# 4894A Errata

## E.1 INTRODUCTION

This Errata contains additional information about operating the 4894A in the S Mode since Revision 9 of the 4894A Manual was printed.

## E.2 ADDED S MODE PROGRAMMING GUIDE

Pages E-2 through E-7 contain new 4894A Manual Pages and include a Quick Basic Program Example.

Rev 06-20-03

## 4.4 PROGRAMMING GUIDELINES

The following section provides information on how to program the 4894A and 4804A in the S Mode. Any comment about the 4894A applies equally to the 4804A.

### 4.4.1 General Concept

In the S Mode, the 4894A and the 4804A act as simple GPIB Bus Controllers to pass serial commands to a GPIB device and to read back device responses and to transmit them serially to the host computer. The command strings and data are passed transparently through the 4894A. The 4894A's internal GPIB Address is the address used to address the device being controlled. The 4894A has no capability to send GPIB Commands like IFC (Abort), Device Clear, Serial Poll, Get or Parallel Poll commands. You can however send all of the IEEE-488.2 Common Commands like *RST, *IDN?, *STB?, *TRG etc. which do many of the same functions. The following suggestions are offered to help you avoid problems:

Try out your device command strings using Hyperterminal or some other terminal emulation program. This way you can be sure of the command syntax and the device responses. When hand typing in the commands, set the 4894A SWAP parameter to LF (linefeed). The SWAP Time setting will cause device errors by sending the device incomplete commands.

When writing your program, check the computer's buffer for a complete response before processing the response. Allow ample time to receive the response string. Remember Windows is not a real time operating system.

### 4.4.2 IDN Query Loop

The Example Program in Figure 4-2 is a Quick Basic example of a loop that queries the IDN message from an IEEE 488.2 compatible device. This example is taken from a larger test program so it contains string tests and error counters to verify long term performance. You can remove these tests and convert it into a Visual Basic, C or LabView program. The steps would be the same in any language.

The program first gives the user a choice of COM ports. The Serflg is the COM port variable. msDelay is a function from our GPIB Controller library that provides delays in milliseconds. You can use any delay mechanism to achieve the same function. Clrlines is a function that clears specific lines on the monitor.

The test loop starts with 'DO WHILE INKEY$ = “”' in the middle of page 4-7 and ends with 'LOOP" at the bottom of page 4-8. The test loop runs until any key is pressed. "*IDN?" is sent out to the 4894A with the 'PRINT #2, CmdStr$' command on page 4-7 . A DO loop is set up to read the input buffer until the terminator is found or until the loop times out. The DO loop is important because you do not want to input a partial message into your program nor do you want to hang up your program if there is no response. If the received message length is not zero, the terminating character is removed and the received message compared against the last received message. In your program, you should test the received message to be sure the message is a nonzero length and then go on to process it. Provide an error recovery process in case you do not receive a response.

The complete message send-receive process can be put into a subroutine that will output the command string and then only read data when your command contains a question mark (?). Alternately the message send-receive process can be written as two routines.

The remainder of the example program displays the message and test statistics. These would not be part of a normal application program.

```
*********************************************************
‘  IDNRead Loop
‘ *********************************************************
IDN.LOOP:
  Er% = 0

IDN.LOOP2: LOCATE 15, 5
  PRINT SPACE64S
  LOCATE 15, 5
  dummy$ = “”
  INPUT “Select COMM port 1 or 2 “, dummy$
  dummy$ = UCASE$(dummy$)
  IF dummy$ = “1” THEN
       Serflg = 1
  ELSEIF dummy$ = “2” THEN
       Serflg = 2
  ELSE
     BEEP
     LOCATE 14, 1
     PRINT SPACE80S
     LOCATE 14, 5
     PRINT “Invalid entry - please re-enter”
     GOTO IDN.LOOP2:
  END IF

  dummy$ = “”
  LOCATE 16, 5
  PRINT “Setting serial port to: 9600 baud, no parity, 8 data
bits, 1 stop bit “
  PRINT “    Using LF terminator.”
  PRINT
  IF Serflg% = 1 THEN
      Setting$ = “COM1:” + “9600,N,8,1,LF”
  ELSE
      Setting$ = “COM2:” + “9600,N,8,1,LF”
  END IF

  msDelay 200
  IF Serflg% = 1 THEN
    OPEN Setting$ FOR RANDOM AS #2 LEN = 4096
    PRINT “     ***COM1 Selected***”
  ELSEIF Serflg% = 2 THEN
    OPEN Setting$ FOR RANDOM AS #2 LEN = 4096
    PRINT “      ***COM2 Selected***”
  END IF
  Rdg$ = INPUT$(LOC(2), #2)      ‘clean up input buffer
  Rdg$ = “”
```

