPUT.VALUE%  'Calculate New Error
510'
520'       Calculate Proportional Term
530'
540 PTERM = PROP.GAIN * NEW.ERROR%
550'
560'       Calculate Integral Term
570'
580 SUM.ERROR = SUM.ERROR + NEW.ERROR% * SAMPLE.TIME
590 ITERM = PROP.GAIN * 1/INT.TIME * SUM.ERROR
600'
```

```
610'       Calculate The Derivative Term
620'
630 DTERM = PROP.GAIN * DERV.TIME * ((NEW.ERROR-OLD.ERROR)/SAMPLE.TIME)
640'
650'
660'
670 PID.VALUE = SET.POINT% + PTERM + ITERM - DTERM
680'
690'       Make Sure Value Is Within Output Range (0 - 4095)
700'
710 IF PID.VALUE < 0 THEN PID.VALUE = 0
720 IF PID.VALUE > 4095 THEN PID.VALUE = 4095
730 RETURN
740'
750'*************************************************************************
760'*
770'*        READ ANALOG INPUT
780'*
790'*************************************************************************
800'
810 COMMAND% = 37                  'Read Analog Inputs Command
820 POSI%(0) = 0                   'Specify Position 0
830 POSI%(1) = -1                  'End Of List
840 GOSUB 2060                     'Call The OPTOMUX Driver
850 INPUT.VALUE% = INFO%(0)            'Store The Input Value
860 RETURN
865'
870'*************************************************************************
880'*
890'*        WRITE ANALOG OUTPUT
900'*
910'*************************************************************************
920'
930 COMMAND% = 35                  'Write Analog Outputs Command
940 POSI%(0) = 1                   'Specify Position 1
950 POSI%(1) = -1                  'End Of List
960 INFO%(0) = PID.VALUE               'New Output Value
970 GOSUB 2060                     'Call The OPTOMUX Driver
980 RETURN
990'
1000      '*************************************************************************
1010      '*
1020      '*     INITIALIZE PID VALUES
1030      '*
1040      '*************************************************************************
1050      '
1060      SET.POINT% = 0.65 * 4095          'Set Setpoint To 65% Of Full Scale
1070      PROP.GAIN = 2                     'Proportional Gain
1080      INT.TIME = 2                      'Reset Rate (2 mins)
1090      DERV.TIME = 1                     'Derivative Time Interval (1 mins)
1100      SUM.ERROR = 0                     'Integral Of Error
1110      SAMPLE.TIME = 0.2/60              'Analog Sampling Time In Mins
1120      '
1130      ' Set up timer interrupt to do PID routine a constant number of
1140      ' times per second.  The variable SAMPLE.TIME contains the
1150      ' time between samples in minutes.
1160      '
1170      ' ON TIMER function requires units of seconds
1180      '
1190      ON TIMER(SAMPLE.TIME*60) GOSUB 260
```

```
1200    TIMER ON
1210    RETURN
1220    '*****************************************************************
1230    '*
1240    '*     DIMENSION AND INITIALIZE VARIABLES
1250    '*
1260    '*****************************************************************
1270    '
1280    ' Initialize OPTOMUX Driver Parameters
1290    '
1300    ADDR% = 255
1310    COMMAND% = 0
1320    DIM POSI%(15),MODI%(1),INFO%(15)
1330    '
1340    FOR I% = 0 TO 15
1350    POSI%(I%) = 0
1360    INFO%(I%) = 0
1370    NEXT
1380    '
1390    MODI%(0) = 0
1400    MODI%(1) = 0
1410    '
1420    ' Define Address Of OPTOMUX Driver
1430    '
1440    OPTOWARE = 4
1450    RETURN
1460    '
1470    '
1480    '*****************************************************************
1490    '*
1500    '*     INITIALIZE OPTOMUX
1510    '*
1520    '*****************************************************************
1530    '
1540    ' Send A Power Up Clear Command
1550    '
1560    COMMAND% = 0                    'Power Up Clear Command
1570    ADDR% = 255                     'OPTOMUX Address Is 255
1580    GOSUB 2060                      'Call The OPTOMUX Driver
1590    '
1600    ' Send A Reset Command
1610    '
1620    COMMAND% = 1                    'Reset Command
1630    GOSUB 2060                      'Call The OPTOMUX Driver
1640    '
1650    ' Configure All Positions Except 0 To Be Outputs
1660    '
1670    ' Position 0 = Level Detector Input
1680    ' Position 1 = Filler Valve Output
1690    '
1700    FOR I% = 0 TO 14
1710    POSI%(I%) = I%+1                'Specify Output Positions
1720    NEXT
1730    POSI%(15) = -1                  '-1 Indicates No More Positions
1740    COMMAND% = 8                    'Command Is Configure As Outputs
1750    GOSUB 2060                      .'Call The OPTOMUX Driver
1760    RETURN
1780    '
```

```
2010    '*********************************************************************
2020    '*
2030    '*     CALL THE OPTOMUX DRIVER
2040    '*
2050    '*********************************************************************
2060    '
2090    CALL OPTOWARE(ERRORS%,ADDR%,COMMAND%,POSI%(0),MODI%(0),INFO%(0))
2100    RETURN
2110    '
2120    '
2130    '*********************************************************************
2140    '*
2150    '*SET UP INTERRUPTS FOR COMMUNICATIONS WITH HOST
2160    '*
2170    '*********************************************************************
2180    '
2190    ' The ON KEY statement is used to interrupt off of the appearance of
2200    ' specific characters at the host communications port.  The letter "S"
2210    ' will be used to request the current status of tank.  The letter "C"
2220    ' will cause execution of the change set point routine.
2230    '
2240    '
2250    ON KEY("S") GOSUB 2420        'Report status when S is sent by host
2260    ON KEY("C") GOSUB 2590        'Change setpoint when C is sent by host
2270    KEY ("S") ON                  'Enable status interrupt
2280    KEY ("C") ON                  'Enable change setpoint interrupt
2290    RETURN
2300    '
2310    '*********************************************************************
2320    '*
2330    '*     HANDLE HOST REQUEST FOR CURRENT STATUS
2340    '*
2350    '*********************************************************************
2360    '
2370    ' This routine is executed when the letter "S" is received at the
2380    ' the host communications port.
2390    '
2400    ' Send The Last Value Read For The Tank Level
2410    '
2420    PRINT INPUT.VALUE%;CHR$(13);
2430    '
2440    ' Send The Last Value Output To Valve Expressed As % Of Full Scale
2450    '
2460    PRINT PID.VALUE/4095;CHR$(13);
2470    RETURN
2480    '
2490    '
2500    '*********************************************************************
2510    '*
2520    '*     HANDLE HOST REQUEST TO CHANGE SETPOINT
2530    '*
2540    '*********************************************************************
2550    '
2560    ' This routine is executed when the letter "C" is received at the
2570    ' the host communications port.
2580    '
2590    GOSUB 2870                    'Get the string sent by the host
2600    NEW.SP% = VAL(HOST.MES$)      'Convert string to an integer
```

```
2610    PRINT HOST.MES$;                        'Echo the new value back to host
2620    GOSUB 2870                              'Get host response
2630    '
2640    ' Test to see if host confirmed setpoint value.  If not return
2650    ' without changing setpoint.
2660    '
2670    IF HOST.MES$ <> "ok"+CHR$(13) THEN RETURN
2680    '
2690    ' Update The Setpoint Value
2700    '
2710    SET.POINT% = NEW.SP%
2720    RETURN
2730    '
2740    '*************************************************************************
2750    '*
2760    '*     Get a string from the host communications port
2770    '*
2780    '*************************************************************************
2790    '
2800    ' This routine is used to get a string from the host port without
2810    ' disabling the ON TIMER interrupt.  The INPUT statement can not
2820    ' be used because the TIMER interrupt is suspended while executing the
2830    ' INPUT statement, and is not enabled until a carriage return is sent to
2840    ' terminate the input string.  This routine will wait for the carriage
2850    ' return for a set period of time and then return.
2860    '
2870    HOST.MES$ = ""                          'Initialize The Input String
2880    CHAR$ = ""                              'Initialize Input Character
2890    WAIT% = 0                               'Initialize Timeout Counter
2900    '
2910    ' Get Characters Until A Carriage Return Is Received Or Time Out Occurs
2920    '
2930    WHILE CHAR$ <> CHR$(13) AND WAIT% < 1000
2940    CHAR$ = INKEY$                          'Get Char If Any At Host Port
2950    HOST.MES$ = HOST.MES$+CHAR$             'Append Character To Input
2960    WAIT% = WAIT%+1                         'Increment Timeout Counter
2970    WEND
2980    RETURN
```

PROGRAMMING NOTE 3

Application

Telemetry systems are used to sense conditions on one end of the link and duplicate those conditions at the other end of the link. The lines are usually multiplexed so that a great number of I/O points can be sensed and their status transmitted on a few wires over large distances.

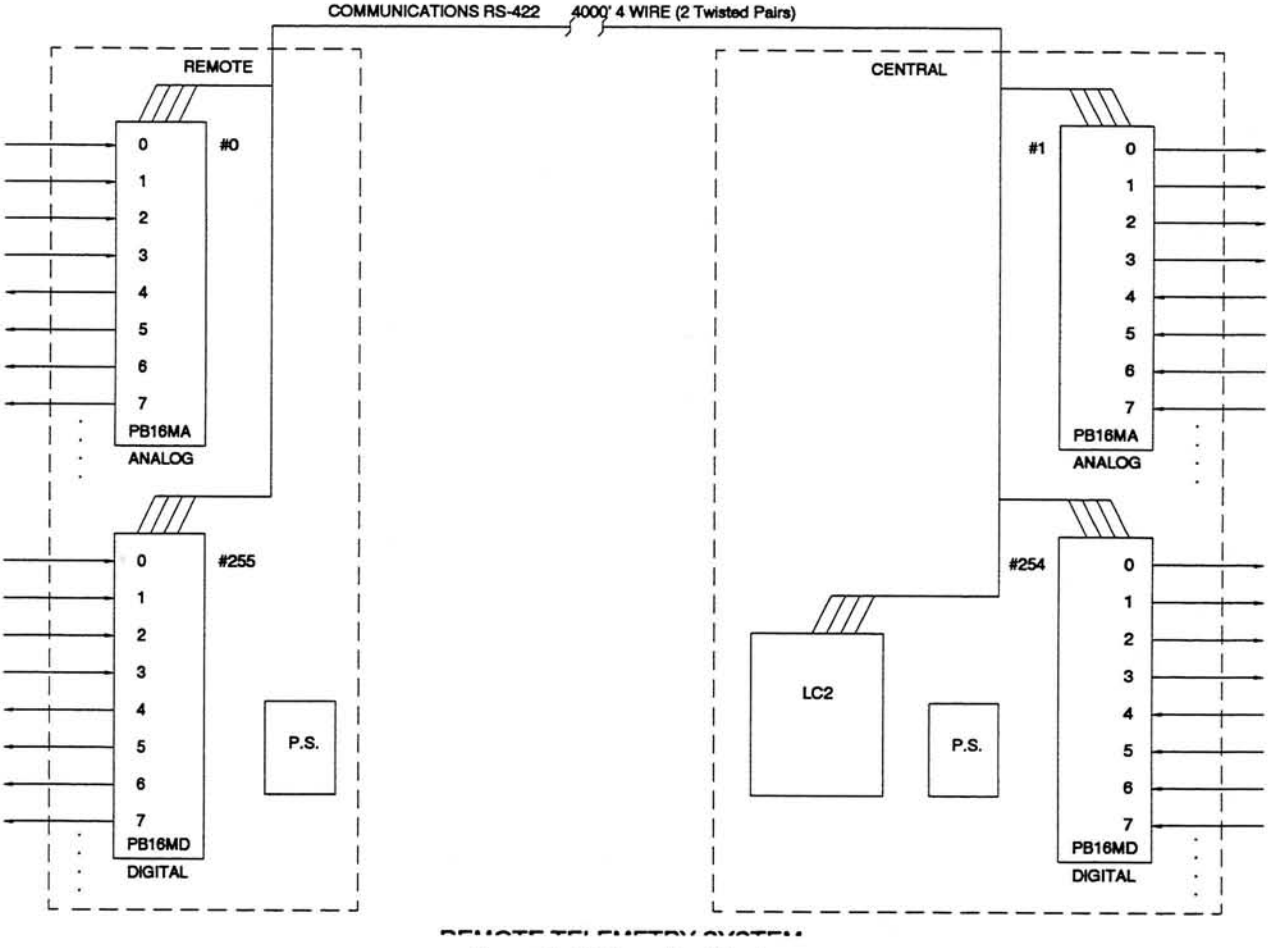
Problem

An example, would be the control center for a petrochemical tank farm, where the farthest tank may be up to one mile from the control room. Each tank may have several control lines to activate pumps and valves, and several signal lines to sense status of limit switches and the level of the tank. The control center may have a large graphic panel indicating the status of each valve and pump. There is also a control panel containing switches to activate pumps and valves. Another panel contains the indicators which display current tank levels. If a set of wires were used for each switch or indicator and then routed to the appropriate tank, a tremendous amount of wiring would need to be installed at a great cost. New additions or changes would require more wires to be added and existing wires to be cut and removed. A telemetry system would greatly reduce the amount of wires while changes can be implemented quickly by connecting to the existing telemetry wires.

Solution

This Example Requires A System Which is flexible, can handle many points, communicate over great distances with few wires, able to transmit analog and discrete signals, allow ease of expansion and installation, and be economical. An Opto 22 OPTOMUX system combined with the LC2/LC4 Local Controller meets all these requirements.

The following example, illustrates a simple telemetry system made up of a remote site and a central site. The remote site is equipped with one B1/PB16H OPTOMUX board for discrete I/O and one B2/PB16AH OPTOMUX board for analog I/O. The central site is consisted of one B1/PB16H OPTOMUX board, one B2/PB16AH OPTOMUX board, and one LC2/LC4 Local Controller. Each OPTOMUX board at the central site corresponds to each OPTOMUX board at the remote site. For clarity and ease of programming, the input channels on each board at a site correspond to the output channels at the equivalent board at the opposite site. The LC2/LC4 is used to poll the OPTOMUX units for the current input status at one site then activate the corresponding outputs at the opposite site.



Remote Telemetry System

```
10 '*****************************************************************
20 '*
30 '*       TELEMETRY PROGRAM FOR USING LC2/LC4 AS A WIRE ELIMINATOR
40 '*
50 '*
60 '*****************************************************************
70 '*****************************************************************
80 '*
90 '*       MAIN PROGRAM
100'*
110'*
120'*****************************************************************
130 GOSUB 2100                                  'Dimension Arrays
140 GOSUB 2490                                  'Assign Constant Values
150 GOSUB 2200                                  'Initialize Variables
160 GOSUB 1710                                  'Send A Power Up Clear
170 GOSUB 1870                                  'Configure The Outputs
180 ON TIMER(0.4) GOSUB 510                         'Do Digitals On Interrupt
190 TIMER ON                                    'Enable The Timer
200 GOSUB 880                                       'Do Analog Boards
210 GOSUB 200                                   'Loop And Read Inputs Again
220 END
230'*****************************************************************
240'*
250'*       SUBROUTINES
260'*
270'*
280'*****************************************************************
290'*****************************************************************
300'*
310'*       LIMIT CHECK FOR ANALOG INPUTS
320'*
330'*
340'*****************************************************************
350 FOR L% = 0 TO 3                             'Remote Inputs
360 IF INFO%(L%)  0 THEN INFO%(L%) = 0              'Lower Limit
370 IF INFO%(L%) 4095 THEN INFO%(L%) = 4095 'Upper Limit
380 NEXT
390 RETURN
400 FOR L% = 4 TO 7                             'Central Inputs
410 IF INFO%(L%)  0 THEN INFO%(L%) = 0              'Lower Limit
420 IF INFO%(L%)  4095 THEN INFO%(L%) = 4095    'Upper Limit
430 NEXT
440 RETURN
450'*****************************************************************
460'*
470'*       PROCESS DIGITAL BOARDS
480'*
490'*
500'*****************************************************************
510'
520'       Remote To Central
530'
540 FOR P% = 0 TO 3                             'Define The Inputs
550 POSITI%(P%) = P%
560 NEXT
570 POSITI%(4) = -1                                 'End Of List
580 ADDRI% = RDBOARD%                               'Read Remote Digital
590 COMMI% = BREAD%                             'Binary Read
600 CALL OPTOWARE (ERRI%,ADDRI%,COMMI%,POSITI%(0),MODIFI%(0),INFOI%(0))
```

```
610 IF ERRI% < 0 THEN GOSUB 1490
620 ADDRI% = CDBOARD%                          'Write To Central

630 COMMI% = BWRITE%                 'Binary Write
640 CALL OPTOWARE (ERRI%,ADDRI%,COMMI%,POSITI%(0),MODIFI%(0),INFOI%(0))
650 IF ERRI% < 0 THEN GOSUB 1490
660 '
670 '       Central To Remote
680 '
690 FOR P% = 0 TO 3                  'Define The Inputs
700 POSITI%(P%) = P%+4
710 NEXT
720 POSITI%(4) = -1                      'End Of List
730 ADDRI% = CDBOARD%                    'Read Central Digital
740 COMMI% = BREAD%                  'Binary Read
750 CALL OPTOWARE (ERRI%,ADDRI%,COMMI%,POSITI%(0),MODIFI%(0),INFOI%(0))
760 IF ERRI% < 0 THEN GOSUB 1490
770 ADDRI% = RDBOARD%                    'Write To Remote
780 COMMI% = BWRITE%                 'Binary Write
790 CALL OPTOWARE (ERRI%,ADDRI%,COMMI%,POSITI%(0),MODIFI%(0),INFOI%(0))
800 IF ERRI% < 0 THEN GOSUB 1490
810 RETURN
820 '*************************************************************************
830 '*
840 '*       PROCESS ANALOG BOARDS
850 '*
860 '*
870 '*************************************************************************
880 '
890 '       Remote To Central
900 '
910 FOR I% = 0 TO 3                  'Define The Inputs
920 POSIT%(I%) = I%
930 NEXT
940 POSIT%(4) = -1                       'End Of List
950 ADDR% = RABOARD%                 'Read Remote Analog
960 COMM% = AREAD%                       'Read Analog
970 CALL OPTOWARE (ERR%,ADDR%,COMM%,POSIT%(0),MODIF%(0),INFO%(0))
980 IF ERR% < 0 THEN GOSUB 1360
990 GOSUB 350                            'Check Limits
1000        ADDR% = CABOARD%                 'Write To Central
1010        COMM% = AWRITE%                  'Write Analog
1020        CALL OPTOWARE (ERR%,ADDR%,COMM%,POSIT%(0),MODIF%(0),INFO%(0))
1030        IF ERR% < 0 THEN GOSUB 1360
1040        '
1050        ' CENTRAL TO REMOTE
1060        '
1070        FOR I% = 0 TO 3                  'Define The Inputs
1080        POSIT%(I%) = I%+4
1090        NEXT
1100        POSIT%(4) = -1                   'End Of List
1110        ADDR% = CABOARD%                 'Read Central Analog
1120        COMM% = AREAD%                   'Analog Read
1130        CALL OPTOWARE (ERR%,ADDR%,COMM%,POSIT%(0),MODIF%(0),INFO%(0))
1140        IF ERR% < 0 THEN GOSUB 1360
1150        GOSUB 400                        'Check Limits
1160        ADDR% = RABOARD%                 'Write To Remote
1170        COMM% = AWRITE%                  'Analog Write
1180        CALL OPTOWARE (ERR%,ADDR%,COMM%,POSIT%(0),MODIF%(0),INFO%(0))
1190        IF ERR% < 0 THEN GOSUB 1360
```

```
1200    RETURN
1210    '*********************************************************************
1220    '*
1230    '*      CALL THE OPTOMUX DRIVER
1240    '*

