



.REM E

IDENTIFICATION  
-----

PRODUCT CODE: AC-FF61A-MC

PRODUCT NAME: CZVSWAO

PRODUCT DATE: 4-JUL-85

MAINTAINER: CSS ENGINEERING

AUTHOR: JOHN BOWSKILL

Copyright (c) 1985 by  
Digital Equipment Corporation, Maynard, Massachusetts  
All Rights Reserved

This software is furnished under a license and may be used and copied only in accordance with the terms of such license and with the inclusion of the above copyright notice. This software or any other copies thereof may not be provided or otherwise made available to any other person. No title to and ownership of the software is hereby transferred.

The information in this software is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation.

Digital assumes no responsibility for the use or reliability of its software on equipment that is not supplied by Digital.

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DEC	PDP	UNIBUS	MASSBUS
DECUS	DECTAPE	VAX	

-----  
} } } } } } } } } }  
} d } i } g } i } t } e } l }  
} } } } } } } } } }  
-----

E

.REM &

## TABLE OF CONTENTS

1.0	GENERAL INFORMATION
1.1	PROGRAM ABSTRACT
1.2	SYSTEM & HARDWARE REQUIREMENTS
2.0	OPERATING INSTRUCTIONS
2.1	COMMANDS
2.2	SWITCHES
2.3	FLAGS
2.4	HARDWARE QUESTIONS
2.7	EXTENDED P-TABLE DIALOGUE
2.8	QUICK STARTUP PROCEDURE
3.0	ERROR INFORMATION
4.0	PERFORMANCE AND PROGRESS REPORTS
5.0	DEVICE INFORMATION TABLES
6.0	TEST SUMMARIES

### 1.0 GENERAL INFORMATION

#### 1.1 PROGRAM ABSTRACT

THIS DIAGNOSTIC VERIFIES THAT DEVICES UNDER TEST ARE VSV21 DEVICES AND CAUSES ALL VSV21 ON-BOARD MICRODIAGNOSTICS TO BE EXECUTED. IT PROVIDES COMPLETE FUNCTION LEVEL COVERAGE AND FRU CALLOUT.

THIS DIAGNOSTIC HAS BEEN WRITTEN FOR USE WITH THE DIAGNOSTIC RUNTIME SERVICES SOFTWARE (SUPERVISOR). THESE SERVICES PROVIDE THE INTERFACE TO THE OPERATOR AND TO THE SOFTWARE ENVIRONMENT. THIS PROGRAM CAN BE USED WITH XXDP+, ACT, APT, SLIDE AND PAPER TAPE. FOR A COMPLETE DESCRIPTION OF THE RUNTIME SERVICES, REFER TO THE XXDP+ USER'S MANUAL. THERE IS A BRIEF DESCRIPTION OF THE RUNTIME SERVICES IN SECTION 2 OF THIS DOCUMENT.

#### 1.2 SYSTEM & HARDWARE REQUIREMENTS

THIS DIAGNOSTIC IS DESIGNED TO RUN ON A 11/23+ OR A 11/73 WITH A MINIMUM MEMORY OF 32 K BYTES.  
THE MODULE UNDER TEST IS A M7656.  
TESTS 4 , 13 , 14 , 16 AND 17 REQUIRE PART NUMBER VSV21-AJ WHICH CONSISTS OF THE FOLLOWING:  
1 OFF 12-15336-01 CONN, LOOPBACK EIA DATA 25-WAY  
2 OFF 70-20130-01 CONN, LOOPBACK EIA DATA 9-WAY  
1 OFF 70-20131-01 CONN, LOOPBACK KEYBOARD JACK

## 2.0 OPERATING INSTRUCTIONS

THIS SECTION CONTAINS A BRIEF DESCRIPTION OF THE RUNTIME SERVICES. FOR DETAILED INFORMATION, REFER TO THE XXDP+ USER'S MANUAL (CHQUS).

### 2.1 COMMANDS

THERE ARE ELEVEN LEGAL COMMANDS FOR THE DIAGNOSTIC RUNTIME SERVICES (SUPERVISOR). THIS SECTION LISTS THE COMMANDS AND GIVES A VERY BRIEF DESCRIPTION OF THEM. THE XXDP+ USER'S MANUAL HAS MORE DETAILS.

COMMAND	EFFECT
START	START THE DIAGNOSTIC FROM AN INITIAL STATE
RESTART	START THE DIAGNOSTIC WITHOUT INITIALIZING
CONTINUE	CONTINUE AT TEST THAT WAS INTERRUPTED (AFTER +C)
PROCEED	CONTINUE FROM AN ERROR HALT
EXIT	RETURN TO XXDP+ MONITOR (XXDP+ OPERATION ONLY!)
ADD	ACTIVATE A UNIT FOR TESTING (ALL UNITS ARE CONSIDERED TO BE ACTIVE AT START TIME)
DROP	DEACTIVATE A UNIT
PRINT	PRINT STATISTICAL INFORMATION (IF IMPLEMENTED BY THE DIAGNOSTIC - SECTION 4.0)
DISPLAY	TYPE A LIST OF ALL DEVICE INFORMATION
FLAGS	TYPE THE STATE OF ALL FLAGS (SEE SECTION 2.3)
ZFLAGS	CLEAR ALL FLAGS (SEE SECTION 2.3)

A COMMAND CAN BE RECOGNIZED BY THE FIRST THREE CHARACTERS. SO YOU MAY, FOR EXAMPLE, TYPE "STA" INSTEAD OF "START".

### 2.2 SWITCHES

THERE ARE SEVERAL SWITCHES WHICH ARE USED TO MODIFY SUPERVISOR OPERATION. THESE SWITCHES ARE APPENDED TO THE LEGAL COMMANDS. ALL OF THE LEGAL SWITCHES ARE TABULATED BELOW WITH A BRIEF DESCRIPTION OF EACH. IN THE DESCRIPTIONS BELOW, A DECIMAL NUMBER IS DESIGNATED BY "DDDD".

SWITCH	EFFECT
/TESTS:LIST	EXECUTE ONLY THOSE TESTS SPECIFIED IN THE LIST. LIST IS A STRING OF TEST NUMBERS, FOR EXAMPLE - /TESTS:1:5:7-10. THIS LIST WILL CAUSE TESTS 1,5,7,8,9,10 TO BE RUN. ALL OTHER TESTS WILL NOT BE RUN.
/PASS:DDDD	EXECUTE DDDDD PASSES (DDDD = 1 TO 64000)
/FLAGS:FLGS	SET SPECIFIED FLAGS. FLAGS ARE DESCRIBED IN SECTION 2.3.
/EOP:DDDD	REPORT END OF PASS MESSAGE AFTER EVERY DDDDD PASSES ONLY. (DDDD = 1 TO 64000)
/UNITS:LIST	TEST/ADD/DROP ONLY THOSE UNITS SPECIFIED IN THE LIST. LIST EXAMPLE - /UNITS:0:5:10-12 USE UNITS 0,5,10,11,12 (UNIT NUMBERS = 0-63)

EXAMPLE OF SWITCH USAGE:

START/TESTS:1-5/PASS:1000/EOP:100

THE EFFECT OF THIS COMMAND WILL BE: 1) TESTS 1 THROUGH 5 WILL BE EXECUTED, 2) ALL UNITS WILL TESTED 1000 TIMES AND 3) THE END OF PASS MESSAGES WILL BE PRINTED AFTER EACH 100 PASSES ONLY. A SWITCH CAN BE RECOGNIZED BY THE FIRST THREE CHARACTERS. YOU MAY, FOR EXAMPLE, TYPE "/TES:1-5" INSTEAD OF "/TESTS:1-5".

BELOW IS A TABLE THAT SPECIFIES WHICH SWITCHES CAN BE USED BY EACH COMMAND.

	TESTS	PASS	FLAGS	EOP	UNITS
START	X	X	X	X	X
RESTART	X	X	X	X	X
CONTINUE		X	X	X	
PROCEED			X		
DROP					X
ADD					X
PRINT					
DISPLAY					X
FLAGS					
ZFLAGS					
EXIT					

### 2.3 FLAGS

FLAGS ARE USED TO SET UP CERTAIN OPERATIONAL PARAMETERS SUCH AS LOOPING ON ERROR. ALL FLAGS ARE CLEARED AT STARTUP AND REMAIN CLEARED UNTIL EXPLICITLY SET USING THE FLAGS SWITCH. FLAGS ARE ALSO CLEARED AFTER A START COMMAND UNLESS SET USING THE FLAG SWITCH. THE ZFLAGS COMMAND MAY ALSO BE USED TO CLEAR ALL FLAGS. WITH THE EXCEPTION OF THE START AND ZFLAGS COMMANDS, NO COMMANDS AFFECT THE STATE OF THE FLAGS; THEY REMAIN SET OR CLEARED AS SPECIFIED BY THE LAST FLAG SWITCH.

FLAG	EFFECT
HOE	HALT ON ERROR - CONTROL IS RETURNED TO RUNTIME SERVICES COMMAND MODE
LOE	LOOP ON ERROR
IER*	INHIBIT ALL ERROR REPORTS
IBR*	INHIBIT ALL ERROR REPORTS EXCEPT FIRST LEVEL (FIRST LEVEL CONTAINS ERROR TYPE, NUMBER, PC, TEST AND UNIT)
IXR*	INHIBIT EXTENDED ERROR REPORTS (THOSE CALLED BY PRINTX MACRO'S)
PRI	DIRECT MESSAGES TO LINE PRINTER
PNT	PRINT TEST NUMBER AS TEST EXECUTES
BOE	"BELL" ON ERROR
UAM	UNATTENDED MODE (NO MANUAL INTERVENTION)
ISR	INHIBIT STATISTICAL REPORTS (DOES NOT APPLY TO DIAGNOSTICS WHICH DO NOT SUPPORT STATISTICAL REPORTING)
IDR	INHIBIT PROGRAM DROPPING OF UNITS
ADR	EXECUTE AUTODROP CODE

LOT LOOP ON TEST  
EVL EXECUTE EVALUATION (ON DIAGNOSTICS WHICH  
HAVE EVALUATION SUPPORT)

\*ERROR MESSAGES ARE DESCRIBED IN SECTION 3.1

SEE THE XXDP+ USER'S MANUAL FOR MORE DETAILS ON FLAGS. YOU MAY SPECIFY MORE THAN ONE FLAG WITH THE FLAG SWITCH. FOR EXAMPLE, TO CAUSE THE PROGRAM TO LOOP ON ERROR, INHIBIT ERROR REPORTS AND TYPE A "BELL" ON ERROR, YOU MAY USE THE FOLLOWING STRING:

/FLAGS:LOE:IER:BOE

#### 2.4 HARDWARE QUESTIONS

WHEN A DIAGNOSTIC IS STARTED, THE RUNTIME SERVICES WILL PROMPT THE USER FOR HARDWARE INFORMATION BY TYPING "CHANGE HW (L) ?" YOU MUST ANSWER "Y" AFTER A START COMMAND UNLESS THE HARDWARE INFORMATION HAS BEEN "PRELOADED" USING THE SETUP UTILITY (SEE CHAPTER 6 OF THE XXDP+ USER'S MANUAL) OR YOU WISH TO USE THE HARDCODED DEFAULTS ( SEE BELOW ) OR YOU HAVE JUST RUN THE DIAGNOSTICS AND WISH TO KEEP THE SAME HARDWARE INFORMATION. WHEN YOU ANSWER THIS QUESTION WITH A "Y", THE RUNTIME SERVICES WILL ASK FOR THE NUMBER OF UNITS (IN DECIMAL). YOU WILL THEN BE ASKED THE FOLLOWING QUESTIONS FOR EACH UNIT.

DEVICE ADDRESS (0) 172010 ?  
VECTOR ADDRESS (0) 320 ?

WHERE DEVICE AND VECTOR ADDRESS ARE DEFINED IN SECTION 5.0.

IN THE FOLLOWING EXAMPLE THE USER SELECTS 1 UNIT WITH 172020 FOR DEVICE ADDRESS INSTEAD OF THE DEFAULT 172010 AND 324 FOR VECTOR ADDRESS INSTEAD OF THE DEFAULT 320.

CHANGE HW (L) ? Y

# UNITS (D) ? 1

DEVICE ADDRESS (0) 172010 ? 172020  
VECTOR ADDRESS (0) 320 ? 324

THE HARDCODED DEFAULTS ARE FOR 4 UNITS WITH THE FOLLOWING ADDRESSES:

UNIT	DEVICE ADDRESS	VECTOR ADDRESS
#1	172010	320
#2	172020	324
#3	172030	330
#4	172040	334

#### 2.5 EXTENDED P-TABLE DIALOGUE

WHEN YOU ANSWER THE HARDWARE QUESTIONS, YOU ARE BUILDING ENTRIES IN A TABLE THAT DESCRIBES THE DEVICES UNDER TEST. THE SIMPLEST

WAY TO BUILD THIS TABLE IS TO ANSWER ALL QUESTIONS FOR EACH UNIT TO BE TESTED. IF YOU HAVE A MULTIPLEXED DEVICE SUCH AS A MASS STORAGE CONTROLLER WITH SEVERAL DRIVES OR A COMMUNICATION DEVICE WITH SEVERAL LINES, THIS BECOMES TEDIOUS SINCE MOST OF THE ANSWERS ARE REPETITIOUS.

TO ILLUSTRATE A MORE EFFICIENT METHOD, SUPPOSE YOU ARE TESTING A FICTIONAL DEVICE, THE XY11. SUPPOSE THIS DEVICE CONSISTS OF A CONTROL MODULE WITH EIGHT UNITS (SUB-DEVICES) ATTACHED TO IT. THESE UNITS ARE DESCRIBED BY THE OCTAL NUMBERS 0 THROUGH 7. THERE IS ONE HARDWARE PARAMETER THAT CAN VARY AMONG UNITS CALLED THE Q-FACTOR. THIS Q-FACTOR MAY BE 0 OR 1. BELOW IS A SIMPLE WAY TO BUILD A TABLE FOR ONE XY11 WITH EIGHT UNITS.

# UNITS (D) ? 8<CR>

UNIT 1

CSR ADDRESS (0) ? 160000<CR>

SUB-DEVICE # (0) ? 0<CR>

Q-FACTOR (0) 0 ? 1<CR>

UNIT 2

CSR ADDRESS (0) ? 160000<CR>

SUB-DEVICE # (0) ? 1<CR>

Q-FACTOR (0) 1 ? 0<CR>

UNIT 3

CSR ADDRESS (0) ? 160000<CR>

SUB-DEVICE # (0) ? 2<CR>

Q-FACTOR (0) 0 ? <CR>

UNIT 4

CSR ADDRESS (0) ? 160000<CR>

SUB-DEVICE # (0) ? 3<CR>

Q-FACTOR (0) 0 ? <CR>

UNIT 5

CSR ADDRESS (0) ? 160000<CR>

SUB-DEVICE # (0) ? 4<CR>

Q-FACTOR (0) 0 ? <CR>

UNIT 6

CSR ADDRESS (0) ? 160000<CR>

SUB-DEVICE # (0) ? 5<CR>

Q-FACTOR (0) 0 ? <CR>

UNIT 7

CSR ADDRESS (0) ? 160000<CR>

SUB-DEVICE # (0) ? 6<CR>

Q-FACTOR (0) 0 ? 1<CR>

UNIT 8

CSR ADDRESS (0) 160000<CR>

SUB-DEVICE # (0) ? 7<CR>

Q-FACTOR (0) 1 ? <CR>

NOTICE THAT THE DEFAULT VALUE FOR THE Q-FACTOR CHANGES WHEN A

NON-DEFAULT RESPONSE IS GIVEN. BE CAREFUL WHEN SPECIFYING MULTIPLE UNITS!

AS YOU CAN SEE FROM THE ABOVE EXAMPLE, THE HARDWARE PARAMETERS DO NOT VARY SIGNIFICANTLY FROM UNIT TO UNIT. THE PROCEDURE SHOWN IS NOT VERY EFFICIENT.

THE RUNTIME SERVICES CAN TAKE MULTIPLE UNIT SPECIFICATIONS HOWEVER. LET'S BUILD THE SAME TABLE USING THE MULTIPLE SPECIFICATION FEATURE.

```
# UNITS (D) ? 8<CR>

UNIT 1
CSR ADDRESS (0) ? 160000<CR>
SUB-DEVICE # (0) ? 0,1<CR>
Q-FACTOR (0) 0 ? 1,0<CR>

UNIT 3
CSR ADDRESS (0) ? 160000<CR>
SUB-DEVICE # (0) ? 2-5<CR>
Q-FACTOR (0) 0 ? 0<CR>

UNIT 7
CSR ADDRESS (0) ? 160000<CR>
SUB-DEVICE # (0) ? 6,7<CR>
Q-FACTOR (0) 0 ? 1<CR>
```

AS YOU CAN SEE IN THE ABOVE DIALOGUE, THE RUNTIME SERVICES WILL BUILD AS MANY ENTRIES AS IT CAN WITH THE INFORMATION GIVEN IN ANY ONE PASS THROUGH THE QUESTIONS. IN THE FIRST PASS, TWO ENTRIES ARE BUILT SINCE TWO SUB-DEVICES AND Q-FACTORS WERE SPECIFIED. THE SERVICES ASSUME THAT THE CSR ADDRESS IS 160000 FOR BOTH SINCE IT WAS SPECIFIED ONLY ONCE. IN THE SECOND PASS, FOUR ENTRIES WERE BUILT. THIS IS BECAUSE FOUR SUB-DEVICES WERE SPECIFIED. THE "-" CONSTRUCT TELLS THE RUNTIME SERVICES TO INCREMENT THE DATA FROM THE FIRST NUMBER TO THE SECOND. IN THIS CASE, SUB-DEVICES 2, 3, 4 AND 5 WERE SPECIFIED. (IF THE SUB-DEVICE WERE SPECIFIED BY ADDRESSES, THE INCREMENT WOULD BE BY 2 SINCE ADDRESSES MUST BE ON AN EVEN BOUNDARY.) THE CSR ADDRESSES AND Q-FACTORS FOR THE FOUR ENTRIES ARE ASSUMED TO BE 160000 AND 0 RESPECTIVELY SINCE THEY WERE ONLY SPECIFIED ONCE. THE LAST TWO UNITS ARE SPECIFIED IN THE THIRD PASS.

THE WHOLE PROCESS COULD HAVE BEEN ACCOMPLISHED IN ONE PASS AS SHOWN BELOW.

```
# UNITS (D) ? 8<CR>

UNIT 1
CSR ADDRESS (0) ? 160000<CR>
SUB-DEVICE # (0) ? 0-7<CR>
Q-FACTOR (0) 0 ? 0,1,0,...,1,1<CR>
```

AS YOU CAN SEE FROM THIS EXAMPLE, NULL REPLIES (COMMAS ENCLOSING A NULL FIELD) TELL THE RUNTIME SERVICES TO REPEAT THE LAST REPLY.

2.6 QUICK START-UP PROCEDURE (XXDP+)



## TO START-UP THIS PROGRAM:

1. BOOT XXDP+
2. TYPE "R ZVSWAO"
3. TYPE "START"
4. ANSWER THE "CHANGE HW" QUESTION WITH "Y" ( IF APPROPRIATE )
5. ANSWER ALL THE HARDWARE QUESTIONS IF ANSWER TO 4 WAS "Y"

WHEN YOU FOLLOW THIS PROCEDURE YOU WILL BE USING ONLY THE DEFAULTS FOR FLAGS. THESE DEFAULTS ARE DESCRIBED IN SECTION 2.3.

## 3.0 ERROR INFORMATION

## 3.1 TYPES OF ERROR MESSAGES

THERE ARE THREE LEVELS OF ERROR MESSAGES THAT MAY BE ISSUED BY A DIAGNOSTIC: GENERAL, BASIC AND EXTENDED. GENERAL ERROR MESSAGES ARE ALWAYS PRINTED UNLESS THE "IER" FLAG IS SET (SECTION 2.3). THE GENERAL ERROR MESSAGE IS OF THE FORM:

```
NAME TYPE NUMBER ON UNIT NUMBER TST NUMBER PC:XXXXXX
ERROR MESSAGE
```

,WHERE; NAME = DIAGNOSTIC NAME  
TYPE = ERROR TYPE (SYS FATAL, DEV FATAL, HARD OR SOFT)  
NUMBER = ERROR NUMBER  
UNIT NUMBER = 0 - N (N IS LAST UNIT IN PTABLE)  
TST NUMBER = TEST AND SUBTEST WHERE ERROR OCCURRED  
PC:XXXXXX = ADDRESS OF ERROR MESSAGE CALL

BASIC ERROR MESSAGES ARE MESSAGES THAT CONTAIN SOME ADDITIONAL INFORMATION ABOUT THE ERROR. THESE ARE ALWAYS PRINTED UNLESS THE "IER" OR "IBR" FLAGS ARE SET (SECTION 2.3). THESE MESSAGES ARE PRINTED AFTER THE ASSOCIATED GENERAL MESSAGE.

EXTENDED ERROR MESSAGES CONTAIN SUPPLEMENTARY ERROR INFORMATION SUCH AS REGISTER CONTENTS OR GOOD/BAD DATA. THESE ARE ALWAYS PRINTED UNLESS THE "IER", "IBR" OR "IXR" FLAGS ARE SET (SECTION 2.3). THESE MESSAGES ARE PRINTED AFTER THE ASSOCIATED GENERAL ERROR MESSAGE AND ANY ASSOCIATED BASIC ERROR MESSAGES.

## 3.2 SPECIFIC ERROR MESSAGES

ERROR 1 : TEST 1 MUST BE PERFORMED FIRST

EXPLANATION: ALL TESTS ( EXCEPT TEST 1 ) REQUIRE TEST 1 TO HAVE BEEN

RUN FIRST TO VERIFY THAT THE DEVICE IS A VSV21.  
TEST 1 SHOULD NOW BE RUN.

OCCURRENCE: ANY TEST EXCEPT TEST 1.

ERROR 2 : TEST 2 MUST BE RUN FIRST TO ENABLE DMA

EXPLANATION: ALL TESTS NEEDING DMA ENABLED REQUIRE TEST 2 TO HAVE  
BEEN RUN FIRST. TEST 2 SHOULD NOW BE RUN.

OCCURRENCE: TESTS 15,16,17 AND 18

ERROR 3 : TIMED OUT WHILE WAITING FOR STATUS VALID

EXPLANATION: THE STATUS VALID BIT IN THE CSR FOR THE CURRENT DEVICE  
HAS NOT BEEN SET WITHIN A SUFFICIENTLY SHORT TIME.  
THIS NORMALLY MEANS THAT THE ON-BOARD FIRMWARE IS  
WAITING FOR SOMETHING AND HAS NOT SET THE BIT.  
AS THIS INDICATES WHEN THE CSR IS IN A VALID STATE TO  
THE HOST FURTHER TESTING OF THIS DEVICE IS NOT POSSIBLE  
UNTIL FIXED.

OCCURRENCE: ANY TEST.

ERROR 4 : THERE HAS BEEN A POWERUP - PLEASE REPEAT TEST 1

EXPLANATION: THE VSV21 HAS BEEN RESET. IT IS NECESSARY TO DO TEST 1  
AGAIN. NOTE THAT DMA WILL ALSO HAVE BEEN DISABLED.

OCCURRENCE: ANY TEST.

ERROR 5 : CONTROLLER READY NOT SET IN CSR

EXPLANATION: THE CONTROLLER READY BIT IS NOT SET IN THE CSR FOR  
THE CURRENT DEVICE. THIS MEANS THAT NO COMMAND CAN  
BE WRITTEN TO IT BY THE HOST. TRY REPEATING TEST 1  
AFTER INVESTIGATION.

OCCURRENCE: ANY TEST.

ERROR 6 : STATUS PACKET SHORTER THAN SPECIFIED LENGTH .

EXPLANATION: IN PROGRAMMED I/O THE HOST PROCESSES A STATUS PACKET  
BY READING THE PARAMETER REGISTER AND EXTRACTING THE  
PACKET WORD BY WORD. IN THIS CASE THE LENGTH SPECIFIED  
IN THE FIRST WORD INDICATES THAT THERE IS MORE DATA TO  
COME BUT THE PARAMETER READY BIT IS NOT SET IN THE CSR  
AND THUS THE PARAMETER REGISTER CANNOT BE READ TO GET  
THE REST OF THE STATUS PACKET.

