Agilent ESG A/B Security Features

NOTE Agilent ESG A/B signal generators were primarily intended for use in the commercial test and measurement market. Consequently, these instruments were not designed to conveniently support the data security measures required for the Department of Defense (DoD). This document is Agilent's best effort at providing procedures to protect the proprietary data of DoD customers choosing to use these products in a secure environment.

This document provides information on how to protect classified proprietary data stored in the following Agilent signal generators:

ESG A	ESG B	ESG B
E4400A	E4400B	E4430B
E4420A	E4420B	E4431B
E4421A	E4421B	E4432B
E4422A	E4422B	E4433B
E4430A	E4423B	E4434B
E4431A	E4424B	E4435B
E4432A	E4425B	E4436B
E4433A	E4426B	E4437B

Agilent Technologies

Part Number: E4400-90631

Printed February 2008

Instrument Memory Types

The ESG A/B comprise several memory types, each used for storing a specific type of data. The following tables describe the memory types for each board in the instrument. A "Yes" in the "Writable During Normal Operation?" column indicates that sensitive user data can reside in that memory type. Refer to the footnotes in the "Purpose/Contents" column for information on removing sensitive user information.

Table 1 A14 CPU/Motherboard Memory

Memory Type/Size	Writable During Normal Operation?	Data Retained When Powered Off?	Purpose/Contents	Data Input Method
EPROM (64 KB)	No	Yes	main firmware image	factory installed or firmware upgrade
IC ROM DS (256 KB)	No	Yes	CPU bootup program and firmware loader/updater	factory programmed
IC ROM F (1 MB)	No	Yes	factory calibration/ configuration data	factory or service only
Battery-backed SRAM (544 KB)	Yes	Yes	stores the following user data: • directory information, such as file names, for the main file system[1, 2] • directory information, such as file names, for the volatile ARB file system[1, 2] • directory information, such as file names, for the non-volatile ARB file system[1, 2] • user data, such as table editor information, stored in the main file system[3] • temporary storage of some cached user data, such as the displayed frequency[3]	firmware operations
Microprocessor Cache, SRAM (3 kB)	Yes	No	CPU data and instruction cache. Can contain fragments of user data.	memory is managed by CPU, not user.

1. Refer to "Zero Overwriting Directory Information" on page 6.
2. Refer to "Sanitizing Directory Information" on page 6.
3. Refer to "Sanitizing the Main File System Memory" on page 7.

Table 2 A5 Dual Arbitrary Waveform Generator Board Memory

Option	Memory Type/Size	Writable During Normal Operation?	Data Retained When Powered Off?	Purpose/Contents	Data Input Method
UND	SRAM (4 MB)	Yes	No	I and Q waveform data[1]	normal user operation
	SRAM (512 KB)	Yes	No	sequencer data	normal user operation
	Flash (1 MB)	No	Yes	firmware image	firmware upgrade
	EEPROM (512 B)	No	Yes	calibration data and board header	firmware upgrade
	SRAM (512 KB)	Yes	No	operating memory for the dual arbitrary waveform generator	During normal operation, some user information, such as payload data, can remain in the memory.
	Flash (4 MB)	Yes	Yes	I and Q waveform data[2]	normal user operation

1. Refer to "Sanitizing Volatile ARB Memory" on page 7.
2. Refer to "Sanitizing Non-Volatile ARB Memory" on page 8.

Table 3 A6 Bit Error Rate Test Board Memory

Option	Memory Type/Size	Writable During Normal Operation?	Data Retained When Powered Off?	Purpose/Contents	Data Input Method
UN7	SRAM (512 KB)	Yes	No	CPU operating memory	memory is managed by CPU, not user
	Flash (2 MB)	No	Yes	CPU program and FPGA configuration	firmware upgrade
	EEPROM (512 B)	No	Yes	calibration data and board header	firmware upgrade
300	SRAM (1 MB)	Yes	No	CPU operating memory	memory is managed by CPU, not user
	Flash (512 KB)	No	Yes	CPU program	firmware upgrade
	EEPROM (512 B)	No	Yes	calibration data and board header	firmware upgrade
	EEPROM (3.2 MB)	No	Yes	FPGA configuration	firmware upgrade

