ICS ELECTRONICS
division of Systems West Inc.

IEEE 488 APPLICATION BULLETIN

USING THE 4896 TO PROVIDE SERIAL INPUTS TO HP's SERIES 700 AND OTHER COMPUTERS

INTRODUCTION

Large numbers of Hewlett-Packard Series 300 computers have been used in industrial applications in the past decade. Some of these applications required multiple serial channels to communicate with devices on the factory floor. As these computers age, they are being replaced with newer computers. However, newer HP computers like the Series 700 computers, do not have the capability to handle multiple serial devices. This application note shows how ICS' Model 4896 GPIB-to-Quad Serial Interface can be used to interface multiple serial devices to the newer Hewlett-Packard computers.

While this application note describes how to add serial channels to HP computers, the concepts can be applied to Sun workstations and other computer systems with a limited number of serial channels.

BACKGROUND

In industrial applications, the computer system often has to collect data from numerous sensors, and communicate with machine controllers and human input devices. A large number of these devices have serial interfaces and use serial ASCII messages to communicate with the computer.

In the early 1980s, a large number of industrial applications were implemented with Hewlett-Packard 9000 Series 300 computers. These HP Series 300 computers could be configured with various interface cards to adapt the computer to a specific application. One of these optional interface cards was a four channel, serial interface card that allowed the computer to interface up to four serial devices. By installing one or more of these cards in the computer, the user created a computer system with multiple serial channels that could input data from factory sensors, collect work status and control devices with serial interfaces.

Nearly a decade later, these computers are wearing out. Maintenance has become difficult as parts are harder to get and new third party programs are not being written for these older computers. Therefore, many users want to replace these older computers to eliminate the maintenance problems, increase computation speed and to take advantage of the newer networks and factory automation programs. When updating the computer system, many users would like to stay with the Hewlett-Packard computers to preserve as much as possible of their existing software. Hewlett-Packard's new Series 700 computer is the ideal replacement candidate for the older Series 300 computers.

The problem with the HP Series 700 computer in this application is that it has only two serial ports on the main board and does not have an optional serial card for interfacing to additional serial devices. It does however have a HP-IB interface that can be used to control GPIB devices. This application note shows how the HP-IB interface can be used with ICS's Model 4896 GPIB-to-Quad Serial Interface to interface multiple serial devices to the HP Series 700 computer.

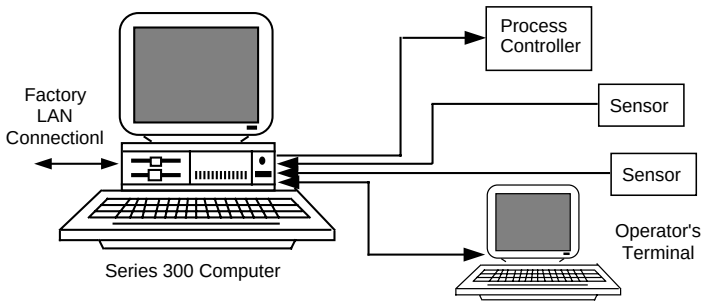

Figure 1 Original HP Series 300 Computer in a Factory Application

TYPICAL FACTORY CONFIGURATION

Figure 1 shows the a HP Series 300 computer in a typical factory data collection system. The computer may be connected to numerous serial devices such as temperature controllers, data terminals. It is also linked to other computers in the factory.

NEW CONFIGURATION

Replacing the computer in the above system with a new HP Series 700 computer without multiple serial interfaces isolates the factory sensors. Figure 2 shows how ICS's 4896 GPIB-to-Quad Serial Interface is used to connect the serial devices in a factory to the HP Series 700 computer. The 4896 connects to the HP 700's HP-IB bus and provides four serial ports. Each port in the 4896 can be set to communicate with RS-232 single ended or with RS-422/RS-485 differential signals. Since each 4896 is only one GPIB load, up to fourteen 4896s can be connected to the computer without using a Bus Expander[1].

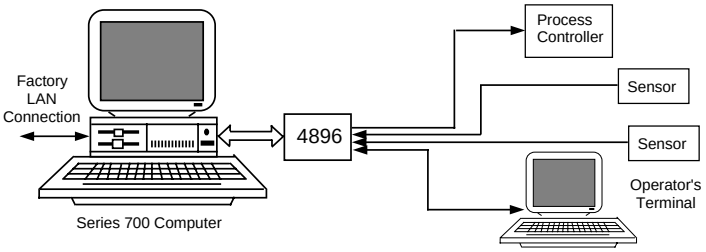

Figure 2 HP 700 using a 4896 for Factory Communication

Each 4896 has its own primary GPIB address and uses secondary addresses to address the serial data channels and to change their configuration. Secondary addresses 01 to 04 are the data channels addresses. Secondary addresses 11 to 14 are the channel control addresses. The 4896 has a common IEEE-488.2 Status Reporting Structure that can be accessed or queried at any control address.