1250    '*
1260    '*********************************************************************
1270    CALL OPTOWARE (ERR%,ADDR%,COMM%,POSIT%(0),MODIF%(0),INFO%(0))
1280    IF ERRI% < 0 THEN GOSUB 1360
1290    RETURN
1300    '*********************************************************************
1310    '*
1320    '*      COMMUNICATIONS ERROR MESSAGE ROUTINE
1330    '*
1340    '*
1350    '*********************************************************************
1360    PRINT "*******************************************"
1370    PRINT " DATE: ";DATE$;" TIME: ";TIME$
1380    PRINT " COMMUNICATION ERROR: #";ERR%
1390    PRINT " ADDRESS: ";ADDR%;"  COMMAND: ";COMM%
1400    PRINT "*******************************************"
1410    IF ERR% = -1 THEN GOSUB 1620   'Restart On Power Failure
1420    RETURN
1430    '*********************************************************************
1440    '*
1450    '*      COMMUNICATIONS ERROR MESSAGE ROUTINE ALTERNATE
1460    '*
1470    '*
1480    '*********************************************************************
1490    PRINT "*******************************************"
1500    PRINT " DATE: ";DATE$;" TIME: ";TIME$
1510    PRINT " COMMUNICATION ERROR: #";ERRI%
1520    PRINT " ADDRESS: ";ADDRI%;" COMMAND: ";COMMI%
1530    PRINT "*******************************************"
1540    IF ERR% = -1 THEN GOSUB 1620   'Restart On Power Failure
1550    RETURN
1560    '*********************************************************************
1570    '*
1580    '*      RESTART ROUTINE FOR POWER FAILURES
1590    '*
1600    '*
1610    '*********************************************************************
1620    GOSUB 1710                     'Power Up Clear All
1630    GOSUB 1870                     'Configure All Outputs
1640    RETURN
1650    '*********************************************************************
1660    '*
1670    '*      SEND POWER UP CLEAR COMMAND TO ALL BOARDS
1680    '*
1690    '*
1700    '*********************************************************************
1710    COMMI% = PUC%                  'Power Up Clear Command
1720    ADDRI% = RDBOARD%              'Remote Digital Board
1730    GOSUB 1270                     'Call OPTOMUX Driver
1740    ADDRI% = CDBOARD%              'Central Digital Board
1750    GOSUB 1270                     'Call OPTOMUX Driver
1760    ADDRI% = RABOARD%              'Remote Analog Board
1770    GOSUB 1270                     'Call OPTOMUX Driver
1780    ADDRI% = CABOARD%              'Central Analog Board
```

```
1790    GOSUB 1270                      'Call OPTOMUX Driver
1800    RETURN
1810    '*****************************************************************
1820    '*
1830    '*     CONFIGURE OUTPUTS OF BOARDS
1840    '*
1850    '*
1860    '*****************************************************************

1870    COMMI% = CFGOUT%            'Configure Outputs Command
1880    FOR I% = 0 TO 11            'Set 4 Thru 7 And Spares To Outputs
1890    POSITI%(I%) = I%+4
1900    NEXT
1910    POSITI%(12) = -1            'End Of List
1920    ADDRI% = RABOARD%           'Configure Remote Analog Board
1930    GOSUB 1270                  'Call OPTOMUX Driver
1940    ADDRI% = RDBOARD%           'Configure Remote Digital Board
1950    GOSUB 1270                  'Call OPTOMUX Driver
1960    FOR I% = 0 TO 3             'Set 1 Thru 3 And Spares To Outputs
1970    POSITI%(I%) = I%
1980    NEXT
1990    ADDRI% = CDBOARD%           'Configure Central Digital Board
2000    GOSUB 1270                  'Call OPTOMUX Driver
2010    ADDRI% = CABOARD%           'Configure Central Analog Board
2020    GOSUB 1270                  'Call OPTOMUX Driver
2030    RETURN
2040    '*****************************************************************
2050    '*
2060    '*     DIMENSION ARRAYS ROUTINE
2070    '*
2080    '*
2090    '*****************************************************************
2100    DIM POSIT%(15), POSITI%(15)  '16 Element Array
2110    DIM INFO%(15), INFOI%(15)    '16 Element Array
2120    DIM MODIF%(1), MODIFI%(1)    '2 Element Array
2130    RETURN
2140    '*****************************************************************
2150    '*
2160    '*     ROUTINE TO INITIALIZE VARIABLES
2170    '*
2180    '*
2190    '*****************************************************************
2200    ERR% = 0                    'Set Everybody Initially To 0
2210    ADDR% = 0
2220    COMM% = 0
2230    MODIF%(0) = 0
2240    MODIF%(1) = 0
2250    FOR I% = 0 TO 15
2260    POSIT%(I%) = 0
2270    INFO%(I%) = 0
2280    NEXT
2290    ERR% = 0                    'Set Everybody Initially To 0
2300    ADDRI% = 0
2310    COMMI% = 0
2320    MODIFI%(0) = 0
2330    MODIFI%(1) = 0
2340    FOR I% = 0 TO 15
2350    POSITI%(I%) = 0
2360    INFOI%(I%) = 0
2370    NEXT
```

```
2380    '
2390    ' Other System Variables
2400    '
2410    ERRFLG% = 0                     'No Errors With OPTOMUX Driver
2420    RETURN
2430    '*************************************************************************
2440    '*
2450    '*     ASSIGN CONSTANTS ROUTINE
2460    '*
2470    '*
2480    '*************************************************************************

2490    ' Define Addresses
2500    RDBOARD% = 255                  'Remote Digital Board
2510    CDBOARD% = 254                  'Central Digital Board
2520    RABOARD% = 0                    'Remote Analog Board
2530    CABOARD% = 1                    'Central Analog Board
2540    '
2550    ' OPTOWARE Command Constants
2560    '
2570    OPTOWARE = 4                    'Address Of OPTOMUX Driver
2580    PUC% = 0                        'Power Up Clear
2590    RESET% = 1                      'Reset
2600    CFGOUT% = 8                     'Configure Outputs
2610    BWRITE% = 65                    'Binary Write Command
2620    BREAD% = 64                     'Binary Read Command
2630    AREAD% = 37                     'Read Analog Inputs
2640    AWRITE% = 46                    'Update Analog Outputs
2650    RETURN
```

```
(        TELEMETRY SYSTEM PROGRAM IN FORTH                )
( This program is useful for a remote telemetry system where )
( digital and analog inputs at a remote location are sensed  )
( and corresponding outputs at the local site are activated. )
( Inputs at the local site are also sensed and corresponding )
( outputs at the remote site activated. One LC2/LC4, 2 PB16H, )
( and 2 PB16AH units makeup the system.                      )

( To run this program first use LCTERM and call up           )
( FORTH {see manual}, download this program, then            )
( type RUN. Pressing a key will stop the program.            )


( Define a parameter block for the digital boards            )

PARAMETER-BLOCK DCENTRAL
PARAMETER-BLOCK DREMOTE
PARAMETER-BLOCK ACENTRAL
PARAMETER-BLOCK AREMOTE

(  Define the most used commands as constants                )

65 CONSTANT B_WRITE       ( Binary Write                     )
64 CONSTANT B_READ              ( Binary Read                )
8 CONSTANT CF_OUT         ( Configure Outputs Command        )
37 CONSTANT A_READ              ( Read Analog Inputs         )
35 CONSTANT A_WRITE T     ( Configure Outputs Command        )
37 CONSTANT A_READ              ( Read Analog Inputs         )
35 CONSTANT A_WRITE       ( Write Analog Outputs             )


( Define the analog boards at address 0 and 1                )

: ABOARD AREMOTE PARAMETERS ! 0 ADDRESS !
        ACENTRAL PARAMETERS ! 1 ADDRESS ! ;


( Define the digital boards at address 255 and 254)

:
 DBOARD1 DCENTRAL PARAMETERS ! 254 ADDRESS !;
: DBOARD2 DREMOTE PARAMETERS ! 255 ADDRESS ! ;


( Setup the positions array for outputs of analog board)


: OUT_POSA0
AREMOTE PARAMETERS !
4 0 POSITIONS ! 5 1 POSITIONS ! 6 2 POSITIONS ! 7 3 POSITIONS !
8 4 POSITIONS ! 9 5 POSITIONS ! 10 6 POSITIONS ! 11 7 POSITIONS !
12 8 POSITIONS ! 13 9 POSITIONS ! 14 10 POSITIONS !
15 11 POSITIONS ! -1 12 POSITIONS ! ;

: OUT_POSA1

ACENTRAL PARAMETERS !
0 0 POSITIONS ! 1 1 POSITIONS ! 2 2 POSITIONS ! 3 3 POSITIONS !
8 4 POSITIONS ! 9 5 POSITIONS ! 10 6 POSITIONS ! 11 7 POSITIONS !
12 8 POSITIONS ! 13 9 POSITIONS ! 14 10 POSITIONS !
```

```
15 11 POSITIONS ! -1 12 POSITIONS ! ;

( Set up the positions to read on the analog board)

: IN_POSA0
AREMOTE PARAMETERS !
0 0 POSITIONS ! 1 1 POSITIONS !
2 2 POSITIONS ! 3 3 POSITIONS ! -1 4 POSITIONS ! ;

: IN_POSA1
ACENTRAL PARAMETERS !
4 0 POSITIONS ! 5 1 POSITIONS !
6 2 POSITIONS ! 7 3 POSITIONS ! -1 4 POSITIONS ! ;

( Setup the positions array for outputs of digital board 1)

: OUT_POSD1
2 0 POSITIONS ! 3 1 POSITIONS ! -1 2 POSITIONS ! ;

( Setup the positions array for outputs of digital board 2)

: OUT_POSD2
2 0 POSITIONS ! 3 1 POSITIONS ! -1 2 POSITIONS ! ;

( Word to send a power up clear command)

: PUC 0 COMMAND ! OPTOWARE ;

( Word to send power up clears to all boards)

: PUC_ALL DBOARD1 PUC DBOARD2 PUC ABOARD PUC ;

( Word to send a configures output command)

: CONFIG CF_OUT COMMAND ! OPTOWARE ;

( Word to configure outputs on all boards)

: CONFIG_ALL ABOARD
        DBOARD1 OUT_POSD1 CONFIG
        DBOARD2 OUT_POSD2 CONFIG
        OUT_POSA0 CONFIG
        OUT_POSA1 CONFIG ;

( Word to send a binary read command                    )

: DG_READ B_READ COMMAND ! OPTOWARE ;


( Word to send a binary write command                   )
( value is shifted over twice to correspond with the output )
( modules                                               )

: DG_WRITE B_WRITE COMMAND !
        0 INFO @ 2* 2* 0 INFO ! OPTOWARE ;

( Read Analog Inputs )

: ANL_READ A_READ COMMAND ! OPTOWARE ;
```

```
( Write Analog Outputs                                          )

: ANL_WRITE A_WRITE COMMAND ! OPTOWARE ;

( Process the digital boards                                    )
( Read inputs of board1 and turn on corresponding outputs of board2)
( Read inputs of board2 and turn on corresponding outputs of board1)

: DODIG1 DBOARD1 DG_READ DBOARD2 DG_WRITE ;

: DODIG2 DBOARD2 DG_READ DBOARD1 DG_WRITE ;

( Process the Analog Boards                                     )
( Value in INFO 2 is put in INFO 0 and written out              )

: DOANL ANALOG_IN PARAMETERS !
        ANL_READ 2 INFO @
        ANALOG_OUT PARAMETERS !
        0 INFO ! ANL_WRITE ;

( The main program                                              )

: RUN PUC_ALL                ( Send power up clears             )
        CONFIG_ALL   ( Configure Outputs                        )
        IN_POSA      ( Set up positions array for parm block 1)
        OUT_POSA     ( Set up positions array for parm block 2)
        BEGIN
        DODIG1       ( Process digital board 1                  )
        DOANL        ( Process analog board                     )
        DODIG2       ( Process digital board 2                  )
        DOANL        ( Process analog board                     )

        ?KEY
        UNTIL ;      ( End program when key is pressed          )
```

PROGRAMMING NOTE 4

Application

Alarm/status monitoring systems are used in a variety of industries from water treatment plants to manufacturing plants. The purpose of a monitoring system is to sense discrete events or measure analog values and report changes in status or when boundary limits are reached. Some systems require some sort of action to take place automatically when an alarm condition occurs. Both local and remote systems may be linked to a computer or printer for storing or reporting the occurrence of an alarm condition.

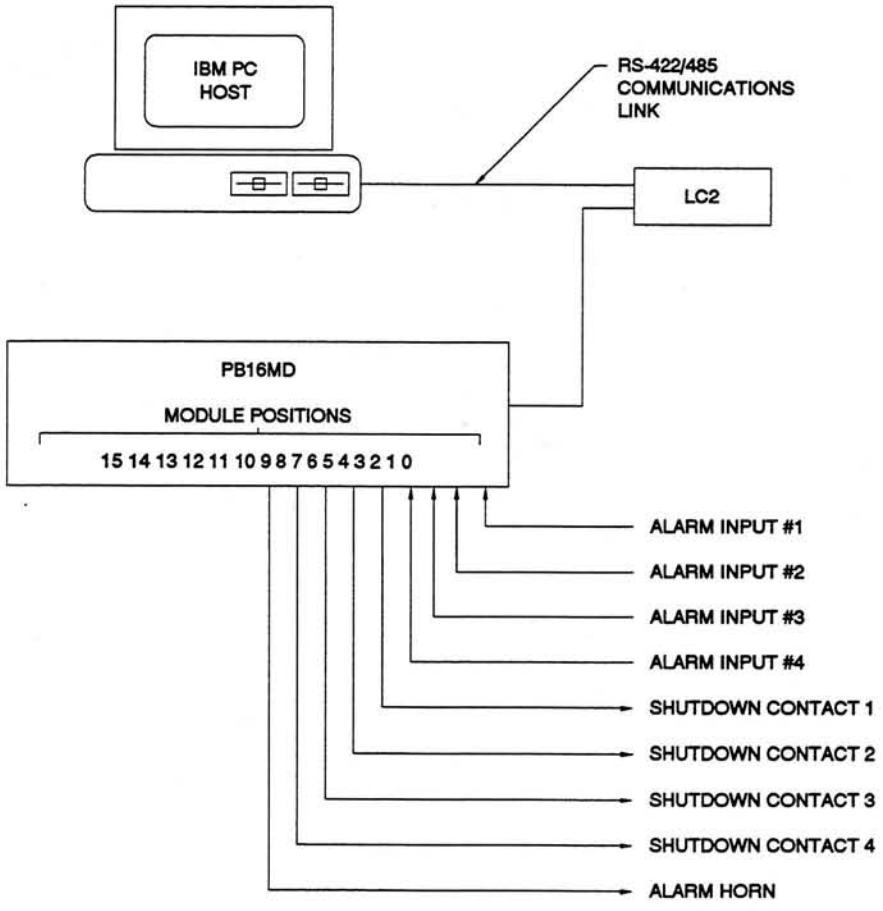
Problem

An example in a manufacturing plant may be the monitoring of machinery that may be operating unattended 24 hours per day. In this example, each machine has a limit switch that is activated during a jam condition. A system is needed that can monitor each switch and upon sensing a jam condition at a particular machine, activate an output which will shutdown that machine until the jam condition is corrected. The monitoring system must also be able to activate a horn and send a message to a host computer which will display the message on a screen located at an operator's desk. An operator can acknowledge the alarm by pressing a key on a keyboard at his desk. Upon detecting the proper key press, the host computer will send an alarm acknowledge message to the monitoring system. When the system receives the alarm acknowledge message, the horn will be disabled. If the jam condition does not go away after a preset time, the horn will be re-enabled. The operator can also select keys on the host for routing the messages to a printer or a disk file.

Solution

The LC2/LC4 Local Controller combined with OPTOMUX makes an ideal, low cost, and versatile system for monitoring alarm conditions and performing local control. The purpose of this programming note is to illustrate the techniques necessary for reporting fault conditions or status changes to a supervisory host computer. These same techniques can be used to record or log data on the host's mass storage device.

The example uses an IBM Personal Computer connected to a LC2/LC4 Local Controller. The LC2/LC4 is connected to an OPTOMUX unit with the proper I/O modules to connect to the limit switches, shutdown switches, and horn. The LC2/LC4 is programmed to sense and control the I/O at the OPTOMUX unit and also interact with the host computer. For the purposes of this example, we will assume an OPTOMUX unit at address 255 will have four inputs which, when active, will indicate an alarm condition. The same OPTOMUX unit also has five outputs which are used for emergency shutdown and sounding a horn.



Alarm Status Monitoring System

```
10 '*****************************************************************
20 '*
30 '*       ALARM PROGRAM - FOR SENSING INPUTS AND TAKING ACTION
40 '*
50 '*
60 '*****************************************************************
70 '*****************************************************************
80 '*
90 '*       MAIN PROGRAM
100'*
110'*
120'*****************************************************************
130 GOSUB 2180                          'Dimension Arrays
140 GOSUB 2550                          'Assign Constant Values
150 GOSUB 2290                          'Initialize Variables
160 GOSUB 1740                          'Send A Power Up Clear
170 GOSUB 1840                          'Configure The Outputs
180 GOSUB 2090                          'Define Timer Interrupt
190 GOSUB 2000                          'Enable Trapping of Keys From Host
200 GOSUB 570                               'Read Inputs Status And Take Action
210 GOTO 200                            'Loop And Read Inputs Again
220 END
230'*****************************************************************
240'*
250'*       SUBROUTINES
260'*
270'*
280'*****************************************************************
290'*****************************************************************
300'*
310'*       CALL THE OPTOMUX DRIVER
320'*
330'*
340'*****************************************************************
350 CALL OPTOWARE(ERRORS%,ADDR%,CMD%,POSIT%(0),MODIF%(0),INFO%(0))
360 IF ERRORS% < 0 THEN GOSUB 1620
370 RETURN
380'*****************************************************************
390'*
400'*       READ ALARM INPUTS
410'*
420'*
430'*****************************************************************
440 ADDR% = ALRMBRD1%                       'Set Address Of Board
450 CMD% = RDSTAT%                          'Read Status Command
460 GOSUB 350                               'Call The OPTOMUX Driver
470 FOR J% = 0 TO 3
480 B%(J%) = INFO%(J%)                  'Put Input Values In B
490 NEXT
500 RETURN
510'*****************************************************************
520'*
530'*       READ INPUTS AND PROCESS CHANGES
540'*
550'*
560'*****************************************************************
570 GOSUB 440                               'Read Input Status
580 FOR I% = 0 TO 3
590 IF A%(I%) <> B%(I%) THEN GOSUB 710   'If Change, Do It
```

```
600 A%(I%) = B%(I%)                          'Update New Status
610 NEXT

620 ALL% = A%(0)+A%(1)+A%(2)+A%(3)           'Sum To Check For 0
630 IF ALL% = 0 THEN GOSUB 1520              'If No Alarms Clear All
640 RETURN
650 '*************************************************************
660 '*
670 '*       TAKE ACTION ON CHANGES
680 '*
690 '*
700 '*************************************************************
710 IF B%(I%) = 0 THEN GOSUB 1240 ELSE GOSUB 1370   'Turn Off  Or On
720 '*       Send Message To Host
730 PRINT DATE$;" ";TIME$;" ";MESSAGE$(I%);" ";STATUS$(I%)
740 RETURN
750 '*************************************************************
760 '*
770 '*       TURN ON THE HORN
780 '*
790 '*
800 '*************************************************************
810 ADDR% = ALRMBRD1%                        'Set Address Of Board With Horn
820 CMD% = TURNON%                           'Activate Outputs Command
830 POSIT%(0) = HORN%                        'Horn Position
840 POSIT%(1) = -1                           'End Of List
850 GOSUB 350                                'Call The OPTOMUX Driver
860 HORNFLG% = 1                             'Set Flag For Horn
870 RETURN
880 '*************************************************************
890 '*
900 '*       TURN OFF THE HORN
910 '*
920 '*
930 '*************************************************************
940 ADDR% = ALRMBRD1%                        'Set Address Of Board With Horn
950 CMD% = TURNOFF%                          'Deactivate Outputs Command
960 POSIT%(0) = HORN%                        'Horn Position
970 POSIT%(1) = -1                           'End Of List
980 GOSUB 350                                'Call The OPTOMUX Driver
990 RETURN
100 '*************************************************************
1010    '*
1020    '*     HORN TIMER SERVICE ROUTINE
1030    '*
1040    '*
1050    '*************************************************************
1060    IF HORNFLG% <> 0 THEN GOSUB 810      'Turn On Horn If Alarm Still Present
1070    TIMER OFF                            'Turn The Timer Off
1080    RETURN
1090    '*************************************************************
1100    '*
1110    '*     SERVICE ROUTINE FOR ACKNOWLEDGE FROM HOST
1120    '*
1130    '*
1140    '*************************************************************
1150    GOSUB 940                            'Turn Off The Horn
1160    TIMER ON                             'Enable Horn Timer
1170    RETURN
1180    '*************************************************************
```

