ICS ELECTRONICS
division of Systems West Inc.

IEEE 488 APPLICATION BULLETIN

USING THE 4863/2363 STATUS REPORTING STRUCTURE TO MONITOR DIGITAL SIGNALS

INTRODUCTION

IEEE-488.2 compatible devices like ICS's 4863 and 2363 have a Status Reporting Structure that can be used to report internal events and to monitor external signals for changes. The way these registers work and their variety of programmable options is often confusing to the user. This Application Note provides information on how the event registers work and how to program the 4863 or the 2363 to take advantage of their ability to detect signal changes. The program information in this application note is applicable to ICS's Model 4803 and 2303 OEM boards, to all of the 486x and 236x series Interfaces and to other IEEE-488.2 compatible instruments.

BACKGROUND

ICS's 4863 or 2363 Digital Interfaces, like all other IEEE 488.2 devices, have an Event Status Register that reports command and device errors. The bit assignments in the Event Status Register are defined by the IEEE 488.2 Standard. The 4863 and 2363 also have an additional Questionable Register that can be used to report changes in fifteen digital input lines. Bit assignments in the Questionable Register correspond to the modules' first fifteen digital input lines and ultimately to the signals that the modules are connected to. By properly setting the Questionable Transition and Enable Registers, the modules can continuously monitor selected signals for changes and generate an Service Request when a change occurs. This capability relieves the user of having to continuously poll the modules to check the signal status.

STATUS REGISTER OPERATION

Figure 1 shows the 4863/2363's Status Reporting Structure. The 4863/2363's Status Reporting Structure is similar to the standard IEEE-488.2 Structure but includes the Questionable and Conditional Registers added by the SCPI Standard. (The SCPI Standard defined a standard command set for programmable instruments and added two register to the IEEE-488.2 Status Structure.)

EVENT STATUS REGISTER SET

The Event Status Register is shown at the top of Figure 1. In an IEEE-488.2 device, bits in the Event Status Register report command and device errors as specified in the IEEE-488.2 Standard. The some of the Event Status Register bits report command syntax, parameter and value out of range errors. Two of the Event Status Register bits can be used by the device designer. In the 4863/2363, bit 6 is used to indicate that data is available from an external device. Bits in the Event Status Register are set when the corresponding condition occurs. i.e. Bit 5 sets when a unrecognized command is detected. Once an event bit is set, it stays on until the register is read or cleared by the *CLS command.

The Event Status Enable Register is located just below the Event Status Register in Figure 1. The Enable register is used to select which bits will be ORed together to create the ESR summary signal. When an Enable Register bit is set and the corresponding Event bit sets, the summary bit becomes true. The ESR summary signal reports to bit 5 in the Status Byte Register at the bottom of Figure 1. The user can set or query the Event Status Enable register at any time.

QUESTIONABLE REGISTER SET

The Questionable Register set is shown in the center right portion of Figure 1. The Questionable Register Set was added to the IEEE-488.2 structure by the SCPI Standard and consists of a Condition Register, a Transition Register, an Event Register and an Enable Register. SCPI STATUS subsystem commands are used to control and query the Questionable Register.