Table 4 A7 Baseband Generator Board Memory

Option	Memory Type/Size	Writable During Normal Operation?	Data Retained When Powered Off?	Purpose/Contents	Data Input Method
UN3/4	SRAM (64 KB)	Yes	No	burst envelope data	normal user operation
	EEPROM (256 KB)	No	Yes	calibration data and board header	firmware upgrade
UN8/9	SRAM (64 KB)	Yes	No	burst envelope data	normal user operation
	SRAM (512 KB)	Yes	No	PRAM data (framing and payload data information for real-time formats) *This part not loaded onto board.*	normal user operation
	EEPROM (9 KB)	No	Yes	firmware image	firmware upgrade
	EEPROM (512 B)	No	Yes	calibration data and board header	firmware upgrade

Table 5 A8 Data Generator Board Memory

Option	Memory Type/Size	Writable During Normal Operation?	Data Retained When Powered Off?	Purpose/Contents	Data Input Method
UN3 (1 MSa)	SRAM (1024 KB)	Yes	No	PRAM data (framing and payload data information for real-time formats)[1]	normal user operation
	EEPROM (256 KB)	No	Yes	calibration data and board header	firmware upgrade
UN4 (8 MSa)	SRAM (8192 KB)	Yes	No	PRAM data (framing and payload data information for real-time formats)[1]	normal user operation
	EEPROM (256 KB)	No	Yes	calibration data and board header	firmware upgrade
UN8 (1 MSa)	SRAM (1024 KB)	Yes	No	PRAM data (framing and payload data information for real-time formats)[1]	normal user operation
	DRAM (8 MB)	Yes	No	CPU operating memory	memory is managed by CPU, not user.
	SRAM (128 KB)	Yes	No	I/Q data generation	memory is managed by CPU, not user.
	Flash (4 MB)	No	Yes	CPU program and FPGA configuration	firmware upgrade
	Flash (128 KB)	No	Yes	CPU boot ROM	firmware upgrade
	EEPROM (512 B)	No	Yes	calibration data and board header	firmware upgrade

Table 5 A8 Data Generator Board Memory

Option	Memory Type/Size	Writable During Normal Operation?	Data Retained When Powered Off?	Purpose/Contents	Data Input Method
UN9 (8 MSa)	SRAM (8192 KB)	Yes	No	PRAM data (framing and payload data information for real-time formats)[1]	normal user operation
	DRAM (8 MB)	Yes	No	CPU operating memory	memory is managed by CPU, not user.
	SRAM (128 KB)	Yes	No	I/Q data generation	memory is managed by CPU, not user.
	Flash (4 MB)	No	Yes	CPU program and FPGA configuration	firmware upgrade
	Flash (128 KB)	No	Yes	CPU boot ROM	firmware upgrade
	EEPROM (512 B)	No	Yes	calibration data and board header	firmware upgrade

1. Refer to "Sanitizing PRAM Memory" on page 8.

Table 6 Other Boards

Board/Option	Memory Type/Size	Writable During Normal Operation?	Data Retained When Powered Off?	Purpose/Contents	Data Input Method
A9 Output	EEPROM (512 B)	No	Yes	calibration data and board header	firmware upgrade
A11 Reference					
A12 Synthesizer/Doubler					
A20 Down Converter (Option 300)					
A21 Demodulator (Option 300)					
A22 YIG Driver					
A24 Frac-N/Divider					
A23 Sampler					
A1 Front Panel Keyboard	No memory on these boards.				
A15 Daughterboard					
A17 Rear Panel Interface					
A18 BER Rear Panel Interface (Option UN7/300)					

User Data Removal Methods

This section describes the methods for removing various user data types stored in the instrument.

Zero Overwriting Directory Information

Use this procedure to zero overwrite all directory information, such as file names. This procedure *does not* affect the stored user data associated with the file name.

1. Turn off the signal generator.
2. Press and hold **Preset** while turning on the signal generator. Continue holding **Preset** until the ESG fail-safe recovery sequence screen is displayed.
3. Press the **Yes** softkey to erase (zero overwrite) all of the signal generator's directory information.
4. Cycle the signal generator power to reinitialize factory defaults and reset factory-installed options.

NOTE After completing this procedure, an I/Q calibration must be performed prior to using the signal generator as a digital modulation source. Likewise, a DCFM/DCΦM calibration must be performed prior to using the signal generator as an FM/ΦM source. Refer to the *User's Guide* for more information.