Data transmission through the 4896 is transparent. Addressing a 4896's serial data channel is the same as addressing the serial device. Commands from the computer are sent to the serial device with a simple OUTPUT command. Data from the serial devices is buffered in the 4896 and are read into the computer with a ENTER type command. The 4896 can be set to generate a SRQ when it has data in the buffer, when it has received a complete serial message or senses a break. The program can sense the SRQ and serial poll the 4896s to determine which device needs service and then query the 4896's 488.2 Status Registers to determine which channel has data or has detected a break.

1-A Bus Expander lets a GPIB bus drive up to 13 additional devices and another 20 meters of bus cable. Refer to ICS's 4860 data sheet for additional information.

In addition, the 4896 provides the user with the following enhancements over the serial card in the older HP 300 computers:

1. Nonvolatile storage of setup parameters so the setup is not lost during power outage.
2. Each serial channel supports RS-232 or RS-422/RS-485 serial devices.
3. Large 64 Kbyte buffers for each channel.
4. Front panel LCD display that shows data buffer status, channel settings, and serial signal status for troubleshooting serial communication problems.
5. Lower cost than the four channel serial card in the Series 300.

PROGRAM CHANGES AND CONSIDERATIONS

The Series 300 computer's serial cards are addressed at an internal card addresses. At turn on, each serial port is initialized by the program. The actual program depends upon the type of serial device connected to the serial port and the application. In the example program shown in Figure 3, Subroutine GP_configer is called to initialize the serial port at power turn-on. The initialization is a fixed set of parameters. Then the program periodically outputs a query to a serial device. The responses are handled by the Datacomm_intr interrupt routine which reads and displays the response from a vacuum gauge.

The 4896 has an enhanced IEEE-488.2 Status Structure which is shown in Figure 4. The Operation and Questionable Registers show the status of the data buffers and whether a break was detected. The enable register bits for each register let the corresponding bit, if set, be summarized and feed to the Status Byte Register. If the bit in the Status Byte register has its corresponding enable bit set, the 4896 generates a SRQ. When the SRQ occurs, the program is interrupted. The interrupt handler routine starts by reading the Status Byte Register and working backwards to determine the source of the interrupt. In a real system, several events may occur "simultaneously' so the program should be written to handle multiple events.