```
1190    '*
1200    '*      TURN OFF OUTPUT
1210    '*
1220    '*
1230    '*************************************************************************
1240    ADDR% = ALRMBRD1%           'Set Address Of Alarm Board
1250    CMD% = TURNOFF%             'Deactivate Output
1260    POSIT%(0) = SHTDWN%(I%)     'Position To Turn Off
1270    POSIT%(1) = -1              'End Of List
1280    GOSUB 350                   'Call The OPTOMUX Driver
1290    STATUS$(I%) = "NORMAL"      'Set Status To Normal
1300    RETURN
1310    '*************************************************************************
1320    '*
1330    '*      TURN ON OUTPUT
1340    '*
1350    '*
1360    '*************************************************************************
1370    ADDR% = ALRMBRD1%           'Set Address Of Alarm Board
1380    CMD% = TURNON%              'Activate Output
1390    POSIT%(0) = SHTDWN%(I%)     'Position To Turn On
1400    POSIT%(1) = -1              'End Of List
1410    GOSUB 350                   'Call The OPTOMUX Driver
1420    GOSUB 810                   'Turn Horn On
1430    HORNFLG% = 1                'Set The Flag That Horn Is On
1440    STATUS$(I%) = "ACTIVE"      'Status Set To Active
1450    RETURN
1460    '*************************************************************************
1470    '*
1480    '*      RESET ALL FLAGS - TURN HORN OFF
1490    '*
1500    '*
1510    '*************************************************************************
1520    TIMER OFF                   'Stop Timer
1530    HORNFLG% = 0                'Reset Horn Flag
1540    GOSUB 940                   'Shutdown Horn
1550    RETURN
1560    '*************************************************************************
1570    '*
1580    '*      COMMUNICATIONS ERROR MESSAGE ROUTINE
1590    '*
1600    '*
1610    '*************************************************************************
1620    PRINT "*****************************************"
1630    PRINT " DATE: ";DATE$;"    TIME: ";TIME$
1640    PRINT "COMMUNICATION ERROR: #";ERRORS%
1650    PRINT " ADDRESS: ";ADDR%;"  COMMAND: ";CMD%
1660    PRINT "*****************************************"
1670    RETURN
1680    '*************************************************************************
1690    '*
1700    '*      SEND POWER UP CLEAR COMMAND
1710    '*
1720    '*
1730    '*************************************************************************
1740    ADDR% = ALRMBRD1%           'Set Board Address
1750    CMD% = PUC%                 'Power Up Clear Command
1760    GOSUB 350                   'Call The OPTOMUX Driver
1770    RETURN
```

```
1780  '*****************************************************************
1790  '*
1800  '*     CONFIGURE OUTPUTS OF BOARD
1810  '*
1820  '*
1830  '*****************************************************************
1840  ADDR% = ALRMBRD1%           'Set Address Of Board
1850  CMD% = CFGOUT%              'Configure Outputs Command
1860  POSIT%(0) = SHTDWN%(0)      'First Output
1870  POSIT%(1) = SHTDWN%(1)      'Second Output
1880  POSIT%(2) = SHTDWN%(2)      'Third Output
1890  POSIT%(3) = SHTDWN%(3)      'Fourth Output
1900  POSIT%(4) = HORN%           'The Horn Output
1910  POSIT%(5) = -1              'End Of List
1920  GOSUB 350                   'Call The OPTOMUX Driver
1930  RETURN
1940  '*****************************************************************
1950  '*
1960  '*     SET UP KEY TO TRAP FROM HOST
1970  '*
1980  '*
1990  '*****************************************************************
2000  ON KEY ("A") GOSUB 1150     'Do Routine To Acknowledge Alarm
2010  KEY ("A") ON                'Enable The Key
2020  RETURN
2030  '*****************************************************************
2040  '*
2050  '*     INITIALIZE TIMER FOR HORN DELAY
2060  '*
2070  '*
2080  '*****************************************************************
2090  ON TIMER(60) GOSUB 1060     'Horn Delay Routine (60 Sec)
2100  TIMER OFF                   'Start With No Delay
2110  RETURN
2120  '*****************************************************************
2130  '*
2140  '*     DIMENSION ARRAYS ROUTINE
2150  '*
2160  '*
2170  '*****************************************************************
2180  DIM POSIT%(15)              '16 Element Array
2190  DIM INFO%(15)               '16 Element Array
2200  DIM MODIF%(1)               '2 Element Array
2210  DIM A%(15),B%(15)           '16 Element Arrays
2220  RETURN
2230  '*****************************************************************
2240  '*
2250  '*     ROUTINE TO INITIALIZE VARIABLES
2260  '*
2270  '*
2800  '*****************************************************************
2290  ERRORS% = 0                 'Set Everybody Initially To 0
2300  ADDR% = 0
2310  CMD% = 0
2320  MODIF%(0) = 0
2330  MODIF%(1) = 0
2340  FOR I% = 0 TO 15
2350  POSIT%(I%) = 0
2360  INFO%(I%) = 0
```

```
2370    A%(I%) = 0
2380    B%(I%) = 0
2390    NEXT
2400    '*

2410    '*      Set Other System Variables
2420    '*
2430    HORNFLG% = 0                    'OK To Turn On Horn
2440    ERRFLG% = 0                     'No Errors With OPTOMUX Driver
2450    FOR I% = 0 TO 3
2460    STATUS$(I%) = "NORMAL"          'Set Alarm Status To Normal
2470    NEXT
2480    RETURN
2490    '*************************************************************************
2500    '*
2510    '*      ASSIGN CONSTANTS ROUTINE
2520    '*
2530    '*
2540    '*************************************************************************
2550    ALRMBRD1% = 255                 'Address Of Alarm Board #1
2560    '*
2570    '*      Define Module Positions
2580    '*
2590    FOR I% = 0 TO 3
2600    ALMINP%(I%) = I%                'Module Positions Of Inputs (ALARM SENSE)
2610    SHTDWN%(I%) = I%+4              'Module Positions Of Outputs (SHUTDOWN)
2620    NEXT
2630    HORN% = 8                       'Position Of Horn
2640    '*
2650    '* OPTOMUX Command Constants
2660    '*
2670    OPTOWARE = 4                    'Address Of OPTOMUX Driver
2680    PUC% = 0                        'Power Up Clear
2690    RESET% = 1                      'Reset
2700    RDSTAT% = 12                    'Read Status
2710    CFGOUT% = 8                     'Configure Outputs
2720    TURNON% = 10                    'Activate Outputs
2730    TURNOFF% = 11                   'Deactivate Outputs
2740    '*
2750    '*      Alarm Messages To Send To Host
2760    '*
2770    MESSAGE$(0) = "***  ZONE 1 ALARM  ***"
2780    MESSAGE$(1) = "***  ZONE 2 ALARM  ***"
2790    MESSAGE$(2) = "***  ZONE 3 ALARM  ***"
2800    MESSAGE$(3) = "***  ZONE 4 ALARM  ***"
2810    RETURN
```

```
10 '*****************************************************************
20 '*
30 '*       IBM PC HOST PROGRAM FOR LC2/LC4 ALARM MONITORING
40 '*
50 '*****************************************************************
60 '*****************************************************************
70 '*
80 '*       MAIN PROGRAM LOOP
90 '*
100'*****************************************************************
110 DIM STATUS$(5),YCOL(5),XROW(5)
120 GOSUB 1400                          'Initialize Variables
130 GOSUB 1620                          'Set Up Screen
140 GOSUB 420                               'Enable The Keys
150 GOSUB 1860                          'Update Screen
160 GOTO 150
170 END
180'*****************************************************************
190'*
200'*       OPENING THE IBM SERIAL PORT
210'*
220'*****************************************************************
230 OPEN "COM2:19200,N,8,1" AS #1
240 ON COM(2) GOSUB 1290            'COM 2 Input Handler
250 COM(2) ON                       'Enable Interrupt
260 STATUS$(1) = "ACTIVE  "         'Set Status
270 RETURN
280'*****************************************************************
290'*
300'*       CLOSING THE IBM SERIAL PORT
310'*
320'*****************************************************************
330 COM(2) OFF                              'Disable Interrupt
340 CLOSE #1                        'Close The Port
350 STATUS$(1) = "INACTIVE"         'Reset Status
360 RETURN
370'*****************************************************************
380'*
390'*       FUNCTION KEY ASSIGNMENT - COM INTERRUPT ASSIGNMENT
400'*
410'*****************************************************************
420 ON KEY(1) GOSUB 970                 'Enable/Disable COM Input
430 ON KEY(2) GOSUB 1060                'Enable/Disable Logging To Disk
440 ON KEY(3) GOSUB 1140                'Enable/Disable Logging To Printer
450 ON KEY(4) GOSUB 1220                'Acknowledge Alarm
460 ON KEY(10) GOSUB 2060           'Exit The Program
470 KEY(1) ON                       'Activate The Keys
480 KEY(2) ON
490 KEY(3) ON
500 KEY(4) ON
510 KEY(10) ON
520 KEY 1,"COMPRT"                  'Display On Bottom
530 KEY 2,"DISK"
540 KEY 3,"PRINTR"
550 KEY 4,"ACKNLG"
560 KEY 10,"EXIT"
570 FOR P% = 5 TO 9                 'Clear Out The Other Keys
580 KEY P%,"      "
590 NEXT
600 KEY ON
```

```
610 ON ERROR GOTO 2000                          'Trap Errors
620 RETURN
630 '*****************************************************************
640 '*
650 '*       LOGGING AN ALARM MESSAGE TO THE DISK
660 '*
670 '*****************************************************************
680 OPEN "ALARMS" FOR APPEND AS #2'Open A File Called ALARMS
690 PRINT #2,MESSG$;                            'Write Message To It
700 CLOSE #2                                    'Close The File
710 RETURN
720 '*****************************************************************
730 '*
740 '*       STORING CONTENTS OF COM PORT IN A VARIABLE
750 '*
760 '*****************************************************************
770 FOR I% = 1 TO 100                           'Delay To Allow Complete
780 NEXT                                        'Message To Arrive
790 '*       Assign Buffer To MESSG$ If Buffer Has It
800 IF LOC(1) <> 0 THEN MESSG$=INPUT$(LOC(1),#1)
810 '*       Condition Message to Mask Junk On First Interrupt After Port Is Opened
820 MESSG$ = CHR$(ASC(MESSG$) AND 127) + RIGHT$(MESSG$,LEN(MESSG$)-1)
830 STATUS$(4) = MESSG$                         'Update Status With Beep
840 RETURN
850 '*****************************************************************
860 '*
870 '*       LOGGING MESSAGES TO THE PRINTER
880 '*
890 '*****************************************************************
900 LPRINT MESSG$                               'Print Message On Printer
910 RETURN
920 '*****************************************************************
930 '*
940 '*       SUBROUTINE FOR FUNCTION KEY 1 - COM PORT ENABLE/DISABLE
950 '*
960 '*****************************************************************
970 IF COMFLG% = 1 THEN COMFLG% = 0 ELSE COMFLG% = 1   'Toggle Flag
980 '*       If Flag Open Port, Else Close It
990 IF COMFLG% = 1 THEN GOSUB 230 ELSE GOSUB 330
1000    RETURN
1010    '*****************************************************************
1020    '*
1030    '*     SUBROUTINE FOR FUNCTION KEY 2 - DISK LOGGING ENABLE/DISABLE
1040    '*
1050    '*****************************************************************
1060    IF DSKFLG% = 1 THEN DSKFLG% = 0 ELSE DSKFLG% = 1   'Toggle Flag
1070    IF DSKFLG% = 1 THEN STATUS$(2) ="ACTIVE  " ELSE STATUS$(2)="INACTIVE"
1080    RETURN
1090    '*****************************************************************
1100    '*
1110    '*     SUBROUTINE FOR FUNCTION KEY 3 - PRINTER LOGGING ENABLE/DISABLE
1120    '*
1130    '*****************************************************************
1140    IF PRNTFLG% = 1 THEN PRNTFLG% = 0 ELSE PRNTFLG% = 1   'Toggle Flag
1150    IF PRNTFLG% = 1 THEN STATUS$(3) = "ACTIVE  " ELSE STATUS$(3) = "INACTIVE"
1160    RETURN
1170    '*****************************************************************
1180    '*
1190    '*     FUNCTION KEY 4 - SEND AN ACKNOWLEDGE
1200    '*
```

```
1210    '*********************************************************************
1220    IF COMFLG% = 1 THEN PRINT #1,"A"   'Send LC2/LC4 An Acknowledge
1230    RETURN
1240    '*********************************************************************
1250    '*
1260    '*     COM2 INPUT HANDLER
1270    '*
1280    '*********************************************************************
1290    COM(2) OFF                         'Turn Off Interrupt
1300    GOSUB 770                          'Get The Input
1310    IF DSKFLG% = 1 THEN GOSUB 680      'Send To Disk If Flag Set
1320    IF PRNTFLG% = 1 THEN GOSUB 900     'Print If Flag Set
1330    COM(2) ON                          'Enable Interrupts
1340    RETURN
1350    '*********************************************************************
1360    '*
1370    '*     INITIALIZE FLAGS AND VARIABLES
1380    '*
1390    '*********************************************************************
1400    DSKFLG% = 0                        'Disk Logging Inactive
1410    PRNTFLG% = 0                       'Printer Inactive
1420    COMFLG% = 0                        'COM Port Inactive
1430    STATUS$(1) = "INACTIVE"            'COM Port Status
1440    STATUS$(2) = "INACTIVE"            'Disk File Inactive
1450    STATUS$(3) = "INACTIVE"            'Printer Inactive
1460    STATUS$(4) = ""                    'No Alarm
1470    MESSG$ = "  "                      'Initialize Message
1480    YCOL(1) = 21                       'Positions Of Status Messages
1490    YCOL(2) = 21
1500    YCOL(3) = 21
1510    YCOL(4) = 21
1520    XROW(1) = 7                        'Position Of COM Status
1530    XROW(2) = 9                        'Position Of Disk Status
1540    XROW(3) = 11                       'Position Of Print Status
1550    XROW(4) = 14                       'Position Of Alarm Status
1560    RETURN
1570    '*********************************************************************
1580    '*
1590    '*     SCREEN STATUS DISPLAY ROUTINE
1600    '*
1610    '*********************************************************************
1620    COLOR 14,1                         'Yellow Letters On Blue Background
1630    CLS                                'Clear The Screen
1640    LOCATE 1,27,0
1650    PRINT "LC2/LC4 ALARMS STATUS DISPLAY";
1660    LOCATE 4,10,0
1670    PRINT "DATE:";
1680    LOCATE 5,10,0
1690    PRINT "TIME:";
1700    LOCATE 7,10,0
1710    PRINT "COM PORT:";
1720    LOCATE 9,10,0
1730    PRINT "DISK FILE:";
1740    LOCATE 11,10,0
1750    PRINT "PRINTER:"
1760    LOCATE 13,10,0
1770    PRINT "CURRENT";
1780    LOCATE 14,10,0
1790    PRINT "ALARM:";
1800    RETURN
```

```
1810    '*************************************************************
1820    '*
1830    '*     UPDATE SCREEN ROUTINE
1840    '*
1850    '*************************************************************
1860    LOCATE 4,16,0                       'Position Of Date
1870    PRINT DATE$                         'Display It
1880    LOCATE 5,16,0                       'Position Of Time
1890    PRINT TIME$                         'Print The Time
1900    FOR J% = 1 TO 4                     'Loop
1910    LOCATE XROW(J%),YCOL(J%),0          'Locate For Each Element
1920    PRINT STATUS$(J%);                  'Print Status
1930    NEXT
1940    RETURN
1950    '*************************************************************
1960    '*
1970    '*     ERROR HANDLER ROUTINE
1980    '*
1990    '*************************************************************
2000    RESUME                              'Just Resume
2010    '*************************************************************
2020    '*
2030    '*     ERROR HANDLER ROUTINE
2040    '*
2050    '*************************************************************
2060    CLOSE                               'Close All Ports And Files
2070    KEY OFF                             'Turn Menu Off
2080    CLS                                 'Clear The Screen
2090    END                                 'That Is All
2100    RETURN                              'Just To Be Consistent
```

PROGRAMMING NOTE 5

Application

Compressor Monitoring System

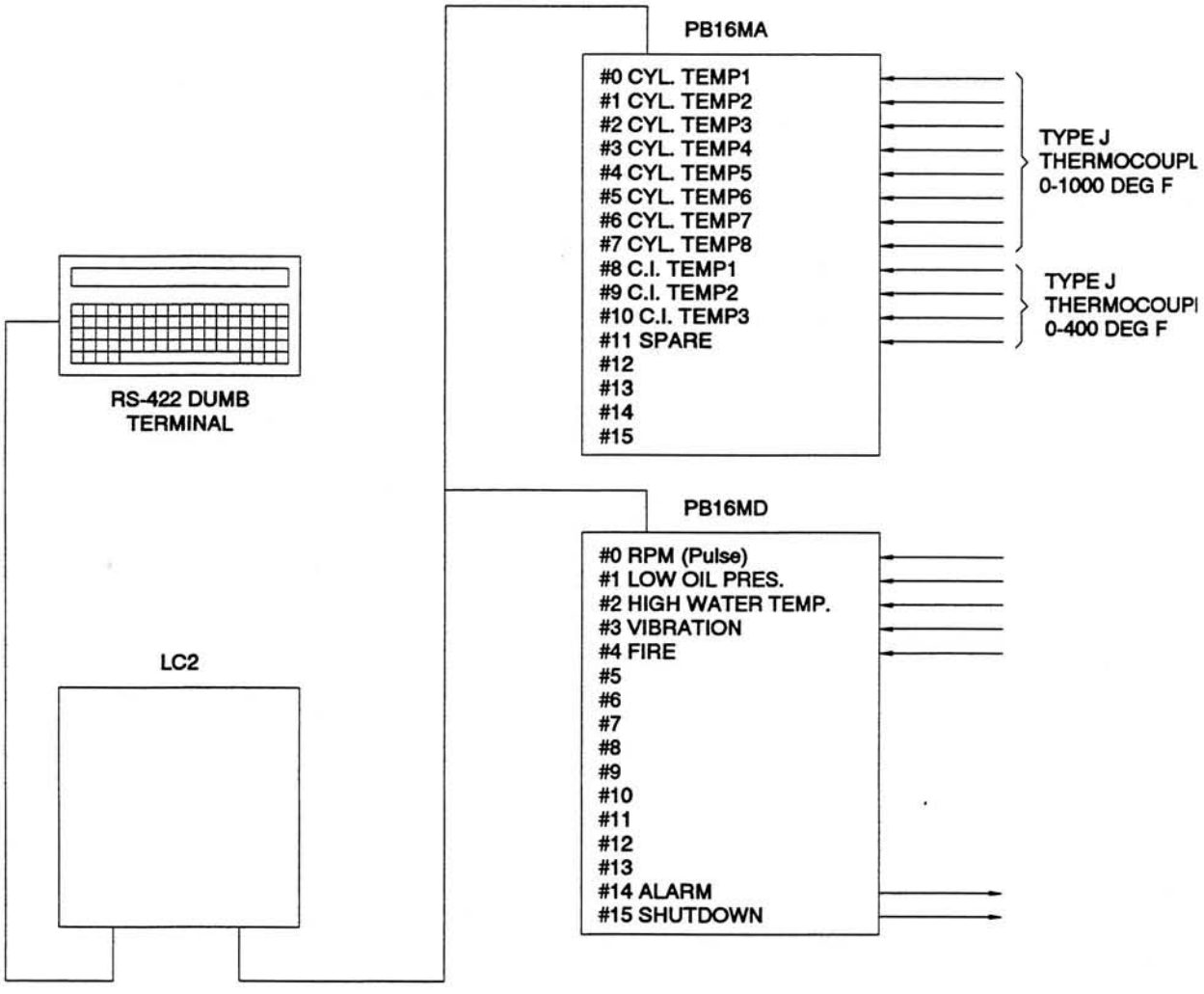
Problem

A monitoring system for a gas compressor used in the petrochemical industry is needed which will monitor cylinder temperatures, speed, and alarm points. Upon sensing conditions of high temperatures, low speed, or any one of four alarm inputs, an output will be activated to alert a maintenance crew. In the event of excessive cylinder temperature, excessive speed, or fire, an output will be activated to shut down the compressor. Provisions are also required for activating the alarm output in the case of a power failure or monitoring system failure. When an alarm occurs, a field technician can connect a portable terminal to the monitoring system, inspect all parameters and active alarms, and change any setpoints.

Solution

This problem can be solved by using one LC2/LC4 Local Controller, one B1/PB16H digital OPTOMUX unit, and one B2/PB16AH analog OPTOMUX unit. The actual I/O consists of one pulse input to measure RPM, eight cylinder temperature inputs, three interstage temperature inputs, four digital inputs for low oil pressure, vibration, high water temperature, and fire alarm, and two digital outputs; one for alarm condition and one for engine shutdown.

All temperatures are measured using Type J thermocouples, therefore, the PB16AH board uses 11 AD5T modules which are transformer isolated and have built in cold junction compensation. This feature makes installation easy. The program in the LC2/LC4 contains subroutines to linearize the thermocouple values as they are received by the LC2/LC4. In addition to the monitoring algorithm for activating the alarm and shutdown outputs based on comparisons to programmed setpoints, subroutines are also included to communicate with a portable terminal for displaying or modifying system parameters.



Compressor Monitoring System

