ML11 ML11 PROM MAINT PROG AH-S511A-MC AUG 1981
CZMLCA0 FICHE 1 OF 2 COPYRIGHT® 1981
MADE IN USA

ML11 ML11 PROM MAINT PROG AH-S511A-MC AUG 1981
CZMLCA0 FICHE 2 OF 2

```
MODULE BSKEL1 =
%TITLE  'CZMLCA ML-11 PROM MAINTENANCE PROGRAM'
```

SEQ 0001

```
%:
```

IDENTIFICATION

PRODUCT CODE: AC-S509A-MC

PRODUCT NAME: CZMLCAO ML-11 PROM MAINTENANCE PROGRAM

PRODUCT DATE: APRIL 22, 1981

MAINTAINER: DIAGNOSTIC ENGINEERING

AUTHOR: D.W.NEALE

TABLE OF CONTENTS

1.0 GENERAL INFORMATION

1.1 PROGRAM ABSTRACT

THE ML-11 MEMORY SYSTEM WITH ITS MOSTLY ARRAY TECHNOLOGY HAS THE FACILITY TO OFFSET AROUND KNOWN BAD MEMORY LOCATIONS IN ITS MEMORY ARRAYS.

INITIALLY, THESE MEMORY ARRAYS ARE TESTED FOR BAD ROW AND COLUMN ADDRESS LOCATIONS AND THE SPECIFIC OFFSETTING INFORMATION IS STORED IN PROM ON THE ARRAY MODULE. THIS TESTING IS DONE ON A SPECIAL 2224 MEMORY TESTER BY MEMORY MANUFACTURING.

THE DESIGN OF THE ML11 HAS ALSO PROVIDED LOGIC THAT WHEN UNDER SOFTWARE CONTROL WILL UPDATE AN ARRAY MODULES OFFSETTING INFORMATION. THIS LOGIC IS TO BE UTILIZED WHEN ADDITIONAL MEMORY CELLS ARE DISCOVERED BAD AFTER THE SYSTEM HAS LEFT THE MANUFACTURING FACILITY.

THIS HAS REQUIRED THE CREATION OF A PROGRAM THAT WHEN RUN WILL TEST A GIVEN ML-11 SYSTEM FOR ANY ADDITIONAL BAD MEMORY CELLS AND UPDATE THE OFFSET INFORMATION SUCH THAT THE BAD LOCATIONS WILL BE MASKED OUT.

THIS PROGRAM WILL SELECTIVELY UPDATE BAD MEMORY CELL OFFSETING FOR AN ENTIRE ML-11 SYSTEM, A SINGLE ARRAY MODULE OR A SINGLE BANK.

THE PROGRAM WILL EXERCISE THE ML-11 WITH ALL ONES, ALL ZEROES AND RANDOM DATA PATTERNS TO FIND ANY ADDITIONAL FAILING MEMORY CELLS.

ONCE THE ADDITIONAL FAILING CELLS HAVE BEEN MASKED OUT THE PROGRAM WILL GO BACK INTO THE FAILING CELLS AND VERIFY THAT THESE BAD CELLS HAVE BEEN MASKED OUT.

THE OPERATOR WILL BE NOTIFIED OF ANY ERROR CONDITIONS WHICH MIGHT OCCURE DURING THE EXECUTION OF THE PROGRAM. ADDITIONAL INFORMATION PERTAINING TO THESE ERROR MESSAGES CAN BE FOUND IN SECTION 3.0 OF THIS DOCUMENT.

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 REQUIREMENTS

1. PDP-11 CENTRAL PROCESSOR WITH A MINIMUM OF 28K USABLE MAIN MEMORY.
2. CONSOLE TERMINAL.
3. RH11 OR RH70 DISK CONTROLLER.
4. A MINIMUM OF ONE ML-11 SYSTEM ATTACHED TO THE ABOVE RH CONTROLLER.
5. XXDP+ LOAD MEDIA.

1.3 RELATED DOCUMENTS AND STANDARDS

1. SUPPRGC.DOC
2. SUPINT.MEM
3. SUPFUN.C
4. XXDPPLUS.DOC
5. BLISS LANGUAGE GUIDE
6. BLISS-16 USER'S GUIDE

1.4 DIAGNOSTIC HIERARCY PREREQUISITES

IT WILL BE ASSUMED THAT PRIOR TO RUNNING OF THIS PROGRAM THAT ALL APPROPRIATE CPU, MAIN MEMORY, ML-11 LOGIC TEST AND ML-11 SYSTEM EXERCISER HAS BEEN SUCCESSFULLY RUN ON THE ML-11 SYSTEM AND THAT THE SYSTEM EXERCISER SPECIFICALLY CALLS OUT THE RUNNING OF THIS PROGRAM.

THE SYSTEM EXERCISER WHEN IT CALLS FOR THE RUNNING OF THIS PROGRAM WILL INDICATE WHICH ARRAYS AND BANKS OF THE ML-11 SYSTEM NEED TO BE PROM MAINTENANCED. THIS INFORMATION SHOULD BE THEN INPUTED INTO THIS PROGRAM.