Figure 5 shows an example program for a serial channel in a 4896. The program is more complex than the serial program in Figure 3 because of the extra capabilities in the 4896 and the ability to pass parameters to the setup routine. The example program uses address 703 for data and address 713 for controlling the channel. At power turn-on, Subroutine GP316_config is passed the communication parameters for the channel. If a passed parameter is different from the previously saved value, the SAVE_FLG is set. Next, subroutine ICS4896_config is passed the values for setting up the Enable registers in the 4896's Status Structure. Again, if a passed value is different than the saved value, the SAVE_FLG

```
10! RE-SAVE “DATACOM_GP:HFS”
20!
30      REAL Vacuum(1:6)
40      DIM Temp$[500],Spoll$[8],Err$[80]
50!
60      ABORT 7
70      CLEAR 7
80!
90      ASSIGN @Gp TO 24
110     GOSUB Gp_config
130     ON TIMEOUT SC(@Gp),1 GOSUB Gp_timeout
140     ON INTR 7,10 GOSUB Datacomm_intr
150     ENABLE INTR 7;202               !intr on BREAK,EOL,MODEM LINE,PROMPT
160!
170 Vpc_read:!
180!
190     LOOP
200        OUTPUT @Gp;”DS CG1"
210        WAIT 1
220     END LOOP
230!
240 Datacomm_intr:!
250!
260     ENTER @Gp;Vac$
270     IF NOT LEN(Err$) THEN
280        PRINT “vacuum1=”;VAL(Vac$)
290     END IF
300     ENABLE INTR SC(@Gp)
310     RETURN
320!
330 Gp_config:!
340!
350     CALL Gp_configer(@Gp)
360     RETURN
370!
380 Gp_timeout:!
390!
400     DISP “GP Timeout”
410     RETURN
420     END
430!*******************************************************************************
440     SUB Gp_configer(@Gp)
450!CREATED:05/11/87  BY:GERRY HUCK        LAST REV:04/12/88  BY:GERRY HUCK
460!ROUTINE TO INITIALIZE A GP VACUUM GAUGE
480!WIRE CONNECTIONS;  HP300 SERIES/HP2397A= 1/1,2/2,3/3,6/6,7/7,20/20
520!
530        ON TIMEOUT SC(@Hp2397),5 GOTO Exit
540        CONTROL SC(@Hp2397),0;1       !RESET CARD
550        CONTROL SC(@Hp2397),8;2       !DRIVERS ALL OFF
560        CONTROL SC(@Hp2397),13;202    !INTR=BREAK,EOL,MODEM LINE,PROMPT
570        CONTROL SC(@Hp2397),14;3      !CONTROL BLOCK=EOL,PROMPT
580        CONTROL SC(@Hp2397),15;8      !INTR ON DTR LINE CHANGE
590        CONTROL SC(@Hp2397),16;0      !NO TIMEOUT LIMIT
600        CONTROL SC(@Hp2397),17;0      !NO ACTIVITY TIMEOUT DISABLED
610        CONTROL SC(@Hp2397),18;0      !NO LOST OF CARRIER TIMEOUT
620        CONTROL SC(@Hp2397),19;0      !NO XMIT TIMEOUT
630        CONTROL SC(@Hp2397),20;14     !XMIT 9600 BAUD
640        CONTROL SC(@Hp2397),21;14     !RMIT 9600 BAUD
650        CONTROL SC(@Hp2397),22;0      !NO XON/XOFF
660        CONTROL SC(@Hp2397),23;0      !NO HARDWARE HANDSHAKE
670        CONTROL SC(@Hp2397),24;60     !PASS=255,DEL,NUL,PROMPTS
680        CONTROL SC(@Hp2397),28;1,13,10 !EOL=CR
690        CONTROL SC(@Hp2397),31;1,30   !PROMPT=RS
700        CONTROL SC(@Hp2397),34;2      !7 BITS
710        CONTROL SC(@Hp2397),35;0      !1 STOP BIT
720        CONTROL SC(@Hp2397),36;1      !EVEN PARITY
730        CONTROL SC(@Hp2397),39;255    !BREAK TIME   WAS 150
740        CONTROL SC(@Hp2397),12;1      !CONNECT
750        ABORTIO @Hp2397
760     SUBEND
770  !*******************************************************************
```

Figure 3 Sample Program for HP Series 300 Serial Channel

is set. If the SAVE_FLG is set at the end of the ICS4896_config routine, the *SAV 0 command is used to save the current configuration. Note that the 4896 does not respond to the *PSC commands and its enable registers are saved by the *SAV 0 command.

Interrupt routine HPIB_intr is a general purpose handler that checks the Status Byte bits. If bit 3 is set, the Questionable register is read to see which channel has a received data message. If bit 4 is set, there is a message in the 4896's output queue. If bit 5 is set, the ESR Register is checked for error conditions. The errors are displayed on the monitor. If bit 7 is set, the Operation Register is read to check the Transmit Buffers and to if a break was detected. (Break detection was added to the 4896 after the sample program was written.) In a real application, the break is generated by a factory worker when he or she wants the system to 'wake up' or change its response to inputs at the terminal. When the interrupt has been serviced, the program resumes its normal operation.

SUMMARY

This application note has described how the 4896 GPIB-to-Quad Serial Interface can be used to interface multiple serial devices to an HP Series 700 computer in a factory environment. Sample programs are included to show the user how to program the 4896 to take advantage of some of its features. The 4896's IEEE-488.2 Status Reporting Structure provides the user with a great deal of flexibility and increased capability for handling multiple serial channels in factory applications. The concepts in this application note can be used to add serial channels to any computer that has a GPIB or HP-IB interface. Copies of the sample programs are available as text files from ICS's web site. They may be used as a starting point for your own application.