```
10 '*****************************************************************
20 '*
30 '*       COMPRESSOR MONITORING SYSTEM
40 '*
50 '*
60 '*****************************************************************
70 '*****************************************************************
80 '*
90 '*       MAIN PROGRAM
100'*
110'*
120'*****************************************************************
130 GOSUB 3360                          'Dimension Arrays
140 GOSUB 3530                          'Initialize Variables
150 GOSUB 3910                          'Assign Constant Values
160 GOSUB 2200                          'Send A Power Up Clear
170 GOSUB 2330                          'Configure The Outputs
180 GOSUB 3260                          'Define Timer Interrupt
190 GOSUB 2460                          'Start The RPM Counter
200 GOSUB 2580                          'Enable Trapping Of Keys From Host
210 GOSUB 850                           'Read Inputs Status And Take Action
220 GOSUB 990                           'Read Analog Temperature
230 GOSUB 1160                          'Check Against Setpoints And Alarm
240 IF PWRFLG% = 1 THEN GOTO 160        'Start Over On Power Failure
250 GOTO 210                            'Loop And Read Inputs Again
260 END
270'*****************************************************************
280'*
290'*       SUBROUTINES
300'*
310'*
320'*****************************************************************
330'*****************************************************************
340'*
350'*       INTERRUPT PROCEDURE FOR READING RPM INPUT
360'*
370'*
380'*****************************************************************
390 CALL OPTOWARE (ERR%,ADR%,CMD%,POS%(0),MOD%(0),INF%(0))
400 IF ERR% < 0 THEN GOSUB 2070         'OPTOMUX Error
410 RPMSTAT% = 60 * INF%(RPMPOS%)/RFRSH%        'Do RPM Calculation
420 RETURN
430'*****************************************************************
440'*
450'*       J TYPE THERMOCOUPLE
460'*
470'*       TEMPERATURE RANGE: 0 TO 760 DEG C
480'*       VOLTAGE RANGE: 0 TO 49.922 Millivolts (Absolute)
490'*
500'*       INPUT: V - Thermocouple Voltage in Volts
510'*       (Absolute with Cold Junction Compensation)
520'*
530'*       If an AD5 or AD5T Thermocouple input module is used, the following
540'*       equation will calculate V from the actual counts value read:
550'*
560'*           V = COUNTS% * 9.553E-6
570'*
580'*       The MV equation is accurate from 0 to 700 Deg C (0-4095 counts)
590'*
600'*       OUTPUT: TC - Temperature in degrees Centigrade
```

```
610 '*
620 '*
630 '*************************************************************************
640 V = COUNTS% * 9.553E-6                    'Counts To Voltage Conversion
650 TC = JT(5)                                'Start With Last Coefficient
660 FOR I% = 4 TO 0 STEP -1                   'Polynomial Calculation
670 TC = TC * V+JT(I%)
680 NEXT
690 RETURN
700 '*************************************************************************
710 '*
720 '*       CALL THE OPTOWARE DRIVER
730 '*
740 '*
750 '*************************************************************************
760 CALL OPTOWARE (ERRS%,ADDR%,COMD%,POSIT%(0),MODI%(0),INFO%(0))
770 IF ERRS% < 0 THEN GOSUB 1940
780 RETURN
790 '*************************************************************************
800 '*
810 '*       READ ALARM INPUTS
820 '*
830 '*
840 '*************************************************************************
850 ADDR% = ALRMBRD1%                         'Set Address Of Board
860 COMD% = RDSTAT%                           'Read Status Command
870 CALL OPTOWARE (ERRS%,ADDR%,COMD%,POSD%(0),MODI%(0),INFO%(0))
880 IF ERRS% < 0 THEN GOSUB 1940              'Process Errors If Any
890 FOR I% = 0 TO LASTPOS%                    'Store All Data
900 INPUTS%(I%) = INFO%(I%)                   'Store Inputs In Data Array
910 NEXT
920 RETURN
930 '*************************************************************************
940 '*
950 '*       READ ANALOG TEMPERATURES
960 '*
970 '*
980 '*************************************************************************
990 ADDR% = TMPBOARD%                         'Set Address Of Board
1000      COMD% = RDTEMP%                     'Read Temperatures Command
1010      CALL OPTOWARE (ERRS%,ADDR%,COMD%,POSA%(0),MODI%(0),INFO%(0))
1020      IF ERRS% < 0 THEN GOSUB 1940        'Process Errors If Any
1030      FOR P% = 0 TO MAXSENS%              'Convert All Analog Values
1040      COUNTS% = INFO%(P%)                 'Temp Value For Conversion
1050      GOSUB 640                           'Do J Type TC Conversion
1060      TEMPS(P%) = TC                      'Store The Temperature
1070      TEMPS(P%) = INT(TEMPS(P%)*10+.5)/10
1080      NEXT
1090      RETURN
1100      '*************************************************************************
1110      '*
1120      '*     CHECK LIMITS AND PROCESS ALARMS
1130      '*
1140      '*
1150      '*************************************************************************
1160      LRMFLG% = 0                         'Clear Alarms
1170      SDFLG% = 0
1180      IF RPMSTAT% > RPMOVER% THEN GOSUB 1370
1190      IF RPMSTAT% < RPMUNDER% THEN ALRMFLG% = 1
1200      FOR M% = 0 TO MAXSENS%              'Do The Temperature Alarm & Shutdown First
```

```
1210    IF TEMPS(M%) > TEMP.HL(M%) THEN ALRMFLG% = 1
1220    IF TEMPS(M%) > TEMP.HHL(M%) THEN SDFLG% = 1
1230    NEXT
1240    FOR I% = 0 TO LASTPOS%              'Check Discrete Inputs
1250    IF INPUTS%(I%) <> 0 THEN ALRMFLG% = 1
1260    NEXT
1270    IF INPUTS%(FIREPOS%) <> 0 THEN SDFLG% = 1
1280    IF SDFLG% = 1 THEN GOSUB 1700 ELSE GOSUB 1820       'Shutdown On Alarm
1290    IF ALRMFLG% = 1 THEN GOSUB 1460 ELSE GOSUB 1580  'Alarm
1300    RETURN
1310    '*********************************************************************
1320    '*
1330    '*     SET BOTH ALARM AND SHUTDOWN FLAGS
1340    '*
1350    '*
1360    '*********************************************************************
1370    ALRMFLG% = 1                        'Set Alarm Flag
1380    SDFLG% = 1
1390    RETURN
1400    '*********************************************************************
1410    '*
1420    '*     TURN ON THE ALARM
1430    '*
1440    '*
1450    '*********************************************************************
1460    ADDR% = ALRMBRD1%                   'Set Address Of Board With Horn
1470    COMD% = TURNON%                     'Activate Outputs Command
1480    POSIT%(0) = ALRMPOS%                'Alarm Position
1490    POSIT%(1) = -1                      'End Of List
1500    GOSUB 760                           'Call The OPTOMUX Driver
1510    RETURN
1520    '*********************************************************************
1530    '*
1540    '*     RESET THE ALARM
1550    '*
1560    '*
1570    '*********************************************************************
1580    ADDR% = ALRMBRD1%                   'Set Address Of Board With Horn
1590    COMD% = TURNOFF%                    'Deactivate Outputs Command
1600    POSIT%(0) = ALRMPOS%                'Alarm Position
1610    POSIT%(1) = -1                      'End Of List
1620    GOSUB 760                           'Call The OPTOMUX Driver
1630    RETURN
1640    '*********************************************************************
1650    '*
1660    '*     SHUTDOWN THE SYSTEM
1670    '*
1680    '*
1690    '*********************************************************************
1700    ADDR% = ALRMBRD1%                   'Set Address Of Board With Horn
1710    COMD% = TURNON%                     'Activate Outputs Command
1720    POSIT%(0) = SDPOS%                  'Shutdown Position
1730    POSIT%(1) = -1                      'End Of List
1740    GOSUB 760                           'Call The OPTOMUX Driver
1750    RETURN
1760    '*********************************************************************
1770    '*
1780    '*     RESET THE SHUTDOWN OUTPUT
1790    '*
1800    '*
```

```
1810    '*****************************************************************
1820    ADDR% = ALRMBRD1%                   'Set Address Of Board With Horn
1830    COMD% = TURNOFF%                    'Deactivate Outputs Command
1840    POSIT%(0) = SDPOS%                  'Shutdown Position
1850    POSIT%(1) = -1                      'End Of List
1860    GOSUB 760                           'Call The OPTOMUX Driver
1870    RETURN
1880    '*****************************************************************
1890    '*
1900    '*     COMMUNICATIONS ERROR MESSAGE ROUTINE
1910    '*
1920    '*
1930    '*****************************************************************
1940    PRINT "*************************************"
1950    PRINT " DATE: ";DATE$;"    TIME: ";TIME$
1960    PRINT " COMMUNICATION ERROR: #";ERRS%
1970    PRINT " ADDRESS: ";ADDR%;"  COMMAND: ";COMD%
1980    PRINT "*************************************"
1990    IF ERRS% = -1 THEN PWRFLG% = 1      'Power Fail On OPTOMUX
2000    RETURN
2010    '*****************************************************************
2020    '*
2030    '*     ERROR MESSAGE FOR INTERRUPT ROUTINE
2040    '*
2050    '*
2060    '*****************************************************************
2070    PRINT "*************************************"
2080    PRINT " DATE: ";DATE$;"    TIME: ";TIME$
2090    PRINT " OPTOWARE INTERRUPT ERROR: #";ERR%
2100    PRINT " ADDRESS: ";ADR%;"  COMMAND: ";CMD%
2110    PRINT "*************************************"
2120    IF ERR% = -1 THEN PWRFLG% = 1       'Power Fail On OPTOMUX
2130    RETURN
2140    '*****************************************************************
2150    '*
2160    '*     SEND POWER UP CLEAR COMMAND
2170    '*
2180    '*
2190    '*****************************************************************
2200    PWRFLG% = 0                         'Reset Power Fail Flag
2210    ADDR% = ALRMBRD1%                   'Set Board Address
2220    COMD% = PUC%                        'Power Up Clear Command
2230    GOSUB 760                           'Call The OPTOMUX Driver
2240    ADDR% = TMPBOARD%                   'Set Analog Board Address
2250    GOSUB 760                           'Call The OPTOMUX Driver
2260    RETURN
2270    '*****************************************************************
2280    '*
2290    '*     CONFIGURE OUTPUTS OF BOARD
2300    '*
2310    '*
2320    '*****************************************************************
2330    ADDR% = ALRMBRD1%                   'Set Address Of Board
2340    COMD% = CFGOUT%                     'Configure Outoput Command
2350    POSIT%(0) = ALRMPOS%                'First Output
2360    POSIT%(1) = SDPOS%                  'Second Output
2370    POSIT%(2) = -1                      'End Of List
2380    GOSUB 760                           'Call The OPTOMUX Driver
2390    RETURN
2400    '*****************************************************************
```

```
2410    '*
2420    '*      INITIALIZE AND START COUNTERS
2430    '*
2440    '*
2450    '*************************************************************************
2460    CMD% = STRTCNT%                 'Start The RPM Counter
2470    CALL OPTOWARE(ERR%,ADR%,CMD%,POS%(0),MOD%(0),INF%(0))
2480    IF ERR% <> 0 THEN GOSUB 2070    'OPTOMUX Error
2490    RPMSTAT% = RPMSP%               'Set RPM Variable To Set Point
2500    CMD% = RDCOUNT%                 'Set Command To Read Counter
2510    RETURN
2520    '*************************************************************************
2530    '*
2540    '*      SET UP KEY TO TRAP FROM HOST
2550    '*
2560    '*
2570    '*************************************************************************
2580    ON KEY("A") GOSUB 2710          'Do Routine To Acknowledge Alarm
2590    ON KEY("D") GOSUB 2820          'Routine To Display Parameters
2600    ON KEY("S") GOSUB 2980          'Routine To Prompt For Set Points
2610    KEY("A") ON                     'Enable The "A" Key
2620    KEY("D") ON                     'Enable The "D" Key
2630    KEY("S") ON                     'Enable The "S" Key
2640    RETURN
2650    '*************************************************************************
2660    '*
2670    '*      KEY INTERRUPT HANDLER "A"
2680    '*
2690    '*
2700    '*************************************************************************
2710    GOSUB 1580                      'Disable Alarm
2720    GOSUB 1820                      'Disable Shutdown
2730    ALRMFLG% = 0                    'Reset Alarm Flag
2740    SDFLG% = 0                      'Reset Shutdown Flag
2750    RETURN
2760    '*************************************************************************
2770    '*
2780    '*      KEY INTERRUPT HANDLER "D"
2790    '*
2800    '*
2810    '*************************************************************************
2820    PRINT "RPM: ";RPMSTAT%;"  ALARM: ";ALRMFLG%;"  SHUTDOWN: ";SDFLG%
2830    PRINT "LOWOIL: ";INPUTS%(OILPOS%);"  FIRE: ";INPUTS%(FIREPOS%)
2840    PRINT "VIBRATION: ";INPUT%(VIBRPOS%);"  WATER TEMP: ";INPUT%(WATRPOS%)
2850    FOR V% = 0 TO MAXSENS% STEP 3
2860    FOR W% = V% TO V%+2
2870    PRINT "  #";W%;"=";TEMPS(W%);
2880    NEXT
2890    PRINT
2900    NEXT
2910    RETURN
2920    '*************************************************************************
2930    '*
2940    '*      KEY INTERRUPT HANDLER "S"
2950    '*
2960    '*
2970    '*************************************************************************
2980    INPUT "RPM, TEMPERATURE, OR EXIT (R,T,E): ";K$
2990    IF K$ = "R" THEN GOSUB 3020     Prompt For RPM
3000    IF K$ = "T" THEN GOSUB 3110     'Prompt For Temperature
```

```
3010    IF K$ = "E" THEN RETURN ELSE GOTO 2980   'Exit On An "E"
3020    PRINT "CURRENT RPM OVERSPEED LIMIT: ";RPMOVER%
3030    INPUT "NEW VALUE: ";RPMOVER%
3040    PRINT
3050    PRINT "CURRENT RPM UNDERSPEED LIMIT: ";RPMUNDER%
3060    INPUT "NEW VALUE: ";RPMUNDER%
3070    PRINT
3080    PRINT "CURRENT RPM SETPOINT: ";RPMSP%
3090    INPUT "NEW VALUE: ";RPMSP%
3100    RETURN
3110    PRINT
3120    INPUT "WHICH MODULE POSITION (0-15): ";U%
3130    PRINT
3140    PRINT "CURRENT HIGH LIMIT DEG C: ";TEMP.HL(U%)
3150    INPUT "NEW VALUE: ";TEMP.HL(U%)
3160    PRINT
3170    PRINT "CURRENT HIGH-HIGH LIMIT DEG C: ";TEMP.HHL(U%)
3180    INPUT "NEW VALUE: ";TEMP.HHL(U%)
3190    RETURN
3200    '*************************************************************
3210    '*
3220    '*     INITIALIZE INTERRUPT TIMER
3230    '*
3240    '*
3250    '*************************************************************
3260    TIMER OFF                                  'Turn Timer Off To Start
3270    ON TIMER(RFRSH%) GOSUB 390                 'Interrupt Every RFRSH% Seconds
3280    TIMER ON                                   'Enable Timer
3290    RETURN
3300    '*************************************************************
3310    '*
3320    '*     DIMENSION ARRAYS ROUTINE
3340    '*
3350    '*************************************************************
3360    DIM POSIT%(15),POS%(15)                    '16 Element Array
3370    DIM POSD%(15),POSA%(15)                    '16 Element Array
3380    DIM INFO%(15),INF%(15)                     'INFO Arrays For OPTOWARE Calls
3390    DIM MOD%(1)                                'MODIFIER Array For OPTOWARE Interrupt
3400    DIM MODI%(1)                               'MODIFIER Array For OPTOWARE
3410    DIM TEMPS(15)                              'Temporary 16 Element Array
3420    DIM TEMP.HL(15)                            'Temp High Limit
3430    DIM TEMP.HHL(15)                           'Temp High High Limit
3440    DIM JT(5)                                  'Coefficients For J Type Thermocouple
3450    DIM INPUTS%(15)                            'Discrete Input Status
3460    RETURN
3470    '*************************************************************
3480    '*
3490    '*     ROUTINE TO INITIALIZE VARIABLES
3510    '*
3520    '*************************************************************
3530    ERRS% = 0                                  'Set Everybody Initially To 0
3540    ADDR% = 0
3550    COMD% = 0
3560    MODI%(0) = 0
3570    MODI%(1) = 0
3580    MOD%(0) = 0
3600    FOR I% = 0 TO 15
```