HOWEVER THIS PROGRAM IS DESIGNED TO GIVE THE OPERATOR THE OPTION TO PROM MAINTENANCE EITHER THE ARRAY AND BANK THAT THE SYSTEM EXERCISER CALLS OUT FOR PM'ING OR SELECT PROM MAINTENANCE FOR AN ENTIRE ARRAY MODULE (FOUR BANKS) OR SELECT PROM MAINTENANCE FOR THE ENTIRE ML-11 SYSTEM (ALL PRESENT ARRAY MODULES).

1.5 ASSUMPTIONS

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".

THIS PROGRAM USES THE DIAGNOSTIC RUN TIME SERVICE FOR PROGRAM PARAMETER INPUT, ERROR REPORTING AND MESSAGE PRINTING.

IT IS DESINGED TO TEST ONE ML-11 SYSTEM AND IS EXPECTED TO RUN FROM START TO FINISH WITH NO OPERATOR INTERRUPTIONS (ie. ^C).

THEREFOR THE ONLY RECOGNIZED DRS> COMMAND BY THIS PROGRAM IS THE 'START' COMMAND AND CONTROL C AND ANY OTHER DRS> COMMAND MUST BE AVOIDED.

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 'DDDDD'.

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:DDDDD	EXECUTE DDDDD PASSES (DDDDD = 1 TO 64000)
/FLAGS:FLGS	SET SPECIFIED FLAGS. FLAGS ARE DESCRIBED IN SECTION 2.3.
/EOP:DDDDD	REPORT END OF PASS MESSAGE AFTER EVERY DDDDD PASSES ONLY. (DDDDD = 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 BE 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					

AS MENTIONED BEFORE THE PROGRAM IS DESIGNED TO TEST ONE ML-11 SYSTEM AND IS EXPECTED TO RUN FROM START TO FINISH WITH NO OPERATOR INTERRUPTIONS.

THEREFOR USAGE OF ANY SWITCHES WOULD PROVE MEANINGLESS TO THE PROGRAM AND SHOULD BE AVOIDED.