OCCURRENCE: ANY TEST WHEN DMA IS NOT ENABLED.

ERROR 7 : STATUS AVAILABLE EXPECTED BUT NOT FOUND  
EXPLANATION: THE HOST HAS SENT A COMMAND TO THE VSV21 ( WHICH HAS DMA ENABLED ) AND IS THUS EXPECTING A 'STATUS AVAILABLE' MESSAGE IN THE CSR WHEN THE VSV21 HAS FINISHED PROCESSING. THIS HAS NOT HAPPENED.  
OCCURRENCE: ANY TEST WHEN DMA IS ENABLED.

ERROR 8 : ERROR PACKET RECEIVED FROM VSV21  
EXPLANATION: THE HOST HAS RECEIVED AN ERROR PACKET FROM THE VSV21.  
OCCURRENCE: ANY TEST.

ERROR 9 : FAILED TO TRANSMIT STATUS ACKNOWLEDGE  
EXPLANATION: THE HOST HAS ATTEMPTED TO SEND A 'STATUS ACKNOWLEDGE' COMMAND VIA THE CSR BUT THE CONTROLLER READY BIT HAS NOT BEEN SET AFTERWARDS.  
OCCURRENCE: ANY TEST.

ERROR 10: TIMEOUT WAITING FOR STATUS  
EXPLANATION: THE HOST HAS SENT A COMMAND TO THE VSV21 AND IS EXPECTING A STATUS ACKNOWLEDGE VIA THE PARAMETER REGISTER. THE PARAMETER READY BIT IN THE CSR HAS NOT BEEN SET WITHIN A SUFFICIENTLY SHORT TIME AS THIS INDICATES WHEN THE HOST CAN READ THE PARAMETER REGISTER.  
OCCURRENCE: ANY TEST.

ERROR 11: DMA DATA MATCH ERROR  
EXPLANATION: DATA RECEIVED BACK FROM THE VSV21 IS DIFFERENT TO THAT SENT. THERE IS SOMETHING WRONG WITH DMA.  
OCCURRENCE: TESTS 2,16,17

ERROR 12: FAILED TO GET MICRO DIAGNOSTICS RETURN STATUS PACKET  
EXPLANATION: THE HOST HAS FAILED TO GET A RETURN STATUS PACKET AFTER INVOKING A MICRODIAGNOSTICS TEST.  
OCCURRENCE: TESTS 4-19

ERROR 13: MICRO DIAGNOSTIC TEST UNSUCCESSFUL  
EXPLANATION: THE MICRODIAGNOSTIC TEST HAS FAILED.

OCCURRENCE: TESTS 4-19.

ERROR 14: ERROR PACKET EXPECTED BUT NOT RECEIVED

EXPLANATION: AN INVALID COMMAND IS SENT TO THE VSV21 IN TEST 3 AND SO AN ERROR PACKET IS EXPECTED. THIS DID NOT HAPPEN.

OCCURRENCE: TEST 3.

ERROR 15: TIMED OUT WAITING TO RECEIVE AN INTERRUPT

EXPLANATION: AN INTERRUPT HAS NOT BEEN RECEIVED WITHIN A SUFFICIENTLY SHORT TIME.

OCCURRENCE: TEST 20.

#### 4.0 PERFORMANCE AND PROGRESS REPORTS

AT THE END OF EACH PASS, THE PASS COUNT IS GIVEN ALONG WITH THE TOTAL NUMBER OF ERRORS REPORTED SINCE THE DIAGNOSTIC WAS STARTED. THE "EOP" SWITCH CAN BE USED TO CONTROL HOW OFTEN THE END OF PASS MESSAGE IS PRINTED. SECTION 2.2 DESCRIBES SWITCHES.

#### 5.0 DEVICE INFORMATION TABLES

THE VSV21 MODULE INTERFACES TO THE HOST PROCESSOR VIA THE PROGRAMMED I/O INTERFACE. THIS USES THREE REGISTERS WITHIN A BLOCK OF FOUR ADDRESSABLE WORD LOCATIONS ON THE HOST BUS. THE BLOCK IS ASSIGNED AN ADDRESS BEGINNING ON A MODULO-4 WORD BOUNDARY VIA HARDWARE ADDRESS SELECT SWITCHES. THIS ADDRESS IS TERMED THE DEVICE ADDRESS. THE BLOCK LOOKS LIKE THIS:

ADDRESS	READ	WRITE
177XXXX0	DMA ADDRESS REGISTER	HARDWARE INITIALIZE
177XXXX2	STATUS REGISTER	COMMAND REGISTER (CSR)
177XXXX4	PARAMETER REGISTER	PARAMETER REGISTER
177XXXX6	(UNDEFINED)	(UNDEFINED)

ADDRESS 177XXXX0 IS THE DEVICE ADDRESS. THE HARDWARE P-TABLE CONSISTS OF TWO VALUES FOR EACH DEVICE. THE FIRST IS THE DEVICE ADDRESS, THE SECOND IS THE VECTOR ADDRESS.

THE DMA ADDRESS REGISTER CONTAINS THE LEAST SIGNIFICANT 16 BITS OF THE PHYSICAL ADDRESS OF THE LAST DMA OPERATION PERFORMED BY THE CONTROLLER

WHEN DMA HAS BEEN ENABLED.

THE HARDWARE INITIALIZE REGISTER , WHEN WRITTEN TO , CAUSES THE VSV21 TO BE RESET.

THE STATUS REGISTER CONTAINS STATUS INFORMATION ON THE OPERATIONAL STATUS OF THE CONTROLLER AND THE PROGRESS OF OPERATIONS WHICH ARE BEING PERFORMED BY THE CONTROLLER. THE REGISTER CONTAINS THE FOLLOWING STATUS BITS.

BIT	DESCRIPTION
12	CONTROLLER READY BIT : THIS BIT IS A LOGICAL 1 WHEN THE CONTROLLER IS READY TO ACCEPT A NEW COMMAND FROM THE HOST. A LOGICAL 0 IN THIS BIT INDICATES THAT THE CONTROLLER CANNOT ACCEPT A NEW COMMAND.
13	PARAMETER READY BIT : THIS BIT IS A LOGICAL 1 WHEN THE CONTROLLER HAS INFORMATION FOR THE HOST. IT IS A LOGICAL 0 WHEN THERE IS NO MORE INFORMATION.
14	CONTROLLER ERROR BIT : THIS BIT IS A LOGICAL 1 WHEN THE VSV21 BOARD HAS BEEN RESET. IT WILL BE CLEARED WHEN A COMMAND IS WRITTEN TO THE COMMAND REGISTER OR THE PARAMETER REGISTER IS READ ( PROVIDING THE PARAMETER READY BIT IS SET ). THIS BIT IS A LOGICAL 0 WHEN THE VSV21 BOARD HAS NOT JUST BEEN RESET.
15	STATUS VALID BIT : THIS BIT IS A LOGICAL 1 WHEN THE STATUS OF THE ABOVE DEFINED BITS IS VALID. A LOGICAL 0 INDICATES THAT THE STATUS OF THE ABOVE DEFINED BITS IS NOT VALID AND MAY BE UPDATED. THE HOST MUST WAIT UNTIL THIS BIT IS SET BEFORE READING THE STATUS OF THE BITS AND TAKING ANY SUBSEQUENT ACTION.

THE COMMAND REGISTER IS THE REGISTER FOR PASSING COMMANDS FROM THE HOST TO THE VSV21.

THE PARAMETER REGISTER IS A READ/WRITE REGISTER. IT IS WRITTEN TO WHEN THE HOST NEEDS TO SEND PARAMETERS PRIOR TO WRITING TO THE COMMAND REGISTER. IT IS READ BY THE HOST WHEN THE VSV21 HAS SET THE PARAMETER READY BIT IN THE STATUS REGISTER. THE HOST KEEPS READING THE PARAMETER REGISTER FOR MORE INFORMATION UNTIL THE PARAMETER READY BIT IS CLEAR.

## 6.0 TEST SUMMARIES

### TEST 1: VERIFY VSV21 PRESENCE

This test verifies that there is a VSV21 device at the current CSR. On the first pass it displays the version of the VSV21 and a message

warning the operator to attach external loopback connectors if he wishes tests 4 , 13 , 14 , 16 or 17 to be done. The test also disables interrupts.

#### TEST 2: IN DEPTH Q22 BUS TEST

This test performs an in-depth test of the Q22 bus by means of exercising the DMA mechanism in both directions. It sets up the DMA protocol and must be done before other tests requiring DMA.

#### TEST 3: FORCED ERROR TEST

This test sends an invalid command to the VSV21 to provoke an error packet.

#### TEST 4: FULL ON BOARD TESTS

This test invokes full on-board VSV21 microdiagnostics tests. If the HOE flag is specified the VSV21 will halt on error. If the LOE flag is specified the operator will be given the choice on the first pass of whether he wishes the on-board tests to loop on error. If he selects yes the test may hang indefinitely as the on board will not return control to the host but will continue looping.

#### TEST 5: ROM CHECKSUM TEST

This test invokes the on-board VSV21 ROM CHECKSUM test. If the HOE flag is specified the VSV21 will halt on error. If the LOE flag is specified the operator will be given the choice on the first pass of whether he wishes the on-board tests to loop on error. If he selects yes the test may hang indefinitely as the on board will not return control to the host but will continue looping.

#### TEST 6: NVRAM CHECKSUM TEST

This test invokes the on-board VSV21 NVRAM CHECKSUM test. If the HOE flag is specified the VSV21 will halt on error. If the LOE flag is specified the operator will be given the choice on the first pass of whether he wishes the on-board tests to loop on error. If he selects yes the test may hang indefinitely as the on board will not return control to the host but will continue looping.

#### TEST 7: RAM TEST

This test invokes the on-board VSV21 RAM test. If the HOE flag is specified the VSV21 will halt on error. If the LOE flag is specified the operator will be given the choice on the first pass of whether he wishes the on-board tests to loop on error. If he selects yes the test may hang indefinitely as the on board will not return control to the host but will continue looping.

#### TEST 8: RAM ADDRESSING TEST

This test invokes the on-board VSV21 RAM ADDRESSING test. If the HOE flag is specified the VSV21 will halt on error. If the LOE flag is specified the operator will be given the choice on the first pass of whether he wishes the on-board tests to loop on error.

If he selects yes the test may hang indefinitely as the on board will not return control to the host but will continue looping.

#### TEST 9: 68K PROCESSOR TEST

This test invokes the on-board VSV21 68K PROCESSOR test.  
If the HOE flag is specified the VSV21 will halt on error.  
If the LOE flag is specified the operator will be given the choice on the first pass of whether he wishes the on-board tests to loop on error.  
If he selects yes the test may hang indefinitely as the on board will not return control to the host but will continue looping.

#### TEST 10: INTERNAL EXCEPTIONS TEST

This test invokes the on-board INTERNAL EXCEPTIONS test.  
If the HOE flag is specified the VSV21 will halt on error.  
If the LOE flag is specified the operator will be given the choice on the first pass of whether he wishes the on-board tests to loop on error.  
If he selects yes the test may hang indefinitely as the on board will not return control to the host but will continue looping.

#### TEST 11: ACRCT INTERNAL TEST

This test invokes the on-board ACRCT INTERNAL test.  
If the HOE flag is specified the VSV21 will halt on error.  
If the LOE flag is specified the operator will be given the choice on the first pass of whether he wishes the on-board tests to loop on error.  
If he selects yes the test may hang indefinitely as the on board will not return control to the host but will continue looping.

#### TEST 12: ACRCT EXTERNAL TEST

This test invokes the on-board VSV21 ACRCT EXTERNAL test.  
If the HOE flag is specified the VSV21 will halt on error.  
If the LOE flag is specified the operator will be given the choice on the first pass of whether he wishes the on-board tests to loop on error.  
If he selects yes the test may hang indefinitely as the on board will not return control to the host but will continue looping.

#### TEST 13: DUART BASIC TEST

This test invokes the on-board VSV21 DUART BASIC test on ports 0-3.  
If the HOE flag is specified the VSV21 will halt on error.  
If the LOE flag is specified the operator will be given the choice on the first pass of whether he wishes the on-board tests to loop on error.  
If he selects yes the test may hang indefinitely as the on board will not return control to the host but will continue looping.

#### TEST 14: DUART FULL TEST

This test invokes the on-board VSV21 DUART FULL test on ports 0-3.  
If the HOE flag is specified the VSV21 will halt on error.  
If the LOE flag is specified the operator will be given the choice on the first pass of whether he wishes the on-board tests to loop on error.  
If he selects yes the test may hang indefinitely as the on board will not return control to the host but will continue looping.

## TEST 15: PERIPHERAL CONFIDENCE TEST

This test is not done if /FLA:UAM is specified on the START command.  
This test allows the operator to select a port for either input or output.  
If he selects output an ASCII string is written to that port.  
If he selects input the first 16 characters read from that port are displayed in octal on the console running the tests.  
If the HOE flag is specified the VSV21 will halt on error.  
If the LOE flag is specified the operator will be given the choice on the first pass of whether he wishes the on-board tests to loop on error.  
If he selects yes the test may hang indefinitely as the on board will not return control to the host but will continue looping.

## TEST 16: INTERNAL LOOPBACK TEST

This test causes the on-board software to perform an INTERNAL LOOPBACK test on ports 0-3.  
If the HOE flag is specified the VSV21 will halt on error.  
If the LOE flag is specified the operator will be given the choice on the first pass of whether he wishes the on-board tests to loop on error.  
If he selects yes the test may hang indefinitely as the on board will not return control to the host but will continue looping.

## TEST 17: EXTERNAL I/O LOOPBACK TEST

This test causes the on-board software to perform an EXTERNAL LOOPBACK test on ports 0-3.  
If the HOE flag is specified the VSV21 will halt on error.  
If the LOE flag is specified the operator will be given the choice on the first pass of whether he wishes the on-board tests to loop on error.  
If he selects yes the test may hang indefinitely as the on board will not return control to the host but will continue looping.

## TEST 18: SCREEN TEST

This test is not done if /FLA:UAM is specified on the START command.  
This test displays different screen test pictures as selected by the operator. If the HOE flag is specified the VSV21 will halt on error.  
If the LOE flag is specified the operator will be given the choice on the first pass of whether he wishes the on-board tests to loop on error.  
If he selects yes the test may hang indefinitely as the on board will not return control to the host but will continue looping.

## TEST 19: NVRAM READ/WRITE TEST

This test is not done if /FLA:UAM is specified on the START command.  
This test tests NVRAM read/write. On each pass the operator will be asked if he wishes to continue as NVRAM has a limited life in terms of read/write cycles.  
If the HOE flag is specified the VSV21 will halt on error.  
If the LOE flag is specified the operator will be given the choice on the first pass of whether he wishes the on-board tests to loop on error.  
If he selects yes the test may hang indefinitely as the on board will not return control to the host but will continue looping.

## TEST 20: ENABLE INTERRUPTS TEST



This test verifies that interrupts are received by the host when interrupts are enabled.

&

```
869          .TITLE PROGRAM HEADER AND TABLES
870          .SBTTL PROGRAM HEADER
871
872          .MCALL SVC
873 000000    SVC                                ; INITIALIZE SUPERVISOR MACROS
874
875          ;*****
876          ; IF STRUCTURED MACROS ARE TO BE USED, ADD ".MCALL STRUCT" AND "STRUCT"
877          ; TO INITIALIZE THE STRUCTURED MACROS.
878
879          177777 SVCINS= -1          ; LIST INSTRUCTIONS, SHIFTED RIGHT
880          177777 SVCTST= -1         ; LIST TEST TAGS, SHIFTED RIGHT
881          177777 SVCSUB= -1        ; LIST SUBTEST TAGS, SHIFTED RIGHT
882          177777 SVCGBL= -1        ; LIST GLOBAL TAGS, SHIFTED RIGHT
883          177777 SVCTAG= -1        ; LIST OTHER TAGS, SHIFTED RIGHT
884
885          ; CHANGE THE VALUES OF THE SVC... SYMBOLS TO BE ZERO IF YOU WISH
886          ; TO ALIGN THE MACRO CALLS AND THEIR EXPANSIONS. CHANGE THE
887          ; SYMBOLS TO BE MINUS-ONE TO NOT LIST THE EXPANSIONS. YOU MAY
888          ; CHANGE THE SYMBOLS AT ANY POINT IN YOUR PROGRAM.
889          ;*****
890
891 000000    .ENABL ABS,AMA
892          002000    = 2000
893
894 002000    BGNMOD
895
896          ;++
897          ; THE PROGRAM HEADER IS THE INTERFACE BETWEEN
898          ; THE DIAGNOSTIC PROGRAM AND THE SUPERVISOR.
899          ;--
900
901 002000    POINTER ERR_TBL,BGNSETUP
902
903 002000    HEADER CZVSWAO,A,0,15,0
904
```

## DISPATCH TABLE

```
916          .SBTTL DISPATCH TABLE
917
918          ;++
919          ; THE DISPATCH TABLE CONTAINS THE STARTING ADDRESS OF EACH TEST.
920          ; IT IS USED BY THE SUPERVISOR TO DISPATCH TO EACH TEST.
921          ;--
922
923          000004          MAXUNIT== 4          ; maximum number of units
924          000024          MAXTST == 20.        ; maximum number of tests
925
926 002122          DISPATCH MAXTST
927
```

## DEFAULT HARDWARE P-TABLE

```
929          .SBTTL  DEFAULT HARDWARE P-TABLE
930
931          ;++
932          ; THE DEFAULT HARDWARE P-TABLE CONTAINS DEFAULT VALUES OF
933          ; THE TEST-DEVICE PARAMETERS.  THE STRUCTURE OF THIS TABLE
934          ; IS IDENTICAL TO THE STRUCTURE OF THE HARDWARE P-TABLES,
935          ; AND IS USED AS A "TEMPLATE" FOR BUILDING THE P-TABLES.
936          ;--
937
938 002174          BGNHW  DFPTBL
939
949
950 002176 172010  .WORD  172010
951 002200 000320  .WORD  320
952
953 002202          ENDDHW
```

DEFAULT HARDWARE P-TABLE

955  
 956  
 957  
 958  
 959  
 960  
 961  
 962  
 963  
 964 002202

.TITLE GLOBAL AREAS  
 .SBTTL GLOBAL EQUATES SECTION

;++  
 ; THE GLOBAL EQUATES SECTION CONTAINS PROGRAM EQUATES THAT  
 ; ARE USED IN MORE THAN ONE TEST.  
 ;--

EQUALS

; BIT DIFINITIONS

100000	BIT15== 100000
040000	BIT14== 40000
020000	BIT13== 20000
010000	BIT12== 10000
004000	BIT11== 4000
002000	BIT10== 2000
001000	BIT09== 1000
000400	BIT08== 400
000200	BIT07== 200
000100	BIT06== 100
000040	BIT05== 40
000020	BIT04== 20
000010	BIT03== 10
000004	BIT02== 4
000002	BIT01== 2
000001	BIT00== 1

; BIT9== BIT09  
 BIT8== BIT08  
 BIT7== BIT07  
 BIT6== BIT06  
 BIT5== BIT05  
 BIT4== BIT04  
 BIT3== BIT03  
 BIT2== BIT02  
 BIT1== BIT01  
 BIT0== BIT00

; EVENT FLAG DEFINITIONS  
 ; EF32:EF17 RESERVED FOR SUPERVISOR TO PROGRAM COMMUNICATION

000040	EF.START== 32.	; BIT POSITION IN SECOND STATUS WORD
000037	EF.RESTART== 31.	; (100000) START COMMAND WAS ISSUED
000036	EF.CONTINUE== 30.	; (040000) RESTART COMMAND WAS ISSUED
000035	EF.NEW== 29.	; (020000) CONTINUE COMMAND WAS ISSUED
000034	EF.PWR== 28.	; (010000) A NEW PASS HAS BEEN STARTED
		; (004000) A POWER-FAIL/POWER-UP OCCURRED

; PRIORITY LEVEL DEFINITIONS

000340	PRI07== 340
000300	PRI06== 300
000240	PRI05== 240

## GLOBAL EQUATES SECTION

```

000200          PRI04== 200
000140          PRI03== 140
000100          PRI02== 100
000040          PRI01== 40
000000          PRI00== 0
;
;OPERATOR FLAG BITS
;
000004          EVL==      4
000010          LOT==     10
000020          ADR==     20
000040          IDU==     40
000100          ISR==    100
000200          UAM==    200
000400          BOE==    400
001000          PNT==   1000
002000          PRI==   2000
004000          IXE==   4000
010000          IBE==  10000
020000          IER==  20000
040000          LOE==  40000
100000          HOE== 100000

965
966
967          ;CSR bit meanings
968          VS.SVL == BIT15      ; status valid
969          VS.ERR == BIT14      ; error ( powerup only )
970          VS.CRY == BIT12      ; controller ready
971          VS.PRY == BIT13      ; parameter ready
972
973
974          ;Parallel interface commands ( these are written to the CSR by the host )
975
976          002400          PI.CMA == 5*400+0      ; command available
977          003000          PI.ACK == 6*400+0      ; status ack
978          001000          PI.SEN == 2*400+0      ; buffers sent
979          001001          PI.CAN == 2*400+1      ; buffers cancelled
980          000400          PI.IDM == 1*400+0      ; check ident. protocol.
981          002000          PI.TST == 4*400+0      ; invoke microdiagnostics
982          001400          PI.IDS == 3*400+0      ; disable interrupts
983          001403          PI.IEN == PI.IDS+3      ; enable interrupts
984          177777          PI.RUB == 177777        ; rubbish command to provoke error
985
986          ;This one is written to the parameter register by the on-board
987
988          100000          PI.AVA == BIT15          ; status available
989
990          ;These are DMA command codes that are put in the first word of the command
991          ;buffer by the host
992
993          002101          DI.RDA == 2101          ; read data
994          002102          DI.WDD == 2102          ; write dats.
995          002103          DI.RRA == 2103          ; read ram protocol code.
996          002104          DI.WRA == 2104          ; write ram protocol code.
997
998          ;return from the on-board
999

```

## GLOBAL EQUATES SECTION

```

1000      000011      DI.MIC == 9.           ; micro diagnostic result code
1001      000005      DI.ERP == 5.           ; error reply packet code
1002      000176      DI.ERS == 126.        ; error log and stop packet code
1003      000177      DI.ERL == 127.        ; error log packet code
1004
1005      ;Fills in error table details
1006      .MACRO ERR.FILLIN      NUMBER,MESSAGE
1007      MOV #2,ERRTYP          ; assume hard error for now
1008      MOV #NUMBER,ERRNBR     ; get error number
1009      MOV #MESSAGE,ERRMSG    ; get error message
1010      MOV #ERRCODE,ERRBLK   ; get error routine
1011      .ENDM
1012
1013      ;Error return
1014      .MACRO ERR.RETURN
1015      SEC
1016      RETURN
1017      .ENDM
1018
1019      ;OK return
1020      .MACRO OK.RETURN
1021      CLC
1022      RETURN
1023      .ENDM
1024
1025      ;Put at start of each test.
1026      ;Ensures that TEST1 must have been performed first.
1027      .MACRO ST.TEST,?LABEL
1028      BGNTST
1029      CALL STTEST
1030      BCC LABEL
1031      EXIT TST
1032 LABEL:
1033      .ENDM
1034
1035      ;Sets up POLCNT for a delay
1036      .MACRO SET.POLCNT
1037      MOV #1500.,POLCNT
1038      .ENDM
1039
1040      ;Writes a command to the CSR
1041      .MACRO DO.COMMAND      COM,LABEL
1042      MOV #COM,COMAND
1043      SET.POLCNT
1044      CALL DOCOM
1045      BCC LABEL
1046      .ENDM
1047
1048      ;Tests manual intervention and exits test if not allowed
1049      .MACRO MAN.IGNORE     ?LABEL
1050      MANUAL
1051      BCOMPLETE LABEL
1052      EXIT TST
1053 LABEL:
1054      .ENDM
1055
1056      ;Tests dma done and exits test if not