```
3610    POSIT%(I%) = 0
3620    POS%(I%) = 0
3630    POSD%(I%) = 0
3640    POSA%(I%) = 0
3650    INFO%(I%) = 0
3660    INF%(I%) = 0
3670    TEMPS(I%) = 0
3680    INPUTS%(I%) = 0
3690    NEXT
3700    '
3710    '       OTHER SYSTEM VARIABLES
3720    '
3730    ALRMFLG% = 0                    'OK To Turn On Alarm
3740    SDFLG% = 0                      'Shutdown Flag
3750    ERRFLG% = 0                     'No Errors With OPTOMUX
3760    RPMSTAT% = 0                    'Current RPM Reading
3770    LOWOIL% = 0                     'Low Oil Pressure Variable
3780    HIGHWTR% = 0                    'High Water Temp Variable
3790    VIBRATION% = 0                  'Vibration Indicator Variable
3800    FIRE% = 0                       'Fire Input Variable
3810    ALARM% = 0                      'Alarm Output
3820    SHUTDOWN% = 0                   'Shutdown Variable
3830    PWRFLG% = 0                     'Power Failure Indicator
3840    RETURN
3850    '*************************************************************************
3860    '*
3870    '*      ASSIGN CONSTANTS ROUTINE
3880    '*
3890    '*
3900    '*************************************************************************
3910    ALRMBRD1% = 255                 'Address Of Alarm Board #1
3920    TMPBOARD% = 239                 'Address Of Analog Board
3930    '
3940    '       DEFINE MODULE POSITIONS
3950    '
3960    RPMPOS% = 0                     'Position Of RPM Input
3970    OILPOS% = 1                     'Position Of Low Oil Pressure
3980    WATRPOS% = 2                    'Position Of Water Temp High Switch
3990    VIBRPOS% = 3                    'Position Of Vibration Sensor
4000    FIREPOS% = 4                    'Position Of Fire Sense Input
4010    LASTPOS%= FIREPOS%              'Last Input Position On Board
4020    ALRMPOS% = 14                   'Position Of Alarm
4030    SDPOS% = 15                     'Position Of Shutdown Output
4040    '
4050    '       OPTOWARE COMMAND CONSTANTS
4060    '
4070    OPTOWARE = 4                    'Address Of OPTOMUX Driver
4080    PUC% = 0                        'Power Up Clear
4090    RESET% = 1                      'Reset
4100    RDSTAT% = 12                    'Read Status
4110    CFGOUT% = 8                     'Configure Outputs
4120    STRTCNT% = 20                   'Start Counter
4130    RDCOUNT% = 23                   'Read And Clear Counter
4140    TURNON% = 10                    'Activate Outputs
4150    TURNOFF% = 11                   'Deactivate Outputs
4160    RDTEMP% = 37                    'Read Analog Temps
4170    '
4180    '       INTERRUPT OPTOWARE VARIABLES
4190    '
4200    CMD% = RDCOUNT%                 'Read And Clear Counters Command
```

```
4210    ADR% = ALRMBRD1%            'Address Of RPM Input
4220    POS%(0) = RPMPOS%           'Position Of RPM Input
4230    POS%(1) = -1                'End Of List
4240    '
4250    '     POSITIONS OF TEMPERATURE INPUTS
4260    '
4270    MAXSENS% = 10               'Number Of Temperature Inputs On Board
4280    FOR I% = 0 TO MAXSENS%
4290    POSA%(I%) = I%
4300    NEXT
4310    POSA%(11) = -1              'End Of List
4320    '
4330    '     POSITIONS OF DISCRETE INPUTS
4340    '
4350    POSD%(0) = OILPOS%          'Position Of Oil Press Input
4360    POSD%(1) = WATRPOS%         'Position Of Water Temp Input
4370    POSD%(2) = VIBRPOS%         'Position Of Vibration Sensor
4380    POSD%(3) = FIREPOS%         'Position Of Fire Alarm
4390    POSD%(4) = -1               'End Of List
4400    '
4410    '     J TYPE THERMOCOUPLE COEFFICIENTS
4420    '
4430    JT(0) = 4.886826E-02
4440    JT(1) = 19873.15
4450    JT(2) = -218614.6
4460    JT(3) = 1.15692E+07
4470    JT(4) = -2.649175E+08
4480    JT(5) = 2.018441E+09
4490    '
4500    '     SETPOINTS AND SYSTEM VARIABLES
4510    '
4520    RFRSH% = 10                 'Seconds To Perform Timer Interrupt
4530    RPMSP% = 600                'RPM Setpoint
4540    RPMOVER% = 690              '15% Overspeed Setpoint
4550    RPMUNDER% = 510             '15% Underspeed Setpoint
4560    '*
4570    '*    INITIALIZE SETPOINTS FOR TEMPERATURES
4580    '*
4590    '*
4600    '*    ALL TEMPERATURE SETPOINTS ARE IN °C THEN THE PROGRAM CONVERTS
4610    '*    TO EQUIVALENT COUNT VALUE FOR COMPARISON WITH ACTUAL INPUT
4620    '*
4630    FOR J% = 0 TO 7
4640    TEMP.HL(J%) = 104           'Cylinder High Limit
4650    TEMP.HHL(J%) = 110          'Cylinder High High Limit
4660    NEXT
4670    FOR J% = 8 TO 10
4680    TEMP.HL(J%) = 149           'Compressor High Limit
4690    TEMP.HHL(J%) = 160          'Compressor High High Limit
4700    NEXT
4710    RETURN
```

PROGRAMMING NOTE 6

Application

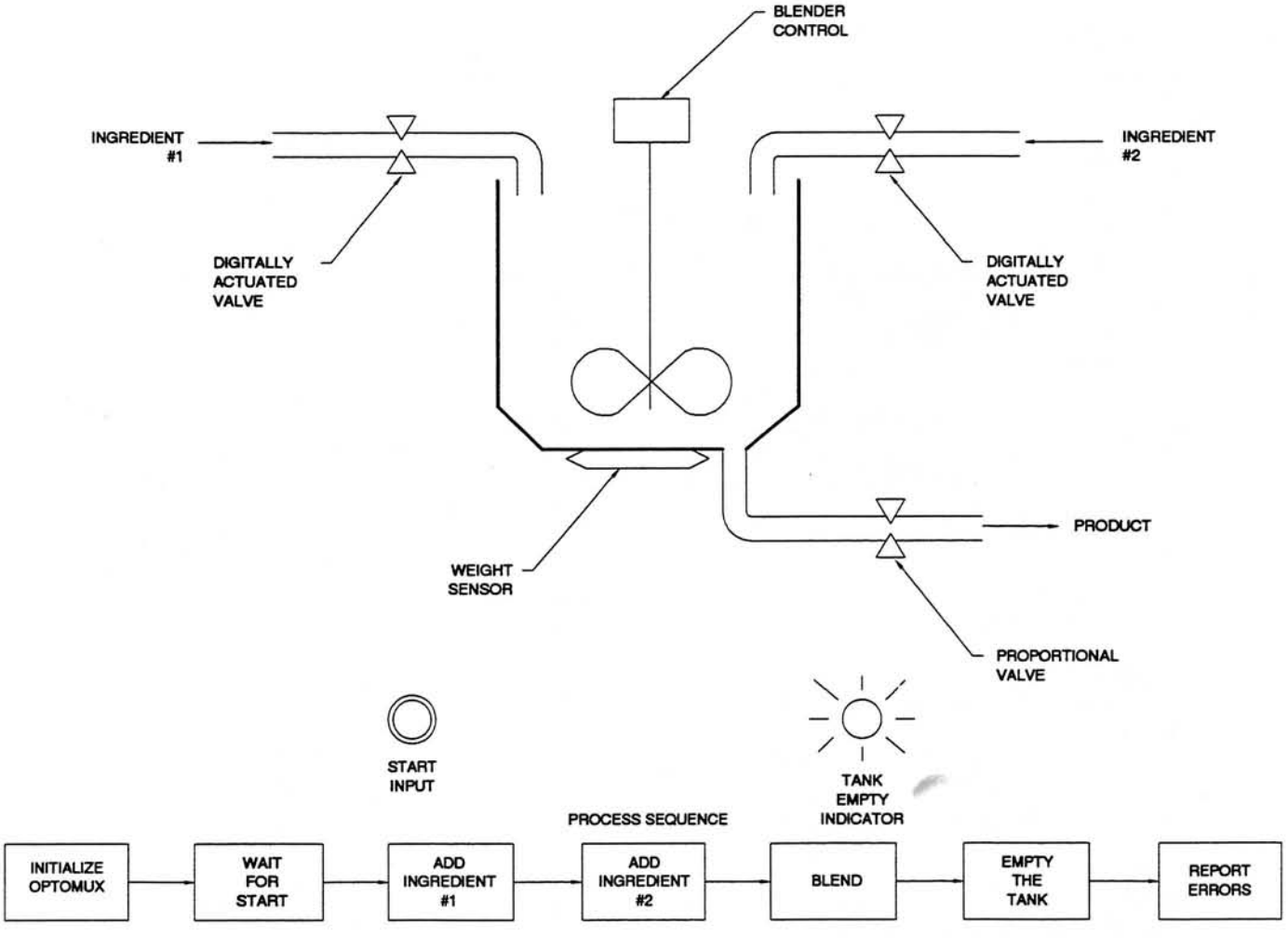
Weighing and Mixing

Problem

A system is needed to dispense set amounts of two substances into a mixing vessel. The contents of the vessel are then mixed for a specified time period. The resulting mixture is then drained from the vessel and the process is repeated.

Solution

A LC2/LC4, a four point analog OPTOMUX board B2/PB4AH, and a 16 point digital OPTOMUX board B1/PB16H will be used to control the blending process. The start button, indicator light, and digitally activated valves used to fill the tank will be controlled by the digital OPTOMUX units. The analog board will be used to read the weight of tank and to control the tank's proportional drain valve.



Weighing and Mixing System

```
10 '*****************************************************************
20 '*
30 '*       BLENDING EXAMPLE
40 '*
50 '*****************************************************************
60 '
70 '       Main Calling Routine
80 '
90 GOSUB 2920                          'Go Initialize Variables
100 GOSUB 2370                         'Go Initialize OPTOMUX
110 IF ERRORS% < 0 THEN GOTO 250           'Test For Errors
120 GOSUB 440                              'Wait For The Start Button
130 IF ERRORS% < 0 THEN GOTO 250           'Test For Errors
140 GOSUB 690                              'Add Ingredient #1
150 IF ERRORS% < 0 THEN GOTO 250           'Test For Errors
160 GOSUB 1140                         'Add Ingredient #2
170 IF ERRORS% < 0 THEN GOTO 250           'Test For Errors
180 GOSUB 1510                         'Blend Ingredients
190 IF ERRORS% < 0 THEN GOTO 250           'Test For Errors
200 GOSUB 1730                         'Empty The Tank
210 IF ERRORS% < 0 THEN GOTO 250           'Test For Errors
220 GOTO 120                           'Go Look For Start Again
230'
240'
250 GOSUB 2370                         'Reinitialize OPTOMUX
260 IF ERRORS% < 0 THEN GOTO 250           'If Errors, Try Again
270 GOTO 120                           'Go Start Process Over
280'
290'*****************************************************************
300'*
310'*       DRIVER CALLING ROUTINE
320'*
330'*****************************************************************
340'
350 CALL OPTOWARE (ERRORS%,ADDR%,CMD%,POSITIONS%(0),MODIFIER%(0),INFO%(0))
360 RETURN
370'
380'*****************************************************************
390'*
400'*       START BUTTON SUBROUTINE
410'*
420'*****************************************************************
430'
440 ADDR% = DIGITAL1%                  'Address The Digital Board
450 CMD% = READLATCHES%                    'Read Latches Command
460'
470 GOSUB 350                              'Call The OPTOMUX Driver
480 IF ERRORS% < 0 THEN RETURN         'Test For Errors
490'
500'       Test Latch To See If Start Button Has Been Pressed
510'
520 IF INFO%(START%) = 0 THEN GOTO 470
530
540'       Clear The Latch
550'
560 CMD% = CLEARLATCHES%                   'Clear Latches Command
570 POSITIONS%(0) = START%                 'Specify The Start Button
580 POSITIONS%(1) = -1                     'End Of List
590 GOSUB 350                              'Call The OPTOMUX Driver
600 RETURN
```

```
610'
620'*************************************************************************
630'*
640'*       ADD INGREDIENT # 1 SUBROUTINE
650'*
660'*************************************************************************
670'
680'
690 ADDR% = DIGITAL1%                    'Address The Digital Board
700 CMD% = ACTIVATE%                     'Activate Outputs Command
710 POSITIONS%(0) = INGREDIENT1%             'Specify Ingredient #1 Valve
720 POSITIONS%(1) = -1                       'End Of List
730 GOSUB 350                                'Call The OPTOMUX Driver
740 IF ERRORS% < 0 THEN RETURN           'Test For Errors
750'
760'        READ THE WEIGHT OF TANK
770'
780 ADDR% = ANALOG%                      'Address The Analog Board
790 CMD% = READANLIN%                        'Read Analog Inputs Command
800 POSITIONS%(0) = WEIGHT%              'Specify The Tank Weight Input
810 POSITIONS%(1) = -1                       'End Of List
820 GOSUB 350                                'Call The OPTOMUX Driver
830 IF ERRORS% < 0 THEN RETURN           'Test For Errors
840'
850'        If Not At Ingredient # Target Level Wait Until It is
860'
870 IF INFO%(WEIGHT%) < FULL1% THEN GOTO 820
880'
890'        Close Ingredient #1 Filler Valve
900'
910 ADDR% = DIGITAL1%                    'Address The Digital Board
920 CMD% = DEACTIVATE%                   'Deactivate Outputs Command
930 POSITIONS%(0) = INGREDIENT1%             'Specify Ingredient #1 Valve
940 POSITIONS%(1) = -1                       'End Of List
950 GOSUB 350                                'Call The OPTOMUX Driver
960 IF ERRORS% < 0 THEN RETURN           'Test For Errors
970'
980'        Turn Off Tank Empty Indicator Lamp
990'
1000    ADDR% = DIGITAL1%                    'Address The Digital Board
1010    CMD% = DEACTIVATE%                   'Deactivate Relays Command
1020    POSITIONS%(0) = TANKEMPTY%           'Specify Indicator Lamp Output
1030    POSITIONS%(1) = -1                   'End Of List
1040    GOSUB 350                            'Call The OPTOMUX Driver
1050    IF ERRORS% < 0 THEN RETURN           'Test For Errors
1060    RETURN
1070    '
1080    '*************************************************************************
1090    '*
1100    '*     ADD INGREDIENT # 2 SUBROUTINE
1110    '*
1120    '*************************************************************************
1130    '
1140    ADDR% = DIGITAL1%                    'Address The Digital Board
1150    CMD% = ACTIVATE%                     'Activate Outputs Command
1160    POSITIONS%(0) = INGREDIENT2%         'Specify Ingredient #2 Valve
1170    POSITIONS%(1) = -1                   'End Of List
1180    GOSUB 350                            'Call The OPTOMUX Driver
1190    IF ERRORS% < 0 THEN RETURN           'Test For Errors
1200    '
```

```
1210    '       Read The Weight Of Tank
1220    '
1230    ADDR% = ANALOG%                     'Address The Analog Board
1240    CMD% = READANLIN%                   'Read Analog Inputs Command
1250    POSITIONS%(0) = WEIGHT%             'Specify The Tank Weight Input
1260    POSITIONS%(1) = -1                  'End Of List
1270    GOSUB 350                           'Call The OPTOMUX Driver
1280    IF ERRORS% < 0 THEN RETURN          'Test For Errors
1290    '
1300    '       If Not At Ingredient #2 Target Level Wait Until It Is
1310    '
1320    IF INFO%(WEIGHT%) < FULL2% THEN GOTO 1270
1330    '
1340    '       Close Ingredient #2 Filler Valve
1350    '
1360    ADDR% = DIGITAL1%                   'Address The Digital Board
1370    CMD% = DEACTIVATE%                  'Deactivate Outputs
1380    POSITIONS%(0) = INGREDIENT2%        'Specify Ingredient #2 Valve
1390    POSITIONS%(1) = -1                  'End Of List
1400    GOSUB 350                           'Call The OPTOMUX Driver
1410    IF ERRORS% < 0 THEN RETURN          'Test For Errors
1420    RETURN
1430    '
1440    '*************************************************************************
1450    '*
1460    '*      BLENDING SUBROUTINE
1470    '*
1480    '*************************************************************************
1490    '
1500    '
1510    ADDR% = DIGITAL1%                   'Address The Digital Board
1520    CMD% = ACTIVATE%                    'Activate Relays
1530    POSITIONS%(0) = BLEND%              'Activate Blender Motor
1540    POSITIONS%(1) = -1                  'End Of List
1550    GOSUB 350                           'Call The OPTOMUX Driver
1560    IF ERRORS% < 0 THEN RETURN          'Test For Errors
1570    '
1580    CMD% = READONOFF%                   'Read On/Off Status Command
1590    GOSUB 350                           'Call The OPTOMUX Driver
1600    IF ERRORS% < 0 THEN RETURN          'Test For Errors
1610    '
1620    '       Check For Blender To Turn Off
1630    '
1640    IF INFO%(BLEND%) = 1 THEN GOTO 1590
1650    RETURN
1660    '
1670    '*************************************************************************
1680    '*
1690    '*      SUBROUTINE TO EMPTY TANK
1700    '*
1710    '*************************************************************************
1720    '
1730    ADDR% = ANALOG%                     'Address The Analog Board
1740    CMD% = SETWAVE%                     'Set Improve Waveform Command
1750    MODIFIER%(0) = 2                    'Specify Ramp Up Waveform
1760    POSITIONS%(0) = PRODUCT%            'Specify Tank Empty Valve
1770    POSITIONS%(1) = -1                  'End Of List
1780    INFO%(0) = 4095                     'Set High Limit = Full Scale
1790    INFO%(1) = 0                        'Set Low Limit = Zero Scale
1800    INFO%(2) = OPENRATE%                'Specify Waveform Period
```

```
1810    GOSUB 350                           'Call The OPTOMUX Driver
1820    IF ERRORS% < 0 THEN RETURN          'Test For Errors
1830    '
1840    '       Read The Tank Level
1850    '
1860    CMD% = READANLIN%                   'Read Analog Inputs Command
1870    POSITIONS%(0) = WEIGHT%             'Read The Tank Weight Input
1880    POSITIONS%(1) = -1                  'End Of List
1890    GOSUB 350                           'Call The OPTOMUX Driver
1900    IF ERRORS% < 0 THEN RETURN          'Test For Errors
1910    '
1920    '       Keep Reading Until Empty
1930    '
1940    IF INFO%(WEIGHT%) > EMPTY% THEN GOTO 1890
1950    '
1960    '       Turn On Tank Empty Indicator Lamp
1970    '
1980    ADDR% = DIGITAL1%                   'Address The Digital Board
1990    CMD% = ACTIVATE%                    'Activate Outputs Command
2000    POSITIONS%(0) = TANKEMPTY%          'Specify What Positions
2010    POSITIONS%(1) = -1                  'End Of List
2020    GOSUB 350                           'Call The OPTOMUX Driver
2030    IF ERRORS% < 0 THEN RETURN          'Test For Errors
2040    '
2050    '       Ramp The Valve to Closed
2060    '
2070    ADDR% = ANALOG%                     'Address The Analog Board
2080    CMD% = SETWAVE%                     'Set Improved Waveform Command
2090    MODIFIER%(0) = 6                    'Specify Ramp Down Waveform
2100    POSITIONS%(0) = PRODUCT%            'Start At Tank Empty Valve
2110    POSITIONS%(1) = -1                  'End Of List
2120    INFO%(0) = 4095                     'Specify High Limit
2130    INFO%(1) = 0                        'Specify Low Limit
2140    INFO%(2) = CLOSERATE%               'Specify Waveform Period
2150    GOSUB 350                           'Call The OPTOMUX Driver
2160    IF ERRORS% < 0 THEN RETURN          'Test For Errors
2170    '
2180    '       Check If Valve Is Fully Closed
2190    '
2200    CMD% = READANLOUT%                  'Read Analog Output Command
2210    GOSUB 350                           'Call The OPTOMUX Driver
2220    IF ERRORS% < 0 THEN RETURN          'Test For Errors
2230    '
2240    '       If Valve Is Not Closed, Check It Again
2250    '
2260    IF INFO%(PRODUCT%) > CLOSED% GOTO 2210
2270    RETURN
2280    '
2290    '*************************************************************************
2300    '*
2310    '*      SUBROUTINE TO INITIALIZE OPTOMUXES
2320    '*
2330    '*************************************************************************
2340    '
2350    '       Initialize Digital OPTOMUX
2360    '
2370    ADDR% = DIGITAL1%                   'Address Of Digital Board
2380    CMD% = PWRUPCLR%                    'Power Up Clear Command
2390    GOSUB 350                           'Call The OPTOMUX Driver
2400    IF ERRORS% < 0 THEN RETURN          'Test For Errors
```