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
```

AS MENTIONED BEFORE THIS PROGRAM IS DESIGNED TO TEST ONE ML-11 AND IS EXPECTED TO RUN FROM START TO FINISH WITH NO OPERATOR INTERRUPTIONS.

THEREFOR USAGE OF ANY FLAGS WOULD PROVE MEANINGLESS TO THIS PROGRAM AND SHOULD BE AVOIDED.

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). WHEN YOU ANSWER THIS QUESTION WITH A "Y", THE RUNTIME SERVICES WILL ASK FOR THE NUMBER OF UNITS (IN DECIMAL). ONLY ONE DRIVE IS PERMITTED TO BE PROM MAINTENANCED PER EXECUTION OF THE PROGRAM THEREFOR ANSWER THIS

QUESTION WITH '1'. YOU WILL THEN BE ASKED THE FOLLOWING QUESTIONS.

OPTION 1 'IS ENTIRE ML-11 SYSTEM TO BE MASKED'
          TRANSFER 'SYS' IF TRUE

OPTION 2 'IS A SINGLE ARRAY TO BE MASKED'
          TRANSFER 'BOARD' IS TRUE

OPTION 3 'IS A SINGLE BANK TO BE MASKED'
          TRANSFER 'DONE' IF FALSE

          'ENTER BANK NUMBER TO BE MASKED'

'BOARD'   'ENTER BOARD NUMBER TO BE MASKED'

'SYS'   'STARTING RH REGISTER ADDRESS'

          'DRIVE UNDER TEST NUMBER'

'DONE'   'ARE YOUR INPUTED PARAMETERS CORRECT'

2.5 SOFTWARE QUESTIONS

SOFTWARE QUESTIONS ARE NOT USED DURING THIS PROGRAM AND THIS QUESTION SHOULD BE ANSWERED WITH A 'NO' RESPONCE.

HOWEVER IF A YES RESPONCE IS GIVEN THE FOLLOWING MESSAGE WILL BE PRINTED:

'NOT USED TYPE <CR>'

2.6 EXTENDED P-TABLE DIALOGUE

TRADITIONALLY DRS> PROVIDES YOU WITH THE ABILITY TO BUILD P-TABLES FOR MULTIPLE DRIVE TESTING. BECAUSE OF THE IMPACT OF THIS PROGRAM ON A ML-11 SYSTEM AND THE LENGTHY RUNTIME ONLY ONE DRIVE WILL BE PROM MAINTENANCED PER EXECUTION OF THIS PROGRAM.

HOWEVER THE NATURE OF DRS> WILL STILL ALLOW YOU TO BUILD MULTIPLE DRIVE SELECTION FOR TESTING. THIS PROGRAM WILL TREAT THIS AS A SYSTEM ERROR AND SELECT THE FIRST P-TABLE BUILT FOR THE RUN TIME PARAMETERS.

2.7 START-UP PROCEDURE (XXDP+)

TO START-UP THIS PROGRAM:

1. BOOT XXDP+
2. GIVE THE DATE AND ANSWER THE LSI AND 50HZ (IF THERE IS A CLOCK) QUESTIONS
3. TYPE 'R CZMLCA''
4. TYPE ''START''

THE START COMMAND WILL BE THE ONLY COMMAND
ACCEPTED BY THIS PROGRAM AND TYPING ANY OTHER
COMMAND MUST BE AVOIDED. FOR TESTING MULTIPLE
DRIVES REPEAT STEPS 4 THRU 7 FOR EACH DRIVE.

5. ANSWER THE ''CHANGE HW'' QUESTION WITH ''Y''

6. ANSWER ALL THE HARDWARE QUESTIONS

7. ANSWER THE ''CHANGE SW'' QUESTION WITH 'N''

WHEN YOU FOLLOW THIS PROCEDURE YOU WILL BE USING ONLY THE
DEFAULTS FOR FLAGS AND SOFTWARE PARAMETERS. THESE DEFAULTS
ARE DESCRIBED IN SECTIONS 2.3 AND 2.5.

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 NUMBER	ERROR DESCRIPTION
ERR_1	ONLY THE DRS> START COMMAND IS RECOGNIZED TO START THE PROGRAM EXECUTION. ANY OTHER DRS> COMMAND WILL CAUSE THIS ERROR
ERR_2	DURING THE HARDWARE QUESTIONS THE OPERATOR IS ASKED IF A SINGLE BANK, A SINGLE ARRAY OR THE ENTIRE ML-11 SYSTEM IS TO BE PROM

MAINTENANCED.

HE/SHE IS THEN ASKED IF HIS/HER INPUTS ARE CORRECT.

THIS ERROR DETECTS A NO ANSWER FOR SELECTING SYSTEM, ARRAY OR BANK AND A YES ANSWER TO 'ARE YOUR INPUTED PARAMETERS CORRECT'.

ERR_3

EVEN THOUGH DRS> WILL BUILD MULTIPLE P-TABLES THIS PROGRAM WILL USE AS ITS RUN TIME PARAMETERS THE LUN 0'S ENTRIES.

THIS ERROR DETECTS THE ABSENCE OF THIS FIRST P-TABLE 'LUN 0'.

ERR_4

THIS ERROR DETECTS UNCONFIRMED FAILING CHIPS.

ie. CHIPS THAT FAILED DURING MASS BUS WRITE CHECK TRANSFERS BUT THE FAILURE DID NOT REOCCURE DURING DATA DIAGNOSTIC MODES. THIS MAY INDICATE THAT THIS FAILURE MAY BE A SOFT ERROR OR POSSIBLE HARDWARE ERRORS.

ERR_5

CONDITION A

THIS INDICATES THAT 'ONE' ALL BAD CHIP (GREATER THAN 10 ALL BAD ROWS AND OR COLUMNS) HAS BEEN DETECTED IN A CHIP AT A GIVEN BANK.

THIS CHIP IS NOT PROM MAINTENANCED AND THE ERROR CORRECTION IS EXPECTED TO CORRECT THE FAILING DATA FROM THIS CHIP UNTIL FIELD SERVICE CAN REPLACE THE ARRAY FROM WHICH THIS CHIP RESIDES.

ERR_6

CONDITION B

THIS INDICATES THAT A SECOND ALL BAD CHIP HAS BEEN DETECTED IN A GIVEN BANK.

THE BAD CHIP IS NOT PROM MAINTENANCED AND FURTHER TESTING OF THIS ARRAY IS ABORTED.

FIELD SERVICE SHOULD REPLACE THIS ARRAY BEFORE LEAVING THE SITE.

ERR_7

CONDITION C

INDICATES THAT BAD NIBBLE OFFSETS HAS EXCEEDED 14 OFFSETS RESULTING IN UNSAFE ERRORS.

THE ARRAYS NIBBLE OFFSETS FOR THIS BANK ARE MASKED UP TO 14 OFFSETS. THE ERROR CORRECTION IS EXPECTED TO CORRECT THE UNMASKED ERRORS WHICH WERE LEFT BEHIND.

FIELD SERVICE SHOULD REPLACE THIS ARRAY MODULE AT THE EARLIEST POSSIBLE DATA.

ERR_8

CONDITION D

THIS INDICATES THAT PROM BLAST ERRORS WERE DETECTED.

THIS MEANS THAT THE SELECTED NEW PROM DATA WAS NOT WRITTEN INTO THE PROMS CORRECTLY. EITHER THE SELECTED PROM LOCATIONS WERE NOT WRITTEN CORRECTLY OR OTHER PROM LOCATIONS WERE INADVERTENTLY WRITTEN INTO.

THIS COULD RESULT IN EITHER BAD LOCATIONS ARE NOT BEING MASKED OR GOOD LOCATIONS ARE BEING MASKED.

LATER ROUTINES WILL DETERMINE IF THIS SITUATION WARRENTS THE ARRAY TO BE REPLACED.

ERR_9

AS MENTIONED BEFORE ONLY THE FIRST P-TABLE BUILT WILL BE USED AS THE PROGRAM PARAMETERS.

THIS ERROR DETECTS GREATER THAN ONE P-TABLE WAS BUILT DURING THE HARDWARE QUESTIONS.

ERR_10

DURING WRITING NEW PROM DATA TO THE PROMS THE DATA CLOCK BIT LOCATED ON THE ARRAY DATA MODULE IS TESTED FOR COMPLETION OF THE PROM WRITE.

THIS ERROR DETECTS THE FAILURE OF THIS BIT TO CLEAR AFTER WRITING TO THE PROMS.

ERR_11

AFTER THE PROMS HAVE BEEN WRITTEN WITH NEW PROM DATA THE PROGRAM VERIFIES THAT ALL NEWLY FAILING CHIPS HAVE BEEN SUCCESSFULLY MASKED OUT.

THIS ERROR DETECTS THE OCCURANCE OF UNCORRECTABLE ERROR DURING THIS VERIFY PASS.

THIS ERROR CAN NOT BE TOLERATED AND THE ARRAY IS CALLED OUT FOR REPLACEMENT.

FIELD SERVICE MUST REPLACE THIS ARRAY MODULE BEFORE LEAVING THE SITE.

IF THIS ERROR STILL EXITS AFTER THE ARRAY IS REPLACED THEN POSSIBLE HARDWARE ERRORS MAY EXITS IN THE DRIVE. THE LOGIC TEST AND EXERCISER SHOULD BE RUN AGAIN.

ERR_12

AGAIN DURING THE VERIFY PASS THE DRIVE IS EXAMINED FOR ERRORS AFTER BLASTING.

SINGLE BIT ERRORS ARE TOLERATED IN BANKS WHICH ARE RUNNING DEGRADE MODE (EITHER ONE ALL BAD CHIP WAS LEFT BEHIND OR NIBBLE OFFSETS GREATER THAN WERE DETECTED). HOWEVER NON - DEGRADE MODE BANKS SHOULD BE RUNNING ERROR FREE AFTER BLASTING.