```

## GLOBAL EQUATES SECTION

```
1057          .MACRO DMA.IGNORE      ?LABEL
1058          TST   DMASET(R4)
1059          BNE   LABEL              ; DMA enabled ?
1060          ERRHRD 2,ERR2,ERRCODE    ; branch if yes
1061          EXIT   TST                ; error exit if not
1062 LABEL:
1063          .ENDM
1064
1065 ;Calls writeram routine
1066          .MACRO DO.WRAMS          BUFADD,BUFLEN
1067          .IF   NC                 BUFADD
1068          .IFT
1069          MOV   BUFADD,COMBUF+4
1070          MOV   BUFLN,COMBUF+6
1071          .IFF
1072          MOV   #DMAOBUF,COMBUF+4
1073          MOV   #DMALEN,COMBUF+6
1074          .IFTF
1075          CALL  WRAMS
1076          .ENDC
1077          .ENDM
1078
1079 ;Calls readram routine
1080          .MACRO DO.RRAMS          BUFADD,BUFLEN
1081          .IF   NB                 BUFADD
1082          .IFT
1083          MOV   BUFADD,COMBUF+4
1084          MOV   BUFLN,COMBUF+6
1085          .IFF
1086          MOV   #DMAIBUF,COMBUF+4
1087          MOV   #DMALEN,COMBUF+6
1088          .IFTF
1089          CALL  RRAMS
1090          .ENDC
1091          .ENDM
```



GLOBAL DATA SECTION

```

1093          .SBTTL GLOBAL DATA SECTION
1094
1095          ;++
1096          ; THE GLOBAL DATA SECTION CONTAINS DATA THAT ARE USED
1097          ; IN MORE THAN ONE TEST.
1098          ;--
1099
1100 002202          ERRTBL
1100 002202 000000  ERRRTYP:: .WORD 0
1100 002204 000000  ERRNBR:: .WORD 0
1100 002206 000000  ERRMSG:: .WORD 0
1100 002210 000000  ERRBLK:: .WORD 0
1101
1102 002212 000000  TEMP1:: .WORD 0 ; temporary storage
1103 002214 000000  TEMP2:: .WORD 0 ; temporary storage
1104 002216 000000  POLCNT:: .WORD 0 ; <0 = poll CSR forever , >0 = poll CSR
1105          ; ; required number
1106 002220 000000  LOGUNT:: .WORD 0 ; current unit number
1107 002222          PBLOC:: .BLKW 4 ; storage for sending microdiagnostic parameters
1108 002232 000000  CURCSR:: .WORD 0 ; CSR address for current unit
1109 002234 000000  CURPAR:: .WORD 0 ; parameter register address for current unit
1110 002236 000000  CURVEC:: .WORD 0 ; vector address for current unit
1111 002240 000000  INTFLG:: .WORD 0 ; <>0 = got interrupt , 0 = not got interrupt
1112 002242 000000  NOTEST:: .WORD 0 ; 0 = tests can be done , <>0 = tests cannot
1113 002244 000001 000002 000004  BITS:: .WORD BIT0,BIT1,BIT2,BIT3,BIT4,BIT5,BIT6,BIT7,BIT8
1113 002252 000010 000020 000040
1113 002260 000100 000200 000400
1114 002266 001000 002000 004000          .WORD BIT9,BIT10,BIT11,BIT12,BIT13,BIT14,BIT15
1114 002274 010000 020000 040000
1114 002302 100000
1115
1116          ;Flags per unit (globals are offset from start of flags for that unit)
1117          ;*****
1118          ; N.B. R4 POINTS TO THE FLAGS FOR THE CURRENT UNIT
1119          ;*****
1120
1121          000000          T1DONE == 0 ; 1 WORD: 0 = TEST1 not done , <>0 = TEST1 DONE
1122          000002          DMASET == T1DONE+2 ; 1 WORD: 0 = DMA disabled , <>0 = DMA enabled
1123          000004          FTHRU == DMASET+2 ; 2 WORDS: bit 'n' is clear if TEST 'n' not yet
1124          ; ; done
1125          000010          TMASKS == FTHRU+4 ; MAXTST WORDS : test masks
1126          ; ; (only applicable to Microdiagnostic tests)
1127          000030          LENFLAGS == TMASKS/2+MAXTST ; length of flags per unit in words
1128 002304 000030  FLAGS:: .BLKW <LENFLAGS*MAXUNIT> ; the flags indexed by unit
1129
1130          ;Command and status buffers
1131
1132          000024          COMLEN = 24
1133          000036          STALEN = 36
1134 002604 012000 002616 017000  BUFDES: .WORD <COMLEN*400> , COMBUF , <STALEN*400> , STABUF ; buffer descriptors
1134 002612 002666
1135 002614          COMAND: .BLKW 1 ; current command
1136 002616          COMBUF: .BLKW COMLEN ; command packet buffer for DMA
1137 002666          STABUF: .BLKW STALEN ; status packet buffer for DMA.
1138
1139          ; the data to be written to RAM
1140

```

GLOBAL DATA SECTION

1141	002762		DMAOBUF::		
1142	002762	177776		.WORD	177776
1143	002764	177775		.WORD	177775
1144	002766	177774		.WORD	177774
1145	002770	177773		.WORD	177773
1146	002772	177772		.WORD	177772
1147	002774	177771		.WORD	177771
1148	002776	177770		.WORD	177770
1149	003000	177760		.WORD	177760
1150	003002	177750		.WORD	177750
1151	003004	177740		.WORD	177740
1152	003006	177730		.WORD	177730
1153	003010	177720		.WORD	177720
1154	003012	177710		.WORD	177710
1155	003014	177700		.WORD	177700
1156	003016	177600		.WORD	177600
1157	003020	177500		.WORD	177500
1158	003022	177400		.WORD	177400
1159	003024	177300		.WORD	177300
1160	003026	177200		.WORD	177200
1161	003030	177100		.WORD	177100
1162	003032	177000		.WORD	177000
1163	003034	000000		.WORD	0
1164					
1165		000054	DMALEN =	.-DMAOBUF	; length of above data
1166	003036		DMAIBUF::		
1167	003036			.BLKW DMALEN	; buffer to read back into .
1168					
1169					

## GLOBAL TEXT SECTION

```

1171          .SBTTL GLOBAL TEXT SECTION
1172
1173          ;**
1174          ; THE GLOBAL TEXT SECTION CONTAINS FORMAT STATEMENTS,
1175          ; MESSAGES, AND ASCII INFORMATION THAT ARE USED IN
1176          ; MORE THAN ONE TEST.
1177          ;--
1178          .NLIST BEX
1179
1180 003166      124      105      123  ERR1:: .ASCIZ /TEST 1 MUST BE PERFORMED FIRST/
1181 003225      124      105      123  ERR2:: .ASCIZ /TEST 2 MUST BE RUN FIRST TO ENABLE DMA/
1182 003274      124      111      115  ERR3:: .ASCIZ /TIMED OUT WHILE WAITING FOR STATUS VALID/
1183 003345      124      110      105  ERR4:: .ASCIZ /THERE HAS BEEN A POWERUP - PLEASE REPEAT TEST 1/
1184 003425      103      117      116  ERR5:: .ASCIZ /CONTROLLER READY NOT SET IN CSR/
1185 003465      123      124      101  ERR6:: .ASCIZ /STATUS PACKET SHORTER THAN SPECIFIED LENGTH/
1186 003541      123      124      101  ERR7:: .ASCIZ /STATUS AVAILABLE EXPECTED BUT NOT FOUND/
1187 003611      105      122      122  ERR8:: .ASCIZ /ERROR PACKET RECEIVED FROM VSV21/
1188 003652      106      101      111  ERR9:: .ASCIZ /FAILED TO TRANSMIT STATUS ACKNOWLEDGE/
1189 003720      124      111      115  ERR10:: .ASCIZ /TIMEOUT WAITING FOR STATUS/
1190 003753      104      115      101  ERR11:: .ASCIZ /DMA DATA MATCH ERROR/
1191 004000      106      101      111  ERR12:: .ASCIZ /FAILED TO GET MICRO DIAGNOSTICS RETURN STATUS PACKET/
1192 004065      115      111      103  ERR13:: .ASCIZ /MICRO DIAGNOSTIC TEST UNSUCCESSFUL/
1193 004130      105      122      122  ERR14:: .ASCIZ /ERROR PACKET EXPECTED BUT NOT RECEIVED/
1194 004177      124      111      115  ERR15:: .ASCIZ /TIMED OUT WAITING TO RECEIVE AN INTERRUPT/
1195
1196          ;
1197          ; NAMES OF DEVICES SUPPORTED BY PROGRAM
1198          ;
1199 004251          DEVTYP <VSV21>
1200          ; TEST DESCRIPTION
1201          ;
1202 004260          DESCRIPT      <VSV21 DIAGNOSTIC>
1203          .EVEN
1204
1205          ;
1206          ; FORMAT STATEMENTS USED IN PRINT CALLS
1207          ;
1208
1209

```

GLOBAL ERROR REPORT SECTION

1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1220 004302  
1221  
1222 004302 005237 002242  
1223  
1224 004306  
1225  
1226  
1227  
1228 004312  
1229

.SBTTL GLOBAL ERROR REPORT SECTION

;++  
; THE GLOBAL ERROR REPORT SECTION CONTAINS MESSAGE PRINTING AREAS  
; USED BY MORE THAN TEST TO OUTPUT ADDITIONAL ERROR INFORMATION. PRINTB  
; (BASIC) AND PRINTX (EXTENDED) CALLS ARE USED TO CALL PRINT SERVICES.  
;--

BGNMSG ERRCODE

INC NOTEST

;say no further tests allowed

EXIT MSG

.EVEN

ENDMSG

GLOBAL SUBROUTINES SECTION

```

1231      .SBTTL GLOBAL SUBROUTINES SECTION
1232
1233      ;**
1234      ; THE GLOBAL SUBROUTINES SECTION CONTAINS THE SUBROUTINES
1235      ; THAT ARE USED IN MORE THAN ONE TEST.
1236      ;--
1237
1238      ;**
1239      ; FUNCTIONAL DESCRIPTION:
1240      ;   SUBROUTINE TO POLL THE CSK OR STATUS VALID BIT TO BE SET.
1241
1242      ; INPUTS:
1243      ;   POLCNT          - NUMBER OF TIMES TO POLL , -1 = INFINITE
1244
1245      ; IMPLICIT INPUTS:
1246      ;   CURCSR          - CSR ADDRESS TO POLL
1247      ;   T1DONE         - TEST 1 DONE FLAG
1248
1249      ; OUTPUTS:
1250      ;   CONDITION CODE - CLEAR = SUCCESS
1251      ;                   - SET   = FAILURE
1252
1253      ; IMPLICIT OUTPUTS:
1254      ;   ERROR BLOCK    - FILLED IN ON FAILURE
1255      ;   T1DONE         - CLEARED IF POWERUP
1256      ;   DMASET        - CLEARED IF POWERUP
1257
1258      ; SUBORDINATE ROUTINES USED:
1259      ;   NONE
1260
1261      ; FUNCTIONAL SIDE EFFECTS:
1262      ;   IF POWERUP OF VSV21 FOUND FORCES TEST1 TO BE DONE AGAIN BY CLEARING
1263      ;   T1DONE. ALSO CLEARS DMASET TO SAY THAT DMA IS NOT ENABLED.
1264
1265      ; CALLING SEQUENCE:
1266      ;   MOV    #1500.,POLCNT    ;SET UP POLL COUNT
1267      ;   CALL  POLL              ;GO TO ROUTINE
1268      ;   BCS   ERROR            ;CARRY SET IF ROUTINE HAD ERROR
1269      ;--
1270 004314 032777 100000 175710 POLL:: BIT    #VS.SVL,@CURCSR      ; SVL ?
1271 004322 001040          BNE    2$              ; branch if yes
1272 004324          DELAY  10.                ; delay 10 units
1273 004354 005737 002216          TST    POLCNT          ; poll forever ?
1274 004360 002755          BLT    POLL              ; yes
1275 004362 005337 002216          DEC    POLCNT          ; no - reduce count
1276 004366 003352          BGT    POLL              ; try again if still some left
1277
1278 004370          ERR.FILLIN    3,ERR3
1279 004370 012737 000002 002202      MOV    #2,ERRTYP      ; assume hard error for now
1280 004376 012737 000003 002204      MOV    #3,ERRNBR     ; get error 3
1281 004404 012737 003274 002206      MOV    #ERR3,ERRMSG  ; get error ERR3
1282 004412 012737 004302 002210      MOV    #ERRCODE,ERRBLK ; get error routine
1283 004420          ERR.RETURN
1284 004420 000261          SEC
1285 004422 000207          RETURN
1286
1287 1281 004424 032777 040000 175600 2$: BIT    #VS.ERR,@CURCSR      ; ERR ?

```

GLOBAL SUBROUTINES SECTION

```

1282 004432 001425          BEQ      3$          ; branch if not
1283
1284          ; powerup - initialise
1285 004434 005064 000002    CLR      DMASET(R4)        ; say DMA disabled
1286 004440 005764 000000    TST      T1DONE(R4)       ; Test 1 done yet ?
1287 004444 001420          BEQ      3$          ; branch if not
1288 004446 005064 000000    CLR      T1DONE(R4)       ; say Test 1 not done
1289 004452          ERR.FILLIN 4,ERR4
      004452 012737 000002 002202  MOV      #2,ERR4TYP      ; assume hard error for now
      004460 012737 000004 002204  MOV      #4,ERR4NBR      ; get error 4
      004466 012737 003345 002206  MOV      #ERR4,ERRMSG    ; get error ERR4
      004474 012737 004302 002210  MOV      #ERRCODE,ERRBLK ; get error routine
1290 004502          ERR.RETURN
      004502 000261          SEC
      004504 000207          RETURN
1291
1292 004506          3$:      OK.RETURN
      004506 000241          CLC
      004510 000207          RETURN
1293

```

GLOBAL SUBROUTINES SECTION

```

1295      ;**
1296      ; FUNCTIONAL DESCRIPTION:
1297      ;     SUBROUTINE TO POLL THE CSR FOR STATUS VALID AND CONTROLLER READY BITS
1298      ;     TO BE SET.
1299
1300      ; INPUTS:
1301      ;     NONE
1302
1303      ; IMPLICIT INPUTS:
1304      ;     CURCSR          - CSR ADDRESS TO POLL
1305
1306      ; OUTPUTS:
1307      ;     CONDITION CODE - CLEAR = SUCCESS
1308      ;                   - SET   = FAILURE
1309
1310      ; IMPLICIT OUTPUTS:
1311      ;     ERROR BLOCK    - FILLED IN ON FAILURE
1312
1313      ; SUBORDINATE ROUTINES USED:
1314      ;     POLL
1315
1316      ; FUNCTIONAL SIDE EFFECTS:
1317      ;     NONE
1318
1319      ; CALLING SEQUENCE:
1320      ;     CALL  POLLCRY          ;GO TO ROUTINE
1321      ;     BCS  ERROR            ;CARRY SET IF ROUTINE HAD ERROR
1322      ;--
1323      POLLCRY::
1324      CALL  POLL                ; poll for SVL
1325      BCC  10$                 ; branch if SVL
1326      RETURN                   ; else return if not
1327      BIT  #VS.CRY,@CURCSR     ; CRY ?
1328      BNE  20$                 ; branch if yes
1329      ERR.FILLIN 5,ERR5        ; error return if not
1330      MOV  #2,ERPTYP           ; assume hard error for now
1331      MOV  #5,ERRN@R           ; get error 5
1332      MOV  #ERR5,ERRMSG        ; get error ERR5
1333      MOV  #ERRCODE,ERRBLK     ; get error routine
1334      ERR.RETURN
1335      SEC
1336      RETURN
1337      OK.RETURN                ; success return
1338      CLC
1339      RETURN

```

## GLOBAL SUBROUTINES SECTION

```

1334      ;**
1335      ; FUNCTIONAL DESCRIPTION:
1336      ;   SUBROUTINE TO WRITE A COMMAND TO THE CURRENT CSR.
1337
1338      ; INPUTS:
1339      ;   POLCNT           - NUMBER OF TIMES TO POLL , -1 = INFINITE
1340
1341      ; IMPLICIT INPUTS:
1342      ;   CURCSR          - CSR ADDRESS TO POLL
1343
1344      ; OUTPUTS:
1345      ;   CONDITION CODE  - CLEAR = SUCCESS
1346      ;                   - SET   = FAILURE
1347
1348      ; IMPLICIT OUTPUTS:
1349      ;   ERROR BLOCK    - FILLED IN ON FAILURE
1350
1351      ; SUBORDINATE ROUTINES USED:
1352      ;   NONE
1353
1354      ; FUNCTIONAL SIDE EFFECTS:
1355      ;   THIS DEPENDS ON THE COMMAND. E.G. IF THE COMMAND IS 'BUFFERS SENT'
1356      ;   DMA WILL THEN BE ENABLED AND STATUS WILL COME BACK VIA DMA.
1357
1358      ; CALLING SEQUENCE:
1359      ;   MOV     #1500.,POLCNT   ;SET UP POLL COUNT
1360      ;   CALL   DOCOM           ;GO TO ROUTINE
1361      ;   BCS    ERROR           ;CARRY SET IF ROUTINE HAD ERROR
1362      ;--
1363
1364 004572 004737 004314  DOCOM:: CALL POLL                ; ensure SVL
1365 004576 103003          BCC     10$                ; branch if OK
1366 004600          ERROR
1367 004602          ERR.RETURN
1368          004602 000261  SEC
1369          004604 000207  RETURN
1370
1371 1369 004606 032777 020000 175416 10$: BIT     #VS.PRY,@CURCSR      ; PRY ?
1372 004614 001404          BEQ     20$                ; branch if not
1373 004616 004737 005066          CALL   GETST1          ; get status
1374 004622 103001          BCC     20$                ; branch if status OK
1375 004624 000207          RETURN                    ; return if not
1376
1377 1375 004626 032777 010000 175376 20$: BIT     #VS.CRY,@CURCSR      ; CRY ?
1378 004634 001017          BNE     30$                ; branch if yes
1379 004636          ERR.FILLIN      5,ERR5
1380          004636 012737 000002 002202  MOV     #2,ERRTYP      ; assume hard error for now
1381          004644 012737 000005 002204  MOV     #5,ERRNBR      ; get error 5
1382          004652 012737 003425 002206  MOV     #ERR5,ERRMSG   ; get error ERR5
1383          004660 012737 004302 002210  MOV     #ERRCODE,ERRBLK ; get error routine
1384          1378 004666          ERROR
1385          1379 004670          ERR.RETURN
1386          004670 000261  SEC
1387          004672 000207  RETURN
1388
1389 1381 004674 013777 002614 175330 30$: MOV     COMAND,@CURCSR      ; write comand to the CSR
1390 1382

```



GLOBAL SUBROUTINES SECTION

```

1383 004702 004737 004314      35$: CALL POLL           ; wait for SVL
1384 004706 103003              BCC 40$
1385 004710                      ERROR
1386 004712                      ERR.RETURN
      004712 000261              SEC
      004714 000207              RETURN
1387
1388 004716 032777 020000 175306 40$: BIT #VS.PRY,@CURCSR      ; PRY ?
1389 004724 001055              BNE 50$           ; branch if yes
1390
1391                      ;interrupt mask and buffer sent commands do not expect status from the VSV21
1392
1393 004726 022737 001400 002614      CMP #PI.IDS,COMAND      ; interrupt disable command ?
1394 004734 001453              BEQ 60$           ; branch if yes
1395 004736 022737 001403 002614      CMP #PI.IEN,COMAND      ; interrupt enable command ?
1396 004744 001447              BEQ 60$           ; branch if yes
1397 004746 022737 001000 002614      CMP #PI.SEN,COMAND      ; buf sent command ?
1398 004754 001443              BEQ 60$           ; branch if yes
1399
1400                      ;try again to get status
1401 004756                      DELAY 10.           ; delay 10 units
1402 005006 005737 002216      TST POLCNT           ; poll forever ?
1403 005012 002733              BLT 35$           ; yes
1404 005014 005337 002216      DEC POLCNT           ; no - reduce count
1405 005020 003330              BGT 35$           ; try again if still some left
1406 005022                      ERR.FILLIN 10.,ERR10      ; timeout
      005022 012737 000002 002202      MOV #2,ERRTYP          ; assume hard error for now
      005030 012737 000012 002204      MOV #10.,ERRNBR        ; get error 10.
      005036 012737 003720 002206      MOV #ERR10,ERRMSG      ; get error ERR10
      005044 012737 004302 002210      MOV #ERRCODE,ERRBLK    ; get error routine
1407 005052                      ERROR
1408 005054                      ERR.RETURN
      005054 000261              SEC
      005056 000207              RETURN
1409
1410 005060 004737 005462      50$: CALL GETST2           ; get status
1411 005064 000207      60$: RETURN
1412

```

GLOBAL SUBROUTINES SECTION

```

1414      ;++
1415      ; FUNCTIONAL DESCRIPTION:
1416      ;   SUBROUTINE TO GET A STATUS PACKET FROM THE VSV21 INTO STATUS BUFFER.
1417      ;   THIS INCLUDES BOTH PROGRAMMED I/O AND DMA MODES.
1418      ;   THIS MODULE IS VERY SIMILAR TO GETST2 BUT IS CALLED BEFORE A COMMAND IS
1419      ;   SENT TO THE VSV21.
1420      ;
1421      ; INPUTS:
1422      ;   NONE
1423
1424      ; IMPLICIT INPUTS:
1425      ;   DMASET          - DMA ENABLED FLAG
1426      ;   STABUF         - STATUS BUFFER
1427      ;   CURCSR        - CSR ADDRESS
1428
1429      ; OUTPUTS:
1430      ;   CONDITION CODE - CLEAR = SUCCESS
1431      ;                   - SET   = FAILURE
1432
1433      ; IMPLICIT OUTPUTS:
1434      ;   ERROR BLOCK   - FILLED IN ON FAILURE
1435      ;   STABUF        - STATUS BUFFER
1436
1437      ; SUBORDINATE ROUTINES USED:
1438      ;   POLL
1439      ;   POLLCRY
1440
1441      ; FUNCTIONAL SIDE EFFECTS:
1442      ;   NONE
1443
1444      ; CALLING SEQUENCE:
1445      ;   CALL   GETST1      ;GO TO ROUTINE
1446      ;   BCS   ERROR      ;CARRY SET IF ROUTINE HAD ERROR
1447      ;--
1448
1449 005066 GETST1::
1450 005066 005764 000002      TST   DMASET(R4)      ; DMA enabled ?
1451 005072 001060              BNE   30$              ; branch if yes
1452
1453      ;DMA not enabled - get status packet via parameter register
1454
1455      ; special processing if ERR set - only one word in status packet
1456 005074 032777 040000 175130 BIT   #VS.ERR,@CURCSR ; ERR ?
1457 005102 001407              BEQ   5$              ; branch if no
1458 005104 012737 000001 002666 MOV   #1,STABUF       ; pretend status packet has a header
1459 005112 017737 175116 002670 MOV   @CURPAR,STABUF+2 ; get rest of packet
1460 005120 000444              BR    25$
1461
1462 005122 017737 175106 002666 5$: MOV   @CURPAR,STABUF ; get status header word
1463 005130 113701 002666      MOVB  STABUF,R1      ; get number of following status words
1464 005134 001436              BEQ   25$              ; branch if none
1465 005136 012702 002670      MOV   #<STABUF+2>,R2 ; get address of where to put them
1466 005142 004737 004314 10$: CALL  POLL              ; poll
1467 005146 103003              BCC   20$
1468 005150              ERROR
1469 005152              ERR.RETURN
1469 005152 000261              SEC