```
2410    '
2420    CMD% = RESETOPMX%               'Reset Command
2430    GOSUB 350                       'Call The OPTOMUX Driver
2440    IF ERRORS% < 0 THEN RETURN      'Test For Errors
2450    '
2460    CMD% = CONFIGURE%               'Configure Input/Output Command
2470    POSITIONS%(0) = INGREDIENT1%    'Control Valve #1 Is An Output
2480    POSITIONS%(1) = INGREDIENT2%    'Control Valve #2 Is An Output
2490    POSITIONS%(2) = BLEND%          'Blender Control Is Output
2500    POSITIONS%(3) = TANKEMPTY%      'Empty Indicator Light Is An Output
2510    POSITIONS%(4) = -1              'End Of List
2520    GOSUB 350                       'Call The OPTOMUX Driver
2530    IF ERRORS% < 0 THEN RETURN      'Test For Errors
2540    '
2550    CMD% = SETIMEDLY%               'Time Delay Command
2560    POSITIONS%(0) = BLEND%          'Blender Motor
2570    POSITIONS%(1) = -1              'End Of List
2580    MODIFIER%(0) = 0                'Specify Pulsed On Delay
2590    INFO%(0) = BLENDTIME%           'Specify Delay Length (in 10 ms units)
2600    GOSUB 350                       'Call The OPTOMUX Driver
2610    IF ERRORS% < 0 THEN RETURN      'Test For Errors
2620    '
2630    '       Initialize Analog OPTOMUX
2640    '
2650    ADDR% = ANALOG%                 'Address Of Analog Board
2660    CMD% = PWRUPCLR%                'Power Up Clear Command
2670    GOSUB 350                       'Call The OPTOMUX Driver
2680    IF ERRORS% < 0 THEN RETURN      'Test For Errors
2690    '
2700    CMD% = RESETOPMX%               'Reset Command
2710    GOSUB 350                       'Call The OPTOMUX Driver
2720    IF ERRORS% < 0 THEN RETURN      'Test For Errors
2730    '
2740    CMD% = CONFIGURE%               'Configure Command
2750    FOR I% = 0 TO 15                'Configure All Positions As Outputs
2760    POSITIONS%(I%) = I%             'To Make Sure All Unused Positions
2770    NEXT                            'Will Be Outputs
2780    IF ERRORS% < 0 THEN RETURN      'Test For Errors
2790    GOSUB 350                       'Call The OPTOMUX Driver
2800    CMD% = CONFIGUREI%              'Configure As Inputs Command
2810    POSITIONS%(0) = WEIGHT%         'Specify Which Positions Are Inputs
2820    POSITIONS%(1) = -1              'End Of List
2830    GOSUB 350                       'Call The OPTOMUX Driver
2840    RETURN
2850    '
2860    '*****************************************************************
2870    '*
2880    '*      INITIALIZE VARIABLES AND CONSTANTS
2890    '*
2900    '*****************************************************************
2910    '
2920    DIGITAL1% = 255                 'Digital Board Is At Address 255
2930    START% = 0                      'Start Button Is At Position 0
2940    INGREDIENT1% = 1                'Valve For Ingredient #1 Is At Position 1
2950    INGREDIENT2% = 2                'Valve For Ingredient #2 Is At Position 2
2960    BLEND% = 3                      'Blender Motor Output Is At Position 3
2970    TANKEMPTY% = 4                  'Tank Empty Indicator Light Is At Position 4
2980    '
2990    ANALOG% = 239                   'Analog Board Is At Address 239
3000    WEIGHT% = 2                     'Weight Sensor Is At Positon 0
```

```
3010    PRODUCT% = 3                    'Proportional Valve Is At Positon 1
3020    '
3030    FULL1% = 0.3 * 4095             'Ingredient #1 Fill Level Is 30% Of Full Scale
3040    FULL2% = 0.7 * 4095             'Ingredient #2 Fill Level Is 70% Of Full Scale
3050    BLENDTIME% = 1000               'Blend Time Is 10 Seconds
3060    EMPTY% = 0.1 * 4095             'Tank Is Empty At 10% Of Full Scale
3070    CLOSED% = 0.1 * 4095            'Valve Closed At 10% Full Scale
3080    OPENRATE% = 20                  'Period For Valve Ramp Open In 100 ms Units
3090    CLOSERATE% = 50                 'Period For Valve Ramp Close In 100 ms Units
3100    '
3110    '     Set Up Variables For OPTOMUX Commands To Make Program Easy To Read
3120    '
3130    PWRUPCLR% = 0                   'Command 0 Is Power Up Clear
3140    RESETOPMX% = 1                  'Command 1 Is Reset
3150    CONFIGURE% = 6                  'Command 6 Is Configure As Inputs And Outputs
3160    CONFIGUREI% = 7                 'Command 7 Is Configure As Inputs
3170    ACTIVATE% = 10                  'Command 10 Is Activate Outputs
3180    DEACTIVATE% = 11                'Command 11 Is Deactivate Outputs
3190    READONOFF% = 12                 'Command 12 Is Read On/Off Status
3200    READLATCHES% = 16               'Commant 16 Is Read Latches
3210    CLEARLATCHES% = 18              'Command 18 Is Clear Latches
3220    SETIMEDLY%  = 25                'Command 25 Is Set Time Delay
3230    READANLOUT% = 36                'Command 36 Is Read Analog Outputs
3240    READANLIN% = 37                 'Command 37 Is Read Analog Inputs Command
3250    SETLIMITS% = 40                 'Command 40 Is Set High/Low Limits
3260    TESTLIMITS% = 41                'Command 41 Is Read Out Of Range Latches
3270    CLEARLIMITS% = 42               'Command 42 Is Clear Out Of Range Latches
3280    SETWAVE% = 50                   'Command 50 Is Set Improved Output Waveform
3290    '
3300    '     Dimension Arrays To Be Passed To The OPTOMUX Driver
3310    '
3320    DIM POSITIONS%(15)              'Positions Array Parameter Has 16 Elements
3330    DIM MODIFIER%(1)                'Modifier Array Parmeter Has 2 Elements
3340    DIM INFO%(15)                   'Info Array Parameter Has 16 Elements
3350    FOR I% = 0 TO 15
3360    POSITIONS%(I%) = 0              'Set All Position And Info Arrays
3370    INFO%(I%) = 0                   'Array Elements To 0
3380    NEXT
3390    MODIFIER%(0) = 0                'Set Modifier Array Elements To 0
3400    MODIFIER%(1) = 0
3410    ERRORS% = 0                     'Initialize Errors To 0
3420    ADDR% = 0                       'Initialize Address To 0
3430    CMD% = 0                        'Initialize Command To 0
3440    '
3450    OPTOWARE = 4
3460    RETURN
```

PROGRAMMING NOTE 7

LC2/LC4 Basic Subroutines

This programming note contains program listings of subroutines to perform functions which are important in data acqusition and control applications. The following is a list and brief description of each subroutine:

LISTING	DESCRIPTION
1	A method of password protection to prevent unauthorized access to a BASIC program currently executing in LC2/LC4.
2	A method of storing data in a non-volatile area which will not be initialized by BASIC if the program is restarted after a power failure.
3	Subroutines for linearizing thermocouple inputs. The polynomial expansion method is used.
4	Dialing a Hayes 1200 Smartmodem with LC2/LC4.
5	A routine to load assembly language subroutines into memory for later access by BASIC programs.

LISTING #1

```
10  '*************************************************************************
20  '*
30  '*      SUBROUTINES FOR PASSWORD PROTECTION TO PREVENT AN
40  '*      UNAUTHORIZED USER FROM HALTING A PROGRAM
50  '*
60  '*      The following routines trap a CTRL/C character which
70  '*      would normally halt program execution, and then prompt
80  '*      the operator for a legal password which is stored in
90  '*      non-volatile memory.  If the operator responds with the
100 '*      proper password, the program will be halted; otherwise,
110 '*      the program continues.  Of course, an operator can stop
120 '*      the program by turning power off, removing the autostart
130 '*      jumper, then turning power back on.  Subroutines are
140 '*      also included for POKEing the password into memory and
150 '*      for changing the password.
160 '*
170 '*      IMPORTANT:  DO NOT USE CTRL/C OR CTRL/P IN YOUR PASSWORD.
180 '*
190 '*************************************************************************
200 '*
210 '*      MAIN PROGRAM LOOP FOR AN EXAMPLE
220 '*
230 CLEAR 20                        'Reserve 20 Bytes As Non-Volatile Memory
240 GOSUB 440                           'Initialize The Password
250 GOSUB 360                           'Initialize Interrupts
260 GOTO 260                        'Wait For Interrupt
270 END
280 '*************************************************************************
290 '*
300 '*      SUBROUTINES
310 '*
320 '*************************************************************************
330 '
340 '       Initialize Interrupt Keys
350 '
360 ON KEY(CHR$(3)) GOSUB 530       'Password Ctrl/C
370 KEY(CHR$(3)) ON                 'Enable Trap
380 ON KEY(CHR$(16)) GOSUB 630          'Change Password CTRL/P
390 KEY(CHR$(16)) ON                'Enable Trap
400 RETURN
410 '
420 '       Place Password In Non-Volatile Memory Then Delete Routine
430 '
440 P% = &H8000                         'Start Of Non-Volatile Area
450 PSWD$ = "OPTO22"                    'Password Which Is 6 Characters Long
460 FOR I% = 0 TO 6                     'Store Password Byte By Byte
470 POKE P%+I%,PEEK(VARPTR(PSWD$)+I%)
480 NEXT
490 RETURN
500 '
```

```
510 '        Password Handler Routine
520 '
530 IF INKEY$ <> "" THEN GOTO 530                    'Flush Buffer
540 INPUT "ENTER PASSWORD: ";PSSWRD$                 'Prompt For PSWD$
550 FOR I% = 0 TO 6                                  'Get Password From Memory
560 POKE VARPTR(PSWD$)+I%,PEEK(&H8000+I%)
570 NEXT
580 IF PSSWRD$ = PSWD$ THEN STOP ELSE PRINT "Security Violation - No Access Granted"
590 RETURN
600 '
610 '        Routine To Change An Existing Password
620 '
630 IF INKEY$ <> "" THEN GOTO 630                    'Flush Buffer
640 INPUT "ENTER OLD PASSWORD: ";PSSWRD$             'Prompt For PSWD$
650 FOR I% = 0 TO 6                                  'Get Password From Memory
660 POKE VARPTR(PSWD$)+I%,PEEK(&H8000+I%)
670 NEXT
680 IF PSSWRD$=PSWD$ THEN GOTO 700 ELSE PRINT "Security Violation - No Access Granted"
690 RETURN
700 INPUT "ENTER NEW PASSWORD (6 CHARACTERS): ";PSWD$
710 GOSUB 460                                        'Save The New Password
720 RETURN
```

LISTING #2

```
10  '*************************************************************************
20  '*
30  '*      SAVING ALARM MESSAGES IN NON-VOLATILE MEMORY
40  '*
50  '*      These subroutines will build a 26 byte message which
60  '*      includes the time, date, identifier, value, and event code
70  '*      then store it in a previously cleared area in memory.
80  '*      A status byte is also maintained for use by a program
90  '*      to record types of events such as power failures.  By
100 '*      storing values in a previously cleared area, BASIC will
110 '*      not overwrite or alter these values in the event of a
120 '*      power failure or restart.
130 '*
140 '*      Each message can be considered a record and has the
150 '*      following structure:
160 '*
170 '*          DATE         :10 Bytes
180 '*          TIME         : 8 Bytes
190 '*          EVENT CODE   : 2 Bytes
200 '*          IDENTIFIER   : 2 Bytes
210 '*          VALUE        : 4 Bytes
220 '*
230 '*      Besides each record, the first two bytes in the cleared
240 '*      RAM area are used for status to record unique alarms and
250 '*      not duplicates.  The third byte keeps a count of the number
260 '*      of records in memory.  Records are stored sequentially
270 '*      starting at the fourth byte.  As a record is created, a
280 '*      check is made to see if the maximum number of records has
290 '*      been exceeded.  If the maximum number has not been exceeded,
300 '*      the record is appended at the end of the list and the count
310 '*      byte is incremented.  If the maximum number of records has
320 '*      been exceeded, the counter is reset to 0, and the record
330 '*      is inserted in the first position, overwriting the earliest
340 '*      record.
350 '*
360 '*
370 '*      BEFORE A PROGRAM CONTAINING THESE ROUTINES CAN BE EXECUTED
380 '*      THE "NEW" COMMAND MUST BE ISSUED, FOLLOWED BY "CLEAR n"
390 '*      WHERE n IS THE NUMBER OF BYTES TO ALLOCATE.  n CAN BE
400 '*      CALCULATED USING THE FOLLOWING EQUATION:
410 '*
420 '*          n = (Maximum # of Records) * 26 + 3
430 '*
440 '*      AFTER THE CLEAR n, THE PROGRAM CAN BE LOADED AND RUN.
450 '*      PLACING THE CLEAR n STATEMENT INSIDE A PROGRAM WILL ALSO
460 '*      WORK AND WILL NOT AFFECT PREVIOUSLY SAVED VALUES DURING A
470 '*      RESTART.  THIS IS BECAUSE CLEAR n ONLY MOVES THE BEGINNING
480 '*      OF BASIC POINTER n ABSOLUTE BYTES AND DOES NOT MODIFY ANY
490 '*      DATA STORED BELOW n.  VARIABLES WITHIN THE BASIC PROGRAM
500 '*      WILL BE REINITIALIZED AFTER USING THE CLEAR n STATEMENT.
510 '*
520 '*************************************************************************
530 '*
540 '*      SUBROUTINES
550 '*
560 '*************************************************************************
```

```
570 '*
580 '*        INITIALIZE THE VARIABLES
590 '*
600 '*
610 '*************************************************************************
620 STAT1% = &H8000                 'Address Of Status Byte 1
630 STAT2% = &H8001                 'Address Of Status Byte 2
640 REC.CNT% = &H8002               'Address Of Record Count
650 BASE.PNT% = &H8003              'Start Of Records
660 N% = 0                          'Memory Pointer Variable
670 MDATE$ = ""                         'Temporary Date Storage
680 MTIME$ = ""                         'Temporary Time Storage
690 CODE% = 0                           'Event Code Variable
700 ID% = 0                             'Identification Integer
710 VALUE = 0                       'Value Variable
720 REC.ADR% = 0                    'Variable For Message Address
730 MAXREC% = 25                        'Maximum Number Of Records
740 POKE STAT1%,0                       'Clear First Status Byte
750 POKE STAT2%,0                       'Clear Second Status Byte
760 POKE REC.CNT%,0                     'Clear # Of Records
770 REC.NUM% = 0                        'Record Number To Read
780 RETURN
790 '*************************************************************************
800 '*
810 '*        STORE A RECORD IN NON VOLATILE MEMORY
820 '*
830 '*
840 '*************************************************************************
850 IF PEEK(REC.CNT%) > MAXREC% THEN POKE REC.CNT%,0   'Reset Counter
860 REC.ADR% = BASE.PNT% + (26 * PEEK(REC.CNT%))          'Calculate Address
870 '
880 '         Store The Time And Date
890 '
900 MDATE$ = DATE$
910 MTIME$ = TIME$
920 '
930 '         Save The Date Into Non-Volatile Memory
940 '
950 FOR S% = 0 TO 9
960 POKE REC.ADR%+S%,PEEK(VARPTR(MDATE$)+S%)
970 NEXT
980 '
990 '         Save The Time Into Non-Volatile Memory
1000        '
1010        FOR S% = 0 TO 7
1020        POKE REC.ADR%+10+S%,PEEK(VARPTR(MTIME$)+S%)
1030        NEXT
1040        '
1050        '       Save EVENT CODE Into Non-Volatile Memory
1060        '
1070        POKE REC.ADR%+18,PEEK(VARPTR(CODE%))
1080        POKE REC.ADR%+19,PEEK(VARPTR(CODE%)+1)
1090        '
1100        '       Save IDENTIFIER Into Non-Volatile Memory
1110        '
1120        POKE REC.ADR%+20,PEEK(VARPTR(ID%))
1130        POKE REC.ADR%+21,PEEK(VARPTR(ID%)+1)
1140        '
1150        '       Save VALUE Into Non-Volatile Memory
```

```
1160    '
1170    FOR S% = 0 TO 3
1180    POKE REC.ADR%+22+S%,PEEK(VARPTR(VALUE)+S%)
1190    NEXT
1200    '
1210    '       Increment The Record Number
1220    '
1230    POKE REC.CNT%,PEEK(REC.CNT%)+1
1240    RETURN
1250    '*************************************************************************
1260    '*
1270    '*      READ A RECORD FROM NON-VOLATILE MEMORY
1280    '*
1290    '*
1300    '*************************************************************************
1310    IF REC.NUM% > MAXREC% THEN GOTO 1680  'Out Of Range Error
1320    REC.ADR% = BASE.PNT% + (26 * REC.NUM%)  'Calculate Address
1330    '
1340    '       Restore The Date
1350    '
1360    FOR R% = 0 TO 9
1370    POKE VARPTR(MDATE$)+R%,PEEK(REC.ADR%+R%)
1380    NEXT
1390    '
1400    '       Restore The Time
1410    '
1420    FOR R% = 0 TO 7
1430    POKE VARPTR(MTIME$)+R%,PEEK(REC.ADR%+10+R%)
1440    NEXT
1450    '
1460    '       Read The EVENT CODE
1470    '
1480    POKE VARPTR(CODE%),PEEK(REC.ADR%+18)
1490    POKE VARPTR(CODE%)+1,PEEK(REC.ADR%+19)
1500    '
1510    '       Read The IDENTIFIER
1520    '
1530    POKE VARPTR(ID%),PEEK(REC.ADR%+20)
1540    POKE VARPTR(ID%)+1,PEEK(REC.ADR%+21)
1550    '
1560    '       Read The VALUE
1570    '
1580    FOR R% = 0 TO 3
1590    POKE VARPTR(VALUE)+R%,PEEK(REC.ADR%+22+R%)
1600    NEXT
1610    RETURN
1620    '*************************************************************************
1630    '*
1640    '*      RECORD NUMBER OUT OF RANGE MESSAGE
1650    '*
1660    '*
1670    '*************************************************************************
1680    PRINT "RECORD NUMBER IS OUT OF RANGE"
1690    RETURN
```

LISTING #3