THIS ERROR DETECTS THE OCCURANCE OF SINGLE BITS IN THE BANKS WHICH ARE NOT IN DEGRADE MODE.

THIS COULD INDICATE THAT THE PROGRAM FAILED TO

TO FIND AND MASK OUT ADDITIONAL ERROR IN THE BANK
OR THAT POSSIBLE HARDWARE ERRORS EXIST.

THE PROGRAM SHOULD BE RUN ON THIS BANK AGAIN.
IF THE ERROR STILL EXISTS THEN ISOLATE WHERE
THE PROBLEM LIES. IF THIS PROGRAM IS SUSPECTED
THEN CONTACT DIAGNOSTIC ENGINGERING.

ERR_13 THIS ERROR DETECTS THE PRESENTS OF UNEXPECTED
DRIVE ERRORS DURING OR AFTER A MASS BUS TRANS-
FER. ATER THE A MESSAGE IS PRINTED STATING THE ERROR
ALL THE DIRECTLY READ ML-11 REGISTERS ARE DUMPED
TO THE TERMINAL.

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.

THIS PROGRAM WILL BE EXECUTED ONE TIME PER 'START' COMMAND ISSUED.
THE DRS> REPORT OF HOW MANY OF ERRORS DETECTED HAS NO MEANING DURING
EXECUTION OF THIS PROGRAM.

ONCE THE PROGRAM EXECUTION HAS COMPLETED THIS JUST PM'ED UNIT IS
DROPPED TO SUPPRESS FURTHER PROGRAM EXECUTION AND CONTROL IS PASSED
TO DRS>.

DROPPING OF A UNIT RESULTS IN A DRS> MESSAGE OF 'PASS ABORTED FOR
THIS UNIT'. THIS MESSAGE HAS NO SIGNIFICANCE ON THE PROGRAMS
EXECUTION AND SHOULD BE IGNORED.

5.0 DEVICE INFORMATION TABLES

HARDWARE P-TABLE ENTRY DEFINTITION

TABLE LOCATION	DESCRIPTION
1. A_HWTBL : INITIAL(0)	STORES A TRUE OR FALSE VALUE AND HAS NO DEFAULT VALUE. SELECTS WHETHER THE ENTIRE ML-11 SYSTEM (ALL PRESENT ARRAYS) ARE TO BE PM'ED.
2. B_HWTBL : INITIAL(0)	STORES A TRUE OR FALSE VALUE AND HAS NO DEFAULT VALUE. SELECTS WHETHER A SINGLE ARRAY MODULE IS TO BE PM'ED.
3. C_HWTBL : INITIAL(0,	STORES A TRUE OR FALSE VALUE AND HAS NO DEFAULT VALUE. SELECTS WHETHER A SINGLE BANK IS TO BE PM'ED.
4. D_HWTBL : INITIAL(0)	STORES THE SELECTED BANK NUMBER TO BE PM'ED IF PM'ING SINGLE BANKS.

THERE IS NO DEFAULT VALUE.

5. E_HWTBL : INITIAL(0)

STORES THE SELECTED ARRAY MODULE TO BE PM'ED IF PM'ING SINGLE ARRAYS.

THERE IS NO DEFAULT VALUE.

6. F_HWTBL : INITIAL(176400)

STORES THE RH CONTROLLER BASE REGISTER ADDRESS.

THE DEFAULT ADDRESS IS %O'176400'.

7. G_HWTBL : INITIAL(0)

STORES THE DRIVE SELECTION NUMBER OF THE SELECTED DRIVE TO BE PM'ED.

THERE IS NO DEFAULT VALUE.

8. H_HWTBL : INITIAL(0)

STORES THE DRIVE OPTION CODE FOR THE SELECTED DRIVE.

CODE 1 - 16K MOS RAMS
CODE 2 - 64K MOS RAMS

9. I_HWTBL : INITIAL(TRUE)

STORES A TRUE OF FALSE VALUE.

THIS FORCES THE OPERATOR TO REVIEW HIS/HER PARAMETER INPUTS FOR CORRECTNESS BEFORE PERFORMING PROM MAINTENANCE.

THE DEFAULT VALUE IS TRUE.

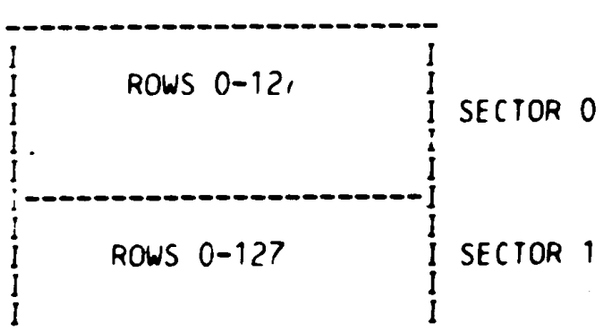
6.0 PROM MAINTENANCE TABLE REPRESENTATIONS

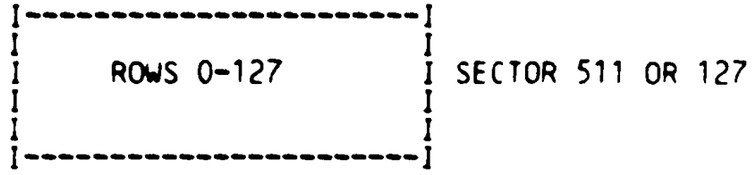
ERROR MAP

THE ERROR MAP IS A BLOCK VECTOR OF 512 BLOCKS. THESE BLOCKS REPRESENT THE SECTORS IN A CHIP. EACH BLOCK HAS 8 WORDS. THESE BIT POSITIONS IN THE WORDS REPRESENT ROW ADDRESSES 0 TO 127.

THE ADJACENT COLUMN ADDRESS FOR EACH ROW ADDRESS CAN BE CALCULATED BY ADDING THE ROW ADDRESS TO THE ROWS SECTOR NUMBER.

ERROR_MAP

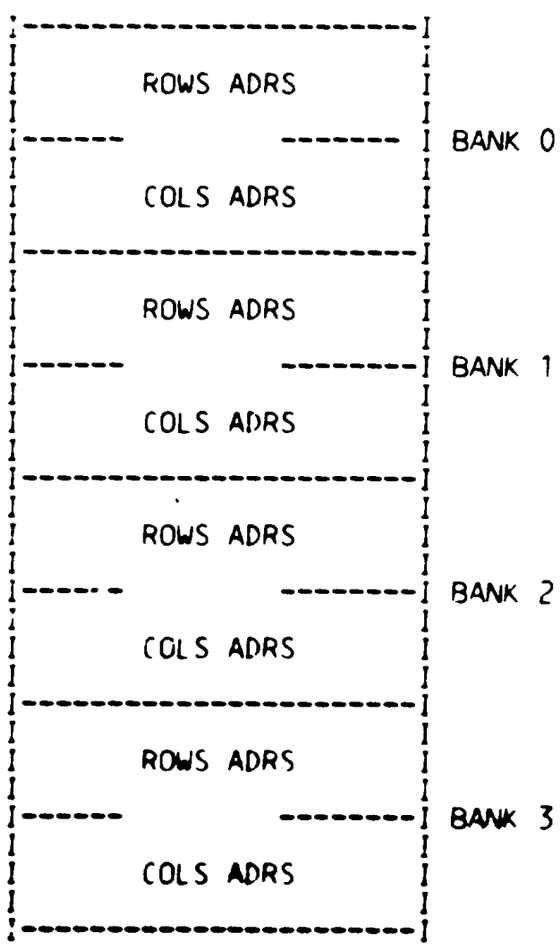




BLAST TABLE

THE BLAST TABLE IS A BLOCK VECTOR OF FOUR BLOCKS.
EACH BLOCK REPRESENTS A PROMS BANK 0 TO 3.
EACH BLOCK HAS 512 WORDS. EACH BLOCK IS FURTHER
DIVIDED INTO TWO SECTIONS. THE UPPER SECTION REPRESENTS
ROW PROM DATA 0 TO 127 (OR 0 TO 256 IF 64K CHIPS)
AND THE LOWER SECTION COLUMN PROM DATA 0 TO 127
(OR 0 TO 256 IF 64K CHIPS).

BLAST TABLE

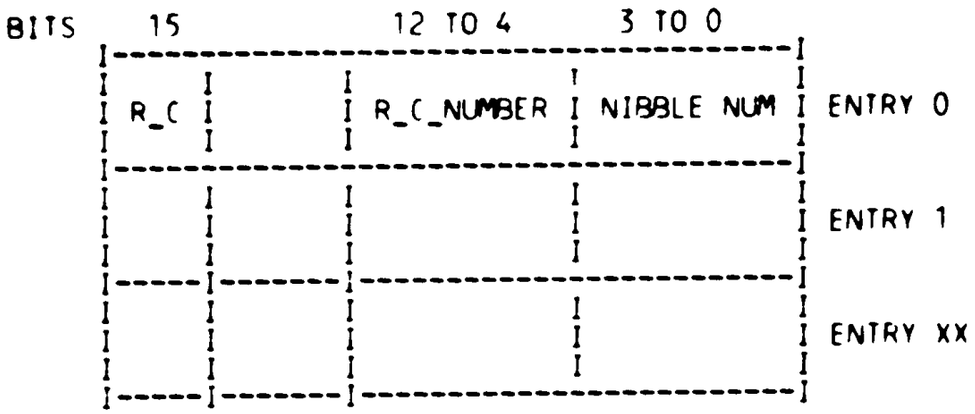


THE BLAST TABLE IS LOADED WITH THE NEW PROM DATA TO MASK
OUT THE NEWLY FAILING ROWS AND COLUMNS AND IS ALSO LOADED WITH
THE OLD PROM DATA FROM PREVIOUS PROM BLASTING.

TEMP BLAST TABLE

THE TEMP BLAST TABLE IS A BLOCK OF 10 WORDS. IN EACH WORD IS STORED A BAD ROW OR COLUMNS RESPECTIVE NIBBLE NUMBER, ITS ROW OR COLUMN NUMBER AND A FLAG TO TELL WHETHER THIS TABLE ENTRY IS A ROW ADRS OR A COLUMN ADRS.

THIS TABLE IS NECESSARY BECAUSE ONCE A BAD ROW OR COLUMNS ADDRESS IS STORED INTO THE MAIN BLAST TABLE THERE IS NO WAY OF KNOWING OF WHICH NIBBLE THE ROW OR COLUMN COMES FROM. THIS INFORMATION IS NEEDED WHEN THIS PART IS DETERMINED TO BE ALL BAD ' > 10 ALL BAD ROW OR COLUMNS) IN THIS EVENT THIS CHIPS NEWLY FAILING ROWS AND COLUMNS ARE NOT BLASTED. TO ACCOMPLISH THIS THE TEMPORARY BLAST TABLE IS SIMPLY NOT TRANSFERED INTO THE MAIN BLAST TABLE.



TABLES USED IN CALCULATING SELECTED ROWS AND COLUMNS FOR BLASTING.

COLUMN COUNT TABLE
THIS TABLE KEEPS A COUNT OF THE NUMBER OF TIMES THE COLUMNS FAILS WITH A PARTICULAR FAILING ROW.

ROW COUNT
THIS SINGLE VARIABLE COUNTS HOW MANY TIMES A ROW NUMBER IS FOUND BAD WHEN SEARCHING THE ERROR MAP. EACH TIME A ROW IS FOUND BAD THE COUNT IS INCREMENTED. IF THIS COUNT GETS > 10 THEN THIS ROW IS CALLED ALL BAD AND IS SELECTED FOR BLASTING. WHEN THIS HAPPENS THE COUNTS OF THE ADJACENT FAILING COLUMN IS DECREMENTED BY ONE. THE COLUMN POINTER TABLE POINTS TO THESE ADJACENT FAILING COLUMN COUNTS.

IF THE ROW COUNT DOES NOT REACH > 10 AFTER SEARCHING THRU THE ERROR MAP THEN THIS ROW IS NOT SELECTED FOR BLASTING AT THIS TIME AND THE ADJACENT FAILING COLUMN COUNTS ARE NOT DECREMENTED.

THE ERROR MAP IS THEN SEARCHED AT THE NEXT ROW NUMBER.

COLUMN POINTER TABLE
AS MENTIONED BEFORE THIS TABLE POINTS TO THE ADJACENT FAILING COLUMN NUMBERS WHICH FAILED WITH THE ROW SEARCH.

REMAINDER TABLE
THE PROGRAM EXECUTION WILL FIRST SEARCH THE ERROR MAP FOR ROWS

WITH GREATER THAN 10 COUNTS AND THE ADJACENT FAILING COLUMNS COUNTS ARE INCREMENTED. NOW ONCE THE ERROR MAP HAS BEEN SEARCH FOR ROWS COUNTS > 10 THEN COLUMN COUNT TABLE IS SEARCHED FOR COLUMN COUNTS > 10. IF A COLUMN COUNT IS > 10 THEN THIS COLUMN NUMBER IS SELECTED FOR BLASTING AND ITS COUNT IS ZEROED.

ALL THAT IS LEFT NOW ARE RANDOM FAILING ROWS AND COLUMNS SCATTERED THROUGH THE CHIP. THESE SCATTERED ROW COLUMN PAIRS ARE TRANSFERED INTO THE REMAINDER TABLE WHERE THEY CAN BE INTERIGATED AND SELECTED FOR BLASTING.

COLUMN COUNT TABLE

COL 0
COL 1
COL 2
COL 3
COL 4
COL 5
COL 6
COL 7
COL 8
COL 9
COL 10
COL 11
COL XX

REMAINDER TABLE

ROW ADRS	COL ADRS

ROW COUNT

COLUMN COUNT POINTER TABLE

POINTER 0
POINTER 1
POINTER 2