Sanitization Procedures

This section describes sanitization procedures for the various user memory locations in the signal generator. You must use SCPI commands to perform these procedures. Alternatively, you can use the "Example Sanitization Utility" on page 9, which randomly overwrites memory a specified number of times.

Sanitizing Directory Information

Use this procedure to sanitize all directory information, such as file names. This procedure *does not* affect the stored user data associated with the directory information. You must use remote SCPI commands to perform this procedure. For more information, refer to the *Programming Guide* and the "Example Sanitization Utility" on page 9 of this document.

1. Delete all files (:MEM:DEL:ALL).

NOTE All user files are deleted with the possible exception of a 16-byte file named SAV_RCL_CONFIG@STATE. This is normal, as this file is used internally to the operation of the signal generator and contains no user information.

2. Write as many small files as the system allows with file name patterns of 23 characters. This fills the directory table with random names, but does not fill all user memory
3. For increased security, repeat steps 1 and 2 as many times as you wish, changing the file name patterns each time.

NOTE Another method for sanitizing the directory information is to open the box and remove the battery from the A14 CPU/Mother board.

Sanitizing the Main File System Memory

Use this procedure to sanitize all user data in the main file system memory. This procedure *does not* affect ARB, NVARB, or PRAM data. You must use remote SCPI commands to perform this procedure. For more information, refer to the *Programming Guide*.

1. Delete all files (:MEM:DEL:ALL).
2. Write a binary file that fills all of the main file system memory.

NOTE A.xx.xx firmware will show up to 26 bytes free. B.xx.xx firmware will show up to 4 bytes free. These free bytes do not contain any user-accessible information.

---

3. For increased security, repeat steps 1 and 2 as many times as you wish, changing the binary file patterns each time.

NOTE Another method for sanitizing the main file system is to open the box and remove the battery from the A14 CPU/Mother board.

Sanitizing Volatile ARB Memory

Use this procedure to sanitize all user data in the ARB memory. This procedure *does not* affect NVARB, PRAM, or main file system data. You must use remote SCPI commands to perform this procedure. For more information, refer to the *Programming Guide*.

1. Delete all files (:MEM:DEL:ARB).
2. Write a binary file that fills all of the ARB memory.
3. For increased security, repeat steps 1 and 2 as many times as you wish, changing the binary file patterns each time.

Sanitizing Non-Volatile ARB Memory

Use this procedure to sanitize all user data in the NVARB memory. This procedure *does not* affect ARB, PRAM, or main file system data. You must use remote SCPI commands to perform this procedure. For more information, refer to the *Programming Guide*.

1. Delete all files (:MMEM:DEL:NVARB).
2. Write a binary file that fills all of the NVARB memory.
3. For increased security, repeat steps 1 and 2 as many times as you wish, changing the binary file patterns each time.

Sanitizing PRAM Memory

Use this procedure to sanitize all user data in the PRAM memory. This procedure *does not* affect ARB, NVARB or main file system data. You must use remote SCPI commands to perform this procedure. For more information, refer to the *Programming Guide*.

1. Delete all files (:MEM:DEL:NVARB).
2. Write a binary file that fills all of the PRAM memory.
3. For increased security, repeat steps 1 and 2 as many times as you wish, changing the binary file patterns each time.

If Your Instrument is Not Functioning

If the signal generator is not functioning and you are unable to erase the memory, using the procedures in this document, you may physically remove the boards that contain sensitive user data (A14 CPU/mother board, A8 Data Generator board, and A5 Dual Arbitrary Waveform Generator board) and do one of the following options:

- Discard the boards and send the instrument to a repair facility. New boards will be installed and the instrument will be repaired and calibrated. If the instrument is still under warranty, you will not be charged for the new boards.
- If you have another working instrument, install the boards into that instrument and erase the memory. Then reinstall the boards back into the non-working instrument and send it to a repair facility for repair and calibration. If you discover that one or more of the boards cause the working instrument to fail, discard the non-working boards and note that they caused the instrument failure on the repair order. If the instrument is still under warranty, you will not be charged for the new boards.

For procedures on removing and replacing boards, refer to the *Service Guide*.