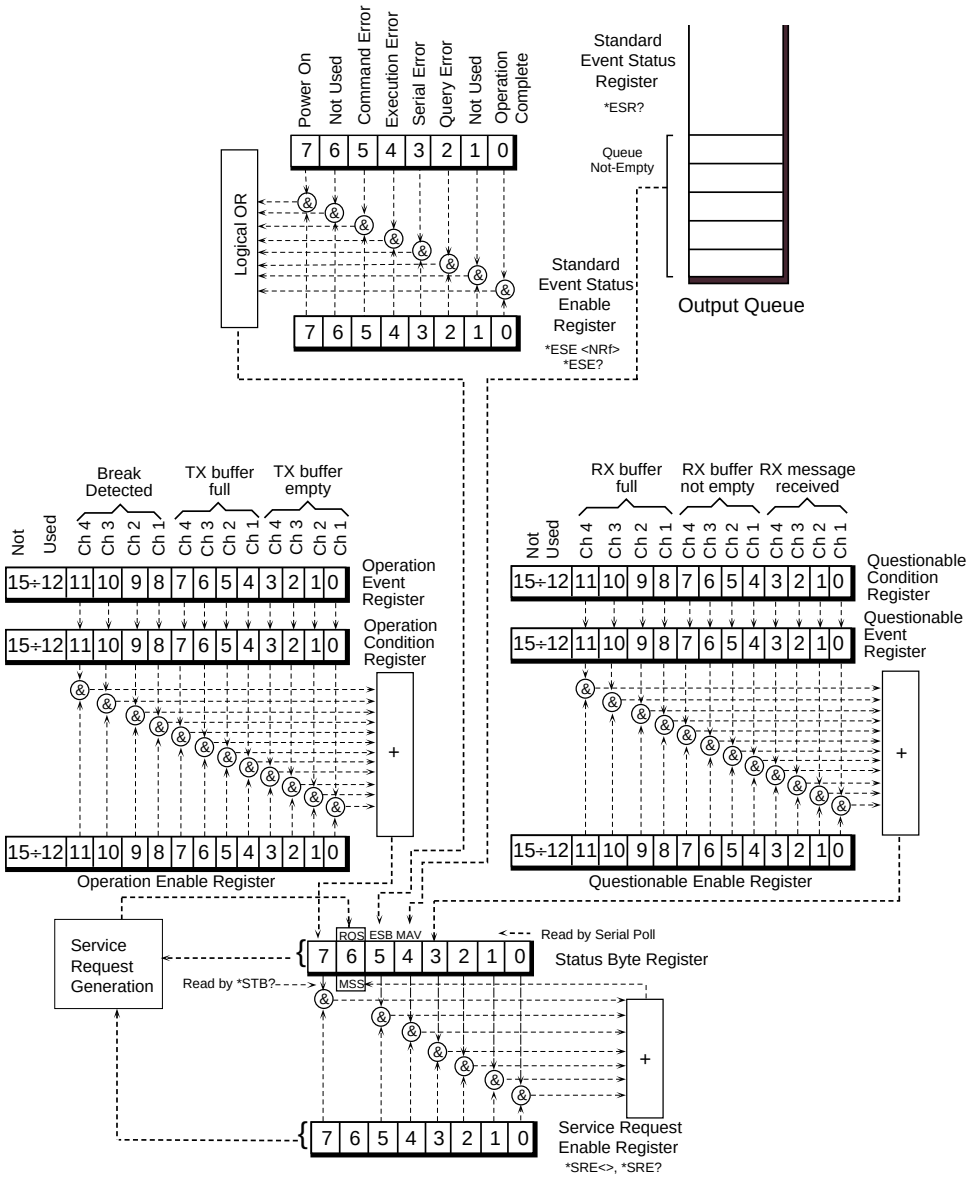

Figure 4 4896's IEEE 488.2 Status Reporting Structure

```
10! RE-SAVE “ISC4896_GP:HFS”
20!
30      INTEGER Spoll7,Ics_save_flg
40      REAL Vacuum(1:6)
50      DIM Temp$[500],Spoll$[8],Err$[80]
60!
70      ABORT 7
80      CLEAR 7
90      ASSIGN @Ics_gp316_conf TO 70413
100     ASSIGN @Ics_gp316_data TO 70403
110!
120     GOSUB Gp316_config
130     GOSUB Ics4896_config
140     OUTPUT 70413;”SYSTEM:DISPLAY:SELECT 1"
150!
160     ON TIMEOUT SC(@Ics_gp316),1 GOSUB Ics_timeout
170     ON INTR 7,10 GOSUB Hpib_intr
180     ENABLE INTR 7;2                 !Intr on SRQ
190!
200 Vpc_read:!
210!
220     LOOP
230        OUTPUT @Ics_gp316;”DS CG1"
240        WAIT 1
250     END LOOP
260!
270 Vpc_in:!
280!
290     ENTER @Ics_gp316;”DS CG1"
300     PRINT “vacuum1=”;VAL(Vac$)
310     RETURN
320!
330 Hpib_intr:!                 !!!!General purpose HPIB intr handler
340!
350     Spoll7=SPOLL(@Ics_gp307_conf)
360     IF BIT(Spoll7,6) THEN
370        Spoll7$=IVAL$(Spoll7,2)
380        PRINT “Intr. from ICS4896    SPOLL=”;Spoll7;”or “;Spoll7$[9]
390        IF BIT(Spoll7,3) THEN
400           OUTPUT @Ics_gp307_conf;”STATUS:QUESTIONABLE:EVENT?”
410           ENTER @Ics_gp307_conf;Ics_status
420           Ics_status$=IVAL$(Ics_status,2)
430           PRINT “Ics4896 Rx buffers intr. Value=”;Ics_status$[5]
440           IF BIT(Ics_status,3) THEN
450              GOSUB Vpc_in
460           END IF
470        END IF
480        IF BIT(Spoll7,4) THEN
490           PRINT “Ics4896 Msg buffers caused a intr.”
500        END IF
510        IF BIT(Spoll7,5) THEN
520           OUTPUT @Ics_gp307_conf;”*ESR?”
530           ENTER @Ics_gp307_conf;Ics_status
540           Ics_status$=IVAL$(Ics_status,2)
550           PRINT “Ics4896 Events intr. Value=”;Ics_status$[9]
560           IF BIT(Ics_status,0) THEN PRINT “>>>>>>> Operation Completed”
570           IF BIT(Ics_status,1) THEN PRINT “>>>>>>> Query Error”
580           IF BIT(Ics_status,3) THEN PRINT “>>>>>>> Serial Error”
590           IF BIT(Ics_status,4) THEN PRINT “>>>>>>> Execution Error”
600           IF BIT(Ics_status,4) THEN PRINT “>>>>>>> Command Error”
610           IF BIT(Ics_status,7) THEN PRINT “>>>>>>> Powered up”
620           GOSUB Gp316_config
630           GOSUB Ics4896_config
640        END IF
650        IF BIT(Spoll7,7) THEN
660           OUTPUT @Ics_gp307_conf;”STATUS:OPERATION:EVENT?”
670           ENTER @Ics_gp307_conf;Ics_status
680           Ics_status$=IVAL$(Ics_status,2)
690           PRINT “Ics4896 Tx buffer intr. Value=”;Ics_status$[9]
700        END IF
710     END IF
720     ENABLE INTR 7;2                 !Intr on SRQ
730     RETURN
740 !
```

Figure 5 Example 4896 Program for Channel 3