```
|             | POINTER 3
|-------------|
|             | POINTER XX
|-------------|
```

BAD CHIP TABLE

THE BAD CHIP TABLE STORES AWAY THE FAILING CHIPS DISCOVERED BAD IN A BANK DURING WRITE CHECK TRANSFERS AND DURING SINGLE STEP DMA MODES. THE FAILING DATA PATTERN FOR EACH CHIP IS ALSO STORED.

THIS TABLE IS A BLOCK OF 39 BYTES AND THE BIT DEFINITIONS ARE AS FOLLOWS:

BIT 8 IS A FAULT INDICATOR
BIT 7 INDICATES ZEROES DATA FAILURE
BIT 6 INDICATES ONES DATA FAILURE
BIT 5 INDICATES RANDOM DATA 0 FAILURE
BIT 4 INDICATES RANDOM DATA 1 FAILURE
BIT 3 INDICATES RANDOM DATA 2 FAILURE
BIT 2 INDICATES RANDOM DATA 3 FAILURE
BIT 1 INDICATES RANDOM DATA 4 FAILURE
BIT 0 INDICATES RANDOM DATA 5 FAILURE

BAD CHIP TABLE

```
|-------|
|       | CHIP 0
|-------|
|       | CHIP 1
|-------|
|       | CHIP 2
|-------|
|       | CHIP 3
|-------|
|       | CHIP 4
|-------|
|       | CHIP 5
|-------|
|       | CHIP XX
|-------|
|       | CHIP 38
|-------|
```

7.0 TEST SUMMARIES

THIS PROM MAINTENANCE PROGRAM CONTAINS ONLY ONE TEST. THIS ONE TEST IS THE MAIN CONTROL LOOP WHICH LOOPS THE PROGRAM EXECUTION ON THE SELECTED ARRAYS AND BANKS FOR PROM MAINTENANCE.

)%
ELJDOM