Example Sanitization Utility

The following program, created in Rocky Mountain Basic (RMB), can be used to sanitize the following user memory locations in your signal generator:

- directory information
- main file system memory
- volatile ARB memory
- non-volatile ARB memory
- PRAM memory

Copy and paste the programming code from this document into a text editor. After pasting the code into the text editor, you may need to manually delete the copied header and footer text from each page (approximately every 25 lines) before saving the program. This example program can also be used to convert into a programming language of your choice. Before using this program, do the following:

- Verify the signal generator firmware is revision A.01.20 or later.
- If your signal generator is warming up and displays the OVEN COLD annunciator, wait approximately five minutes and then clear all errors by pressing **Utility** > **Error Info** > **Clear Error Queue (s)**.
- Set the signal generator address and number of overwrites you want in lines 110 and 120 of the code.

```
10 !
20 ! Sanitization program version 1.01 dated Feb 12, 2008
30 !
40    DIM Err$[100],Optlist$[100]
50 !
60    PRINT "This program overwrites all user memory in an"
70    PRINT "Agilent E44xxA or E44xxB Signal Generator"
80    PRINT "following Document E4400-90631 dated February 2008"
90    PRINT
100!
110   Address=719       ! Set this to your Signal Generator address
120   Num_overwrites=3  ! Set this to the number of overwrites you require
130   Verbose=0         ! Set this to 1 to see the program's progress
140 !
150   PRINT "Preparing to overwrite data in instrument at GPIB address";Address
160   PRINT
170   PRINT "Data will be overwritten";Num_overwrites;"times"
180   PRINT
190   PRINT "You should ignore error messages that appear on the instrument while the program runs"
200   PRINT
```

```
210   PRINT "Enter 'y' to continue, 'n' to abort"
220   INPUT A$
230   IF A$<>"y" THEN
240     PRINT "Stopped.  No data was overwritten."
250     STOP
260   END IF
270 !
280 !
290   Gpib=INT(Address/100)
300   ASSIGN @Inst TO Address
310   OUTPUT @Inst;"*RST"
320   WAIT 3
330   No_err$="+0,""No error"""
340   Clear_error_buf(@Inst,No_err$)
350   OFF TIMEOUT
360 !
370   OUTPUT @Inst;":DIAG:INFO:OPT?"
380   ENTER @Inst;Optlist$
390   Has_opt_und=(POS(Optlist$,"UND")>0)
400   Has_opt_un3=(POS(Optlist$,"UN3")>0)
410   Has_opt_un4=(POS(Optlist$,"UN4")>0)
420   Has_opt_un8=(POS(Optlist$,"UN8")>0)
430   Has_opt_un9=(POS(Optlist$,"UN9")>0)
440   Has_opt_1eh=(POS(Optlist$,"1EH")>0)
450   IF Has_opt_un4 OR Has_opt_un9 THEN
460     Pram_size=2^23 ! This is 8 MB
470   ELSE
480     IF Has_opt_un3 OR Has_opt_1eh OR Has_opt_un8 THEN
490       Pram_size=2^20 ! This is 1 MB
500     ELSE
510       Pram_size=0
520     END IF
530   END IF
540 !
550   RANDOMIZE
560   FOR Iteration=1 TO Num_overwrites
570     PRINT
580     PRINT "Starting iteration";Iteration;"of";Num_overwrites
590 !
600 !
610     PRINT "Sanitizing Directory Information"
620     OUTPUT @Inst;":MEM:DEL:ALL"
630     Clear_error_buf(@Inst,No_err$)
640     ON TIMEOUT Gpib,5 GOTO Sanitize_dir
650 Sanitize_dir:     !
660     Clear_error_buf(@Inst,No_err$)
```

```
670     Err$=No_err$
680     WHILE Err$=No_err$ ! Write small files until out of memory
690       PRINT ".";
700       Wrt_random_file(@Inst,"",23,1)
710       OUTPUT @Inst;":SYST:ERR?"
720       ENTER @Inst;Err$