```
750 Ics4896_config: !
760 !
770     CALL Ics4896_init(@Ics_gp307_conf,0,189,0,32,Ics_save_flg)
780     RETURN
790 !
800 Gp316_config: !
810 !
820     CALL Ics4896_232init(@Ics_gp316_conf,9600,”EVEN”,”NONE”,7,1,-1,-1,0,0,0,1,0,Ics_save_flg)
830     RETURN
840 !
850 Ics_timeout: !
860 !
870     DISP “ICS/GP TIMEOUT”
880     RETURN
890     END
900 !*****************************************************************************
910     SUB Ics4896_init(@Ics,INTEGER Operation,Event,Questionable,Srq,Save_flg)
920 !Created 02/16/95  BY:Gerry Huck    Last Rev:—/—/—     BY:
930 !Configures ICS4896 SRQ intr registers.
940 !If Save_flg is set all parm’s will be saved in E2PROM
950 !This routine must be called after Comm ports have been set.
960 !
970         INTEGER Status
980         REAL Ics_addr
990         DIM Out$[30]
1000        ON TIMEOUT SC(@Ics),1 GOTO Timeout
1010!
1020        STATUS @Ics,3;Ics_addr
1030        OUTPUT @Ics
1040        OUTPUT @Ics;”*CLS”
1050!
1060        OUTPUT @Ics;”STATUS:OPERATION:ENABLE?”
1070        ENTER @Ics;Status
1080        IF Status<>Operation THEN
1090           OUTPUT @Ics USING “K,X,K”;”STATUS:OPERATION:ENABLE”,Operation
1100           PRINT “ICS4896 @”;Ics_addr;”Operation Status Reg. Incorrect Was=”;Status;” Needed=”;Operation
1110           Save_flg=1
1120        END IF
1130        OUTPUT @Ics;”*ESE?”
1140        ENTER @Ics;Status
1150        IF Status<>Event THEN
1160           OUTPUT @Ics USING “4A,X,K”;”*ESE”,Event
1170           PRINT “ICS4896 @”;Ics_addr;”Event Status Reg. Incorrect Was=”;Status;” Needed=”;Event
1180           Save_flg=1
1190        END IF
1200        OUTPUT @Ics;”STATUS:QUESTIONABLE:ENABLE?”
1210        ENTER @Ics;Status
1220        IF Status<>Questionable THEN
1230           OUTPUT @Ics USING “K,X,K”;”STATUS:QUESTIONABLE:ENABLE”;Questionable
1240           PRINT “ICS4896 @”;Ics_addr;”Questionable Status Reg. Incorrect Was=”;Status;” Needed=”;Questionable
1250           Save_flg=1
1260        END IF
1270        OUTPUT @Ics;”*SRE?”
1280        ENTER @Ics;Status
1290        IF Status<>Srq THEN
1300           OUTPUT @Ics USING “4A,X,K”;”*SRE”,Srq
1310           PRINT “ICS4896 @”;Ics_addr;”SRQ Status Reg. Incorrect Was=”;Status;” Needed=”;Srq
1320           Save_flg=1
1330        END IF
1340        IF Save_flg THEN
1350           OUTPUT @Ics;”SYSTEM:DISPLAY:SELECT 50"
1360           OUTPUT @Ics;”SYSTEM:DISPLAY:CREATE CONFIGURATION   SAVED @ “&TIME$(TIMEDATE)
1370           OUTPUT @Ics;”*SAV 0"
1380           Save_flg=0
1390        ELSE
1400           OUTPUT @Ics;”SYSTEM:DISPLAY:SELECT 1"        !1=Comms 1-17=valid 40=SCPI 50=user
1410        END IF
1420        SUBEXIT
1430!
```