**Figure 4-2 Quick Basic Example Program**

```
  CALL Clrlines(15, 20)
  LOCATE 15, 5
  PRINT “Reading Device until any key pressed”
  LOCATE 16, 5
   PRINT “Test Loop, Delay Count, TermChar Position, Error
Count, Lp x 10K, Reading”

  I = 0
  Lpnum = 0
  lasttcnt = 0
  lastrdg$ = “”
  Termchr$ = CHR$(10)

  DO WHILE INKEY$ = “”
    CmdStr$ = “*IDN?”
    PRINT #2, CmdStr$      ‘output query
    msDelay 100
    termcnt% = 0
    count = 0
    Rdg$ = “”
    DO
      Rdg$ = Rdg$ + INPUT$(LOC(2), #2)  ‘input until
                                         terminator found
      termcnt% = INSTR(Rdg$, Termchr$)
      count = count + 1
      IF count > 5000 THEN
       BEEP
       PRINT “Timeout-did not receive device IDN response”
       INPUT “    Reset devices and restart test.  Press
                                Enter when done “, dummy$
       RETURN
      END IF
    LOOP UNTIL termcnt% <> 0
    L = LEN(Rdg$)
    IF L <> 0 THEN
      Rdg$ = LEFT$(Rdg$, L - 1)
    ELSE
      Rdg$ = “No message received”
    END IF
    LOCATE 18, 1
    PRINT SPACE64S
```

**Figure 4-2 Quick Basic Example Program Continued**

```
    LOCATE 17, 5
    IF termcnt% < 80 THEN
    PRINT STR$(I) + "  " + STR$(count) + "  " + STR$(termcnt%)
+ "  " + STR$(Er%) + "   " + STR$(Lpnum)
      LOCATE 18, 5
      PRINT Rdg$              'display IDN message"
    ELSE
      BEEP
    PRINT STR$(I) + "  " + STR$(count) + "  " + STR$(termcnt%)
+ "  " + STR$(Er%) + "   " + STR$(Lpnum) + " Term Cnt Error"
      LOCATE 18, 5
      PRINT Rdg$                'display IDN message"
      Er% = Er% + 1
    END IF
    IF (Rdg$ <> lastrdg$) THEN
      IF (I > 1) THEN
       LOCATE 20, 5
       PRINT "Prior Rdg = "
       LOCATE 21, 5
       PRINT lastrdg$
       LOCATE 22, 5
       IF Stopflg = 1 THEN
         INPUT "    Press Enter to continue", dummy$
       ELSE
         msDelay 2000
       END IF
       LOCATE 20, 5
       PRINT SPACE64S
       PRINT SPACE64S
       PRINT SPACE64S
       Er% = Er% + 1
      END IF
    END IF
    lastrdg$ = Rdg$
    lasttcnt = termcnt%'save last terminator position count
    I = I + 1
    IF I > 10000 THEN
      I = 1
      Lpnum = Lpnum + 1
      LOCATE 17, 1
      PRINT SPACE64S
    END IF
  LOOP
RETURN
```

**Figure 4-2 Quick Basic Example Program Continued**