```
10  '*************************************************************************
20  '*
30  '*       OPTO 22
40  '*
50  '*       THERMOCOUPLE LINEARIZATION SUBROUTINES
60  '*       FOR TYPE J, K, S, E, R, AND T TYPE THERMOCOUPLES
70  '*
80  '*       The following routines are useful for calculating the
90  '*       temperature based on the thermocouple voltage and type.  The
100 '*       relationship is based on a power series expansion polynomial of
110 '*       the type: T = A0 + A1 * X + A2 * X^2 + A3 * X^3...+ An * X^n
120 '*
130 '*       where: T is temperature, X is voltage, and A and n are constants
140 '*
150 '*       The equations vary from 5th order to 9th order depending on type.
160 '*
170 '*
180 '*       The thermocouple voltage must be a cold junction compensated
190 '*       value.  Refer to a book on thermocouples for setting up a cold
200 '*       junction compensated system.  For type J and K thermocouples,
210 '*       the AD5 and AD8 modules have built-in cold junction compensation.
220 '*       Refer to the type J and K subroutines to calculate the actual
230 '*       thermocouple voltage from the count value returned by the module.
240 '*       For the AD8 module, the equation is only accurate for temperatures
250 '*       above 0 degrees centigrade (module reading = 349).  Module readings
260 '*       up to 5713 will work accurately.
270 '*
280 '*************************************************************************
290 '
300 '*************************************************************************
310 '*
320 '*       LIST OF COEFFICIENTS FOR J, K, S, R, T, AND E THERMOCOUPLES
330 '*
340 '*       USED IN SOLVING A nth ORDER POLYNOMIAL EXPRESSION
350 '*
360 '*************************************************************************
370  DIM JT(5), KT(8), ST(9), TT(7), RT(8), ET(9)
380 '
390 '       J Type Thermocouple Coefficients
400 '
410 JT(0) = 4.886826E-02
420 JT(1) = 19873.15
430 JT(2) = -218614.6
440 JT(3) = 1.15692E+07
450 JT(4) = -2.649175E+08
460 JT(5) = 2.018441E+09
470 '
480 '       K Type Thermocouple Coefficients
490 '
500 KT(0) = 2.265846E-1
510 KT(1) = 2.415211E4
520 KT(2) = 6.723342E4
530 KT(3) = 2.210341E6
540 KT(4) = -8.609639E8
550 KT(5) = 4.83506E10
560 KT(6) = -1.18452E12
570 KT(7) = 1.38690E13
```

```
580 KT(8) = -6.33708E13
590 '
600 '          S Type Thermocouple Coefficients
610 '
620 ST(0) = 9.277632E-1
630 ST(1) = 1.695265E5
640 ST(2) = -3.156836E7
650 ST(3) = 8.990731E9
660 ST(4) = -1.63565E12
670 ST(5) = 1.88027E14
680 ST(6) = -1.37241E16
690 ST(7) = 6.17501E17
700 ST(8) = -1.56105E19
710 ST(9) = 1.69535E20
720 '
730 '          T Type Thermocouple Coefficients
740 '
750 TT(0) = 1.008609E-1
760 TT(1) = 2.572794E4
770 TT(2) = -7.673458E5
780 TT(3) = 7.802559E7
790 TT(4) = -9.247486E9
800 TT(5) = 6.97688E11
810 TT(6) = -2.66192E13
820 TT(7) = 3.94078E14
830 '
840 '          R Type Thermocouple Coefficients
850 '
860 RT(0) = 2.636329E-1
870 RT(1) = 1.790754E5
880 RT(2) = -4.884034E7
890 RT(3) = 1.90002E10
900 RT(4) = -4.82704E12
910 RT(5) = 7.62091E14
920 RT(6) = -7.20026E16
930 RT(7) = 3.71496E18
940 RT(8) = -8.03104E19
950 '
960 '          E Type Thermocouple Coefficients
970 '
980 ET(0) = 1.049672E-1
990 ET(1) = 1.718945E4
1000        ET(2) = -2.826391E5
1010        ET(3) = 1.269534E7
1020        ET(4) = -4.487031E8
1030        ET(5) = 1.10866E10
1040        ET(6) = -1.76807E11
1050        ET(7) = 1.71842E12
1060        ET(8) = -9.19278E12
1070        ET(9) = 2.06132E13
1080        RETURN
1090        '*************************************************************************
1100        '*
1110        '*       J TYPE THERMOCOUPLE
1120        '*
1130        '*       TEMPERATURE RANGE : 0 TO 760 DEG C
1140        '*       VOLTAGE RANGE : 0 TO 49.922 Millivolts (Absolute)
1150        '*
1160        '*       INPUT : V - Thermocouple Voltage in Volts
1170        '*                   (Absolute with Cold Junction Compensation)
```

```
1180    '*
1190    '*      If an AD5 or AD5T Thermocouple input module is used, the following
1200    '*      equation will calculate V from the actual counts value read:
1210    '*
1220    '*          V = COUNTS% * 9.553E-6
1230    '*
1240    '*      The MV equation is accurate from 0 to 700 Deg C (0-4095 counts)
1250    '*
1260    '*      OUTPUT : TC - Temperature in degrees Centigrade
1270    '*
1280    '*
1290    '************************************************************************
1300    TC = JT(5)
1310    FOR I% = 4 TO 0 STEP -1
1320    TC = TC * V+JT(I%)
1330    NEXT
1340    RETURN
1350    '************************************************************************
1360    '*
1370    '*      K TYPE THERMOCOUPLE
1380    '*
1390    '*      TEMPERATURE RANGE : 0 TO 1370 DEG C
1400    '*      VOLTAGE RANGE : 0 TO 54.807 Millivolts (Absolute)
1410    '*
1420    '*      INPUT : V - Thermocouple Voltage in Volts
1430    '*              (Absolute with Cold Junction Compensation)
1440    '*
1450    '*      If an AD8 or AD8T Thermocouple input module is used, the following
1460    '*      equation will calculate V from the actual counts value read:
1470    '*
1480    '*          V = (COUNTS% * 1.021E-5)-3.553E-3
1490    '*
1500    '*      The MV equation is accurate from 0 to 1370 Deg C (349-5713 counts)
1510    '*
1520    '*      OUTPUT : TC - Temperature in degrees Centigrade
1530    '*
1540    '*
1550    '************************************************************************
1560    TC = KT(8)
1570    FOR I% = 7 TO 0 STEP -1
1580    TC = TC * V+KT(I%)
1590    NEXT
1600    RETURN
1610    '************************************************************************
1620    '*
1630    '*      S TYPE THERMOCOUPLE
1640    '*
1650    '*      TEMPERATURE RANGE : 0 TO 1750 DEG C
1660    '*      VOLTAGE RANGE : 0 TO 18.504 Millivolts (Absolute)
1670    '*
1680    '*      INPUT : V - Thermocouple Voltage in Volts
1690    '*              (Absolute with Cold Junction Compensation)
1700    '*
1710    '*      OUTPUT : TC - Temperature in degrees Centigrade
1720    '*
1730    '*
1740    '************************************************************************
1750    TC = ST(9)
1760    FOR I% = 8 TO 0 STEP -1
1770    TC = TC * V+ST(I%)
```

```
1780    NEXT
1790    RETURN
1800    '*************************************************************************
1810    '*
1820    '*      R TYPE THERMOCOUPLE
1830    '*
1840    '*      TEMPERATURE RANGE : 0 TO 1000 DEG C
1850    '*      VOLTAGE RANGE : 0 TO 10.503 Millivolts (Absolute)
1860    '*
1870    '*      INPUT : V - Thermocouple Voltage in Volts
1880    '*              (Absolute with Cold Junction Compensation)
1890    '*
1900    '*      OUTPUT : TC - Temperature in degrees Centigrade
1910    '*
1920    '*
1930    '*************************************************************************
1940    TC = RT(8)
1950    FOR I% = 7 TO 0 STEP -1
1960    TC = TC * V+RT(I%)
1970    NEXT
1980    RETURN
1990    '*************************************************************************
2000    '*
2010    '*      T TYPE THERMOCOUPLE
2020    '*
2030    '*      TEMPERATURE RANGE : -160 TO 400 DEG C
2040    '*      VOLTAGE RANGE : -4.865 TO 20.869 Millivolts (Absolute)
2050    '*
2060    '*      INPUT : V - Thermocouple Voltage in Volts
2070    '*              (Absolute with Cold Junction Compensation)
2080    '*
2090    '*      OUTPUT : TC - Temperature in degrees Centigrade
2100    '*
2110    '*
2120    '*************************************************************************
2130    TC = TT(7)
2140    FOR I% = 6 TO 0 STEP -1
2150    TC = TC * V+TT(I%)
2160    NEXT
2170    RETURN
2180    '*************************************************************************
2190    '*
2200    '*      E TYPE THERMOCOUPLE
2210    '*
2220    '*      TEMPERATURE RANGE : -100 TO 1000 DEG C
2230    '*      VOLTAGE RANGE : -5.237 TO 76.358 Millivolts (Absolute)
2240    '*
2250    '*      INPUT : V - Thermocouple Voltage in Volts
2260    '*              (Absolute with Cold Junction Compensation)
2270    '*
2280    '*      OUTPUT : TC - Temperature in degrees Centigrade
2290    '*
2300    '*
2310    '*************************************************************************
2320    TC = ET(9)
2330    FOR I% = 8 TO 0 STEP -1
2340    TC = TC * V+ET(I%)
2350    NEXT
2360    RETURN
```

LISTING #4

```
10  '*************************************************************************
20  '*
30  '*       DIALING A HAYES SMARTMODEM WITH LC2/LC4
40  '*
50  '*
60  '*       The following are subroutines for dialing, answering,
70  '*       configuring, and communicating through the Hayes
80  '*       Smartmodem 1200 device.  Since the host port on the
90  '*       LC2 is a RS-422 port, it must first be converted to
100 '*       a RS-232 port by using an Opto 22 AC7A/B card.
110 '*
120 '*       The Hayes Smartmodem 1200 has eight dip switches used
130 '*       for configuring the modem.  These subroutines assume
140 '*       all the switches are in the DOWN position except
150 '*       switches 2 and 6 which are in the UP position.
160 '*       Refer to the Hayes manual for more information.
170 '*
180 '*       When accessing the Smartmodem 1200 or communicating
190 '*       with a remote modem, make sure sufficient delays are
200 '*       used where appropriate.  The modem at the other end may
210 '*       be expecting certain characters at first so it can
220 '*       determine baud rate, etc.
230 '*
240 '*       The main program loop illustrates how LC2/LC4 can dial the
250 '*       OPTO 22 Bulletin Board, log on, get to the main menu,
260 '*       then log off.
270 '*
280 '*
290 '*************************************************************************
300 GOSUB 1360                  'Initialize Variables
310 PRINT                     'Send CRs To Clear Modem Buffer
320 GOSUB 1530                  'Reset And Initialize Modem
330 PHONE$="7148928375"         'OPTO 22 Bulletin Board System
340 GOSUB 1070                  'Dial The Phone
350 PRINT CHR$(13);             'Send A Carriage Return
360 GOSUB 1650                  'Wait
370 PRINT "JOHN"                'Print First Name
380 GOSUB 1730                  'Wait
390 PRINT "DOE"                     'Print Last Name
400 GOSUB 1730                  'Wait
410 PRINT "Y";                  'Name Is Correct
420 GOSUB 1730                  'Wait
430 PRINT "SMUDGE"              'Password
440 GOSUB 1650                  'Wait
450 PRINT "N"                   'No More Bulletin
460 GOSUB 1730                  'Wait
470 PRINT "G"                   'Say Goodbye
480 GOSUB 1730                  'Wait
490 PRINT "N"                   'No Message For Sysop
500 END
510 '*************************************************************************
520 '*
530 '*       PLACE MODEM IN LOCAL COMMAND STATE
540 '*
550 '*************************************************************************
560 FOR D% = 1 TO DLY           'Delay One Second
570 NEXT
```

```
580 PRINT "+++";                    'Get Modem's Attention
590 FOR D% = 1 TO DLY               'Delay One Second
600 NEXT
610 RETURN
620 '*************************************************************************
630 '*
640 '*      CHECK RESULT CODE
650 '*
660 '*************************************************************************
670 INPUT "",CR$                            'Get Leading CR LF
680 INPUT "",RC$                            'Get Result Code
690 '
700 '       Check result code.  OK is no errors.  Modem must be set to
710 '       return word codes by setting the V command switch.
720 '
730 IF RC$ <> "OK" THEN GOTO 750
740 RETURN
750 IF RC$ = "CONNECT" THEN CONNECT% = 1 ELSE CONNECT% = 0
760 IF RC$ = "RING" THEN RING% = 1 ELSE RING% = 0
770 IF RC$ = "NO CARRIER" THEN NOCARR% = 1 ELSE NOCARR% = 0
780 IF RC$ = "ERROR" THEN COMERR% = 1 ELSE COMERR% = 0
790 RETURN
800 '*************************************************************************
810 '*
820 '*      SET UP TO ANSWER THE PHONE
830 '*
840 '*************************************************************************
850 GOSUB 560                               'Get Into Command State
860 PRINT "AT S0=2"                         'Set Number Of Rings To 2
870 GOSUB 670                               'Get Error Code
880 RETURN
890 '*************************************************************************
900 '*
910 '*      DIALING THE PHONE
920 '*
930 '*      Input:   PHONE$   contains the phone number to dial
940 '*                        legal chars include: 0...9 # * ( ) - . /
950 '*
960 '*      The Hayes dialing command is of the form AT Ds
970 '*      where s can be any of the legal characters plus
980 '*      the following letter codes:
990 '*
1000    '*      P = Pulse dialing
1010    '*      R = Dial "originate only" modem
1020    '*      T = Touch tone dial
1030    '*      , = Pause when dialing
1040    '*      ; = Return to command state after dialing
1050    '*
1060    '*************************************************************************
1070    PRINT "AT DT";PHONE$            'Dial The Number Touch Tone
1080    GOSUB 670                       'Check Result Code
1085    '
1090    '       Wait Until Connection
1095    '
1100    WHILE CONNECT% = 0
1110    GOSUB 670
1120    WEND
1130    '
1140    '       This section sends two carriage returns so the remote modem
1150    '       can determine baud rate and parity to finalize the connection.
```

```
1160    '
1170    GOSUB 1730                      'Wait Before Sending First Character
1180    PRINT CHR$(13);                 'Send A Carriage Return
1190    GOSUB 1730                      'Wait Before Sending Next Character
1200    PRINT CHR$(13);                 'Send Another
1210    CONNECT% = 0                    'Reset Connect Flag
1220    RETURN
1230    '*************************************************************************
1240    '*
1250    '*      HANG UP THE PHONE
1260    '*
1270    '*************************************************************************
1280    GOSUB 560                       'Put Modem In Command State
1290    PRINT "AT H0 O"                 'Hang Up And Return On Line
1300    RETURN
1310    '*************************************************************************
1320    '*
1330    '*      VARIABLES
1340    '*
1350    '*************************************************************************
1360    CONNECT% = 0                    'Hayes Result Code 1 Flag
1370    RING% = 0                       'Hayes Result Code 2 Flag
1380    NOCARR% = 0                     'Hayes Result Code 3 Flag
1390    COMERR% = 0                     'Hayes Result Code 4 Flag (ERROR)
1400    PHONE$ = ""                     'Phone Number For Dialing
1410    '
1420    '       These Delay Constants Give More Than Enough Time
1430    '
1440    DLY = 2000                      'Delay For Command Wait
1450    DELAY% = 10000                  'Delay Between Screens
1460    DELAY1% = 5000                  'Delay Between Messages
1470    RETURN
1480    '*************************************************************************
1490    '*
1500    '*      INITIALIZE THE MODEM
1510    '*
1520    '*************************************************************************
1530    GOSUB 560                       'Get Into Command State
1540    PRINT "AT Z"                    'Reset
1550    GOSUB 670                       'Check Result Code
1560    GOSUB 1730                      'Wait Before Sending Commands After Reset
1570    PRINT "AT V1 E0 M0"             'Word Results, No Echo, Speaker Off
1580    GOSUB 670                       'Check Result Code
1590    RETURN
1600    '*************************************************************************
1610    '*
1620    '*      DELAY SUBROUTINE
1630    '*
1640    '*************************************************************************
1650    FOR ZZZ% = 1 TO DELAY%
1660    NEXT
1670    RETURN
1680    '*************************************************************************
1690    '*
1700    '*      DELAY SUBROUTINE
1710    '*
1720    '*************************************************************************
1730    FOR ZZZ% = 1 TO DELAY1%
1740    NEXT
1750    RETURN
```

LISTING #5

```
1000    '*************************************************************************
1010    '*
1020    '*
1030    '*      This program is used to download an Intel format Hex file into
1040    '*      LC2/LC4's memory (this routine doesn't verify checksums).
1050    '*
1060    '*
1070    '*************************************************************************
1080    ABORT% = -1                                          'Set Up Flag
1090    WHILE ABORT%                                         'Do Until End
1100    INPUT "",INSTRING$                                   'Get One Line
1110    GOSUB 1210                                           'Process The Line
1120    WEND
1130    END
1140    '
1150    '
1160    '
1170    '                                                    Process On Line
1180    '
1190    '
1200    '
1210    IF LEFT$(INSTRING$,1) <> ":" THEN RETURN             'Get Out If Junk
1220    IF LEN(INSTRING$) < 11 THEN RETURN                   'Not Valid Line
1230    OFFSET% = 9                                          'Pointer To First Data Character
1240    LENGTH% = VAL("&H"+RIGHT$(LEFT$(INSTRING$,3),2))     'Get Number Of Characters
1250    ADDRESS% = VAL("&H"+RIGHT$(LEFT$(INSTRING$,7),4))    'Get Address To Store Data At
1260    IF LENGTH% = 0 THEN ABORT% = 0                       'If End, Get Out
1270    WHILE LENGTH%
1280    POKE ADDRESS%, VAL("&H"+RIGHT$(LEFT$(INSTRING$,OFFSET%+2),2))
1290    ADDRESS% = ADDRESS% + 1                              'Increment Address Pointer
1300    OFFSET% = OFFSET% + 2                                'Increment String Pointer
1310    LENGTH% = LENGTH% - 1                                'Decrement Length Counter
1320    WEND
1330    RETURN
```

PROGRAMMING NOTE 8

LC2/LC4 FORTH SUBROUTINES

This programming note contains program listings of subroutines to perform functions which are important in data acqusition and control applications. The following is a list and brief description of each subroutine:

LISTING	DESCRIPTION
1	A FORTH task scheduler for prioritizing and executing multiple tasks.
2	A Hex/ASCII dump utility to view the contents of memory.
3	A FORTH assembler for generating in-line assembly code within FORTH definitions.
4	Equates for all the OPTOMUX commands.

NOTE: In listing #1, the program must be modified to run on the LC4 Local Controller. The variable "CLOCK.TICK" must be changed to "CLICK.TICK + 2" to work properly on the LC4. The "CLOCK.TICK" variable is referenced in four different locations in the program. This is required because the resolution of the real time clock interrupt on the LC4 is 0.01 seconds instead of 0.1 seconds.

The other three FORTH program listings will work on the LC2 or LC4 without modifications.

LISTING #1

```
(                                                                    )
(       This is an example of a simple task scheduler                )
(                                                                    )
(       The resolution of the real time clock interrupt              )
(       on the LC2 is 0.1 seconds.  This will allow us               )
(       to start a new task every 0.1 seconds.  If a                 )
(       task takes longer than 0.1 seconds, other tasks              )
(       can not be performed until the culprit is done.              )
(                                                                    )
(       Each task is assigned a tick count with a 0.1 sec            )
(       resolution.  A task will be performed every                  )
(       time its tick count comes around.                            )
(                                                                    )
(       EXAMPLE:    A task with a tick count of 3 will               )
(                   be performed each time the tick                  )
(                   equals 3, 6, 9, 12, etc.                         )
(                                                                    )

: TASK ;            ( easy forget                                    )


(       QUE will contain the list of tasks and tick                  )
(       counts for each task                                         )
(                                                                    )
(       QUE + 0 has the current clock.tick                           )
(       QUE + 2 has the tick value for the first word                )
(       QUE + 4 has the address of the first word                    )
(       QUE + 6 has the tick value for the second word               )
(               .                                                    )
(               .                                                    )
(               .                                                    )
(       QUE + n-1 has address of last word                           )
(       QUE + n has a zero to indicate the end of que                )
(                                                                    )

    VARIABLE QUE 42 ALLOT       ( make space for 10 words            )


(       TASK.POINTER will keep track of where you are in             )
(       que                                                          )

    VARIABLE TASK.POINTER


(       The following group of words are just some simple            )
(       tasks to schedule                                            )