Figure 5 Example 4896 Program for Channel 3 - Continued

```
1440 Timeout:!
1450!
1460        DISP “Timeout in SUB Ics4896_init @”;Ics_addr
1470        BEEP
1480        PAUSE
1490     SUBEND
1500!*****************************************************************************
1510     SUB Ics4896_232init(@Ics,Baud,Parity$,Pace$,INTEGER
Bits,Sbits,Eom,Add_char,Eoi,Echo,Loopback,Xmit,Extclk,Save_flg)
1520!Created 02/16/95  BY:Gerry Huck    Last Rev:—/—/—     BY:
1530!Configures ICS4896 serial port
1540!Save_flg is set if any parm has been modified.
1550!
1560        INTEGER Itemp
1570        REAL Address,Rtemp
1580        DIM Temp$[10]
1590        ON TIMEOUT SC(@Ics),1 GOTO Timeout
1600!
1610        STATUS @Ics,3;Address
1620        SELECT Baud
1630        CASE 300,600,1200,2400,4800,9600,19200,38400,57600,76800,115200
1640           OUTPUT @Ics;”SYSTEM:COMM:SERIAL:BAUD?”
1650           ENTER @Ics;Rtemp
1660           IF Rtemp<>Baud THEN
1670              OUTPUT @Ics USING “K,X,K”;”SYSTEM:COMM:SERIAL:BAUD”;Baud
1680              PRINT “ICS4896 @”;Address;”Baud Incorrect Was=”;Rtemp;” Needed=”;Baud
1690              Save_flg=1
1700           END IF
1710        CASE ELSE
1720           PRINT “ICS4896 @”;Address;”Asked for Invailed Baud of”;Baud
1730        END SELECT
1740        OUTPUT @Ics;”SYSTEM:COMM:SERIAL:PARITY:CHECK?”
1750        ENTER @Ics;Itemp
1760        OUTPUT @Ics;”SYSTEM:COMM:SERIAL:PARITY:TYPE?”
1770        ENTER @Ics;Temp$
1780        SELECT Parity$
1790        CASE “NONE”,”ODD”,”EVEN”
1800           IF Temp$<>Parity$ THEN
1810              OUTPUT @Ics;”SYSTEM:COMM:SERIAL:PARITY:TYPE “;Parity$
1820              PRINT “ICS4896 @”;Address;”Parity Incorrect Was=”;Temp$;” Needed=”;Parity$
1830              Save_flg=1
1840           END IF
1850           IF Parity$=”NONE” AND Itemp THEN
1860              OUTPUT @Ics;”SYSTEM:COMM:SERIAL:PARITY:CHECK OFF”
1870              PRINT “ICS4896 @”;Address;”Parity Switch Incorrect Was=”;Itemp;” Needed=1"
1880              Save_flg=1
1890           ELSE
1900              IF Parity$<>”NONE” AND NOT Itemp THEN
1910                 OUTPUT @Ics;”SYSTEM:COMM:SERIAL:PARITY:CHECK ON”
1920                 PRINT “ICS4896 @”;Address;”Parity Switch Incorrect Was=”;Itemp;” Needed=0"
1930                 Save_flg=1
1940              END IF
1950           END IF
1960        CASE ELSE
1970           PRINT “ICS4896 @”;Address;”Asked for Invailed Parity of”;Parity$
1980        END SELECT
1990        SELECT Bits
2000        CASE 7,8
2010           OUTPUT @Ics;”SYSTEM:COMM:SERIAL:BITS?”
2020           ENTER @Ics;Itemp
2030           IF Itemp<>Bits THEN
2040              OUTPUT @Ics USING “K,X,K”;”SYSTEM:COMM:SERIAL:BITS”,Bits
2050              PRINT “ICS4896 @”;Address;”Baud Incorrect Was=”;Itemp;” Needed=”;Bits
2060              Save_flg=1
2070           END IF
2080        CASE ELSE
2090           PRINT “ICS4896 @”;Address;”Asked for Invailed # of Bits”;Bits
2100        END SELECT
```