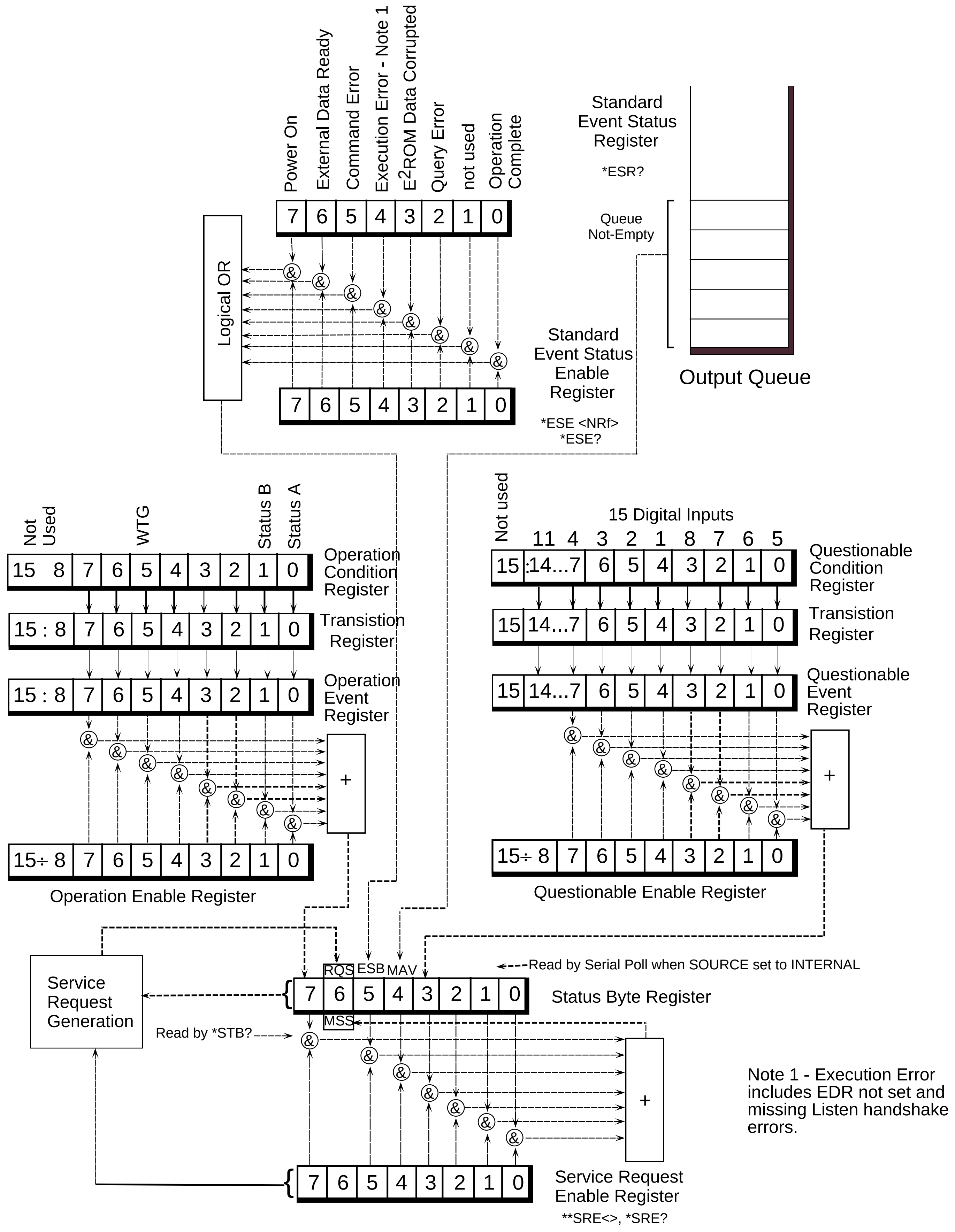

Figure 1 4863/2363 Status Structure

The Questionable Condition Register reports the current status of any signals connected to it. In the case of the 4863 and 2363, this is the state of the first fifteen digital input lines. A Conditional Register can be queried at any time to check on the state of its input signals with the STAT:QUES:COND? query.

The Questionable Transition Register is a two stage filter that lets only selected signal changes pass onto the Questionable Event Register. The Transition register stages can sense positive and/or negative going changes in the Conditional Register. Since the Condition register always reflects the status of its input signals, the transition register is effectively operating on the input signals. To sense a positive going signal change (0->1), set the corresponding PTR bit in the Transition Register. To sense a negative going signal change (1->0), set the corresponding NTR bit in the Transition Register. When a transition is sensed, the corresponding bit in the Questionable Event Register becomes set. The following example sets bit 0 for a positive signal transition, bit 1 for a negative signal transition and bit 2 for a signal change in either direction.

```
e.g.    STAT:QUES:PTR 5
        STAT:QUES:NTR 6
```

Sets the Transition Register bits as shown in the figure on the right

2	1	0	Bits
+		+	
-	-		

The Questionable Event and Enable Registers are similar to those in the Event Status Register. An input change from the Transition Register sets the corresponding bit in the Questionable Event Register. Bits in the Questionable Event Register stay set until the register is read or cleared with the *CLS command. The Questionable Enable register is used to select which Event bits will be ORed together to create the Questionable summary signal. When a Questionable Enable Register bit is set and the corresponding Questionable Event bit sets, the summary bit becomes true. The Questionable summary signal reports to bit 3 in the Status Byte Register at the bottom of Figure 1. The user can set or query the Questionable Enable register at any time. The Questionable Enable Register is set with the STAT:QUES:ENAB command.