```

GLOBAL SUBROUTINES SECTION

```

005154 000207 RETURN
1470 005156 032777 020000 175046 20$: BIT #VS.PRY,@CURCSR ; PRY ?
1471 005164 001017 BNE 22$ ; branch if yes
1472 005166 ERR.FILLIN 6,ERR6
005166 012737 000002 002202 MOV #2,ERRTYP ; assume hard error for now
005174 012737 000006 002204 MOV #6,ERRNBR ; get error 6
005202 012737 003465 002206 MOV #ERR6,ERRMSG ; get error ERR6
005210 012737 004302 002210 MOV #ERRCODE,ERRBLK ; get error routine
1473 005216 ERROR
1474 005220 ERR.RETURN
005220 000261 SEC
005222 000207 RETURN
1475 005224 017722 175004 22$: MOV @CURPAR,(R2)+ ; get a status word
1476 005230 077134 SOB R1,10$ ; again
1477 005232 000434 25$: BR 60$
1478
1479 ;DMA enabled - status packet is written straight to status buffer
1480
1481 005234 022777 100000 174772 30$: CMP #PI.AVA,@CURPAR ; is there a packet available ?
1482 005242 001417 BEQ 40$ ; branch if yes
1483 005244 ERR.FILLIN 7,ERR7
005244 012737 000002 002202 MOV #2,ERRTYP ; assume hard error for now
005252 012737 000007 002204 MOV #7,ERRNBR ; get error 7
005260 012737 003541 002206 MOV #ERR7,ERRMSG ; get error ERR7
005266 012737 004302 002210 MOV #ERRCODE,ERRBLK ; get error routine
1484 005274 ERROR
1485 005276 ERR.RETURN
005276 000261 SEC
005300 000207 RETURN
1486
1487 005302 004737 004512 40$: CALL POLLCRY ; poll for CRY
1488 005306 103003 BCC 50$ ; branch if OK
1489 005310 ERROR
1490 005312 ERR.RETURN ; return if error
005312 000261 SEC
005314 000207 RETURN
1491 005316 012777 003000 174706 50$: MOV #PI.ACK,@CURCSR ; send status ack
1492
1493 005324 004737 004512 60$: CALL POLLCRY ; poll for CRY
1494 005330 103017 BCC 70$ ; branch if OK
1495 005332 ERR.FILLIN 9.,ERR9 ; error
005332 012737 000002 002202 MOV #2,ERRTYP ; assume hard error for now
005340 012737 000011 002204 MOV #9.,ERRNBR ; get error 9.
005346 012737 003652 002206 MOV #ERR9,ERRMSG ; get error ERR9
005354 012737 004302 002210 MOV #ERRCODE,ERRBLK ; get error routine
1496 005362 ERROR
1497 005364 ERR.RETURN
005364 000261 SEC
005366 000207 RETURN
1498
1499 005370 122737 000005 002667 70$: CMPB #DI.ERP,STABUF+1 ; error status packet ?
1500 005376 001410 BEQ 75$ ; branch if yes
1501 005400 122737 000176 002667 CMPB #DI.ERS,STABUF+1 ; error status packet ?
1502 005406 001404 BEQ 75$ ; branch if yes
1503 005410 122737 000177 002667 CMPB #DI.ERL,STABUF+1 ; error status packet ?
1504 005416 001017 BNE 80$ ; branch if not
1505 005420 75$: ERR.FILLIN 8.,ERR8
    
```

GLOBAL SUBROUTINES SECTION

	005420	012737	000002	002202	MOV	#2,ERRTYP	; assume hard error for now
	005426	012737	000010	002204	MOV	#8.,ERRNBR	; get error 8.
	005434	012737	003611	002206	MOV	#ERR8,ERRMSG	; get error ERR8
	005442	012737	004302	002210	MOV	#ERRCODE,ERRBLK	; get error routine
1506	005450				ERROR		
1507	005452				ERR.RETURN		
	005452	000261			SEC		
	005454	000207			RETURN		
1508							
1509	005456				80\$:	OK.RETURN	
	005456	000241				CLC	
	005460	000207				RETURN	
1510							

GLOBAL SUBROUTINES SECTION

```

1512 ;**
1513 ; FUNCTIONAL DESCRIPTION:
1514 ; SUBROUTINE TO GET A STATUS PACKET FROM THE VSV21 INTO STATUS BUFFER.
1515 ; THIS INCLUDES BOTH PROGRAMMED I/O AND DMA MODES.
1516 ; THIS MODULE IS VERY SIMILAR TO GETST1 BUT IS CALLED AFTER A COMMAND IS SENT
1517 ; TO THE VSV21. -----
1518
1519 ; INPUTS:
1520 ; NONE
1521
1522 ; IMPLICIT INPUTS:
1523 ; DMASET - DMA ENABLED FLAG
1524 ; STABUF - STATUS BUFFER
1525 ; CURCSR - CSR ADDRESS
1526 ; L$TEST - CURRENT TEST
1527
1528 ; OUTPUTS:
1529 ; CONDITION CODE - CLEAR = SUCCESS
1530 ; SET = FAILURE
1531
1532 ; IMPLICIT OUTPUTS:
1533 ; ERROR BLOCK - FILLED IN ON FAILURE
1534 ; STABUF - STATUS BUFFER
1535
1536 ; SUBORDINATE ROUTINES USED:
1537 ; POLL
1538 ; POLLCRY
1539
1540 ; FUNCTIONAL SIDE EFFECTS:
1541 ; NONE
1542
1543 ; CALLING SEQUENCE:
1544 ; CALL GETST2 ;GO TO ROUTINE
1545 ; BCS ERROR ;CARRY SET IF ROUTINE HAD ERROR
1546 ;--
1547
1548 005462 GETST2::
1549 005462 005764 000002 TST DMASET(R4) ; DMA enabled ?
1550 005466 001060 BNE 30$ ; branch if yes
1551
1552 ;DMA not enabled - get status packet via parameter register
1553
1554 ; special processing if ERR set - only one word in status packet
1555 005470 032777 040000 174534 BIT #VS.ERR,@CURCSR ; ERR ?
1556 005476 001407 BEQ 5$ ; branch if no
1557 005500 012737 000001 002666 MOV #1,STABUF ; pretend status packet has a header
1558 005506 017737 174522 002670 MOV @CURPAR,STABUF+2 ; get rest of packet
1559 005514 000444 BR 25$
1560
1561 005516 017737 174512 002666 5$: MOV @CURPAR,STABUF ; get status header word
1562 005524 113701 002666 MOVB STABUF,R1 ; get number of following status words
1563 005530 001436 BEQ 25$ ; branch if none
1564 005532 012702 002670 MOV #<STABUF+2>,R2 ; get address of where to put them
1565 005536 004737 004314 10$: CALL POLL ; poll
1566 005542 103003 BCC 20$
1567 005544 ERROR
1568 005546 ERR.RETURN

```

## GLOBAL SUBROUTINES SECTION

```

005546 000261 SEC
005550 000207 RETURN
1569 005552 032777 020000 174452 20$: BIT #VS.PRY,@CURCSR ; PRY ?
1570 005560 001017 BNE 22$ ; branch if yes
1571 005562 ERR.FILLIN 6,ERR6
005562 012737 000002 002202 MOV #2,ERRTYP ; assume hard error for now
005570 012737 000006 002204 MOV #6,ERRNBR ; get error 6
005576 012737 003465 002206 MOV #ERR6,ERRMSG ; get error ERR6
005604 012737 004302 002210 MOV #ERRCODE,ERRBLK ; get error routine
1572 005612 ERROR
1573 005614 ERR.RETURN
005614 000261 SEC
005616 000207 RETURN
1574 005620 017722 174410 22$: MOV @CURPAR,(R2)+ ; get a status word
1575 005624 077134 SOB R1,10$ ; again
1576 005626 000434 25$: BR 60$
1577
1578 ;DMA enabled - status packet is written straight to status buffer
1579
1580 005630 022777 100000 174376 30$: CMP #PI.AVA,@CURPAR ; is there a packet available ?
1581 005636 001417 BEQ 40$ ; branch if yes
1582 005640 ERR.FILLIN 7,ERR7
005640 012737 000002 002202 MOV #2,ERRTYP ; assume hard error for now
005646 012737 000007 002204 MOV #7,ERRNBR ; get error 7
005654 012737 003541 002206 MOV #ERR7,ERRMSG ; get error ERR7
005662 012737 004302 002210 MOV #ERRCODE,ERRBLK ; get error routine
1583 005670 ERROR
1584 005672 ERR.RETURN
005672 000261 SEC
005674 000207 RETURN
1585
1586 005676 004737 004512 40$: CALL POLLCRY ; poll for CRY
1587 005702 103003 BCC 50$ ; branch if OK
1588 005704 ERROR
1589 005706 ERR.RETURN ; return if error
005706 000261 SEC
005710 000207 RETURN
1590 005712 012777 003000 174312 50$: MOV #PI.ACK,@CURCSR ; send status ack
1591
1592 005720 004737 004512 60$: CALL POLLCRY ; poll for CRY
1593 005724 103017 BCC 70$ ; branch if OK
1594 005726 ERR.FILLIN 9,ERR9 ; error
005726 012737 000002 002202 MOV #2,ERRTYP ; assume hard error for now
005734 012737 000011 002204 MOV #9,ERRNBR ; get error 9.
005742 012737 003652 002206 MOV #ERR9,ERRMSG ; get error ERR9
005750 012737 004302 002210 MOV #ERRCODE,ERRBLK ; get error routine
1595 005756 ERROR
1596 005760 ERR.RETURN
005760 000261 SEC
005762 000207 RETURN
1597
1598 005764 122737 000005 002667 70$: CMPB #DI.ERP,STABUF+1 ; error status packet ?
1599 005772 001414 BEQ 75$ ; branch if yes
1600 005774 122737 000176 002667 CMPB #DI.ERS,STABUF+1 ; error status packet ?
1601 006002 001410 BEQ 75$ ; branch if yes
1602 006004 022737 000003 002114 CMP #3,L$TEST ; test 3 (error expected) ?
1603 006012 001423 BEQ 80$ ; branch if yes

```

GLOBAL SUBROUTINES SECTION

```

1604 006014 122737 000177 002667      CMPB    #DI.ERL,STABUF+1      ; error status packet ?
1605 006022 001017                      BNE     80$                   ; branch if not
1606 006024                      75$:  ERR.FILLIN      8.,ERR8
      006024 012737 000002 002202      MOV     #2,ERRTYP            ; assume hard error for now
      006032 012737 000010 002204      MOV     #8.,ERRNBR          ; get error 8.
      006040 012737 003611 002206      MOV     #ERR8,ERRMSG        ; get error ERR8
      006046 012737 004302 002210      MOV     #ERRCODE,ERRBLK    ; get error routine
1607 006054      ERROR
1608 006056      ERR.RETURN
      006056 000261      SEC
      006060 000207      RETURN
1609
1610 006062                      80$:  OK.RETURN
      006062 000241      CLC
      006064 000207      RETURN
1611

```

GLOBAL SUBROUTINES SECTION

```

1613      ;**
1614      ; FUNCTIONAL DESCRIPTION:
1615      ;   SUBROUTINE TO WRITE DATA USING DMA.
1616
1617      ; INPUTS:
1618      ;   NONE
1619
1620      ; IMPLICIT INPUTS:
1621      ;   COMBUF           - COMMAND BUFFER CONTAINING :
1622      ;                                     DMA BUFFER ADDRESS IN COMBUF+4
1623      ;                                     DMA BUFFER LENGTH IN COMBUF+6
1624
1625      ; OUTPUTS:
1626      ;   CONDITION CODE - CLEAR = SUCCESS
1627      ;                   - SET   = FAILURE
1628
1629      ; IMPLICIT OUTPUTS:
1630      ;   NONE
1631
1632      ; SUBORDINATE ROUTINES USED:
1633      ;   DOCOM
1634
1635      ; FUNCTIONAL SIDE EFFECTS:
1636      ;   NONE
1637
1638      ; CALLING SEQUENCE:
1639      ;   MOV     #BUFADD,COMBUF+4      ;GET DMA BUFFER ADDRESS
1640      ;   MOV     #BUFLEN,COMBUF+6     ;GET DMA BUFFER LENGTH
1641      ;   CALL    WRAMS                 ;GO TO ROUTINE
1642      ;   BCS     ERROR                 ;CARRY SET IF ROUTINE HAD ERROR
1643      ;--
1644 006066 012737 002104 002616 WRAMS:: MOV     #DI.WRA,COMBUF      ; load write ram command code protocol.
1645 006074 005037 002620          CLR     COMBUF+2          ; increment address to bottom 16 bits
1646 006100 012737 000170 002626          MOV     #120.,COMBUF+8.    ; page number in ram of where to write.
1647
1648 006106          DO.COMMAND      PI.CMA,1$      ; do the DMA
1649 006106 012737 002400 002614          MOV     #PI.CMA,COMAND
1650 006114 012737 002734 002216          MOV     #1500.,POLCNT
1651 006122 004737 004572          CALL    DOCOM
1652 006126 103000          BCC     1$
1653 006130 000207          1$:   RETURN          ; return failed

```



GLOBAL SUBROUTINES SECTION

```

1652      ;**
1653      ; FUNCTIONAL DESCRIPTION:
1654      ;   SUBROUTINE TO READ DATA USING DMA.
1655
1656      ; INPUTS:
1657      ;   NONE
1658
1659      ; IMPLICIT INPUTS:
1660      ;   COMBUF           - COMMAND BUFFER CONTAINING :
1661      ;                                     DMA BUFFER ADDRESS IN COMBUF+4
1662      ;                                     DMA BUFFER LENGTH IN COMBUF+6
1663
1664      ; OUTPUTS:
1665      ;   CONDITION CODE - CLEAR = SUCCESS
1666      ;                                     - SET   = FAILURE
1667
1668      ; IMPLICIT OUTPUTS:
1669      ;   DMA BUFFER CONTAINING THE DMA'D DATA
1670      ;   CORRUPTS REGISTERS R1,R2
1671
1672      ; SUBORDINATE ROUTINES USED:
1673      ;   DOCOM
1674
1675      ; FUNCTIONAL SIDE EFFECTS:
1676      ;   NONE
1677
1678      ; CALLING SEQUENCE:
1679      ;   MOV     #BUFADD,COMBUF+4      ;GET DMA BUFFER ADDRESS
1680      ;   MOV     #BUFLN,COMBUF+6      ;GET DMA BUFFER LENGTH
1681      ;   CALL    RRAMS                 ;GO TO ROUTINE
1682      ;   BCS     ERROR                 ;CARRY SET IF ROUTINE HAD ERROR
1683      ;--
1684
1685      RRAMS::
1686      ;clear read buffer
1687      MOV     COMBUF+4,R1              ; get address of read buffer
1688      MOV     COMBUF+6,R2              ; get length in bytes to clear
1689      10$:   CLRB    (R1)+              ; clear read buffer
1690      SOB     R2,10$                  ; next byte
1691
1692      ;read into read buffer
1693      MOV     #DI.RRA,COMBUF           ; load write ram command code protocol.
1694      CLR     COMBUF+2                 ; increment to bottom 16 bits
1695      MOV     #120.,COMBUF+8.          ; page number in ram of where to read
1696
1697      DO.COMMAND      PI.CMA,20$      ; do the DMA
1698      MOV     #PI.CMA,COMAND
1699      MOV     #1500.,POLCNT
1700      CALL    DOCOM
1701      BCC     20$
1702      20$:   RETURN                    ; return failed

```

## GLOBAL SUBROUTINES SECTION

```

1700
1701      ;**
1702      ; FUNCTIONAL DESCRIPTION:
1703      ;   SUBROUTINE TO COMPARE DATA WRITTEN BY WRAMS WITH DATA READ BACK BY RRAMS.
1704
1705      ; INPUTS:
1706      ;   NONE
1707
1708      ; IMPLICIT INPUTS:
1709      ;   DMAOBUF      - ASSUMED OUTPUT BUFFER USED BY A PREVIOUS CALL TO WRAMS
1710      ;   DMAIBUF      - ASSUMED INPUT BUFFER USED BY A PREVIOUS CALL TO RRAMS
1711      ;   DMALEN       - LENGTH OF ABOVE BUFFERS
1712
1713      ; OUTPUTS:
1714      ;   CONDITION CODE - CLEAR = SUCCESS
1715      ;                   - SET   = FAILURE
1716
1717      ; IMPLICIT OUTPUTS:
1718      ;   CORRUPTS REGISTERS R1,R2
1719
1720      ; SUBORDINATE ROUTINES USED:
1721      ;   NONE
1722
1723      ; FUNCTIONAL SIDE EFFECTS:
1724      ;   NONE
1725
1726      ; CALLING SEQUENCE:
1727      ;   CALL   CRAMS      ;GO TO ROUTINE
1728      ;   BCS   ERROR      ;CARRY SET IF ROUTINE HAD ERROR
1729      ;--
1730 006212 012701 002762  CRAMS:: MOV    #DMAOBUF,R1      ; get start address of data
1731 006216 012702 003036      MOV    #DMAIBUF,R2      ; get start address of data read back
1732 006222 012703 000054      MOV    #DMALEN,R3      ; get length in bytes
1733 006226 122122          10$:  CMPB  (R1)+,(R2)+      ; are we o.k
1734 006230 001417          BEQ   20$              ; branch if yes
1735 006232          ERR.FILLIN 11.,ERR11 ; error return if not
1736 006232 012737 000002 002202      MOV    #2,ERRPTYP      ; assume hard error for now
1737 006240 012737 000013 002204      MOV    #11.,ERRNBR     ; get error 11.
1738 006246 012737 003753 002206      MOV    #ERR11,ERRMSG   ; get error ERR11
1739 006254 012737 004302 002210      MOV    #ERRCODE,ERRBLK ; get error routine
1740 006262          ERROR
1741 006264          ERR.RETURN
1742 006264 000261          SEC
1743 006266 000207          RETURN
1744 005270 077322          20$: SOB    R3,10$          ; branch if we have not finished
1745 006272          OK.RETURN
1746 006272 000241          CLC
1747 006274 000207          RETURN

```

GLOBAL SUBROUTINES SECTION

```

1742
1743 ; **
1744 ; FUNCTIONAL DESCRIPTION:
1745 ; SUBROUTINE TO SET UP TEST MASK FOR EVERY MICRODIAGNOSTIC TEST.
1746
1747 ; INPUTS:
1748 ; NONE
1749
1750 ; IMPLICIT INPUTS:
1751 ; L$TEST - CURRENT TEST
1752
1753 ; OUTPUTS:
1754 ; NONE
1755
1756 ; IMPLICIT OUTPUTS:
1757 ; CORRUPTS REGISTERS R1,R2
1758 ; PBLOC+2 - LOCATION TO STORE TEST MASK ( TEMPORARY )
1759 ; TMASKS - LOCATION TO STORE TEST MASK ( PERMANENT )
1760
1761 ; SUBORDINATE ROUTINES USED:
1762 ; NONE
1763
1764 ; FUNCTIONAL SIDE EFFECTS:
1765 ; NONE
1766
1767 ; CALLING SEQUENCE:
1768 ; CALL TSTMSK ;GO TO ROUTINE
1769 ;--
1770 TSTMSK::
1771 MOV L$TEST,R2 ; get current test ( 1 - MAXTST )
1772 DEC R2 ; Get offset from 0
1773 ASL R2 ; in bytes.
1774 ADD R4,R2 ; Get offset from FLAGS
1775 CALL TSTONCE ; First time through this test ?
1776 BNE 40$ ; branch if not
1777 MOV #<1*400>,PBLOC+2 ; test count = 1 ( top byte )
1778 RFLAGS R1 ; get DRS operator flag settings
1779 BIT #HOE,R1 ; HOE bit set ?
1780 BEQ 10$ ; branch if not
1781 BISB #BIT6,PBLOC+2 ; set the HOE bit
1782 BIT #LOE,R1 ; LOE bit set ?
1783 BEQ 30$ ; branch if not
1784 PRINTF #TM1 ; print warning message
1785 MANUAL ; manual allowed ?
1786 BNCOMPLETE 20$ ; no - do same as op answer yes
1787 GMANIL TM2,TEMP1,177777,NO ; op wish LOE set ?
1788 TST TEMP1
1789 BEQ 30$ ; branch if no
1790 BISB #BIT7,PBLOC+2 ; set the LOE bit
1791 MOV PBLOC+2,TMASKS(R2) ; save the mask
1792 MOV TMASKS(R2),PBLOC+2 ; copy saved mask into parameter block
1793 RETURN
1794 006444 045 116 045 TM1: .ASCIZ /*N%A** AN ON-BOARD ERROR WILL HANG THIS TEST INDEFINITELY **/
1795 006541 104 125 105 TM2: .ASCIZ /DUE TO ON-BOARD LOE FLAG SET - DO YOU WISH IT SET ?/
1796
1797 .EVEN
    
```

GLOBAL SUBROUTINES SECTION

```

1799
1800 ; **
1801 ; FUNCTIONAL DESCRIPTION:
1802 ; SUBROUTINE TO MOVE MICRODIAGNOSTIC PARAMETERS INTO THE PARAMETER REGISTER.
1803 ; INPUTS:
1804 ; NONE
1805
1806 ; IMPLICIT INPUTS:
1807 ; PBLOC - BUFFER CONTAINING MICRODIAGNOSTIC PARAMETERS
1808 ; CURPAR - CURRENT PARAMETER REGISTER ADDRESS
1809
1810 ; OUTPUTS:
1811 ; CONDITION CODE - CLEAR = SUCCESS
1812 ; - SET = FAILURE
1813
1814 ; IMPLICIT OUTPUTS:
1815 ; CORRUPTS REGISTERS R1,R2
1816
1817 ; SUBORDINATE ROUTINES USED:
1818 ; POLLCRY
1819
1820 ; FUNCTIONAL SIDE EFFECTS:
1821 ; NONE
1822
1823 ; CALLING SEQUENCE:
1824 ; CALL MVAR ;GO TO ROUTINE
1825 ; BCS ERROR ;CARRY SET IF ROUTINE HAD ERROR
1826 ;--
1827
1828 006626 012701 000004 MVAR:: MOV #4,R1 ; get length to move
1829 006632 012702 002222 MOV #PBLOC,R2 ; get start address of Pbloc
1830 006636 SET.POLCNT
1831 006636 012737 002734 002216 MOV #1500.,POLCNT
1832 006644 004737 004512 10$: CALL POLLCRY ; wait for SVL CRY
1833 006650 103003 BCC 20$ ; branch if OK
1834 006652 ERROR
1834 006654 ERR.RETURN
1834 006654 000261 SEC
1834 006656 000207 RETURN
1835 006660 012277 173350 20$: MOV (R2)+,@CURPAR ; move parameter into parameter reg
1836 006664 077111 SOB R1,10$ ; next
1837 006666 OK.RETURN
1837 006666 000241 CLC
1837 006670 000207 RETURN
1838