Figure 5 Example 4896 Program for Channel 3 - Continued

```
2110    SELECT Sbits
2120    CASE 1,2
2130      OUTPUT @Ics;"SYSTEM:COMM:SERIAL:SBITS?"
2140      ENTER @Ics;Itemp
2150      IF Itemp<>Sbits THEN
2160        OUTPUT @Ics USING "K,X,K";"SYSTEM:COMM:SERIAL:SBITS";Sbits
2170        PRINT "ICS4896 @";Address;"Baud Incorrect Was=";Itemp;" Needed=";Sbits
2180        Save_flg=1
2190      END IF
2200    CASE ELSE
2210      PRINT "ICS4896 @";Address;"Asked for Invailed # of stop bits";Sbits
2220    END SELECT
2230    SELECT Pace$
2240    CASE "NONE","XON"
2250      OUTPUT @Ics;"SYSTEM:COMM:SERIAL:PACE?"
2260      ENTER @Ics;Temp$
2270      IF Temp$<>Pace$ THEN
2280        OUTPUT @Ics USING "K,X,K";"SYSTEM:COMM:SERIAL:PACE";Pace$
2290        PRINT "ICS4896 @";Address;"Baud Incorrect Was=";Temp$;" Needed=";Pace$
2300        Save_flg=1
2310      END IF
2320    CASE ELSE
2330      PRINT "ICS4896 @";Address;"Asked for Invailed Pace of ";Pace$
2340    END SELECT
2350    SELECT Eom
2360    CASE 0 TO 255
2370      OUTPUT @Ics;"SYSTEM:COMM:SERIAL:EOM?"
2380      ENTER @Ics;Itemp
2390      IF Itemp<>Eom THEN
2400        OUTPUT @Ics USING "K,X,K";"SYSTEM:COMM:SERIAL:EOM";Eom
2410        PRINT "ICS4896 @";Address;"EOM was incorrect was=";Itemp;" Needed=";Eom
2420        Save_flg=1
2430      END IF
2440    CASE -1
2450      !Do nothing switch
2460    CASE ELSE
2470      PRINT "ICS4896 @";Address;"Asked for Invailed EOM of ";VAL$(Eom)
2480    END SELECT
2490    SELECT Add_char
2500    CASE -1                 !OFF
2510      OUTPUT @Ics;"SYSTEM:COMM:SERIAL:ADD:ENABLE?"
2520      ENTER @Ics;Temp
2530      IF Temp THEN
2540        OUTPUT @Ics;"SYSTEM:COMM:SERIAL:ADD:ENABLE OFF"
2550        PRINT "ICS4896 @";Address;"ADD ENABLE was incorrect was=ON Needed=OFF"
2560        Save_flg=1
2570      END IF
2580    CASE 0-255              !ON
2590      OUTPUT @Ics;"SYSTEM:COMM:SERIAL:ADD:ENABLE?"
2600      ENTER @Ics;Temp
2610      IF Temp THEN
2620        OUTPUT @Ics;"SYSTEM:COMM:SERIAL:ADD:ENABLE ON"
2630        PRINT "ICS4896 @";Address;" ADD ENABLE was incorrect was=OFF Needed=ON"
2640        Save_flg=1
2650      END IF
2660      OUTPUT @Ins;"SYSTEM:COMM:SERIAL:ADD:CHARACTER?"
2670      ENTER @Ics;Itemp
2680      IF Itemp<>Add_char THEN
2690        OUTPUT @Ics USING "K,X,K";"SYSTEM:COMM:SERIAL:ADD:CHAR";Add_char
2700        PRINT "ICS4896 @";Address;" ADD CHAR was incorrect was=";Itemp;" needed=";Add_char
2710        Save_flg=1
2720      END IF
2730    CASE ELSE
2740      PRINT "ICS4896 @";Address;"Asked for Invailed ADD CHARACTER of ";VAL$(Add_char)
2750    END SELECT
2760    SELECT Eoi
2770    CASE 0,1
2780      OUTPUT @Ics;"SYSTEM:COMM:SERIAL:EOI?"
2790      ENTER @Ics;Temp
2800      IF Temp<>Eoi THEN
2810        OUTPUT @Ics USING "K,X,K";"SYSTEM:COMM:SERIAL:EOI";Eoi
2820        PRINT "ICS4896 @";Address;" EOI was incorrect was=";Temp;" needed=";Eoi
2830        Save_flg=1
```