STATUS BYTE REGISTER SET

The Status Byte Registers are shown at the bottom of Figure 1. They consist of a Status register and an Enable register. The Status Register follows the state of its input signals and is used as a point to summarize the signals. It can be queried repeatedly by serial polling or with the *STB? query without changing the status of its inputs. The Status Request Enable Register controls which Status Register bits will set the RQS/MSS bit (bit 6) and generate a Service Request. In the 4863, the Service Request asserts the SRQ line on the GPIB bus. In the 2363, the Service Request sends a serial message to the host computer. The message format is SRMnn where nn is the decimal value of the Status Byte Register. The Status Request Enable Register is controlled by the IEEE-488.2 *SRE command. e.g. *SRE 40 sets Status Request Enable Register bits 3 and 5 which in turn enable either the ESR or the Questionable Register summary bits to generate a Service Request.

When multiple bits or events are enabled, a Service Request is generated by the first bit to become true. A new Service Request will not be generated if a subsequent bit becomes true while the original bit is still on. Put another way, the user should service an enabled event as soon as it is reported to avoid blocking another Service Request.

4863 APPLICATION

The Event63 program demonstrates how the 4863 or 2363 can be programmed to generate Service Requests for command syntax errors and digital input signal changes. A GPIB cable is used to connect a 4863 to a PC with an internal ICS Model 488-PC2 Controller Card as shown in Figure 2. (A null modem cable can be used to connect a 2363 to a PC's COM port.)

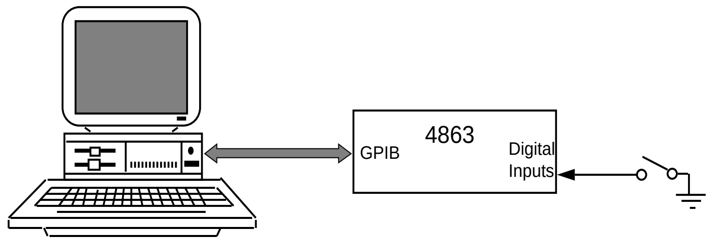

Figure 2 4863 Demonstration Setup

Event63 is a Quick Basic program that runs in DOS or in a DOS window on a PC. The program, shown in Figure 3, lets the user select an internal GPIB card in the PC to drive a 4863 over the GPIB bus or select a COM port to drive a 2363 with an RS-232 interface. The heart of the program is the setup commands that are sent to the 4863 or 2363 and the queries used to check the Questionable Registers. They can be duplicated in your own program and with minor modifications, be expanded to include more input signals, sense other signal transitions and more Event Status conditions.

The Event63 program is longer than a user's application program would be because Event63 is a demonstration program that supports both serial and GPIB devices. A large part of the program deals with the user's selection of the interface and its initialization. This dual usage also complicates the program's I/O functions.

When the program starts, the user selects the interface and the program then initializes the interface. The Device Setup commands configure the first eight digital signals as inputs, set the Questionable NTR register to sense negative signal transitions, and set the Questionable Enable register to OR all eight input bits onto the Questionable Summary line. The setup commands also enable bit 5 in the Event Status Register to report command syntax errors.

In the Test Loop, the program then loops waiting for a key stroke, SRQ or SRM message. Command syntax errors are created by sending a bad command (*EEE) to the module when the user presses the 'E' key. Digital input signal changes are created when the user grounds one of the first eight input lines. (Use CH1 on pin 20) . Either event causes the 4863 to generate an SRQ or the 2363 to output a SRM message. The value of the status byte causes the program to branch to a Questionable register service routine or to the ESR service routine. The function, *msDelay*, is borowed from ICS's IEEE-488.2 GPIB Driver.

In the Questionable register routine, the Status Byte Register, Questionable Event and Condition registers are sampled every 1.5 seconds until the digital line is no longer grounded. Because the Questionable Event register cleared when it was first read, the Status Byte and Questionable Event Registers read zero after the first sample.

In the ESR routine, the Status Byte Register and ESR register are read twice. The first time, the registers display the value for the syntax error, the second time they read zero since the ESR register was cleared when it was first read.