730     END WHILE
740     PRINT
750     Clear_error_buf(@Inst,No_err$)
760     IF Verbose THEN
770       PRINT "Verify memory catalog contains many random filenames."
780       PRINT "Key sequence: Local, Utility, Memory Catalog, Catalog Type, All"
790       PRINT "Then press Enter to continue"
800       INPUT A$
810     END IF
820     OFF TIMEOUT
830 !
840 !
850     PRINT "Sanitizing the Main File System Memory"
860     OUTPUT @Inst;":MEM:DEL:ALL"
870     Clear_error_buf(@Inst,No_err$)
880     Max_size=16384
890     ON TIMEOUT Gpib,10 GOTO Sanitize_main
900 Sanitize_main:    !
910     WHILE Max_size>=1
920       Clear_error_buf(@Inst,No_err$)
930       Err$=No_err$
940       WHILE Err$=No_err$! Write large files until out of memory
950         PRINT ".";
960         Wrt_random_file(@Inst,"",8,Max_size)
970         OUTPUT @Inst;":SYST:ERR?"
980         ENTER @Inst;Err$
990       END WHILE
1000      Clear_error_buf(@Inst,No_err$)
1010      Max_size=INT(Max_size/2)
1020    END WHILE
1030    PRINT
1040    IF Verbose THEN
1050      PRINT "Verify memory catalog is full of large binary files."
1060      PRINT "Verify 4 or fewer free bytes remain in memory for E44xxB"
1070      PRINT "Verify 26 or fewer free bytes remain in memory for E44xxA"
1080      PRINT "Key sequence: Local, Utility, Memory Catalog, Catalog Type, All"
1090      PRINT "Then press Enter to continue"
1100      INPUT A$
1110    END IF
1120    OFF TIMEOUT
```

```
1130!
1140!
1150    IF Has_opt_und THEN
1160!
1170      PRINT "Sanitizing Volatile ARB Memory"
1180      OUTPUT @Inst;":MEM:DEL:ALL"
1190      OUTPUT @Inst;":MMEM:DEL:ARB"
1200      OUTPUT @Inst;":MMEM:DEL:NVARB"
1210      Clear_error_buf(@Inst,No_err$)
1220      ON TIMEOUT Gpib,10 GOTO Sanitize_arb
1230 Sanitize_arb:   !
1240      Clear_error_buf(@Inst,No_err$)
1250      Max_size=16384
1260      Err$=No_err$
1270      WHILE Err$=No_err$! Write large files until out of memory
1280        PRINT ".";
1290        Wrt_random_file(@Inst,"ARBI",10,Max_size)
1300        OUTPUT @Inst;":SYST:ERR?"
1310        ENTER @Inst;Err$
1320      END WHILE
1330      PRINT
1340      Clear_error_buf(@Inst,No_err$)
1350      IF Verbose THEN
1360        PRINT "Verify ARB memory catalog is full of large ARB files."
1370        PRINT "Key sequence: Local, Utility, Memory Catalog,"
1380        PRINT "              Catalog Type, ARB Catalog Types, ARB"
1390        PRINT "Then press Enter to continue"
1400        INPUT A$
1410      END IF
1420      OFF TIMEOUT
1430!
1440!
1450      PRINT "Sanitizing Non-Volatile ARB Memory"
1460      OUTPUT @Inst;":MEM:DEL:ALL"
1470      OUTPUT @Inst;":MMEM:DEL:ARB"
1480      OUTPUT @Inst;":MMEM:DEL:NVARB"
1490      Clear_error_buf(@Inst,No_err$)
1500      ON TIMEOUT Gpib,10 GOTO Sanitize_nvarb
1510 Sanitize_nvarb:   !
1520      Clear_error_buf(@Inst,No_err$)
1530      Max_size=16384
1540      Err$=No_err$
1550      WHILE Err$=No_err$! Write large files until out of memory
1560        PRINT ".";
1570        Wrt_random_file(@Inst,"NVARBI",10,Max_size)
1580        OUTPUT @Inst;":SYST:ERR?"
```

```
1590        ENTER @Inst;Err$
1600      END WHILE
1610      PRINT
1620      Clear_error_buf(@Inst,No_err$)
1630      IF Verbose THEN
1640        PRINT "Verify NVARB memory catalog is full of large NVARB files"
1650        PRINT "Key sequence: Local, Utility, Memory Catalog,"
1660        PRINT "              Catalog Type, ARB Catalog Types, NVARB"