Figure 5 Example 4896 Program for Channel 3 - Continued

```
2840          END IF
2850        CASE ELSE
2860          PRINT “ICS4896 @”;Address;”Asked for Invailed EOI of “;Eoi$
2870        END SELECT
2880        SELECT Echo
2890        CASE 0,1
2900          OUTPUT @Ics;”SYSTEM:COMM:SERIAL:ECHO?”
2910          ENTER @Ics;Temp
2920          IF Temp<>Echo THEN
2930            OUTPUT @Ics USING “K,X,K”;”SYSTEM:COMM:SERIAL:ECHO”;Echo
2940            PRINT “ICS4896 @”;Address;” ECHO was incorrect was=”;Temp;” needed=”;Echo
2950            Save_flg=1
2960          END IF
2970        CASE ELSE
2980          PRINT “ICS4896 @”;Address;”Asked for Invailed ECHO of “;Echo$
2990        END SELECT
3000        SELECT Loopback
3010        CASE 0,1
3020          OUTPUT @Ics;”SYSTEM:COMM:SERIAL:LOOPBACK?”
3030          ENTER @Ics;Temp
3040          IF Temp<>Loopback THEN
3050            OUTPUT @Ics USING “K,X,K”;”SYSTEM:COMM:SERIAL:LOOPBACK”;Loopback
3060            PRINT “ICS4896 @”;Address;” LOOPBACK was incorrect was=”;Temp;” needed=”;Loopback
3070            Save_flg=1
3080          END IF
3090        CASE ELSE
3100          PRINT “ICS4896 @”;Address;”Asked for Invailed LOOPBACK of “;Loopback$
3110        END SELECT
3120        SELECT Xmit
3130        CASE 0,1
3140          OUTPUT @Ics;”SYSTEM:COMM:SERIAL:TRANSMIT?”
3150          ENTER @Ics;Temp
3160          IF Temp<>Xmit THEN
3170            OUTPUT @Ics USING “K,X,K”;”SYSTEM:COMM:SERIAL:TRANSMIT”;Xmit
3180            PRINT “ICS4896 @”;Address;” TRANSMIT was incorrect was=”;Temp;” needed=”;Xmit
3190            Save_flg=1
3200          END IF
3210        CASE ELSE
3220          PRINT “ICS4896 @”;Address;”Asked for Invailed TRANSMIT of “;Xmit$
3230        END SELECT
3240        SELECT Extclk
3250        CASE 0,1
3260          OUTPUT @Ics;”SYSTEM:COMM:SERIAL:EXTCLK?”
3270          ENTER @Ics;Temp
3280          IF Temp<>Extclk THEN
3290            OUTPUT @Ics USING “K,X,K”;”SYSTEM:COMM:SERIAL:EXTCLK”;Extclk
3300            PRINT “ICS4896 @”;Address;” EXTCLK was incorrect was=”;Temp;” needed=”;Extclk
3310            Save_flg=1
3320          END IF
3330        CASE ELSE
3340          PRINT “ICS4896 @”;Address;”Asked for Invalid External Clock of “;Extclk$
3350        END SELECT
3360        SUBEXIT
3370!
3380 Timeout:!
3390!
3400        DISP “Timeout in SUB ics4896_232init @”;Address
3410        BEEP
3420        PAUSE
3430      SUBEND
3440!*****************************************************************************
```

Figure 5 Example 4896 Program for Channel 3 - Continued

ACKNOWLEDGMENT

The author is indebted to Gerry Huck at Precision Cast Parts Inc. for his program examples and help in preparing this Application Bulletin.