SUMMARY

This application note has described how the 4863 and 2363's Status Reporting Structure works, how to program the modules to monitor their digital inputs and how to generate Service Requests when a signals change state. By changing the NTR and PTR Transition Register settings and the Enable Register mask, the user can have the modules monitor any combination of input signals. Source and executable versions of the listed Event63 program are available on disk so the user can try the program before writing one of his own or copy portions of it into the final application program.

```
DECLARE SUB Sout (ADDR%, Soutstr$)
DECLARE SUB Clrlines (Topline!, Bottomline!)
‘
‘****************************************************************
‘ FILE NAME:    Event63.BAS                                     *
‘                                                               *
‘           Event Detection Demonstration Program               *
‘                                                               *
‘ DATE:        02-25-97                                         *
‘ AUTHOR:      G.MERCOLA                                        *
‘ REMARK:      Program uses Questionable Register to capture signal *
‘           changes.                                            *
‘                                                               *
‘ LIBRARY FILE: PC2qbdos.QLB If run from inside the environment *
‘           Place program, include files and libraries in QB45  directory *
‘           From DOS type >QB event63 /l pc2qbdos               *
‘           Use Alt R and Start to run the program              *
‘****************************************************************
‘   N.B.  Change RevDate$ when making changes
‘REVISION       CHANGE                                          *

‘=========================================================
COMMON SHARED EVENT.STATUS$, RDG$, UUT%, Escstr$
COMMON SHARED SPACE64S  AS STRING * 64 ‘space string for initializing input strings

DIM SHARED Serflg AS INTEGER
DIM SHARED L AS INTEGER

‘ $INCLUDE: ‘C:\QB45\PC2INCL.BAS’
‘ $INCLUDE: ‘C:\QB45\PC2COM.BAS’
‘
‘ initialize string variables
‘
RevDate$ = “03-11-97”
```

Figure 3 EVENT63 Program Listing

```
SETTING% = &H100                        ‘select basic compiler
T% = 1000                         ‘set time out period to 1 sec
DEFINT A-Z
EVENT.STATUS$ = “*ESR?”
Escstr$ = “UNL LISTEN 04 UNL LISTEN 04 UNL”      ‘minibox escape sequence
SPACE64S = STRING$(54, “ “)
‘  Set device addresses
UUT% = 4                          ‘GPIB or serial device address
NoAddr% = -1
‘
‘ INITIALIZATION
‘
‘ SELECTION SCREEN
‘
  COLOR 15, 1                       ‘white letters
  CLS
  LOCATE 2, 12                      ‘Title
  PRINT “WELCOME TO THE ICS SIGNAL CHANGE DETECTION PROGRAM”
  LOCATE 3, 28
  PRINT “Revised “ + RevDate$
Ireenter:                         ‘I/O port selection
  LOCATE 5, 5
  dummy$ = “      “
  INPUT “Select 488-PC2 card, 4818, or COMM port (4818/488-PC2/1 or 2)”, dummy$
  dummy$ = UCASE$(dummy$)
  LOCATE 5, 1
  PRINT SPACE80S
  IF dummy$ = “4818” THEN                ‘select GPIB controller type
          IOPORT% = &H378
          CALL ieInit(IOPORT%, MYADDR%, SETTING%)
          CALL ieTimeOut(T%)
          Serflg = 0
          PRINT “    Set test unit to address 4 “
          msDelay 2000
  ELSEIF dummy$ = “488-PC2” THEN
          IOPORT% = &H2E1
          CALL ieInit(IOPORT%, MYADDR%, SETTING%)
          CALL ieTimeOut(T%)
          PRINT “    Set test unit to address 4 “
          Serflg = 0
          msDelay 2000
  ELSEIF dummy$ = “1” THEN
          Serflg = 1
  ELSEIF dummy$ = “2” THEN
          Serflg = 2
  ELSE
      GOTO Ireenter
  END IF

  IF Serflg = 0 THEN
    CALL ieInit(IOPORT%, MYADDR%, SETTING%)
    CALL ieTimeOut(T%)
    CALL ieRemote(NoAddr%)
    CALL ieAbort                   ‘reset the bus and take control
    msDelay 50
    CmdStr$ = “*CLS”
    CALL Sout(UUT%, CmdStr$)
    msDelay 50
  END IF

  IF Serflg <> 0 THEN
   dummy$ = “                        “
   LOCATE 5, 5
   PRINT “Default settings are: 9600 baud, no parity, 8 data bits, 1 stop bit “
   PRINT “    and LF terminator.  Are these correct?   Press Enter for yes or “
```