```
:          0001  MODULE BSKEL2 ( IDENT - 'REV A PATCH 00') -
:          0002  BEGIN
:          0003  REQUIRE 'BLSMAC.REQ';
:          1493  %SBTTL  'PROGRAM HEADER'
:          1494
:          1495
:          1496  LITERAL
:          1497     DS$NBR_OF_TESTS = 1;
:          1498
:          1499
:        C 1500  %(
:        C 1501  ;++
:        C 1502  ; THE PROGRAM HEADER IS THE INTERFACE BETWEEN
:        C 1503  ; THE DIAGNOSTIC PROGRAM AND THE SUPERVISOR.
:        C 1504  ;--
:          1505  )%
:          1506
:          1507  EQUALS;
:          1508
:          1509  POINTER (ALL);
:          1510
:          1511  HEADER  (%ASCII'CZMLCA',%ASCII'A',%ASCII'0',120,0,PRI00);
:          1512
:          1513
:          1514  %SBTTL  'DISPATCH TABLE'
:          1515
:        C 1516  %(
:        C 1517  ;++
:        C 1518  ; THE DISPATCH TABLE CONTAINS THE STARTING ADDRESS OF EACH TEST.
:        C 1519  ; IT IS USED BY THE SUPERVISOR TO DISPATCH TO EACH TEST.
:        C 1520  ;--
:          1521  )%
:          1522
:          1523  DISPATCH (DS$NBR_OF_TESTS);
:          1524
:          1525
:          1526  ERRTBL;
:          1527
:          1528
:          1529
:          1530
:          1531
:          1532
:          1533
:          1534  %SBTTL  'DEFAULT HARDWARE P-TABLE'
:          1535
:        C 1536  %(
:        C 1537  ;++
:        C 1538  ; THE DEFAULT HARDWARE P-TABLE CONTAINS DEFAULT VALUES OF
:        C 1539  ; THE TEST-DEVICE PARAMETERS.  THE STRUCTURE OF THIS TABLE
:        C 1540  ; IS IDENTICAL TO THE STRUCTURE OF THE HARDWARE P-TABLES,
:        C 1541  ; AND IS USED AS A ''TEMPLATE'' FOR BUILDING THE P-TABLES.
```