```

GLOBAL SUBROUTINES SECTION

```

1840 ;**
1841 ; FUNCTIONAL DESCRIPTION:
1842 ;   SUBROUTINE TO INVOKE A MICRODIAGNOSTIC
1843
1844 ; INPUTS:
1845 ;   NONE
1846
1847 ; IMPLICIT INPUTS:
1848 ;   POLCNT           - POLL COUNT
1849 ;   PBLOC            - BUFFER FOR MICRODIAGNOSTIC PARAMETERS
1850 ;   STABUF          - STATUS BUFFER FOR RETURNS FROM VSV21
1851
1852 ; OUTPUTS:
1853 ;   CONDITION CODE  - CLEAR = SUCCESS
1854 ;                   - SET   = FAILURE
1855
1856 ; IMPLICIT OUTPUTS:
1857 ;   NONE
1858
1859 ; SUBORDINATE ROUTINES USED:
1860 ;   DOCOM
1861
1862 ; FUNCTIONAL SIDE EFFECTS:
1863 ;   NONE
1864
1865 ; CALLING SEQUENCE:
1866 ;   CALL   DOMIC           ;GO TO ROUTINE
1867 ;   BCS   ERROR           ;CARRY SET IF ROUTINE HAD ERROR
1868 ;--
1869
1870 006672 012737 002000 002614 DOMIC:: MOV    #PI.TST,COMAND
1871 006700 032737 000200 002224          BIT    #BIT7,PBLOC+2          ; LOE ?
1872 006706 001004          BNE    10$                ; branch if yes
1873 006710          SET.POLCNT                ; set ordinary timeout
1874 006710 012737 002734 002216          MOV    #1500.,POLCNT
1875 006716 000403          BR     20$
1876 006720 012737 177777 002216 10$:  MOV    #-1,POLCNT          ; set infinite timeout
1877 006726 004737 004572 20$:  CALL   DOCOM                ; invoke micro diagnostic.
1878 006732 103001          BCC   30$
1879 006734 000207          RETURN
1880 006736 122737 000011 002667 30$:  CMPB   #DI.MIC,STABUF+1      ; correct status return ?
1881 006744 001417          BEQ    40$                ; branch if yes
1882 006746          ERR.FILLIN    12.,ERR12          ; error if no
1883 006746 012737 000002 002202          MOV    #2,ERRTYP          ; assume hard error for now
1884 006754 012737 000014 002204          MOV    #12.,ERRNBR       ; get error 12.
1885 006762 012737 004000 002206          MOV    #ERR12,ERRMSG     ; get error ERR12
1886 006770 012737 004302 002210          MOV    #ERRCODE,ERRBLK  ; get error routine
1887 006776          ERROR
1888 007000          ERR.RETURN
1889 007000          SEC
1890 007002          RETURN
1891 007004 005737 002670 40$:  TST    STABUF+2            ; test successful ?
1892 007010 001417          BEQ    50$                ; branch if YES
1893 007012          ERR.FILLIN    13.,ERR13          ; error if no
1894 007012 012737 000002 002202          MOV    #2,ERRTYP          ; assume hard error for now
1895 007020 012737 000015 002204          MOV    #13.,ERRNBR       ; get error 13.
1896 007026 012737 004065 002206          MOV    #ERR13,ERRMSG     ; get error ERR13

```

GLOBAL SUBROUTINES SECTION

	007034	012737	004302	002210		MOV	#ERRCODE,ERRBLK ; get error routine
1887	007042					ERPUR	
1888	007044					ERR.RETURN	
	007044	000261				SEC	
	007046	000207				RETURN	
1889	007050				50\$:	OK.RETURN	
	007050	000241				C' C	
	007052	000207				RETURN	
1890							

GLOBAL SUBROUTINES SECTION

```

1892      ;++
1893      ; FUNCTIONAL DESCRIPTION:
1894      ;   SUBROUTINE TO MARK CURRENT TEST AS HAVING BEEN DONE AT LEAST ONCE
1895
1896      ; INPUTS:
1897      ;   NONE
1898
1899      ; IMPLICIT INPUTS:
1900      ;   L$TEST          - CURRENT TEST
1901
1902      ; OUTPUTS:
1903      ;   NONE
1904
1905      ; IMPLICIT OUTPUTS:
1906      ;   CORRUPTS REGISTERS R0,R1
1907      ;   FTHRU          - FLAG BITS INDICATING IF TESTS HAVE BEEN DONE ONCE
1908
1909      ; SUBORDINATE ROUTINES USED:
1910      ;   NONE
1911
1912      ; FUNCTIONAL SIDE EFFECTS:
1913      ;   NONE
1914
1915      ; CALLING SEQUENCE:
1916      ;   CALL   MRKONCE          ;GO TO ROUTINE
1917      ;--
1918 007054 MRKONCE::
1919 007054 013700 002114      MOV     L$TEST,R0          ; get current test
1920 007060 010401           MOV     R4,R1             ; get address of flags
1921 007062 020027 000017      CMP     R0,#15.         ; test 1-15 ?
1922 007066 003404           BLE     10$              ; branch if yes
1923 007070 162700 000017      SUB     #15.,R0         ; get in range for a test > 15
1924 007074 062701 000002      ADD     #2,R1           ; point to second flag word (test > 15)
1925 007100 006300           10$:  ASL     R0           ; get offset from start of BITS
1926 007102 056061 002244 000004  BIS     BITS(R0),FTHRU(R1) ; set appropriate flag bit
1927 007110 000207           RETURN

```

GLOBAL SUBROUTINES SECTION

```

1929
1930 ;++
1931 ; FUNCTIONAL DESCRIPTION:
1932 ;   SUBROUTINE TO TEST IF CURRENT TEST HAS BEEN DONE AT LEAST ONCE
1933 ; INPUTS:
1934 ;   NONE
1935
1936 ; IMPLICIT INPUTS:
1937 ;   L$TEST           - CURRENT TEST
1938 ;   FTHRU           - FLAG BITS INDICATING IF TESTS HAVE BEEN DONE ONCE
1939
1940 ; OUTPUTS:
1941 ;   Z BIT           - SET INDICATES TEST HAS BEEN DONE AT LEAST ONCE
1942 ;                   - CLEAR INDICATES TEST HAS NOT BEEN DONE
1943
1944 ; IMPLICIT OUTPUTS:
1945 ;   CORRUPTS REGISTERS R0,R1
1946
1947 ; SUBORDINATE ROUTINES USED:
1948 ;   NONE
1949
1950 ; FUNCTIONAL SIDE EFFECTS:
1951 ;   NONE
1952
1953 ; CALLING SEQUENCE:
1954 ;   CALL   TSTONCE           ;GO TO ROUTINE
1955 ;   BNE   ATLEASTONCE       ;BRANCH IF TEST DONE AT LEAST ONCE
1956 ;--
1957
1958 007112
1959 007112 013700 002114
1960 007116 010401
1961 007120 020027 000017
1962 007124 003404
1963 007126 162700 000017
1964 007132 062701 000002
1965 007136 006300
1966 007140 036061 002244 000004
1967 007146 000207

TSTONCE::
      MOV     L$TEST,R0           ; get current test
      MOV     R4,R1              ; get address of flags
      CMP     R0,#15.            ; test 1-15 ?
      BLE     10$                ; branch if yes
      SUB     #15.,R0            ; get in range for a test > 15
      ADD     #2,R1              ; point to second flag word (test > 15)
10$:  ASL     R0                  ; get offset from start of BITS
      BIT     BITS(R0),FTHRU(R1) ; test appropriate flag bit
      RETURN

```



GLOBAL SUBROUTINES SECTION

```

1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999 007150
2000 007150 005737 002242
2001 007154 001402
2002 007156
    007156 000261
    007160 000207
2003 007162 013701 002114
2004 007166 005301
2005 007170 006301
2006 007172 005701
2007 007174 001414
2008
2009
2010
2011 007176 005764 000000
2012 007202 001006
2013 007204
2014 007214
    007214 000261
    007216 000207
2015
2016
2017
2018 007220 016137 007232 002222
2019
2020 007226
    007226 000241

```

```

; **
; FUNCTIONAL DESCRIPTION:
;   SUBROUTINE TO BE CALLED AT START OF EACH TEST TO PERFORM INITIALISATION.

; INPUTS:
;   NONE

; IMPLICIT INPUTS:
;   L$TEST      - CURRENT TEST
;   T1DONE     - TEST 1 DONE FLAG

; OUTPUTS:
;   CONDITION CODE - CLEAR = SUCCESS
;                   - SET   = FAILURE

; IMPLICIT OUTPUTS:
;   CORRUPTS REGISTER R1
;   PBLOC       - FILLS IN CORRESPONDING MICRODIAGNOSTIC TEST NUMBER

; SUBORDINATE ROUTINES USED:
;   NONE

; FUNCTIONAL SIDE EFFECTS:
;   NONE

; CALLING SEQUENCE:
;   CALL STTEST      ;GO TO ROUTINE
;   BCS  ERROR      ;BRANCH IF ERROR
;--

STTEST::
    TST     NOTEST          ; can test be done ?
    BEQ     5$             ; branch if yes
    ERR.RETURN            ; error return if not
    SEC
    RETURN
5$:  MOV     L$TEST,R1      ; get current test
    DEC     R1             ; Get in range 0 - 40
    ASL     R1             ; for offset calculations.
    TST     R1             ; is it test 1 ?
    BEQ     20$           ; exit if yes

; Test is not 1 - ensure that test 1 has already been done

    TST     T1DONE(R4)    ; test 1 done ?
    BNE     10$          ; branch if yes
    ERRHRD  1,ERR1,ERRCODE ; error exit if not
    ERR.RETURN
    SEC
    RETURN

; Set up microdiagnostic test number in PBLOC

10$: MOV     MICNUM(R1),PBLOC ; get microdiagnostic test number

20$: OK.RETURN
    CLC

```

GLOBAL SUBROUTINES SECTION

```
007230 000207 RETURN
2021
2022 ;Microdiagnostic test number versus test. ( -1 is null )
2023
2024 007232 177777 177777 177777 MICNUM: .WORD -1.,-1.,-1.,0,1,2,3,4,5,6,7
2025 007260 000010 000011 000012 .WORD 8.,9.,10.,11.,11.,11.,12.,13.,-1.
2026
```

GLOBAL SUBROUTINES SECTION

```
2028          .TITLE MISCELLANEOUS SECTIONS
2029          .SBTTL  PROTECTION TABLE
2030
2031          ;++
2032          ; THIS TABLE IS USED BY THE RUNTIME SERVICES
2033          ; TO PROTECT THE LOAD MEDIA.
2034          ;--
2035
2036 007302          BGNPROT
2037
2038 007302 000000          0          ;OFFSET INTO P-TABLE FOR CSR ADDRESS
2039 007304 177777          -1         ;OFFSET INTO P-TABLE FOR MASSBUS ADDRESS
2040 007306 177777          -1         ;OFFSET INTO P-TABLE FOR DRIVE NUMBER
2041
2042 007310          ENDPROT
2043
```

## INITIALIZE SECTION

```

2058          .SBTTL INITIALIZE SECTION
2059
2060          ;++
2061          ; THE INITIALIZE SECTION CONTAINS THE CODING THAT IS PERFORMED
2062          ; AT THE BEGINNING OF EACH PASS.
2063          ;--
2064
2065 007310          BGNINIT
2066
2090 007310          READEF #EF.START          ; IF started by start THEN
2091 007316          BCOMPLETE 10$           ; do start coding
2092 007320          READEF #EF.RESTART       ; IF not started by restart THEN
2093 007326          BNCOMPLETE 20$          ; skip start coding
2094 007330 012700 002304 10$: MOV #FLAGS,R0 ; get address of FLAGS
2095 007334 012701 000140          MOV #<LENFLAGS*MAXUNIT>,R1 ; get length of FLAGS
2096 007340 005020          15$: CLR (R0)+   ; clear out FLAGS
2097 007342 077102          SOB R1,15$      ; next
2098 007344 005037 002242          CLR NOTEST ; say tests can be done
2099 007350          20$: READEF #EF.CONTINUE ; IF started by continue THEN
2100 007356          BCOMPLETE END          ; don't get P-table
2101 007360          READEF #EF.NEW          ; IF not a new pass THEN
2102 007366          BNCOMPLETE NEXT        ; skip setup
2103
2104 0C7370 012737 177777 002220          MOV #-1,LOGUNT ; initialise unit number
2105 007376 005237 002220          NEXT: INC LOGUNT ; next unit number
2106 007402 023737 002220 002012          CMP LOGUNT,L$UNIT ; IF passed operator's max unit number THEN
2107 007410 001437          BEQ ABORT       ; abort pass
2108 007412 023727 002220 000004          CMP LOGUNT,#MAXUNIT ; IF passed max unit number THEN
2109 007420 002033          BGE ABORT       ; abort pass (N.B. OPERATOR'S MAX COULD BE >)
2110 007422          GPHARD LOGUNT,R1      ; Get hardware P-table address
2111 007432          BNCOMPLETE NEXT        ; if not available get next
2112 007434 011137 002232          MOV (R1),CURCSR ; get current device address from P-table
2113 007440 062737 000002 002232          ADD #2,CURCSR ; add 2 to get current CSR
2114 007446 011137 002234          MOV (R1),CURPAR ; get current device address from P-table
2115 007452 062737 000004 002234          ADD #4,CURPAR ; add 4 to get current Parameter Register
2116 007460 016137 000002 002236          MOV 2(R1),CURVEC ; get current vector address
2117 007466 013701 002220          MOV LOGUNT,R1 ; get unit number
2118 007472 070127 000060          MUL #<LENFLAGS*2>,R1 ; get offset from start of FLAGS
2119 007476 062701 002304          ADD #FLAGS,R1 ; get true address of FLAGS for this unit
2120 007502 010104          MOV R1,R4     ; save in R4
2121 007504          END: EXIT INIT
2122 007510          ABORT: DOCLN
2123 007512          ENDINIT
2124
2125          .EVEN
2126

```

AUTODROP SECTION

2128  
2129  
2130  
2131  
2132  
2133  
2134  
2135  
2136  
2137 007514  
2138  
2145  
2146 007514

.SBTTL AUTODROP SECTION

;++  
; THIS CODE IS EXECUTED IMMEDIATELY AFTER THE INITIALIZE CODE IF  
; THE "ADR" FLAG WAS SET. THE UNIT(S) UNDER TEST ARE CHECKED TO  
; SEE IF THEY WILL RESPOND. THOSE THAT DON'T ARE IMMEDIATELY  
; DROPPED FROM TESTING.  
;--

BGNAUTO

ENDAUTO

CLEANUP CODING SECTION

```

2148          .SBTTL  CLEANUP CODING SECTION
2149
2150          ;**
2151          ; THE CLEANUP CODING SECTION CONTAINS THE CODING THAT IS PERFORMED
2152          ; AFTER THE HARDWARE TESTS HAVE BEEN PERFORMED.
2153          ;--
2154
2155 007516          BGNCLN
2156
2157 007516 023727 002220 000004      CMP    LOGUNT,#MAXUNIT      ; IF greater than max unit THEN
2158 007524 002011                    BGE    10$                  ; exit
2159 007526 013700 002232            MOV    CURCSR,R0          ; get current CSR
2160 007532 005040                    CLR    -(R0)              ; Write to the previous word (i.e. H/W
2161                                     ; register ) to reset VSV21.
2162
2163 007534 005064 000002            CLR    DMASET(R4)        ; say DMA disabled
2164 007540 005064 000000            CLR    T1DONE(R4)       ; say Test 1 not done
2165 007544 005037 002242            CLR    NOTEST           ; say tests can be done
2166
2167 007550          10$:  EXIT    CLN
2168
2169 007554          ENDCLN
2170

```

CLEANUP CODING SECTION

```

2172          .TITLE HARDWARE TESTS
2173
2174          .SBTTL TEST 1: VERIFY VSV21 PRESENCE
2175
2176
2177          ;**
2178          ; This test verifies that there is a VSV21 device at the current CSR.
2179          ; On the first pass it displays the version of the VSV21. It also disables
2180          ; interrupts.
2181          ;--
2182
2183 007556          ST.TEST
                CALL STTEST
                BCC 30000$
007556 004737 007150
007562 103002
2184
2185          ;SUBTEST 1 sends an interrupt mask to disable interrupts.
2186
2187 007570          BGNSUB
2188 007572 005064 000000          CLR T1DONE(R4) ; say that TEST1 has not been done
2189 007576          DO.COMMAND PI.IDS,10$ ; send over interrupt mask.
                MOV #PI.IDS,COMAND
                MOV #1500.,POLCNT
                CALL DOCOM
                BCC 10$
                ESCAPE TST
2190 007620          10$: INC T1DONE(R4) ; say TEST1 done so that if powerup
2191 007624 005264 000000          ; when request firmware id will get
2192          ; error
2193          ;
                ENDSUB
2194 007630
2195          ;SUBTEST 2 requests firmware ID and displays it
2196
2197
2198 007632          BGNSUB
2199 007634          DO.COMMAND PI.IDM,10$ ; request firmware id
                MOV #PI.IDM,COMAND
                MOV #1500.,POLCNT
                CALL DOCOM
                BCC 10$
                CLR T1DONE(R4) ; say that TEST1 has not been done
                ESCAPE TST
2200 007656 005064 000000          10$: INC T1DONE(R4) ; say TEST1 done
2201 007662          CALL TSTONCE ; first time through ?
                BNE 20$ ; branch if not
                CALL MRKONCE ; say not first time through
                SWAB STABUF+2 ; Rearrange
                SWAB STABUF+4 ; firmware ID
                SWAB STABUF+6 ; so that can be printed.
                SWAB STABUF+8. ; This assumes top byte zero.
2202 007666 005264 000000          20$: PRINTX #TS1,#<STABUF+2> ; Print firmware id.
2203 007672 004737 007112          PRINTX #TS2
2204 007676 001044          PRINTX #TS3
2205 007700 004737 007054          20$: ENDSUB
2206 007704 000337 002670          EXIT TST
2207 007710 000337 002672
2208 007714 000337 002674
2209 007720 000337 002676
2210 007724
2211 007750
2212 007770
2213 010010
2214
2215 010012
2216
2217 010016 045 116 045 TS1: .ASCIZ /%N%AFIRMWARE ID IS : %T/
2218 010046 045 116 062 TS2: .ASCIZ /%N2%ATESTS 4,13,14,16 AND 17 WILL ABORT WITH AN ERROR IF/
    
```

TEST 1: VERIFY VSV21 PRESENCE

```
2219 010137      045      116      045 TS3:  .ASCIZ /NAEXTERNAL LOOPBACK CONNECTORS ARE NOT ATTACHED TO PORTS 0-3/
2220
2221
2222
2223 010236      .EVEN
2224
                                ENDTST
```



TEST 1: VERIFY VSV21 PRESENCE

```

2226
2227           .SBTTL TEST 2: IN DEPTH Q22 BUS TEST
2228
2229
2230           ;++
2231           ; This test performs an in-depth test of the Q22 bus by means of exercising the
2232           ; DMA mechanism in both directions. It sets up the DMA protocol and must be
2233           ; done before other tests requiring DMA.
2234           ;--
2235
2236 010240           ST.TEST
                010240 004737 007150          CALL STTEST
                010244 103002                BCC 30001$
2237
2238           ;SUBTEST 1 moves DMA buffer descriptors through the parameter register
2239
2240 010252           BGNSUB
2241 010254 012701 002604          MOV #BUFDES,R1           ; get address of buffer descriptors
2242 010260 012702 000004          MOV #4,R2              ; length to move
2243
2244 010264           10$: SET.POLCNT
                010264 012737 002734 002216  MOV #1500.,POLCNT
2245 010272 004737 004512          CALL POLLCRY           ; wait for SVL CRY
2246 010276 103002                BCC 20$              ; branch if OK
2247 010300          ESCAPE TST
2248 010304 012177 171724          20$: MOV (R1)+,@CURPAR      ; move buffer desc. into parameter reg
2249 010310 077213                SOB R2,10$           ; next descriptor
2250 010312          ENDSUB
2251
2252           ;SUBTEST 2 does the buffer sent command
2253
2254 010314           BGNSUB
2255 010316          DO.COMMAND      PI.SEN,BUFOK      ; do buffers sent command
                010316 012737 001000 002614  MOV #PI.SEN,COMAND
                010324 012737 002734 002216  MOV #1500.,POLCNT
                010332 004737 004572          CALL DOCOM
                010336 103016                BCC BUFOK
2256 010340          ENDSUB
2257
2258           ;SUBTEST 3 does a buffers cancelled command if SUBTEST 2 failed
2259
2260 010342           BGNSUB
2261 010344          DO.COMMAND      PI.CAN,10$       ; do buffers cancelled command
                010344 012737 001001 002614  MOV #PI.CAN,COMAND
                010352 012737 002734 002216  MOV #1500.,POLCNT
                010360 004737 004572          CALL DOCOM
                010364 103000                BCC 10$
2262 010366          10$: ESCAPE TST
2263 010372          ENDSUB
2264
2265           ;SUBTEST 3 writes to RAM using DMA. This is SUBTEST 3 if 2 is successful.
2266
2267 010374           BUFOK: BGNSUB
2268 010376 005264 000002          INC DMASET(R4)         ; record DMA enabled
2269 010402          DO.WRAMS
                010402 012737 002762 002622  MOV #DMAOBUF,COMBUF+4 ; write to RAM
                010410 012737 000054 002624  MOV #DMALEN,COMBUF+6

```

TEST 2: IN DEPTH Q22 BUS TEST

```

010416 004737 006066          CALL  WRAMS
2270 010422 103002          BCC  10$          ; carry on if OK
2271 010424          ESCAPE TST
2272 010430          10$: ENDSUB
2273
2274          ;SUBTEST 4 reads from RAM using DMA.
2275
2276 010432          BGNSUB
2277 010434          DO.RRAMS          ; read from RAM
      010434 012737 003036 002622  MOV  #DMAIBUF,COMBUF+4
      010442 012737 000054 002624  MOV  #DMALEN,COMBUF+6
      010450 004737 006132          CALL  RRAMS
2278 010454 103002          BCC  10$          ; carry on if OK
2279 010456          ESCAPE TST
2280 010462          10$: ENDSUB
2281
2282          ;SUBTEST 5 compares data read back with data written
2283
2284 010464          BGNSUB
2285 010466 004737 006212  CALL  CRAMS          ; compare the patterns
2286 010472 103002          BCC  10$          ; carry on if OK
2287 010474          ESCAPE TST
2288 010500          10$: ENDSUB
2289
2290 010502          EXIT  TST
2291
2292          .EVEN
2293 010506          ENDTST
2294

```

TEST 2: IN DEPTH Q22 BUS TEST

```

2296
2297
2298
2299
2300
2301
2302
2303
2304 010510
      010510 004737 007150
      010514 103002
2305
2306
2307
2308 010522
2309 010524
      010524 012737 177777 002614
      010532 012737 002734 002216
      010540 004737 004572
      010544 103000
2310 010546 122737 000177 002667 10$:
2311 010554 001002
2312 010556
2313
2314 010562
2315 010612
2316 010614
2317 010620
2318
2319 010622

```

```

.SBTTL TEST 3: FORCED ERROR TEST

; ++
; This test sends an invalid command to the VSV21 to provoke an error packet.
; --

ST.TEST
CALL STTEST
BCC 30002$

;SUBTEST 1 sends an invalid command

BGNSUB
DO.COMMAND PI.RUB,10$ ; send a rubbish command
MOV #PI.RUB,COMAND
MOV #1500.,PGLCNT
CALL DOCOM
BCC 10$
CMPB #DI.ERL,STABUF+1 ; correct error return ?
BNE 20$ ; branch if not
EXIT TST ; received an error - exit ok

20$: ERR.FILLIN 14.,ERR14
ERROR
ESCAPE TST
ENDSUB

ENDTST

```

TEST 3: FORCED ERROR TEST

2321  
 2322  
 2323  
 2324  
 2325  
 2326  
 2327  
 2328  
 2329  
 2330  
 2331  
 2332  
 2333  
 2334 010624  
       010624 004737 007150  
       010630 103002  
 2335  
 2336 010636 004737 006276  
 2337 010642 004737 007054  
 2338  
 2339  
 2340  
 2341  
 2342 010646  
 2343 010650 004737 006626  
 2344 010654 103002  
 2345 010656  
 2346 010662  
 2347  
 2348  
 2349  
 2350 010664  
 2351 010666 004737 006672  
 2352 010672 103002  
 2353 010674  
 2354 010700  
 2355  
 2356 010702  
 2357

.SBTTL TEST 4: FULL ON BOARD TESTS

```

; **
; This test invokes full on-board VSV21 microdiagnostic tests.
; If the HOE flag is specified the VSV21 will halt on error.
; If the LOE flag is specified the operator will be given the choice on the
; first pass of whether he wishes the on-board tests to loop on error. If he
; selects yes the test may hang indefinitely as the on board will not return
; control to the host but will continue looping.
; --
    
```

```

ST.TEST
CALL STTEST
BCC 30003$
    
```

```

CALL TSTMSK ; get test mask for microdiagnostic
CALL MRKONCE ; mark test as entered at least once
    
```

```

;SUBTEST 1 moves the parameters for the microdiagnostics test through the
;parameter register
    
```

```

BGNSUB
CALL MVPAR ; move the parameters
BCC 10$
ESCAPE TST
10$: ENDSUB
    
```

```

;SUBTEST 2 invokes the microdiagnostic test
    
```

```

BGNSUB
CALL DOMIC
BCC 10$
ESCAPE TST
10$: ENDSUB

ENDTST
    
```