Figure 3   EVENT63 Program Listing Cont'd

```
  INPUT "    enter new setting formated as '9600,N,8,1,LF' ", dummy$
  PRINT
'
  IF dummy$ = "" THEN
    SETTING$ = "COM1:9600,N,8,1,LF"
    IF Serflg = 1 THEN
         SETTING$ = "COM1:" + "9600,N,8,1,LF"
    ELSE
         SETTING$ = "COM2:" + "9600,N,8,1,LF"
    END IF
  ELSE
    IF Serflg = 1 THEN
         SETTING$ = "COM1:" + dummy$
    ELSE
         SETTING$ = "COM2:" + dummy$
    END IF
  END IF

  CLOSE
  IF Serflg = 1 THEN
    OPEN SETTING$ FOR RANDOM AS #2 LEN = 4096
    PRINT "     ***COM1 Selected***"
    CmdStr$ = CHR$(6) + "*CLS"
    CALL Sout(UUT%, CmdStr$)
  ELSEIF Serflg = 2 THEN
    OPEN SETTING$ FOR RANDOM AS #2 LEN = 4096
    PRINT "      ***COM2 Selected***"
    CmdStr$ = CHR$(6) + "*CLS"
    CALL Sout(UUT%, CmdStr$)
  END IF
 END IF
'
' DEVICE SETUP
'
 PRINT "    Initializing first 8 digital inputs to sense negative signal changes"
 CmdStr$ = "*cls"                   'reset status flags
 CALL Sout(UUT%, CmdStr$)
 CmdStr$ = "CONF:INP (@ 1)"          'set byte #1 to inputs
 CALL Sout(UUT%, CmdStr$)
 CmdStr$ = "STAT:QUES:NTR 255"        'set negative transition event
 CALL Sout(UUT%, CmdStr$)
 CmdStr$ = "STAT:QUES:ENAB 255"        'enable questionable register bits
 CALL Sout(UUT%, CmdStr$)
 CmdStr$ = "*ESE 32"                 'enable ESR register bits
 CALL Sout(UUT%, CmdStr$)
 CmdStr$ = "*cls"                   'reset status flags
 msDelay 100
 CALL Sout(UUT%, CmdStr$)
 CmdStr$ = "*SRE 40"                 'enable questionable and ESR summary bits in Status Reg
 CALL Sout(UUT%, CmdStr$)
 CmdStr$ = "*cls"                   'reset status flags
 msDelay 100
 CALL Sout(UUT%, CmdStr$)

' any change in the input bits will generate an SRQ (GPIB) or SRM (Serial message)
 PRINT "    Ground CH1-CH8 inputs to create a Questionable Register Event"
 PRINT "    (J2 pins 18-20, 39-41, or 60-61"
 PRINT "    Press E for an ESR error, press Q to quit"
 ReadFlag = 0
'
```