```
        ( 1542  ;--
          1543  )%
          1544
          1545  BGNHW   (DFPTBL);
          1546
          1547          GLOBAL
          1548                  A_HWTBL : INITIAL (0),          .MASK ENTIRE SYSTEM FLAG INDICATOR
          1549                  B_HWTBL : INITIAL (0),          .MASK A SINGLE ARRAY FLAG INDICATOR
          1550                  C_HWTBL : INITIAL (0),          .MASK A SINGLE BANK FLAG INDICATORG
          1551                  D_HWTBL : INITIAL (0),          .BANK NO. TO BE MASKED
          1552                  E_HWTBL : INITIAL (0),          .ARRAY NO. TO BE MASKED
          1553                  F_HWTBL : INITIAL (%O'176400'), .RH REGISTER BASE ADRS
          1554                  G_HWTBL : INITIAL (0),          !DRIVE SELECT NO. OF DRIVE TO MASKED
          1555                  H_HWTBL : INITIAL (1),          .OPTION,  1 = 16K MOS RAMS, 0 = 64K MOS RAM
          1556                  I_HWTBL: INITIAL (1);           .PARAMETERS CORRECT INDICATOR
          1557  ENDHW;
          1558  %SBTTL 'SOFTWARE P-TABLE'
          1559
        C 1560  %(
        C 1561  ;++
        C 1562  ;  THE SOFTWARE TABLE CONTAINS VARIOUS DATA USED BY THE
        C 1563  ;  PROGRAM AS OPERATIONAL PARAMETERS.  THESE PARAMETERS ARE
        C 1564  ;  SET UP A ASSEMBLY TIME AND MAY BE VARIED BY THE OPERATOR
        C 1565
        C 1566  ;  AT RUN TIME.  HOWEVER THIS PROGRAM WILL NOT USE THIS OPTIONAL
        C 1567  ;  SECTION.
        C 1568  ;--
          1569  )%
          1570  .
          1571  !
          1572  ! THERE ARE NO SOFTWARE QUESTIONS USED
          1573  ! DURING THIS PROGRAM.  HOWEVER THIS
          1574  ! SOFTWARE TABLE LOCATION IS PROVIDED
          1575  ! IN THE EVENT OF A 'YES' RESPONCE TO
          1576  ! THE DRS> SOFTWARE QUESTION 'Change Software'
          1577  !
          1578  BGNSW (SFPTBL);
          1579
          1580  GLOBAL
          1581      SWTBL$_RET : INITIAL (0);                   !DUMMY ARGUMENT
          1582
          1583  ENDSW;
          1584
          1585
          1586
          1587  %SBTTL  'PROTECTION TABLE'
          1588
        C 1589  %(
        C 1590  ;++
        C 1591  ; THIS TABLE IS USED BY THE RUNTIME SERVICES
        C 1592  ; TO PROTECT THE LOAD MEDIA.
        C 1593  ;--
```