TEST 4: FULL ON BOARD TESTS

2359  
 2360  
 2361  
 2362  
 2363  
 2364  
 2365  
 2366  
 2367  
 2368  
 2369  
 2370  
 2371  
 2372 010704  
       010704 004737 007150  
       010710 103002  
 2373  
 2374 010716 004737 006276  
 2375 010722 004737 007054  
 2376  
 2377  
 2378  
 2379  
 2380 010726  
 2381 010730 004737 006626  
 2382 010734 103002  
 2383 010736  
 2384 010742  
 2385  
 2386  
 2387  
 2388 010744  
 2389 010746 004737 006672  
 2390 010752 103002  
 2391 010754  
 2392 010760  
 2393  
 2394 010762

.SBTTL TEST 5: ROM CHECKSUM TEST

```

; **
; This test invokes the on-board VSV21 ROM CHECKSUM test.
; If the HOE flag is specified the VSV21 will halt on error.
; If the LOE flag is specified the operator will be given the choice on the
; first pass of whether he wishes the on-board tests to loop on error. If he
; selects yes the test may hang indefinitely as the on board will not return
; control to the host but will continue looping.
; --
    
```

```

ST.TEST
CALL STTEST
BCC 30004$
    
```

```

CALL TSTMSK ; get test mask for microdiagnostic
CALL MRKONCE ; mark test as entered at least once
    
```

```

;SUBTEST 1 moves the parameters for the microdiagnostics test through the
;parameter register
    
```

```

BNSUB
CALL MVAR ; move the parameters
BCC 10$
ESCAPE TST
10$: ENDSUB
    
```

```

;SUBTEST 2 invokes the microdiagnostic test
    
```

```

BGNSUB
CALL DOMIC
BCC 10$
ESCAPE TST
10$: ENDSUB
ENDTST
    
```

TEST 5: ROM CHECKSUM TEST

2396  
 2397  
 2398  
 2399  
 2400  
 2401  
 2402  
 2403  
 2404  
 2405  
 2406  
 2407  
 2408  
 2409  
 2410  
 2411  
 2412  
 2413  
 2414  
 2415  
 2416  
 2417  
 2418  
 2419  
 2420  
 2421  
 2422  
 2423  
 2424  
 2425  
 2426  
 2427  
 2428  
 2429  
 2430  
 2431

010764  
 010764 004737 007150  
 010770 103002  
 010776 004737 006276  
 011002 004737 007054  
 011006  
 011010 004737 006626  
 011014 103002  
 011016  
 011022  
 011024  
 011026 004737 006672  
 011032 103002  
 011034  
 011040  
 011042

.SBTTL TEST 6: NVRAM CHECKSUM TEST

```

; **
; This test invokes the on-board VSV21 NVRAM CHECKSUM test.
; If the HOE flag is specified the VSV21 will halt on error.
; If the LOE flag is specified the operator will be given the choice on the
; first pass of whether he wishes the on-board tests to loop on error. If he
; selects yes the test may hang indefinitely as the on board will not return
; control to the host but will continue looping.
; --
    
```

```

ST.TEST
CALL STTEST
BCC 30005$

CALL TSTMSK ; get test mask for microdiagnostic
CALL MRKONCE ; mark test as entered at least once

;SUBTEST 1 moves the parameters for the microdiagnostics test through the
;parameter register

BGNSUB
CALL MVAR ; move the parameters
BCC 10$
ESCAPE TST
10$: ENDSUB

;SUBTEST 2 invokes the microdiagnostic test

BGNSUB
CALL DOMIC
BCC 10$
ESCAPE TST
10$: ENDSUB

ENDTST
    
```

TEST 6: NVRAM CHECKSUM TEST

2433  
 2434  
 2435  
 2436  
 2437  
 2438  
 2439  
 2440  
 2441  
 2442  
 2443  
 2444  
 2445  
 2446  
 2447  
 2448  
 2449  
 2450  
 2451  
 2452  
 2453  
 2454  
 2455  
 2456  
 2457  
 2458  
 2459  
 2460  
 2461  
 2462  
 2463  
 2464  
 2465  
 2466  
 2467  
 2468

011044  
 011044 004737 007150  
 011050 103002  
 011056 004737 006276  
 011062 004737 007054  
 011066  
 011070 004737 006626  
 011074 103002  
 011076  
 011102  
 011104  
 011106 004737 006672  
 011112 103002  
 011114  
 011120  
 011122

.SBTTL TEST 7: RAM TEST

```

; ++
; This test invokes the on-board VSV21 RAM test.
; If the HOE flag is specified the VSV21 will halt on error.
; If the LOE flag is specified the operator will be given the choice on the
; first pass of whether he wishes the on-board tests to loop on error. If he
; selects yes the test may hang indefinitely as the on board will not return
; control to the host but will continue looping.
; --
    
```

```

ST.TEST
CALL STTEST
BCC 30006$
    
```

```

CALL TSTMSK ; get test mask for microdiagnostic
CALL MRKONCE ; mark test as entered at least once
    
```

```

;SUBTEST 1 moves the parameters for the microdiagnostics test through the
;parameter register
    
```

```

BGNSUB
CALL MVPAR ; move the parameters
BCC 10$
ESCAPE TST
10$: ENDSUB
    
```

```

;SUBTEST 2 invokes the microdiagnostic test
    
```

```

BGNSUB
CALL DOMIC
BCC 10$
ESCAPE TST
10$: ENDSUB

ENDTST
    
```

TEST 7: RAM TEST

2470  
 2471  
 2472  
 2473  
 2474  
 2475  
 2476  
 2477  
 2478  
 2479  
 2480  
 2481  
 2482  
 2483  
 2484  
 2485  
 2486  
 2487  
 2488  
 2489  
 2490  
 2491  
 2492  
 2493  
 2494  
 2495  
 2496  
 2497  
 2498  
 2499  
 2500  
 2501  
 2502  
 2503  
 2504  
 2505

011124  
 011124 004737 007150  
 011130 103002  
 011136 004737 006276  
 011142 004737 007054  
 011146  
 011150 004737 006626  
 011154 103002  
 011156  
 011162  
 011164  
 011166 004737 006672  
 011172 103002  
 011174  
 011200  
 011202

.SBTTL TEST 8: RAM ADDRESSING TEST

```

; **
; This test invokes the on-board VSV21 RAM ADDRESSING test.
; If the HOE flag is specified the VSV21 will halt on error.
; If the LOE flag is specified the operator will be given the choice on the
; first pass of whether he wishes the on-board tests to loop on error. If he
; selects yes the test may hang indefinitely as the on board will not return
; control to the host but will continue looping.
; --
    
```

```

ST.TEST
CALL STTEST
BCC 30007$
    
```

```

CALL TSTMSK ; get test mask for microdiagnostic
CALL MRKONCE ; mark test as entered at least once
    
```

```

;SUBTEST 1 moves the parameters for the microdiagnostics test through the
;parameter register
    
```

```

BGNSUB
CALL MVAR ; move the parameters
BCC 10$
ESCAPE TST
10$: ENDSUB
    
```

```

;SUBTEST 2 invokes the microdiagnostic test
    
```

```

BGNSUB
CALL DOMIC
BCC 10$
ESCAPE TST
10$: ENDSUB
ENDTST
    
```



TEST 8: RAM ADDRESSING TEST

2507  
 2508  
 2509  
 2510  
 2511  
 2512  
 2513  
 2514  
 2515  
 2516  
 2517  
 2518  
 2519  
 2520  
 2521  
 2522  
 2523  
 2524  
 2525  
 2526  
 2527  
 2528  
 2529  
 2530  
 2531  
 2532  
 2533  
 2534  
 2535  
 2536  
 2537  
 2538  
 2539  
 2540  
 2541  
 2542

011204  
 011204 004737 007150  
 011210 103002  
 011216 004737 006276  
 011222 004737 007054  
 011226  
 011230 004737 006626  
 011234 103002  
 011236  
 011242  
 011244  
 011246 004737 006672  
 011252 103002  
 011254  
 011260  
 011262

.SBTTL TEST 9: 68K PROCESSOR TEST

```

; **
; This test invokes the on-board VSV21 68K PROCESSOR test.
; If the HOE flag is specified the VSV21 will halt on error.
; If the LOE flag is specified the operator will be given the choice on the
; first pass of whether he wishes the on-board tests to loop on error. If he
; selects yes the test may hang indefinitely as the on board will not return
; control to the host but will continue looping.
; --
    
```

```

ST.TEST
CALL STTEST
BCC 30008$
    
```

```

CALL TSTMSK ; get test mask for microdiagnostic
CALL MRKONCE ; mark test as entered at least once
    
```

```

;SUBTEST 1 moves the parameters for the microdiagnostics test through the
;parameter register
    
```

```

BGNSUB
CALL MVAR ; move the parameters
BCC 10$
ESCAPE TST
10$: ENDSUB
    
```

```

;SUBTEST 2 invokes the microdiagnostic test
    
```

```

BGNSUB
CALL DOMIC
BCC 10$
ESCAPE TST
10$: ENDSUB
ENDTST
    
```

TEST 9: 68K PROCESSOR TEST

2544  
 2545  
 2546  
 2547  
 2548  
 2549  
 2550  
 2551  
 2552  
 2553  
 2554  
 2555  
 2556  
 2557 011264  
 011264 004737 007150  
 011270 103002  
 2558  
 2559 011276 004737 006276  
 2560 011302 004737 007054  
 2561  
 2562  
 2563  
 2564  
 2565 011306  
 2566 011310 004737 006626  
 2567 011314 103002  
 2568 011316  
 2569 011322  
 2570  
 2571  
 2572  
 2573 011324  
 2574 011326 004737 006672  
 2575 011332 103002  
 2576 011334  
 2577 011340  
 2578  
 2579 011342

.SBTTL TEST 10: INTERNAL EXCEPTIONS TEST

```

; **
; This test invokes the on-board INTERNAL EXCEPTIONS test.
; If the MOE flag is specified the VSV21 will halt on error.
; If the LOE flag is specified the operator will be given the choice on the
; first pass of whether he wishes the on-board tests to loop on error. If he
; selects yes the test may hang indefinitely as the on board will not return
; control to the host but will continue looping.
; --
    
```

```

ST.TEST
CALL STTEST
BCC 30009$
    
```

```

CALL TSTMSK ; get test mask for microdiagnostic
CALL MRKONCE ; mark test as entered at least once
    
```

```

;SUBTEST 1 moves the parameters for the microdiagnostics test through the
;parameter register
    
```

```

BGNSUB
CALL MVAR ; move the parameters
BCC 10$
    
```

```

10$: ESCAPE TST
ENDSUB
    
```

```

;SUBTEST 2 invokes the microdiagnostic test
    
```

```

BGNSUB
CALL DOMIC
BCC 10$
ESCAPE TST
    
```

```

10$: ENDSUB
ENDTST
    
```

TEST 10: INTERNAL EXCEPTIONS TEST

2581  
 2582  
 2583  
 2584  
 2585  
 2586  
 2587  
 2588  
 2589  
 2590  
 2591  
 2592  
 2593  
 2594 011344  
 011344 004737 007150  
 011350 103002  
 2595  
 2596 011356 004737 006276  
 2597 011362 004737 007054  
 2598  
 2599  
 2600  
 2601  
 2602 011366  
 2603 011370 004737 006626  
 2604 011374 103002  
 2605 011376  
 2606 011402  
 2607  
 2608  
 2609  
 2610 011404  
 2611 011406 004737 006672  
 2612 011412 103002  
 2613 011414  
 2614 011420  
 2615  
 2616 011422

.SBTTL TEST 11: ACRCT INTERNAL TEST

```

; **
; This test invokes the on-board ACRCT INTERNAL test.
; If the HOE flag is specified the VSV21 will halt on error.
; If the LOE flag is specified the operator will be given the choice on the
; first pass of whether he wishes the on-board tests to loop on error. If he
; selects yes the test may hang indefinitely as the on board will not return
; control to the host but will continue looping.
; --
    
```

```

ST.TEST
CALL STTEST
BCC 30010$
    
```

```

CALL TSTMSK ; get test mask for microdiagnostic
CALL MRKONCE ; mark test as entered at least once
    
```

```

;SUBTEST 1 moves the parameters for the microdiagnostics test through the
;parameter register
    
```

```

BGNSUB
CALL MVAR ; move the parameters
BCC 10$
ESCAPE TST
10$: ENDSUB
    
```

```

;SUBTEST 2 invokes the microdiagnostic test
    
```

```

BGNSUB
CALL DOMIC
BCC 10$
ESCAPE TST
10$: ENDSUB
    
```

```

ENDTST
    
```

TEST 11: ACRCT INTERNAL TEST

.SBTTL TEST 12: ACRCT EXTERNAL TEST

2618  
 2619  
 2620  
 2621  
 2622  
 2623  
 2624  
 2625  
 2626  
 2627  
 2628  
 2629  
 2630  
 2631 011424  
       011424 004737 007150  
       011430 103002  
 2632  
 2633 011436 004737 006276  
 2634 011442 004737 007054  
 2635  
 2636  
 2637  
 2638  
 2639 011446  
 2640 011450 004737 006626  
 2641 011454 103002  
 2642 011456  
 2643 011462  
 2644  
 2645  
 2646  
 2647 011464  
 2648 011466 004737 006672  
 2649 011472 103002  
 2650 011474  
 2651 011500  
 2652  
 2653 011502

```

; **
; This test invokes the on-board VSV21 ACRCT EXTERNAL test.
; If the HOE flag is specified the VSV21 will halt on error.
; If the LOE flag is specified the operator will be given the choice on the
; first pass of whether he wishes the on-board tests to loop on error. If he
; selects yes the test may hang indefinitely as the on board will not return
; control to the host but will continue looping.
; --
    
```

```

ST.TEST
CALL STTEST
BCC 30011$

CALL TSTMSK ; get test mask for microdiagnostic
CALL MRKONCE ; mark test as entered at least once

;SUBTEST 1 moves the parameters for the microdiagnostics test through the
;parameter register

BGNSUB
CALL MVAR ; move the parameters
BCC 10$
ESCAPE TST
10$: ENDSUB

;SUBTEST 2 invokes the microdiagnostic test

BGNSUB
CALL DOMIC
BCC 10$
ESCAPE TST
10$: ENDSUB

ENDTST
    
```

TEST 12: ACRCT EXTERNAL TEST

2655  
 2656  
 2657  
 2658  
 2659  
 2660  
 2661  
 2662  
 2663  
 2664  
 2665  
 2666  
 2667 011504  
       011504 004737 007150  
       011510 103002  
 2668  
 2669 011516 004737 006276  
 2670 011522 004737 007054  
 2671  
 2672  
 2673  
 2674 011526 005037 002226  
 2675  
 2676  
 2677  
 2678  
 2679 011532  
 2680 011534 004737 006626  
 2681 011540 103002  
 2682 011542  
 2683 011546  
 2684  
 2685  
 2686  
 2687 011550  
 2688 011552 004737 006672  
 2689 011556 103002  
 2690 011560  
 2691 011564  
 2692  
 2693  
 2694  
 2695 011566 005237 002226  
 2696  
 2697  
 2698  
 2699  
 2700 011572  
 2701 011574 004737 006626  
 2702 011600 103002  
 2703 011602  
 2704 011606  
 2705  
 2706  
 2707  
 2708 011610  
 2709 011612 004737 006672

.SBTTL TEST 13: DUART BASIC TEST

```

; **
; This test invokes the on-board VSV21 DUART BASIC test on ports 0-3.
; If the HOE flag is specified the VSV21 will halt on error.
; If the LOE flag is specified the operator will be given the choice on the
; first pass of whether he wishes the on-board tests to loop on error. If he
; selects yes the test may hang indefinitely as the on board will not return
; control to the host but will continue looping.
; --
    
```

```

ST.TEST
CALL STTEST
BCC 30012$

CALL TSTMSK ; get test mask for microdiagnostic
CALL MRKONCE ; mark test as entered at least once

;FIRST PORT =====
CLR PBLOC+4 ; get port 0 into parameter block

;SUBTEST 1 moves the parameters for the microdiagnostics test through the
;parameter register
BGNSUB
CALL MVPAR ; move the parameters
BCC 10$
ESCAPE TST
10$: ENDSUB

;SUBTEST 2 invokes the microdiagnostic test
BGNSUB
CALL DOMIC
BCC 10$
ESCAPE TST
10$: ENDSUB

;SECOND PORT =====
INC PBLOC+4 ; get next port into parameter block

;SUBTEST 3 moves the parameters for the microdiagnostics test through the
;parameter register
BGNSUB
CALL MVPAR ; move the parameters
BCC 10$
ESCAPE TST
10$: ENDSUB

;SUBTEST 4 invokes the microdiagnostic test
BGNSUB
CALL DOMIC
    
```

TEST 13: DUART BASIC TEST

```

2710 011616 103002          BCC      10$
2711 011620                ESCAPE  TST
2712 011624                10$:    ENDSUB
2713
2714                ;THIRD PORT =====
2715
2716 011626 005237 002226          INC      PBLOC+4          ; get next port into parameter block
2717
2718                ;SUBTEST 5 moves the parameters for the microdiagnostics test through the
2719                ;parameter register
2720
2721 011632                BGNSUB
2722 011634 004737 006626          CALL    MVAR          ; move the parameters
2723 011640 103002                BCC     10$
2724 011642                ESCAPE  TST
2725 011646                10$:    ENDSUB
2726
2727                ;SUBTEST 6 invokes the microdiagnostic test
2728
2729 011650                BGNSUB
2730 011652 004737 006672          CALL    DOMIC
2731 011656 103002                BCC     10$
2732 011660                ESCAPE  TST
2733 011664                10$:    ENDSUB
2734
2735                ;FOURTH PORT =====
2736
2737 011666 005237 002226          INC      PBLOC+4          ; get next port into parameter block
2738
2739                ;SUBTEST 7 moves the parameters for the microdiagnostics test through the
2740                ;parameter register
2741
2742 011672                BGNSUB
2743 011674 004737 006626          CALL    MVAR          ; move the parameters
2744 011700 103002                BCC     10$
2745 011702                ESCAPE  TST
2746 011706                10$:    ENDSUB
2747
2748                ;SUBTEST 8 invokes the microdiagnostic test
2749
2750 011710                BGNSUB
2751 011712 004737 006672          CALL    DOMIC
2752 011716 103002                BCC     10$
2753 011720                ESCAPE  TST
2754 011724                10$:    ENDSUB
2755
2756
2757 011726                ENDTST

```

TEST 13: DUART BASIC TEST

2759  
 2760  
 2761  
 2762  
 2763  
 2764  
 2765  
 2766  
 2767  
 2768  
 2769  
 2770  
 2771  
 2772 011730  
 011730 004737 007150  
 011734 103002  
 2773  
 2774 011742 004737 006276  
 2775 011746 004737 007054  
 2776  
 2777  
 2778  
 2779 011752 005037 002226  
 2780  
 2781  
 2782  
 2783  
 2784 011756  
 2785 011760 004737 006626  
 2786 011764 103002  
 2787 011766  
 2788 011772  
 2789  
 2790  
 2791  
 2792 011774  
 2793 011776 004737 006672  
 2794 012002 103002  
 2795 012004  
 2796 012010  
 2797  
 2798  
 2799  
 2800 012012 005237 002226  
 2801  
 2802  
 2803  
 2804  
 2805 012016  
 2806 012020 004737 006626  
 2807 012024 103002  
 2808 012026  
 2809 012032  
 2810  
 2811  
 2812  
 2813 012034

.SBTTL TEST 14: DUART FULL TEST

```

; **
; This test invokes the on-board VSV21 DUART FULL test on ports 0-3.
; If the HOE flag is specified the VSV21 will halt on error.
; If the LOE flag is specified the operator will be given the choice on the
; first pass of whether he wishes the on-board tests to loop on error. If he
; selects yes the test may hang indefinitely as the on board will not return
; control to the host but will continue looping.
; --
    
```

```

ST.TEST
CALL STTEST
BCC 30013$

CALL TSTMSK ; get test mask for microdiagnostic
CALL MRKONCE ; mark test as entered at least once

;FIRST PORT =====
CLR PBLOC+4 ; get port 0 into parameter block

;SUBTEST 1 moves the parameters for the microdiagnostics test through the
;parameter register
BGNSUB
CALL MVAR ; move the parameters
BCC 10$
ESCAPE TST
10$: ENDSUB

;SUBTEST 2 invokes the microdiagnostic test
BGNSUB
CALL DOMIC
BCC 10$
ESCAPE TST
10$: ENDSUB

;SECOND PORT =====
INC PBLOC+4 ; get next port into parameter block

;SUBTEST 3 moves the parameters for the microdiagnostics test through the
;parameter register
BGNSUB
CALL MVAR ; move the parameters
BCC 10$
ESCAPE TST
10$: ENDSUB

;SUBTEST 4 invokes the microdiagnostic test
BGNSUB
    
```

TEST 14: DUART FULL TEST

```

2814 012036 004737 006672      CALL    DOMIC
2815 012042 103002             BCC     10$
2816 012044                   .ESCAPE TST
2817 012050      10$:        ENDSUB
2818
2819                          ;THIRD PORT =====
2820
2821 012052 005237 002226      INC     PBLOC+4           ; get next port into parameter block
2822
2823                          ;SUBTEST 5 moves the parameters for the microdiagnostics test through the
2824                          ;parameter register
2825
2826 012056                   BGNSUB
2827 012060 004737 006626      CALL    MVAR             ; move the parameters
2828 012064 103002             BCC     10$
2829 012066                   ESCAPE TST
2830 012072      10$:        ENDSUB
2831
2832                          ;SUBTEST 6 invokes the microdiagnostic test
2833
2834 012074                   BGNSUB
2835 012076 004737 006672      CALL    DOMIC
2836 012102 103002             BCC     10$
2837 012104                   ESCAPE TST
2838 012110      10$:        ENDSUB
2839
2840                          ;FOURTH PORT =====
2841
2842 012112 005237 002226      INC     PBLOC+4           ; get next port into parameter block
2843
2844                          ;SUBTEST 7 moves the parameters for the microdiagnostics test through the
2845                          ;parameter register
2846
2847 012116                   BGNSUB
2848 012120 004737 006626      CALL    MVAR             ; move the parameters
2849 012124 103002             BCC     10$
2850 012126                   ESCAPE TST
2851 012132      10$:        ENDSUB
2852
2853                          ;SUBTEST 8 invokes the microdiagnostic test
2854
2855 012134                   BGNSUB
2856 012136 004737 006672      CALL    DOMIC
2857 012142 103002             BCC     10$
2858 012144                   ESCAPE TST
2859 012150      10$:        ENDSUB
2860
2861
2862 012152                   ENDTST

```



TEST 14: DUART FULL TEST

2864  
2865  
2866  
2867  
2868  
2869  
2870  
2871  
2872  
2873  
2874  
2875  
2876  
2877  
2878  
2879  
2880  
2881  
2882  
2883  
2884  
2885  
2886  
2887  
2888  
2889  
2890  
2891  
2892  
2893  
2894  
2895  
2896  
2897  
2898  
2899  
2900  
2901  
2902  
2903  
2904  
2905  
2906  
2907  
2908  
2909  
2910  
2911  
2912  
2913  
2914  
2915

000010  
012154 004737 007150  
012160 103002  
012166  
012176  
012220  
012240 004737 006276  
012244 004737 007054  
012250  
012270  
012304 005737 013076  
012310 001007  
012312 052737 000001 002224  
012320 012737 000056 002230  
012326 000406  
012330 052737 000002 002224  
012336 012737 000020 002230  
012344  
012344  
012346 005737 013076  
012352 001013  
012354  
012354 012737 013020 002622  
012362 012737 000056 002624  
012370 004737 006066  
012374 103002