Figure 3   EVENT63 Program Listing Cont'd

```
‘ TEST LOOP
‘
 DO
   IF Serflg = 0 THEN
     SRQ% = ieSRQStat
     IF SRQ% = 1 THEN                    ‘if SRQ then set ReadFlag
             CmdStr$ = “*STB?”                  ‘do an *STB? here to set Readflg
             CALL Sout(UUT%, CmdStr$)
             ReadFlag = VAL(RDG$)
     END IF
   ELSEIF Serflg <> 0 THEN
     IF EOF(2) <> -1 THEN                ‘if SRM in Rx buffer then set ReadFlag
             RDG$ = “”
             RDG$ = RDG$ + INPUT$(LOC(2), #2)
             L = INSTR(RDG$, “SRM”)
             IF L <> 0 THEN
               ReadFlag = VAL(MID$(RDG$, (L + 3), (L + 6)))
             END IF
     END IF
   END IF
   IF ReadFlag <> 0 THEN
     DO UNTIL ReadFlag = 0
             BEEP
             IF (ReadFlag AND 8) = 8 THEN              ‘start questionable register loop
               CmdStr$ = “*STB?”
               CALL Sout(UUT%, CmdStr$)
               LOCATE 17, 1
               PRINT SPACE64S
               LOCATE 17, 1
               PRINT “    Status Byte =  “ + RDG$
               CmdStr$ = “STAT:QUES:EVENT?”
               CALL Sout(UUT%, CmdStr$)
               LOCATE 19, 1
               PRINT SPACE64S
               LOCATE 19, 1
               PRINT “    Questionable Event Register =  “ + RDG$
               CmdStr$ = “STAT:QUES:COND?”
               CALL Sout(UUT%, CmdStr$)
               Inputs% = 32767 - VAL(RDG$)
               LOCATE 21, 1
               PRINT SPACE64S
               LOCATE 21, 1
               PRINT “    Questionable Condition Register =  “ + STR$(Inputs%)
               msDelay 1500                      ‘wait 1.5 second
               IF Inputs% = 0 THEN
                 ReadFlag = 0                    ‘stop loop when signal no longer grounded
                 CALL Clrlines(17, 21)               ‘clear display
               END IF
             ELSEIF (ReadFlag AND 32) = 32 THEN        ‘else start ESR loop
               CmdStr$ = “*STB?”
               CALL Sout(UUT%, CmdStr$)
               LOCATE 17, 1
               PRINT SPACE64S
               LOCATE 17, 1
               PRINT “    Status Byte =  “ + RDG$
               CmdStr$ = “*ESR?”
               CALL Sout(UUT%, CmdStr$)
               LOCATE 19, 1
               PRINT SPACE64S
               LOCATE 19, 1
               PRINT “    Event Status Register =  “ + RDG$
               msDelay 1500                      ‘wait 1.5 second
               IF VAL(RDG$) = 0 THEN
                 ReadFlag = 0                    ‘stop loop when ESR condition cleared
                 CALL Clrlines(17, 21)               ‘clear display
```

Figure 3 EVENT63 Program Listing Cont'd

```
                END IF
               ELSE
                ReadFlag = 0
               END IF
      LOOP
     END IF
    A$ = INKEY$
    IF UCASE$(A$) = "E" THEN                  'routine to send a bad command
      BEEP
      LOCATE 17, 5
      PRINT "Sending bad command"
      BadCmd$ = "*eee"
      IF Serflg = 0 THEN                  'GPIB output
       CALL ieOutput(UUT%, BadCmd$)
      ELSE                                'serial output
       IF NET.ENA% = 1 THEN
             BadCmd$ = CHR$(2) + CHR$(48 + ADDR%) + BadCmd$
       END IF
       PRINT #2, BadCmd$
      END IF
      msDelay 500
      LOCATE 17, 5
      PRINT SPACE64S
    ELSE
      IF A$ <> "" THEN EXIT DO            'test for exit
    END IF
    LOOP
   END

   DEFSNG A-Z
   SUB Clrlines (A, B)   'clears selected screen line area
    FOR I% = A TO B
     LOCATE I%, 1
     PRINT SPACE80S     'blank the line
    NEXT I%
   END SUB

   SUB Sout (ADDR%, Soutstr$)
    IF Serflg = 0 THEN                    'GPIB output-input
      CALL ieOutput(ADDR%, Soutstr$)
      IF INSTR(Soutstr$, CHR$(63)) <> 0 THEN    'input if ?
       RDG$ = SPACE64S
       CALL ieEnter(ADDR%, RDG$)
      END IF
    ELSE                                  'serial output
      IF NET.ENA% = 1 THEN
           Soutstr$ = CHR$(2) + CHR$(48 + ADDR%) + Soutstr$
      END IF
      PRINT #2, Soutstr$
      msDelay 25
      msDelay 25
'     RDG$ = SPACE64S
       RDG$ = ""
      Prompt = 0
      DO
           RDG$ = RDG$ + INPUT$(LOC(2), #2)        'input until prompt found
           Prompt = INSTR(RDG$, CHR$(62))
      LOOP UNTIL Prompt <> 0
      RDG$ = RDG$ + INPUT$(LOC(2), #2)         'get last LF
      IF INSTR(RDG$, CHR$(10)) <> 0 THEN       'discard prompt
           RDG$ = LEFT$(RDG$, INSTR(RDG$, CHR$(10)))
      END IF
     END IF
   END SUB
```

Figure 3   EVENT63 Program Listing Cont'd