: ':'

    58 HOLD                     ( stick a : into text buffer         )
;
```

```
: '/'

    47 HOLD                         ( stick a / into text buffer        )
;


    ( the following word prints the time on the screen in the           )
    ( HH:MM:SS format                                                   )
    (                                                                   )
    (       hours is 3rd on the stack                                   )
    (       minutes is 2nd on the stack                                 )
    (       seconds is 1st on the stack                                 )

: .TIME

    SWAP ROT                        ( get into correct order            )
         0 <# ':' # # #>    ( convert hours                             )
    TYPE                            ( type hours                        )
         0 <# ':' # # #>    ( convert minutes                           )
    TYPE                            ( type minutes                      )
         0 <# # # #>                ( convert seconds                   )
    TYPE                            ( type seconds                      )
;


    ( the following word print the date on the screen in the            )
    ( MM/DD/YY format:                                                  )
    (                                                                   )
    (       month is 3rd on the stack                                   )
    (       day is 2nd on the stack                                     )
    (       year is 1st on the stack                                    )

: .DATE

    SWAP ROT                        ( get into correct order            )
         0 <# '/' # # #>    ( convert months                            )
    TYPE                            ( type months                       )
         0 <# '/' # # #>    ( convert days                              )
    TYPE                            ( type days                         )
         0 <# # # #>                ( convert years                     )
    TYPE                            ( type years                        )
;


: TEST

    ." This is just a test" CR
;


: PRINT.TIME

    TIME@                           ( get time from clock chip          )
```

```
    .TIME                   ( print the time                    )
    CR
;

: PRINT.DATE

    DATE@                       ( get date from clock chip      )
    .DATE                   ( print the date                    )
    CR
;

    ( This word is used to stop all the tasks in the que        )
    ( it does this by storing a zero in the tick value for      )
    ( the first word in the que                                 )
    (                                                           )
    ( the tasks are stoped only if a key press is detected      )
    (                                                           )

: STOP.RUN
    ?KEY                    ( check for keypress                )
    IF
        KEY DROP            ( get key and drop it               )
        0 QUE 2+ !          ( make que empty                    )
    THEN
;

: WAIT.FOR.TICK             ( wait for clock tick to change     )

    BEGIN                   ( start loop                        )
        CLOCK.TICK @            ( get current tick value; for LC4: CLOCK.TICK + 2 )
        QUE  @              ( get value to compare with         )
        = NOT               ( see if equal                      )
    UNTIL                   ( if not equal loop                 )
;

: SEE.IF.MINE                   ( see if tick value is active   )

    QUE @                   ( get current tick value            )
    TASK.POINTER @ @        ( get next words tick value         )
    MOD 0=                  ( get modulo value                  )
;

: QUE!
    OVER ! 2+
;

    ( initialize que                                            )
```

```
QUE 2+                                  ( point to first location          )
        9 QUE! ' TEST QUE!              ( do TEST every 9 ticks            )
        3 QUE! ' PRINT.TIME QUE!        ( do PRINT.TIME every 3 ticks      )
        6 QUE! ' PRINT.DATE QUE!        ( do PRINT.DATE every 6 ticks      )
        1 QUE! ' STOP.RUN QUE!          ( do STOP.RUN every 1 ticks        )
        0 QUE!                          ( stick in que terminator          )

: RUN                           ( perform words in QUE                     )

    1 CLOCK.TICK !              ( initialize CLOCK.TICK; for LC4: CLOCK.TICK + 2           )

    BEGIN                       ( setup loop                               )
        QUE 2+ TASK.POINTER !   ( initialize task pointer                  )
        CLOCK.TICK @ DUP        ( get current CLOCK.TICK; for LC4: CLOCK.TICK + 2    )

    0 = IF                      ( if equal to zero                         )
        DROP 1 1 CLOCK.TICK !   ( set equal to an 1; for LC4: CLOCK.TICK + 2       )
    THEN

    QUE !                       ( set que value                            )
    WAIT.FOR.TICK               ( wait for the tick to change              )
    TASK.POINTER @ @            ( get first tick value                     )

    IF                          ( if value isn't zero                      )
        BEGIN

            SEE.IF.MINE         ( compare que with clock.tick              )

            IF
                TASK.POINTER @      ( get a copy of task pointer           )
                2+ @                ( point to word to execute             )
                EXECUTE             ( perform word                         )
            THEN
                TASK.POINTER @      ( get a copy of task pointer           )
                2+ 2+ DUP           ( increment pointer                    )
                TASK.POINTER !      ( save updated task pointer            )
                @ 0 =               ( see if tick is equal to zero         )
            UNTIL

            FALSE ( on stack to continue loop                              )

        ELSE                ( if first item was zero                       )

            TRUE            ( and put a true onto the stack                )
        THEN                ( to terminate the loop                        )

        UNTIL
;
```

LISTING #2

```
    (                                                          )
    (       these words are used for doing a dump of memory    )
    (       top of stack will contain the number of bytes to dump )
    (       and next on the stack will contain the starting address )
    (       of the dump                                        )
    (                                                          )
    (                                                          )

: TASKS ;                   ( easy forget                      )

    BASE @                  ( save old number base             )
    HEX                     ( get into Hex mode                )


    (                                                          )
    ( PRINT.HEX.DATA                                           )
    (                                                          )
    (       this word prints the low order byte of the word    )
    (       on top of the stack in hexidecimal and with        )
    (       leading zeros                                      )
    (                                                          )

: PRINT.HEX.DATA

    S>D                     ( make into double precision       )
    <# # # #> TYPE          ( print with leading zeros         )
;


    (                                                          )
    ( PRINT.HEX.ADDRESS                                        )
    (                                                          )
    (       this word prints the word on top of the stack in   )
    (       hexidecimal with leading zeros                     )
    (                                                          )

: PRINT.HEX.ADDRESS

    S>D                         ( make into double precision   )
    <# # # # # #> TYPE          ( print with leading zeros     )
;


    (                                                          )
    ( PRINT.ASCII.DATA                                         )
    (                                                          )
    (       this word prints the low order byte of the word on top )
    (       of the stack as an ASCII character                 )
    (                                                          )
```

```
: PRINT.ASCII.DATA

    07F AND             ( strip off eight bit                    )
    DUP 20 <            ( check for less than space              )
    IF
        DROP 2E         ( if so, exchange with a period          )
    THEN
    EMIT                ( and print it                           )
;

: DUMP

    BASE @ ROT ROT      ( get current number base                )
    HEX                 ( change base to hex                     )
    0 DO
        CR DUP DUP          ( print CR and dup address           )
        PRINT.HEX.ADDRESS SPACE ( print the address              )

        8 0 DO              ( print data in hex format           )
            DUP C@ SPACE        ( get data                       )
            PRINT.HEX.DATA      ( print the data                 )
    1 + LOOP            ( increment pointer and loop             )
    SPACE               ( print extra space                      )
    8 0 DO              ( print data in hex format               )
        DUP C@ SPACE        ( get data                           )
        PRINT.HEX.DATA      ( print the data                     )
    1 + LOOP DROP           ( increment pointer and loop         )

    3 SPACES            ( seperate hex and ASCII data            )
    10 0 DO
        DUP C@              ( get data                           )
        PRINT.ASCII.DATA    ( print ASCII data                   )
    1 + LOOP                ( increment pointer and loop         )

    10 + LOOP           ( increment pointer and loop             )
    DROP                ( remove pointer                         )
    BASE ! CR           ( restore old number base                )
;

BASE !                      ( restore old number base            )

( ******************** end of dump routine ************************ )
```

LISTING #3

```
(                                                    )
(       This group of words make an 8080 assembler for the   )
(       LC2/LC4 OPTOFORTH                            )
(                                                    )
( ********************* WARNING ************************** )
(                                                    )
(       If you extend this assembler to include the Z80      )
(       instructions, do NOT use the EXX instruction.  The   )
(       alternate register set registers DE and BC are used  )
(       for banging the watchdog timer.  If you trash these  )
(       registers you will get reset.                )
(                                                    )
( ***************************************************** )
(                                                    )

: TASK ;                    ( easy forget                    )

    HEX

    VARIABLE TEMP.VOC ( temporary storage for context        )

    VOCABULARY ASSEMBLER    ( declare assembler vocabulary )


    (                                                    )
    ( 8*                                                 )
    (                                                    )
    (       shift value on stack by 3 bits to the left   )
    (                                                    )
    (                                                    )

: 8* 2* 2* 2* ;

    (                                                    )
    ( CODE                                               )
    (                                                    )
    (       this word is used to start an assembly       )
    (       language definition                          )
    (                                                    )

: CODE

    CREATE                  ( create dictionary header       )
    SMUDGE
    HERE HERE 2- !          ( stick in add. of machine code  )
    CONTEXT @ TEMP.VOC !    ( save context                   )
    ASSEMBLER               ( make assembler current         )
;

    (                                                    )
    ( LABEL                                              )
    (                                                    )

: LABEL

    CREATE                  ( create dictionary header       )
    ASSEMBLER               ( make assembler current         )
```

```
;

    (                                                       )
    ( ;CODE                                                 )
    (                                                       )

: ;CODE

    COMPILE (;CODE)             ( stick in the primitive    )
    [COMPILE] [
    CONTEXT @ TEMP.VOC !        ( save context vocabulary   )
    ASSEMBLER ; IMMEDIATE       ( and make assembler current )

    ASSEMBLER DEFINITIONS       ( get into assembler vocabulary )


    (                                                       )
    ( END-CODE                                              )
    (                                                       )
    (       this word terminates an assembly language definition )
    (                                                       )

: END-CODE

    SMUDGE                      ( make word visible         )
    TEMP.VOC @ CONTEXT !        ( restore context           )
;

: ;C END-CODE ;             ( same as END-CODE              )

    (                                                       )
    (       define constants for register names             )
    (                                                       )

    4 CONSTANT H        5 CONSTANT L        7 CONSTANT A        6 CONSTANT PSW
    2 CONSTANT D        3 CONSTANT E        0 CONSTANT B        1 CONSTANT C
    6 CONSTANT M        6 CONSTANT SP


    (                                                       )
    ( the following words are used for defining the assembler )
    ( instructions                                          )
    (                                                       )

: 1MI
    CREATE              ( build the header and stick in     )
    C,                  ( opcode                            )
    DOES>       ( define runtime definition             )
    C@                  ( get opcode and stick in word      )
    C,                  ( being defined                     )
;

: 2MI

    CREATE              ( build the header and stick in     )
    C,                  ( the opcode                        )
    DOES>       ( define runtime definition             )
    C@ +                ( get opcode and add to register    )
    C,                  ( stick in word being defined       )
;
```

```
: 3MI

    CREATE              ( build the header and stick in      )
    C,                  ( the opcode                         )
    DOES>         ( define runtime definition                )
    C@                  ( get opcode                         )
    SWAP 8* +           ( shift register and add to op.      )
    C,                  ( and stick in definition            )
;

: 4MI

    CREATE              ( build the header and stick in      )
    C,                  ( the opcode                         )
    DOES>         ( define runtime definition                )
    C@                  ( get opcode and put in word         )
    C,                  ( being defined                      )
    C,                  ( put bbit data in word being        )
;                       ( defined                            )

: 5MI

    CREATE              ( build the header and stick in      )
    C,                  ( the opcode                         )
    DOES>         ( define runtime definition                )
    C@                  ( get opcode and put in word         )
    C,                  ( being defined                      )
    ,                   ( put 16 bit data in word being      )
;                       ( defined                            )

    (                                                        )
    ( HPUSH                                                  )
    (                                                        )
    (       this word puts the address of HPUSH onto the stack )
    (                                                        )

: HPUSH NEXT-LINK 1- ;
    (                                                        )
    ( DPUSH                                                  )
    (                                                        )
    (       this word puts the address of DPUSH onto the stack )
    (                                                        )

: DPUSH NEXT-LINK 2- ;

    (                                                        )
    ( NEXT                                                   )
    (                                                        )
    (       this word puts the address of NEXT onto the stack )
    (                                                        )

: NEXT NEXT-LINK ;

    (                                                        )
    (       define the instructions as to opcode and type    )
    (                                                        )
```

```
00 1MI NOP      76 1MI HLT      F3 1MI DI       FB 1MI EI
07 1MI RLC      0F 1MI RRC      17 1MI RAL      1F 1MI RAR
E9 1MI PCHL     F9 1MI SPHL     E3 1MI XTHL     EB 1MI XCHG
27 1MI DAA      2F 1MI CMA      37 1MI STC      3F 1MI CMC
C9 1MI RET      C0 1MI RNZ      C8 1MI RZ       D0 1MI RNC
D8 1MI RC       E0 1MI RPO      E8 1MI RPE      F0 1MI RP
F8 1MI RM

80 2MI ADD      88 2MI ADC      90 2MI SUB      98 2MI SBB
A0 2MI ANA      A8 2MI XRA      B0 2MI ORA      B8 2MI CMP

09 3MI DAD      C1 3MI POP      C5 3MI PUSH     02 3MI STAX
0A 3MI LDAX     04 3MI INR      05 3MI DCR      03 3MI INX
0B 3MI DCX      C7 3MI RST

D3 4MI OUT      DB 4MI IN       C6 4MI ADI      CE 4MI ACI
D6 4MI SUI      DE 4MI SBI      E6 4MI ANI      EE 4MI XRI
F6 4MI ORI      FE 4MI CPI

22 5MI SHLD     2A 5MI LHLD     32 5MI STA      3A 5MI LDA
CD 5MI CALL     C4 5MI CNZ      CC 4MI CZ       D4 5MI CNC
DC 5MI CC       E4 5MI CPO      EC 5MI CPE      F4 5MI CP
FC 5MI CM       C3 5MI JMP      C2 5MI JNZ      CA 5MI JZ
D2 5MI JNC      DA 5MI JC       E2 5MI JPO      EA 5MI JPE
F2 5MI JP       FA 5MI JM

: MOV 8* 40 + + C, ;
: MVI 8* 6 + C, C, ;
: LXI 8* 1+ C, , ;

    0C2 CONSTANT 0=
    0D2 CONSTANT CS
    0E2 CONSTANT PE
    0F2 CONSTANT 0<

: NOT 8 + ;

: THEN HERE SWAP ! ;
: IF C, HERE 0 , ;
: ELSE C3 IF SWAP THEN ;
: BEGIN HERE ;
: UNTIL C, , ;
: WHILE IF ;
: REPEAT SWAP 0C3 C, , THEN ;

    ( some Z80 instructions                                    )

    18 4MI JR     20 4MI JRNZ   28 4MI JRZ    30 4MI JNC
    38 4MI JRC    10 4MI DJNZ

( ********** end of the 8080 assembler **********              )

FORTH DEFINITIONS          ( return to FORTH vocabulary  )
```

LISTING #4

```
    (                                                          )
    ( OPTOMUX command set                                      )
    (                                                          )
    ( ********** WARNING **********                            )
    (                                                          )
    ( before using any of the following commands, make sure that )
    ( you have defined your parameter blocks and have assigned  )
    ( one of them to the variable PARAMETERS                   )
    (                                                          )

    (                                                          )
    ( MAKE                                                     )
    (                                                          )
    (       this is a defining word for creating and performing )
    (       OPTOMUX commands                                   )
    (                                                          )

: MAKE

    CREATE              ( create a new word                    )
    C,                  ( stick number in word                 )
    DOES>
    C@                  ( get command number                   )
    COMMAND !           ( stick in command variable            )
    OPTOWARE            ( call the driver                      )
;

    ( Setup Commmands )

     0      MAKE POWER.UP.CLEAR
     1      MAKE RESET
     3      MAKE WATCHDOG.DELAY
    45      MAKE ANALOG.WATCHDOG.DELAY
     4      MAKE PROTOCOL
     5      MAKE IDENTIFY.TYPE

    ( Configuration Commands )

     6      MAKE CONFIGURE.POSITIONS
     7      MAKE CONFIGURE.INPUTS
     8      MAKE CONFIGURE.OUTPUTS

    ( Gain & Offset Commands )

    52      MAKE CALCULATE.INPUT.OFFSETS
    53      MAKE SET.INPUT.OFFSETS
    54      MAKE CALCULATE.SET.INPUT.OFFSETS
    55      MAKE CALCULATE.GAIN.COEFFICIENTS
    56      MAKE SET.GAIN.COEFFICIENTS
    57      MAKE CALCULATE.SET.COEFFICIENTS
    ( On/Off Commands )

     9      MAKE WRITE.OUTPUTS
    10      MAKE ACTIVATE.OUTPUTS
    11      MAKE DEACTIVATE.OUTPUTS
    12      MAKE READ.STATUS
```

(Latch Commands)

```
13    MAKE SET.LATCH.EDGES
14    MAKE LATCH.OFF.TO.ON
15    MAKE LATCH.ON.TO.OFF
16    MAKE READ.LATCHES
17    MAKE READ.CLEAR.LATCHES
18    MAKE CLEAR.LATCHES
```

(Counting Commands)

```
19    MAKE START.STOP.COUNTERS
20    MAKE START.COUNTERS
21    MAKE STOP.COUNTERS
22    MAKE READ.COUNTERS
23    MAKE READ.CLEAR.COUNTERS
24    MAKE CLEAR.COUNTERS
```

(Time Delay & Pulse Commands)

```
25    MAKE TIME.DELAY
26    MAKE SQUARE.WAVE
27    MAKE CANCEL.DELAY.SQUARE.WAVE
```

(Duration Measurement Commands)

```
28    MAKE TRIGGER.POLARITY
29    MAKE TRIGGER.ON.POSITIVE
30    MAKE TRIGGER.ON.NEGATIVE
31    MAKE READ.DURATION.COMPLETE.BITS
32    MAKE READ.DURATIONS
33    MAKE READ.CLEAR.DURATION.COUNTERS
34    MAKE CLEAR.DURATION.COUNTERS
```

(Analog I/O Commands)

```
35    MAKE WRITE.ANALOG.OUTPUTS
36    MAKE READ.OUTPUTS
37    MAKE READ.INPUTS
38    MAKE AVERAGE.AND.READ.INPUT
46    MAKE UPDATE.OUTPUTS
47    MAKE START.INPUTS.AVERAGING
48    MAKE READ.AVERAGE.COMPLETE.BITS
49    MAKE READ.AVERAGED.INPUTS
```

```
( Input Range Commands )

 39     MAKE SET.RANGE
 40     MAKE READ.OUT.OF.RANGE.LATCHES
 41     MAKE READ.CLEAR.OUT.OF.RANGE.LATCHES
 42     MAKE CLEAR.OUT.OF.RANGE.LATCHES
 58     MAKE READ.LOWEST.VALUES
 60     MAKE READ.CLEAR.LOWEST.VALUES
 59     MAKE CLEAR.LOWEST.VALUES
 61     MAKE READ.PEAK.VALUES
 63     MAKE READ.CLEAR.PEAK.VALUES
 62     MAKE CLEAR.PEAK.VALUES

( Waveform Commands )

 43     MAKE SET.WAVEFORM
 44     MAKE CANCEL.WAVEFORM
 50     MAKE IMPROVED.OUTPUT.WAVEFORMS
 51     MAKE CANCEL.ENHANCED.WAVEFORMS

( Driver Commands )

100     MAKE DRIVER.PROTOCOL
101     MAKE TURN.AROUND.DELAY
103     MAKE NUMBER.OF.RETRIES

( ********** end of OPTOMUX commands **********  )
```

OPTO 22

43044 Business Park Drive • Temecula, CA 92590-3614
Phone: 800/321-OPTO (6786) or 951/695-3000
Fax: 800/832-OPTO (6786) or 951/695-2712
Internet Web site: http://www.opto22.com

Product Support Services:
800/TEK-OPTO (835-6786) or 951/695-3080
Fax: 951/695-3017
E-mail: support@opto22.com
Bulletin Board System (BBS): 951/695-1367
FTP site: ftp.opto22.com