.SBTTL TEST 15: PERIPHERAL CONFIDENCE TEST

```

; **
; This test allows the operator to select a port for either input or output.
; If he selects output an ASCII string is written to that port.
; If he selects input the first 16 characters read from that port are displayed
; in octal on the console running the tests.
; If the HOE flag is specified the VSV21 will halt on error.
; If the LOE flag is specified the operator will be given the choice on the
; first pass of whether he wishes the on-board tests to loop on error. If he
; selects yes the test may hang indefinitely as the on board will not return
; control to the host but will continue looping.
; --
    
```

```

T15RLEN=8.
ST.TEST
CALL STTEST
BCC 30014$

MAN.IGNORE ; ignore test if not manual
DMA.IGNORE ; ensure DMA enabled
GMANID T15M1,PBLOC+4,D,177777,0,3,NO ; get port number
CALL TSTMSK ; get test mask for microdiagnostic
CALL MRKONCE ; mark test as entered at least once

PRINTF #T15M2
GMANIL T15M3,T15INP,177777,NO ; get whether input/output device
TST T15INP ; input device ?
BNE 30$ ; branch if yes
BIS #1,PBLOC+2 ; say normal mode + write
MOV #<T15WEN-T15WBUF>,PBLOC+6 ; put length to be DMA'd
BR 40$
30$: BIS #2,PBLOC+2 ; say normal mode + read
MOV #<T15RLEN*2>,PBLOC+6 ; put length to be DMA'd

;SUBTEST 1 DMAs test data into the VSV21 for an output device
40$: BGNSUB
TST T15INP ; input device ?
BNE 10$ ; branch if yes
DO.WRAMS #T15WBUF,#<T15WEN-T15WBUF> ; DMA the data
MOV #T15WBUF,COMBUF+4
MOV #<T15WEN-T15WBUF>,COMBUF+6
CALL WRAMS
BCC 10$
ESCAPE TST ; exit if DMA fails
10$: ENDSUB

;SUBTEST 2 moves the parameters for the microdiagnostics test through the
;parameter register
    
```

TEST 15: PERIPHERAL CONFIDENCE TEST

```

2916 012404
2917 012406 004737 006626
2918 012412 103002
2919 012414
2920 012420 10$:
2921
2922
2923 ;SUBTEST 3 invokes the microdiagnostic test
2924 012422
2925 012424 004737 006672
2926 012430 103002
2927 012432
2928 012436 10$:
2929
2930 ;SUBTEST 4 DMA's the data back from the VSV21 for an input device
2931
2932 012440
2933 012442 005737 013076
2934 012446 001441
2935 012450
    012450 012737 013000 002622
    012456 012737 000020 002624
    012464 004737 006132
2936 012470 103002
2937 012472
2938 012476 10$:
2939 012516 012701 000010
2940 012522 012702 013000
2941 012526
2942 012550 077112 20$:
2943 012552 30$:
2944
2945 012554
2946
2947 012560 120 114 105 T15M1: .ASCIZ /PLEASE ENTER PORT NUMBER/
2948 012611 045 116 045 T15M2: .ASCIZ /%N%AIS IT AN INPUT DEVICE ?/
2949 012645 120 114 105 T15M3: .ASCIZ /PLEASE ENTER "YES" FOR INPUT DEVICE OR "NO" FOR OUTPUT DEVICE/
2950 012743 045 116 045 T15M4: .ASCIZ /%N%ADATA RECEIVED IS : /
2951 012773 045 117 067 T15M5: .ASCIZ /%07/
2952 .EVEN
2953 013000 T15RBUF:
2954 013000 .BLKW T15RLEN
2955 013020 T15WBUF:
2956 013020 041 100 043 .ASCIZ /!@#%&*+&*( )_+1234567890QWERTYUIOPiOasdfghjkl;' /
2957 013076 T15WEN:
2958 .EVEN
2959 013076 000000 T15INP: .WORD 0 ; <>0 for i/p device , 0 for o/p device
2960
2961 013100 ENDTST
    
```

TEST 15: PERIPHERAL CONFIDENCE TEST

2963  
2964  
2965  
2966  
2967  
2968  
2969  
2970  
2971  
2972  
2973  
2974  
2975  
2976  
2977  
2978  
2979  
2980  
2981  
2982  
2983  
2984  
2985  
2986  
2987  
2988  
2989  
2990  
2991  
2992  
2993  
2994  
2995  
2996  
2997  
2998  
2999  
3000  
3001  
3002  
3003  
3004  
3005  
3006  
3007  
3008  
3009  
3010  
3011  
3012  
3013  
3014

013102  
013102 004737 007150  
013106 103002

013114

013136 004737 006276  
013142 004737 007054

013146 052737 000013 002224  
013154 012737 000054 002230

013162

013164  
013164 012737 002762 002622  
013172 012737 000054 002624  
013200 004737 006066

013204 103002

013206

013212

013214 005037 002226

013220

013222 004737 006626  
013226 103002

013230

013234

013236

013240 004737 006672  
013244 103002

013246

013252

.SBTTL TEST 16: INTERNAL LOOPBACK TEST

```

; **
; This test causes the on-board software to perform an INTERNAL LOOPBACK
; test on ports 0-3.
; If the HOE flag is specified the VSV21 will halt on error.
; If the LOE flag is specified the operator will be given the choice on the
; first pass of whether he wishes the on-board tests to loop on error. If he
; selects yes the test may hang indefinitely as the on board will not return
; control to the host but will continue looping.
; --
    
```

```

ST.TEST
CALL STTEST
BCC 30017$
    
```

```

DMA.IGNORE ; ensure DMA enabled
    
```

```

CALL TSTMSK ; get test mask for microdiagnostic
CALL MRKONCE ; mark test as entered at least once
    
```

```

BIS #13,PBLOC+2 ; say local loop + read + write
MOV #DMALEN,PBLOC+6 ; put length to be DMA'd
    
```

;SUBTEST 1 DMA's test data into the VSV21

```

BGNSUB
DO.WRAMS ; DMA the data
    
```

```

MOV #DMAOBUF,COMBUF+4
MOV #DMALEN,COMBUF+6
    
```

```

CALL WRAMS
BCC 10$
    
```

```

10$: ESCAPE TST ; exit if DMA fails
      ENDSUB
    
```

;FIRST PORT =====

```

CLR PBLOC+4 ; get port 0 into parameter block
    
```

;SUBTEST 2 moves the parameters for the microdiagnostics test through the parameter register

```

BGNSUB
CALL MVPAR ; move the parameters
BCC 10$
    
```

```

10$: ESCAPE TST
      ENDSUB
    
```

;SUBTEST 3 invokes the microdiagnostic test

```

BGNSUB
CALL DOMIC
BCC 10$
    
```

```

10$: ESCAPE TST
      ENDSUB
    
```

TEST 16: INTERNAL LOOPBACK TEST

```

3015
3016           ;SUBTEST 4 DMAs the data back from the VSV21
3017
3018 013254           BGNSUB
3019 013256           DO.RRAMS           ; DMA the data
           012737 003036 002622      MOV #DMAIBUF,COMBUF+4
           013264 012737 000054 002624  MOV #DMALEN,COMBUF+6
           013272 004737 006132      CALL RRAMS
3020 013276 103002      BCC 10$
3021 013300           ESCAPE TST           ; exit if DMA fails
3022 013304           10$: ENDSUB
3023
3024           ;SUBTEST 5 Compares data written and read
3025
3026 013306           BGNSUB
3027 013310 004737 006212      CALL CRAMS           ; DMA the data
3028 013314 103002      BCC 10$
3029 013316           ESCAPE TST           ; exit if DMA fails
3030 013322           10$: ENDSUB
3031
3032           ;SECOND PORT =====
3033
3034 013324 005237 002226      INC PBLOC+4           ; get next port into parameter block
3035
3036           ;SUBTEST 6 moves the parameters for the microdiagnostics test through the
3037           ;parameter register
3038
3039 013330           BGNSUB
3040 013332 004737 006626      CALL MVAR           ; move the parameters
3041 013336 103002      BCC 10$
3042 013340           ESCAPE TST
3043 013344           10$: ENDSUB
3044
3045           ;SUBTEST 7 invokes the microdiagnostic test
3046
3047 013346           BGNSUB
3048 013350 004737 006672      CALL DOMIC
3049 013354 103002      BCC 10$
3050 013356           ESCAPE TST
3051 013362           10$: ENDSUB
3052
3053           ;SUBTEST 8 DMAs the data back from the VSV21
3054
3055 013364           BGNSUB
3056 013366           DO.RRAMS           ; DMA the data
           013366 012737 003036 002622  MOV #DMAIBUF,COMBUF+4
           013374 012737 000054 002624  MOV #DMALEN,COMBUF+6
           013402 004737 006132      CALL RRAMS
3057 013406 103002      BCC 10$
3058 013410           ESCAPE TST           ; exit if DMA fails
3059 013414           10$: ENDSUB
3060
3061           ;SUBTEST 9 Compares data written and read
3062
3063 013416           BGNSUB
3064 013420 004737 006212      CALL CRAMS           ; DMA the data
3065 013424 103002      BCC 10$

```

TEST 16: INTERNAL LOOPBACK TEST

```

3066 013426          ESCAPE TST          ; exit if DMA fails
3067 013432          10$: ENDSUB
3068
3069                ;THIRD PORT =====
3070
3071 013434 005237 002226          INC      PBLOC+4          ; get next port into parameter block
3072
3073                ;SUBTEST 10 moves the parameters for the microdiagnostics test through the
3074                ;parameter register
3075
3076 013440          BGNSUB
3077 013442 004737 006626          CALL     MVPAR          ; move the parameters
3078 013446 103002          BCC      10$
3079 013450          ESCAPE TST
3080 013454          10$: ENDSUB
3081
3082                ;SUBTEST 11 invokes the microdiagnostic test
3083
3084 013456          BGNSUB
3085 013460 004737 006672          CALL     DOMIC
3086 013464 103002          BCC      10$
3087 013466          ESCAPE TST
3088 013472          10$: ENDSUB
3089
3090                ;SUBTEST 12 DMA's the data back from the VSV21
3091
3092 013474          BGNSUB
3093 013476          DO.RRAMS          ; DMA the data
3094 013476 012737 003036 002622          MOV     #DMAIBUF,COMBUF+4
3095 013504 012737 000054 002624          MOV     #DMALEN,COMBUF+6
3096 013512 004737 006132          CALL     RRAMS
3097 013516 103002          BCC      10$
3098 013520          ESCAPE TST          ; exit if DMA fails
3099 013524          10$: ENDSUB
3100
3101                ;SUBTEST 13 Compares data written and read
3102
3103 013526          BGNSUB
3104 013530 004737 006212          CALL     CRAMS          ; DMA the data
3105 013534 103002          BCC      10$
3106 013536          ESCAPE TST          ; exit if DMA fails
3107 013542          10$: ENDSUB
3108
3109                ;FOURTH PORT =====
3110
3111 013544 005237 002226          INC      PBLOC+4          ; get next port into parameter block
3112
3113                ;SUBTEST 14 moves the parameters for the microdiagnostics test through the
3114                ;parameter register
3115
3116 013550          BGNSUB
3117 013552 004737 006626          CALL     MVPAR          ; move the parameters
3118 013556 103002          BCC      10$
3119 013560          ESCAPE TST
3119 013564          10$: ENDSUB
3119
3119                ;SUBTEST 15 invokes the microdiagnostic test

```

TEST 16: INTERNAL LOOPBACK TEST

```

3120
3121 013566          BGNSUB
3122 013570 004737 006672    CALL    DOMIC
3123 013574 103002          BCC     10$
3124 013576          ESCAPE  TST
3125 013602          10$:   ENDSUB
3126
3127          ;SUBTEST 16 DMA's the data back from the VSV21
3128
3129 013604          BGNSUB
3130 013606          DO.RRAMS          ; DMA the data
      013606 012737 003036 002622    MOV     #DMAIBUF,COMBUF+4
      013614 012737 000054 002624    MOV     #DMALEN,COMBUF+6
      013622 004737 006132          CALL    RRAMS
3131 013626 103002          BCC     10$
3132 013630          ESCAPE  TST          ; exit if DMA fails
3133 013634          10$:   ENDSUB
3134
3135          ;SUBTEST 17 Compares data written and read
3136
3137 013636          BGNSUB
3138 013640 004737 006212    CALL    CRAMS          ; DMA the data
3139 013644 103002          BCC     10$
3140 013646          ESCAPE  TST          ; exit if DMA fails
3141 013652          10$:   ENDSUB
3142
3143
3144 013654          ENDTST
    
```

TEST 16: INTERNAL LOOPBACK TEST

.SBTTL TEST 17: EXTERNAL I/O LOOPBACK TEST

3146  
3147  
3148  
3149  
3150  
3151  
3152  
3153  
3154  
3155  
3156  
3157  
3158  
3159

```

; **
; This test causes the on-board software to perform an EXTERNAL LOOPBACK
; test on ports 0-3.
; If the HOE flag is specified the VSV21 will halt on error.
; If the LOE flag is specified the operator will be given the choice on the
; first pass of whether he wishes the on-board tests to loop on error. If he
; selects yes the test may hang indefinitely as the on board will not return
; control to the host but will continue looping.
; --
    
```

3160 013656  
013656 004737 007150  
013662 103002

```

ST.TEST
CALL STTEST
BCC 30019$
    
```

3161  
3162 013670

```

DMA.IGNORE ; ensure DMA enabled
    
```

3163  
3164 013712 004737 006276  
3165 013716 004737 007054

```

CALL TSTMSK ; get test mask for microdiagnostic
CALL MRKONCE ; mark test as entered at least once
    
```

3166  
3167 013722 052737 000003 002224  
3168 013730 012737 000054 002230

```

BIS #3,PBLOC+2 ; say normal mode + read + write
MOV #DMALEN,PBLOC+6 ; put length to be DMA'd
    
```

3169  
3170

;SUBTEST 1 DMA's test data into the VSV21

3171  
3172 013736  
3173 013740  
013740 012737 002762 002622  
013746 012737 000054 002624  
013754 004737 006066

```

BGNSUB
DO.WRAMS ; DMA the data
MOV #DMAOBUF,COMBUF+4
MOV #DMALEN,COMBUF+6
CALL WRAMS
    
```

3174 013760 103002  
3175 013762  
3176 013766

```

BCC 10$
ESCAPE TST ; exit if DMA fails
10$: ENDSUB
    
```

3177  
3178

;FIRST PORT -----

3179  
3180 013770 005037 002226

```

CLR PBLOC+4 ; get port 0 into parameter block
    
```

3181  
3182

;SUBTEST 2 moves the parameters for the microdiagnostics test through the  
;parameter register

3183  
3184  
3185 013774  
3186 013776 004737 006626  
3187 014002 103002

```

BGNSUB
CALL MVPAR ; move the parameters
BCC 10$
ESCAPE TST
    
```

3188 014004  
3189 014010

10\$: ENDSUB

3190  
3191

;SUBTEST 3 invokes the microdiagnostic test

3192  
3193 014012  
3194 014014 004737 006672  
3195 014020 103002  
3196 014022  
3197 014026

```

BGNSUB
CALL DOMIC
BCC 10$
ESCAPE TST
10$: ENDSUB
    
```

TEST 17: EXTERNAL I/O LOOPBACK TEST

```

3198
3199           ;SUBTEST 4 DMA's the data back from the VSV21
3200
3201 014030           BGNSUB
3202 014032           DO.RRAMS           ; DMA the data
           012737 003036 002622       MOV     #DMAIBUF,COMBUF+4
           014040 012737 000054 002624  MOV     #DMALEN,COMBUF+6
           014046 004737 006132       CALL    RRAMS
3203 014052 103002   BCC     10$
3204 014054           ESCAPE  TST           ; exit if DMA fails
3205 014060 10$:     ENDSUB
3206
3207           ;SUBTEST 5 Compares data written and read
3208
3209 014062           BGNSUB
3210 014064 004737 006212       CALL    CRAMS           ; DMA the data
3211 014070 103002   BCC     10$
3212 014072           ESCAPE  TST           ; exit if DMA fails
3213 014076 10$:     ENDSUB
3214
3215           ;SECOND PORT =====
3216
3217 014100 005237 002226       INC     PBLOC+4           ; get next port into parameter block
3218
3219           ;SUBTEST 6 moves the parameters for the microdiagnostics test through the
3220           ;parameter register
3221
3222 014104           BGNSUB
3223 014106 004737 006626       CALL    MVPAR           ; move the parameters
3224 014112 103002   BCC     10$
3225 014114           ESCAPE  TST
3226 014120 10$:     ENDSUB
3227
3228           ;SUBTEST 7 invokes the microdiagnostic test
3229
3230 014122           BGNSUB
3231 014124 004737 006672       CALL    DOMIC
3232 014130 103002   BCC     10$
3233 014132           ESCAPE  TST
3234 014136 10$:     ENDSUB
3235
3236           ;SUBTEST 8 DMA's the data back from the VSV21
3237
3238 014140           BGNSUB
3239 014142           DO.RRAMS           ; DMA the data
           014142 012737 003036 002622  MOV     #DMAIBUF,COMBUF+4
           014150 012737 000054 002624  MOV     #DMALEN,COMBUF+6
           014156 004737 006132       CALL    RRAMS
3240 014162 103002   BCC     10$
3241 014164           ESCAPE  TST           ; exit if DMA fails
3242 014170 10$:     ENDSUB
3243
3244           ;SUBTEST 9 Compares data written and read
3245
3246 014172           BGNSUB
3247 014174 004737 006212       CALL    CRAMS           ; DMA the data
3248 014200 103002   BCC     10$

```



TEST 17: EXTERNAL I/O LOOPBACK TEST

```

3240 014202
3250 014206          ESCAPE TST          ; exit if DMA fails
10$: ENDSUB
3251
3252
3253 ;THIRD PORT =====
3254 014210 005237 002226          INC PBLOC+4          ; get next port into parameter block
3255
3256 ;SUBTEST 10 moves the parameters for the microdiagnostics test through the
3257 ;parameter register
3258
3259 014214          BGNSUB
3260 014216 004737 006626          CALL MVAR          ; move the parameters
3261 014222 103002          BCC 10$
3262 014224          ESCAPE TST
3263 014230 10$: ENDSUB
3264
3265 ;SUBTEST 11 invokes the microdiagnostic test
3266
3267 014232          BGNSUB
3268 014234 004737 006672          CALL DOMIC
3269 014240 103002          BCC 10$
3270 014242          ESCAPE TST
3271 014246 10$: ENDSUB
3272
3273 ;SUBTEST 12 DMA's the data back from the VSV21
3274
3275 014250          BGNSUB
3276 014252          DO.RRAMS          ; DMA the data
          014252 012737 003036 002622          MOV #DMAIBUF,COMBUF+4
          014260 012737 000054 002624          MOV #DMALEN,COMBUF+6
          014266 004737 006132          CALL RRAMS
3277 014272 103002          BCC 10$
3278 014274          ESCAPE TST          ; exit if DMA fails
3279 014300 10$: ENDSUB
3280
3281 ;SUBTEST 13 Compares data written and read
3282
3283 014302          BGNSUB
3284 014304 004737 006212          CALL CRAMS          ; DMA the data
3285 014310 103002          BCC 10$
3286 014312          ESCAPE TST          ; exit if DMA fails
3287 014316 10$: ENDSUB
3288
3289 ;FOURTH PORT =====
3290
3291 014320 005237 002226          INC PBLOC+4          ; get next port into parameter block
3292
3293 ;SUBTEST 14 moves the parameters for the microdiagnostics test through the
3294 ;parameter register
3295
3296 014324          BGNSUB
3297 014326 004737 006626          CALL MVAR          ; move the parameters
3298 014332 103002          BCC 10$
3299 014334          ESCAPE TST
3300 014340 10$: ENDSUB
3301
3302 ;SUBTEST 15 invokes the microdiagnostic test

```

TEST 17: EXTERNAL I/O LOOPBACK TEST

```

3303
3304 014342          BGNSUB
3305 014344 004737 006672    CALL    DOMIC
3306 014350 103002          BCC     10$
3307 014352          ESCAPE  TST
3308 014356          10$:   ENDSUB
3309
3310          ;SUBTEST 16 DMAs the data back from the VSV21
3311
3312 014360          BGNSUB
3313 014362          DO.RRAMS          ; DMA the data
      014362 012737 003036 002622    MOV     #DMAIBUF,COMBUF+4
      014370 012737 000054 002624    MOV     #DMALEN,COMBUF+6
      014376 004737 006132          CALL    RRAMS
3314 014402 103002          BCC     10$
3315 014404          ESCAPE  TST          ; exit if DMA fails
3316 014410          10$:   ENDSUB
3317
3318          ;SUBTEST 17 Compares data written and read
3319
3320 014412          BGNSUB
3321 014414 004737 006212    CALL    CRAMS          ; DMA the data
3322 014420 103002          BCC     10$
3323 014422          ESCAPE  TST          ; exit if DMA fails
3324 014426          10$:   ENDSUB
3325
3326
3327 014430          ENDTST
    
```

TEST 18: SCREEN TEST

```

3329          .SBTTL TEST 18: SCREEN TEST
3330
3331
3332          ;++
3333          ; This test displays different screen test pictures as selected by the
3334          ; operator. If the HOE flag is specified the VSV21 will halt on error.
3335          ; If the LOE flag is specified the operator will be given the choice on the
3336          ; first pass of whether he wishes the on-board tests to loop on error. If he
3337          ; selects yes the test may hang indefinitely as the on board will not return
3338          ; control to the host but will continue looping.
3339          ;--
3340
3341          014432          ST.TEST
3342          014432          004737          007150          CALL          STTEST
3343          014436          103002          BCC          30021$
3344
3345          014444          MAN.IGNORE          ; ignore test if not manual
3346
3347          014454          DMA.IGNORE          ; ensure DMA enabled
3348
3349          014476          004737          006276          CALL          TSTMSK          ; get testmask
3350          014502          004737          007054          CALL          MRKONCE          ; mark test as entered at least once
3351
3352          ;SUBTEST 1 DMAs the operator requested picture to the VSV21
3353
3354          014506          BGNSUB
3355          014510          PRINTF          #T18M1
3356          014530          GMANID          T18M2,TEMP1,D,177777,1,4,NO ; get current picture number
3357          014550          013700          002212          MOV          TEMP1,R0          ; and save it
3358          014554          005300          DEC          R0          ; get in range 0 - 3
3359          014556          006300          ASL          R0          ; get byte offset
3360          014560          016037          014712          002622          DO.WRAMS          PICS(R0),#<PIC2-PIC1> ; write the picture
3361          014566          012737          000042          002624          MOV          PICS(R0),COMBUF+4
3362          014574          004737          006066          MOV          #<PIC2-PIC1>,COMBUF+6
3363          014600          103002          CALL          WRAMS
3364          014602          BCC          20$
3365          014606          ESCAPE          TST
3366          20$:          ENDSUB
3367
3368          ;SUBTEST 2 moves the parameters for the microdiagnostics test through the
3369          ;parameter register
3370
3371          014610          BGNSUB
3372          014612          012737          000042          002226          MOV          #<PIC2-PIC1>,PBLOC+4          ; third parameter is length of picture
3373          014620          004737          006626          CALL          MVPAR          ; move the parameters
3374          014624          103002          BCC          10$
3375          014626          ESCAPE          TST
3376          014632          10$:          ENDSUB
3377
3378          ;SUBTEST 3 invokes the microdiagnostic test
3379
3380          014634          BGNSUB
3381          014636          004737          006672          CALL          DOMIC
3382          014642          103002          BCC          10$
3383          014644          ESCAPE          TST
3384          014650          10$:          ENDSUB
3385
3386
3387
3388
3389
3390