1670        PRINT "Then press Enter to continue"
1680        INPUT A$
1690      END IF
1700      OFF TIMEOUT
1710 !
1720    END IF ! Has_opt_und
1730 !
1740 !
1750    IF Pram_size>0 THEN
1760      PRINT "Sanitizing PRAM Memory"
1770      OUTPUT @Inst;":MEM:DEL:ALL"
1780      IF Has_opt_und THEN
1790        OUTPUT @Inst;":MMEM:DEL:ARB"
1800        OUTPUT @Inst;":MMEM:DEL:NVARB"
1810      END IF
1820      Clear_error_buf(@Inst,No_err$)
1830      Wrt_random_pram(@Inst,Pram_size)
1840      IF Verbose THEN
1850        PRINT "Sanitized";Pram_size;"bytes of PRAM"
1860        PRINT "Verify no errors occurred while writing to PRAM"
1870        PRINT "Key sequence: Local, Utility, Error Info"
1880        PRINT "Then press Enter to continue"
1890        INPUT A$
1900      END IF
1910      OFF TIMEOUT
1920    END IF ! pram_size>0
1930 !
1940     PRINT "Completed iteration";Iteration;"of";Num_overwrites
1950  NEXT Iteration
1960 !
1970  PRINT "Sanitization complete"
1980  PRINT "Done"
1990  END
2000 !
2010 Wrt_random_file:SUB Wrt_random_file(@Inst,Dir$,Name_length,File_length)
2020 ! This subroutine writes a random file with a random name
2030    DIM Filename$[100],Msus$[200]
2040    FOR I=1 TO Name_length
```

```
2050      Char=INT(RND*36)
2060      IF Char<10 THEN
2070        Char$=CHR$(Char+NUM("0"))
2080      ELSE
2090        Char$=CHR$(Char-10+NUM("A"))
2100      END IF
2110      Filename$[I;1]=Char$
2120    NEXT I
2130    IF Dir$="" THEN
2140      Msus$=Filename$
2150    ELSE
2160      Msus$=Dir$&":"&Filename$
2170    END IF
2180    ALLOCATE Buffer$[File_length]
2190    FOR I=1 TO File_length
2200      Char=INT(RND*256)
2210      Buffer$[I;1]=CHR$(Char)
2220    NEXT I
2230    Nbytes$=VAL$(File_length)
2240    Ndigits$=VAL$(LEN(Nbytes$))
2250    OUTPUT @Inst USING "#,K";":MEM:DATA """&Msus$&""", #"
2260    OUTPUT @Inst USING "#,K";Ndigits$
2270    OUTPUT @Inst USING "#,K";Nbytes$
2280    OUTPUT @Inst USING "K";Buffer$
2290    DEALLOCATE Buffer$
2300  SUBEND
2310 !
2320 Wrt_random_pram:SUB Wrt_random_pram(@Inst,Pram_size)
2330 ! This subroutine writes random pram data
2340    DIM Nbytes$[20]
2350    Max_buffer=16384
2360    !
2370    ! convert the Pram_size to an integer in a string
2380    ! even when Pram_size is greater than the max integer in RMB
2390    OUTPUT Nbytes$ USING "#,10D";Pram_size
2400    WHILE Nbytes$[1;1]=" "
2410      Nbytes$=Nbytes$[2]  ! remove leading blanks
2420    END WHILE
2430    !
2440    Ndigits$=VAL$(LEN(Nbytes$))
2450    OUTPUT @Inst USING "#,K";":MEM:DATA:PRAM:BLOCK #"
2460    OUTPUT @Inst USING "#,K";Ndigits$
2470    OUTPUT @Inst USING "#,K";Nbytes$
2480    Bytes_left=Pram_size
2490    WHILE (Bytes_left>0)
2500      Bytes_to_write=MIN(Max_buffer,Bytes_left)
```

```
2510      ALLOCATE Buffer$[Bytes_to_write]
2520      FOR I=1 TO Bytes_to_write
2530        Char=INT(RND*256)
2540        Buffer$[I;1]=CHR$(Char)
2550      NEXT I
2560      OUTPUT @Inst USING "#,K";Buffer$
2570      DEALLOCATE Buffer$
2580      Bytes_left=Bytes_left-Bytes_to_write
2590      PRINT ".";
2600    END WHILE
2610    PRINT ""
2620    OUTPUT @Inst USING "K";""
2630  SUBEND
2640 !
2650 Clear_error_buf:SUB Clear_error_buf(@Dut,No_err$)
2660 ! This subroutine clears the error buffer
2670    DIM Err$[100]
2680    Err$=""
2690    WHILE Err$<>No_err$
2700      OUTPUT @Dut;"SYST:ERR?"
2710      ENTER @Dut;Err$
2720    END WHILE
2730  SUBEND
```