```

TEST 18: SCREEN TEST

```

3381 014652          PRINTF #T18M3          ; Print message
3382 014672          GMANIL T18M4,TEMP1,177777,YES ; and wait.
3383
3384 014706          EXIT  TST
3385
3386 014712 014722 014764 015026 PICS: .WORD PIC1,PIC2,PIC3,PIC4
3387 014722 014001 000001 177777 PIC1: .WORD 014001,000001,177777,004005,000000,004006,000000,004007
3388 014742 000360 004014 040000 .WORD 000360,004014,040000,004015,000000,054000,021042,000531
3389 014762 177001 .WORD 177001
3390 014764 014001 000001 177777 PIC2: .WORD 014001,000001,177777,004005,000000,004006,000000,004007
3391 015004 000360 004014 040000 .WORD 000360,004014,040000,004015,000000,054000,042104,000531
3392 015024 177001 .WORD 177001
3393 015026 014001 000001 177777 PIC3: .WORD 014001,000001,177777,004005,000000,004006,000000,004007
3394 015046 000360 004014 040000 .WORD 000360,004014,040000,004015,000000,054000,10421,000531
3395 015066 177001 .WORD 177001
3396 015070 014001 000001 177777 PIC4: .WORD 014001,000001,177777,004005,000000,004006,000000,004007
3397 015110 000360 004014 040000 .WORD 000360,004014,040000,004015,000000,054000,177777,000531
3398 015130 177001 .WORD 177001
3399
3400 015132 045 116 045 T18M1: .ASCIZ /%N%PLEASE INPUT PICTURE NUMBER WHERE :/
3401 015202 122 105 104 T18M2: .ASCIZ /RED , GREEN , BLUE AND WHITE SCREENS ARE 1,2,3,4 RESP./
3402 015271 045 116 045 T18M3: .ASCIZ /%N%PLEASE ENTER CARRIAGE RETURN WHEN YOU HAVE FINISHED/
3403 015361 127 111 124 T18M4: .ASCIZ /WITH THE PICTURE/
3404
3405 .EVEN
3406
3407 015402          ENDTST
    
```

TEST 18: SCREEN TEST

3409  
3410  
3411  
3412  
3413  
3414  
3415  
3416  
3417  
3418  
3419  
3420  
3421  
3422  
3423  
3424  
3425  
3426  
3427  
3428  
3429  
3430  
3431  
3432  
3433  
3434  
3435  
3436  
3437  
3438  
3439  
3440  
3441  
3442  
3443  
3444  
3445  
3446  
3447  
3448  
3449  
3450  
3451  
3452  
3453  
3454  
3455  
3456  
3457  
3458  
3459  
3460

.SBTTL TEST 19: NVRAM READ/WRITE TEST

```

; **
; This test tests NVRAM read/write. On each pass the operator will be asked if
; he wishes to continue as NVRAM has a limited life in terms of read/write
; cycles.
; If the HOE flag is specified the VSV21 will halt on error.
; If the LOE flag is specified the operator will be given the choice on the
; first pass of whether he wishes the on-board tests to loop on error. If he
; selects yes the test may hang indefinitely as the on board will not return
; control to the host but will continue looping.
; --
    
```

```

ST.TEST
CALL STTEST
BCC 30024$

MAN.IGNORE ; ignore test if not manual

CALL TSTMSK ; get test mask for microdiagnostic
CALL MRKONCE ; mark test as entered at least once

PRINTF #T20M1 ; Ask if operator
GMANIL T20M2,TEMP1,177777,NO ; wants to do test.
TST TEMP1 ; operator want to do test ?
BNE 20$ ; branch if yes
EXIT TST ; else exit if no
    
```

20\$:

```

;SUBTEST 1 moves the parameters for the microdiagnostics test through the
;parameter register
    
```

```

BGNSUB
CALL MVAR ; move the parameters
BCC 10$
ESCAPE TST
ENDSUB
    
```

10\$:

```

;SUBTEST 2 invokes the microdiagnostic test
    
```

```

BGNSUB
CALL DOMIC
BCC 10$
ESCAPE TST
ENDSUB
    
```

10\$:

ENDTST

```

045 T20M1: .ASCIZ /*N%A** WARNING ** WRITING TO NVRAM REDUCES IT'S LIFE/
040 T20M2: .ASCIZ /DO YOU STILL WANT TO DO THIS TEST ?/
    
```

.EVEN

TEST 19: NVRAM READ/WRITE TEST

3462  
3463  
3464  
3465  
3466  
3467  
3468  
3469  
3470

.SBTTL TEST 20: ENABLE INTERRUPTS TEST

```

;+
; This test verifies that interrupts are received by the host when interrupts
; are enabled.
;--
    
```

3471 015674  
015674 004737 007150  
015700 103002

```

ST.TEST
CALL STTEST
BCC 30026$
    
```

3472  
3473 015706  
3474  
3475 015734 005037 002240  
3476

```

SETVEC CURVEC,#INTVEC,#240 ; enable the interrupt service routine
; and give it priority #5
;
CLR INTFLG ; say no interrupts have been received
    
```

3477  
3478

;SUBTEST 1 enables interrupts and expects to get an interrupt while doing it

3479 015740  
3480 015742  
015742 012737 001403 002614  
015750 012737 002734 002216  
015756 004737 004572  
015762 103001

```

BGNSUB
DO.COMMAND PI.IEN,10$ ; enable interrupts
MOV #PI.IEN,COMAND
MOV #1500.,POLCNT
CALL DOCOM
BCC 10$
BR 20$ ; branch on error
MOV #1500.,R1 ; count to poll
TST INTFLG ; have we received an interrupt ?
BNE 20$ ; branch if yes
DELAY 10. ; delay 10 units
SOB R1,15$ ; again
ERR.FILLIN 15.,ERR15 ; time out
MOV #2,ERRTYP ; assume hard error for now
MOV #15.,ERRNBR ; get error 15.
MOV #ERR15,ERRMSG ; get error ERR15
MOV #ERRCODE,ERRBLK ; get error routine
ERROR
ENDSUB
    
```

3481 015764 000437  
3482 015766 012701 002734 10\$:  
3483 015772 005737 002240 15\$:  
3484 015776 001032  
3485 016000

3486 016030 077120  
3487 016032  
016032 012737 000002 002202  
016040 012737 000017 002204  
016046 012737 004177 002206  
016054 012737 004302 002210

3488 016062  
3489 016064  
3490

20\$:

;SUBTEST 2 disables interrupts

3491  
3492  
3493 016066  
3494 016070  
016070 012737 001400 002614  
016076 012737 002734 002216  
016104 004737 004572  
016110 103000

```

BGNSUB
DO.COMMAND PI.IDS,10$ ; disable interrupts
MOV #PI.IDS,COMAND
MOV #1500.,POLCNT
CALL DOCOM
BCC 10$
CLRVEC CURVEC ; disable the interrupt service routine
ENDSUB
    
```

3495 016112  
3496 016120  
3497  
3498 016122  
3499

ENDTST

3500  
3501

;Interrupt service routine

3502 016124  
3503 016124 012737 000001 002240  
3504 016132

```

BGNSRV INTVEC
MOV #1,INTFLG ; say received interrupt
ENDSRV
    
```

TEST 20: ENABLE INTERRUPTS TEST

```

3506
3507      .TITLE PARAMETER CODING
3508
3509      .SBTTL  HARDWARE PARAMETER CODING SECTION
3510
3511
3512      ;++
3513      ; THE HARDWARE PARAMETER CODING SECTION CONTAINS MACROS
3514      ; THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES.  THE
3515      ; MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
3516      ; INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES.  THE
3517      ; MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
3518      ; WITH THE OPERATOR.
3519      ;--
3520
3521 016134      BGNHRD
3522
3523 016136      GPRMA  DEVADD,0,0,160000,177776,YES
3524
3525 016146      GPRMA  VECADD,2,0,0,776,YES
3526
3527 016156      ENDHRD
3528
3529 016156      104    105    126  DEVADD: .ASCIZ  /DEVICE ADDRESS/
3530 016175      126    105    103  VECADD: .ASCIZ  /VECTOR ADDRESS/
3531
3532      .EVEN
3533
3534
3535 016214      $PATCH::
3536 016214      .BLKW  50.
3537
3538 016360      LASTAD
3539 016364      L$LAST::
3539 016364      ENDMOD

```

## HARDWARE PARAMETER CODING SECTION

3541	016364		BGNSETUP	4
3542	016364		BGNPTAB	
3543	016370	172010	.WORD	172010
3544	016372	000320	.WORD	320
3545	016374		ENDPTAB	
3546	016374		BGNPTAB	
3547	016400	172020	.WORD	172020
3548	016402	000324	.WORD	324
3549	016404		ENDPTAB	
3550	016404		BGNPTAB	
3551	016410	172030	.WORD	172030
3552	016412	000330	.WORD	330
3553	016414		ENDPTAB	
3554	016414		BGNPTAB	
3555	016420	172040	.WORD	172040
3556	016422	000334	.WORD	334
3557	016424		ENDPTAB	
3558	016424		ENDSETUP	
3559	000001	.END		



## Symbol table

ABORT	007510	C\$ERDF=	000055	DMA0BU	002762 G	GETST2	005462 G	L\$CO	002032 G
ADR	= 000020 G	C\$ERHR=	000056	DMASET=	000002 G	G\$CNT0=	000200	L\$DEPO	002011 G
ASSEMB=	000010	C\$ERRO=	000060	DOCOM	004572 G	G\$DELM=	000372	L\$DESC	004260 G
BITS	002244 G	C\$ERSF=	000054	DOMIC	006672 G	G\$DISP=	000003	L\$DESP	002076 G
BIT0	= 000001 G	C\$ERSO=	000057	EF.CON=	000036 G	G\$EXCP=	000400	L\$DEVP	002060 G
BIT00	= 000001 G	C\$ESCA=	000010	EF.NEW=	000035 G	G\$HILI=	000002	L\$DISP	002124 G
BIT01	= 000002 G	C\$ESEG=	000005	EF.PWR=	000034 G	G\$LOLI=	000001	L\$DLY	002116 G
BIT02	= 000004 G	C\$ESUB=	000003	EF.RES=	000037 G	G\$NO	= 000000	L\$DTP	002040 G
BIT03	= 000010 G	C\$ETST=	000001	EF.STA=	000040 G	G\$OFFS=	000400	L\$DTYP	002034 G
BIT04	= 000020 G	C\$EXIT=	000032	END	007504	G\$OFFSI=	000376	L\$DUT	002072 G
BIT05	= 000040 G	C\$FREQ=	000101	ERRBLK	002210 G	G\$PRMA=	000001	L\$DVTY	004251 G
BIT06	= 000100 G	C\$FRME=	000100	ERRCOD	004302 G	G\$PRMD=	000002	L\$EF	002052 G
BIT07	= 000200 G	C\$GETB=	000026	ERRMSG	002206 G	G\$PRML=	000000	L\$ENVI	002044 G
BIT08	= 000400 G	C\$GETW=	000027	ERRNBR	002204 G	G\$RADA=	000140	L\$ERRT	002202 G
BIT09	= 001000 G	C\$GMAN=	000043	ERRTYP	002202 G	G\$RADB=	000000	L\$ETP	002102 G
BIT1	= 000002 G	C\$GPHR=	000042	ERR1	003166 G	G\$RADD=	000040	L\$EXP1	002046 G
BIT10	= 002000 G	C\$GPRI=	000040	ERR10	003720 G	G\$RADL=	000120	L\$EXP4	002064 G
BIT11	= 004000 G	C\$INIT=	000011	ERR11	003753 G	G\$RADO=	000020	L\$EXP5	002066 G
BIT12	= 010000 G	C\$INLP=	000020	ERR12	004000 G	G\$XFER=	000004	L\$HARD	016136 G
BIT13	= 020000 G	C\$MANI=	000050	ERR13	004065 G	G\$YES	= 000010	L\$HIME	002120 G
BIT14	= 040000 G	C\$MAP	= 000102	ERR14	004130 G	HELP	= 000000	L\$HPCP	002016 G
BIT15	= 100000 G	C\$MEM	= 000031	ERR15	004177 G	HOE	= 100000 G	L\$HPTP	002022 G
BIT2	= 000004 G	C\$MMU	= 000103	ERR2	003225 G	IBE	= 010000 G	L\$HW	002176 G
BIT3	= 000010 G	C\$MSG	= 000023	ERR3	003274 G	IDU	= 000040 G	L\$ICP	002104 G
BIT4	= 000020 G	C\$OPNR=	000034	ERR4	003345 G	IER	= 020000 G	L\$INIT	007310 G
BIT5	= 000040 G	C\$OPNW=	000104	ERR5	003425 G	INTFLG	002240 G	L\$LADP	002026 G
BIT6	= 000100 G	C\$PNTB=	000014	ERR6	003465 G	INTVEC	016124 G	L\$LAST	016364 G
BIT7	= 000200 G	C\$PNTF=	000017	ERR7	003541 G	ISR	= 000100 G	L\$LOAD	002100 G
BIT8	= 000400 G	C\$PNTS=	000016	ERR8	003611 G	IXE	= 004000 G	L\$LUN	002074 G
BIT9	= 001000 G	C\$PNTX=	000015	ERR9	003652 G	I\$AU	= 000041	L\$MREV	002050 G
BOE	= 000400 G	C\$PUTB=	000072	EVL	= 000004 G	I\$AUTO=	000041	L\$NAME	002000 G
BUFDES	002604	C\$PUTW=	000073	E\$END	= 002100	I\$CLN	= 000041	L\$PRIO	002042 G
BUFOK	010374	C\$QIO	= 000377	E\$LOAD=	000035	I\$DU	= 000041	L\$PROT	007302 G
COMAND	002614	C\$RDBU=	000007	FLAGS	002304 G	I\$HRD	= 000041	L\$PRT	002112 G
COMBUF	002616	C\$REFG=	000047	FTHRU	= 000004 G	I\$INIT=	000041	L\$REPP	002062 G
COMLEN=	000024	C\$REL	= 000077	F\$AU	= 000015	I\$MOD	= 000041	L\$REV	002010 G
CRAMS	006212 G	C\$RESE=	000033	F\$AUTO=	000020	I\$MSG	= 000041	L\$SPC	002056 G
CURCSR	002232 G	C\$REVI=	000004	F\$BGN	= 000040	I\$PROT=	000040	L\$SPCP	002020 G
CURPAR	002234 G	C\$RFLA=	000021	F\$CLEA=	000007	I\$PTAB=	000041	L\$SPTP	002024 G
CURVEC	002236 G	C\$RPT	= 000025	F\$DU	= 000016	I\$PWR	= 000041	L\$STA	002030 G
C\$AU	= 000052	C\$SEFG=	000046	F\$END	= 000041	I\$RPT	= 000041	L\$TEST	002114 G
C\$AUTO=	000061	C\$SPRI=	000041	F\$HARD=	000004	I\$SEG	= 000041	L\$TIML	002014 G
C\$BRK	= 000022	C\$SVEC=	000037	F\$HW	= 000013	I\$SETU=	000041	L\$UNIT	002012 G
C\$BSEG=	000004	C\$TOME=	000076	F\$INIT=	000006	I\$SRV	= 000041	L1000	002202
C\$BSUB=	000002	DEVADD	016156	F\$JMP	= 000050	I\$SUB	= 000041	L10001	004312
C\$CLCK=	000062	DFPTBL	002176 G	F\$MOD	= 000000	I\$TST	= 000041	L10003	007512
C\$CLEA=	000012	DIAGMC=	000000	F\$MSG	= 000011	J\$JMP	= 000167	L10004	007514
C\$CLOS=	000035	DI.ERL=	000177 G	F\$PROT=	000021	LENFLA=	000030 G	L10005	007554
C\$CLP1=	000006	DI.ERP=	000005 G	F\$PWR	= 000017	LOE	= 040000 G	L10006	010236
C\$CPBF=	000074	DI.ERS=	000176 G	F\$RPT	= 000012	LOGUNT	002220 G	L10007	007630
C\$CPME=	000075	DI.MIC=	000011 G	F\$SEG	= 000003	LOT	= 000010 G	L10010	010010
C\$CVEC=	000036	DI.RDA=	002101 G	F\$SOFT=	000005	L\$ACP	002110 G	L10011	010506
C\$DCLN=	000044	DI.RRA=	002103 G	F\$SRV	= 000010	L\$APT	002036 G	L10012	010312
C\$DODU=	000051	DI.WDD=	002102 G	F\$SUB	= 000002	L\$AUT	002070 G	L10013	010340
C\$DRPT=	000024	DI.WRA=	002104 G	F\$SW	= 000014	L\$AUTO	007514 G	L10014	010372
C\$DU	= 000053	DMAIBU	003036 G	F\$TEST=	000001	L\$CCP	002106 G	L10015	010430
C\$EDIT=	000000	DMALEN=	000054	GETST1	005066 G	L\$CLEA	007516 G	L10016	010462

## Symbol table

L10017	010500	L10110	013304	NEXT	007376	TS1	010016	T12.1	011446
L10020	010622	L10111	013322	NOTEST	002242 G	TS2	010046	T12.2	011464
L10021	010620	L10112	013344	O\$APTS=	000000	TS3	010137	T13	011504 G
L10022	010702	L10113	013362	O\$AU =	000000	T\$ARGC=	000001	T13.1	011532
L10023	010662	L10114	013414	O\$BGNR=	000000	T\$CODE=	001031	T13.2	011550
L10024	010700	L10115	013432	O\$BGNS=	000000	T\$ERRN=	000002	T13.3	011572
L10025	010762	L10116	013454	O\$DU =	000000	T\$EXCP=	000000	T13.4	011610
L10026	010742	L10117	013472	O\$ERRT=	000001	T\$FLAG=	000040	T13.5	011632
L10027	010760	L10120	013524	O\$GNSW=	000000	T\$FREE=	016424	T13.6	011650
L10030	011042	L10121	013542	O\$POIN=	000001	T\$GMAN=	000000	T13.7	011672
L10031	011022	L10122	013564	O\$SETU=	000001	T\$HILI=	000776	T13.8	011710
L10032	011040	L10123	013602	PBLOC	002222 G	T\$LAST=	000001	T14	011730 G
L10033	011122	L10124	013634	PICS	014712	T\$LOLI=	000000	T14.1	011756
L10034	011102	L10125	013652	PIC1	014722	T\$LSYM=	010000	T14.2	011774
L10035	011120	L10126	014430	PIC2	014764	T\$LTNO=	000024	T14.3	012016
L10036	011202	L10127	013766	PIC3	015026	T\$NEST=	177777	T14.4	012034
L10037	011162	L10130	014010	PIC4	015070	T\$NSO =	000000	T14.5	012056
L10040	011200	L10131	014026	PI.ACK=	003000 G	T\$NS1 =	000004	T14.6	012074
L10041	011262	L10132	014060	PI.AVA=	100000 G	T\$NS2 =	000002	T14.7	012116
L10042	011242	L10133	014076	PI.CAN=	001001 G	T\$PCNT=	000000	T14.8	012134
L10043	011260	L10134	014120	PI.CMA=	002400 G	T\$PTAB=	010173	T15	012154 G
L10044	011342	L10135	014136	PI.IDM=	000400 G	T\$PTHV=	000004	T15INP	013076
L10045	011322	L10136	014170	PI.IDS=	001400 G	T\$PTNU=	000004	T15M1	012560
L10046	011340	L10137	014206	PI.IEN=	001403 G	T\$SAVL=	177777	T15M2	012611
L10047	011422	L10140	014230	PI.RUB=	177777 G	T\$SEGL=	177777	T15M3	012645
L10050	011402	L10141	014246	PI.SEN=	001000 G	T\$SIZE=	000020	T15M4	012743
L10051	011420	L10142	014300	PI.TST=	002000 G	T\$SUBN=	000002	T15M5	012773
L10052	011502	L10143	014316	PNT =	001000 G	T\$TAGL=	177777	T15RBU	013000
L10053	011462	L10144	014340	POLCNT	002216 G	T\$TAGN=	010175	T15RLE=	000010
L10054	011500	L10145	014356	POLL	004314 G	T\$TEMP=	000000	T15WBU	013020
L10055	011726	L10146	014410	POLLCR	004512 G	T\$TEST=	000024	T15WEN	013076
L10056	011546	L10147	014426	PRI =	002000 G	T\$TSTM=	177777	T15.1	012344
L10057	011564	L10150	015402	PRI00 =	000000 G	T\$TSTS=	000001	T15.2	012404
L10060	011606	L10151	014606	PRI01 =	000040 G	T\$AUT=	010004	T15.3	012422
L10061	011624	L10152	014632	PRI02 =	000100 G	T\$CLE=	010005	T15.4	012440
L10062	011646	L10153	014650	PRI03 =	000140 G	T\$DAT=	010174	T16	013102 G
L10063	011664	L10154	015540	PRI04 =	000200 G	T\$HAR=	010163	T16.1	013162
L10064	011706	L10155	015520	PRI05 =	000240 G	T\$HW =	010000	T16.10	013440
L10065	011724	L10156	015536	PRI06 =	000300 G	T\$INI=	010003	T16.11	013456
L10066	012152	L10157	016122	PRI07 =	000340 G	T\$MSG=	010001	T16.12	013474
L10067	011772	L10160	016064	RRAMS	006132 G	T\$PC =	000004	T16.13	013526
L10070	012010	L10161	016120	STABUF	002666	T\$PRO=	010002	T16.14	013550
L10071	012032	L10162	016132	STALEN=	000036	T\$PTA=	010173	T16.15	013566
L10072	012050	L10163	016156	STTEST	007150 G	T\$SRV=	010162	T16.16	013604
L10073	012072	L10164	016370	SVCGBL=	000000	T\$SUB=	010161	T16.17	013636
L10074	012110	L10165	016400	SVCINS=	177777	T\$TES=	010157	T16.2	013220
L10075	012132	L10166	016374	SVCSUB=	177777	T1	007556 G	T16.3	013236
L10076	012150	L10167	016410	SVCTAG=	177777	T1DONE=	000000 G	T16.4	013254
L10077	013100	L10170	016404	SVCTST=	177777	T1.1	007570	T16.5	013306
L10100	012402	L10171	016420	S\$LSYM=	010000	T1.2	007632	T16.6	013330
L10101	012420	L10172	016414	TEMP1	002212 G	T10	011264 G	T16.7	013346
L10102	012436	L10174	016424	TEMP2	002214 G	T10.1	011306	T16.8	013364
L10103	012552	MAXTST=	000024 G	TMASKS=	000010 G	T10.2	011324	T16.9	013416
L10104	013654	MAXUNI=	000004 G	TM1	006444	T11	011344 G	T17	013656 G
L10105	013212	MICNUM	007232	TM2	006541	T11.1	011366	T17.1	013736
L10106	013234	MRKONC	007054 G	TSTMSK	006276 G	T11.2	011404	T17.10	014214
L10107	013252	MVPAR	006626 G	TSTONC	007112 G	T12	011424 G	T17.11	014232

## Symbol table

T17.12	014250	T18	014432 G	T2.3	010342	T5	010704 G	T9.2	011244
T17.13	014302	T18M1	015132	T2.4	010374	T5.1	010726	UAM	= 000200 G
T17.14	014324	T18M2	015202	T2.5	010432	T5.2	010744	VECADD	016175
T17.15	014342	T18M3	015271	T2.6	010464	T6	010764 G	VS.CRY=	010000 G
T17.16	014360	T18M4	015361	T20	015674 G	T6.1	011006	VS.ERR=	040000 G
T17.17	014412	T18.1	014506	T20M1	015542	T6.2	011024	VS.PRY=	020000 G
T17.2	013774	T18.2	014610	T20M2	015627	T7	011044 G	VS.SVL=	100000 G
T17.3	014012	T18.3	014634	T20.1	015740	T7.1	011066	WRAMS	006066 G
T17.4	014030	T19	015404 G	T20.2	016066	T7.2	011104	X\$ALWA=	000000
T17.5	014062	T19.1	015504	T3	010510 G	T8	011124 G	X\$FALS=	000040
T17.6	014104	T19.2	015522	T3.1	010522	T8.1	011146	X\$OFFS=	000400
T17.7	014122	T2	010240 G	T4	010624 G	T8.2	011164	X\$TRUE=	000020
T17.8	014140	T2.1	010252	T4.1	010646	T9	011204 G	\$PATCH	016214 G
T17.9	014172	T2.2	010314	T4.2	010664	T9.1	011226		

. ABS. 016424 000 (RW,I,GBL,ABS,OVR)  
000000 001 (RW,I,LCL,REL,CON)

Errors detected: 0

## \*\*\* Assembler statistics

Work file reads: 303  
Work file writes: 304  
Size of work file: 35760 Words ( 140 Pages)  
Size of core pool: 19714 Words ( 75 Pages)  
Operating system: RSX-11M/PLUS (Under VAX/VMS)

Elapsed time: 00:07:25.81  
ZVSWAO,ZVSWAO/-SP=SVC40.MLB/ML,ZVSWAO.MAC