```
;         1594  )%
;         1595
;         1596  BGNPROT(-1,-1,-1);
;         1597
;         1598
;         1599          .1ST ARG =      OFFSET INTO P-TABLE FOR CSR ADDRESS
;         1600          .2ND ARG =      OFFSET INTO P-TABLE FOR MASSBUS ADDRESS
;         1601          .3RD ARG =      OFFSET INTO P-TABLE FOR DRIVE NUMBER
;         1602
;         1603  ENDPROT;
;         1604
;         1605
;         1606  END
;         1607  ELUDOM
```

```
                                .TITLE  BSKEL2
                                .IDENT  /REV A /

                                .PSECT  $CODE$
000000                  L$NAME::.ASCII  /CZM/
000000  103     132     115             .ASCII  /LCA/
000003  114     103     101             .BYTE   0
000006  000                             .BYTE   0
000007  000                     L$REV::
000010                                  .ASCII  /A/
000010  101                             .ASCII  /0/
000011  060                     L$UNIT::.WORD   T$PTHV
000012  000000G                 L$TIML::.WORD   170
000014  000170                  L$HPCP::.WORD   L$HARD
000016  000000G                 L$SPCP::.WORD   L$SOFT
000020  000000G                 L$HPTP::.WORD   L$HW
000022  000140'                  L$SPTP::.WORD   L$SW
000024  000166'                  L$LADP::.WORD   L$LAST
000026  000000G                  L$STA:: .WORD   0
000030  000000                   L$CO::  .WORD   0
000032  000000                   L$DT P::.WORD   0
000034  000000                   L$APT:: .WORD   0
000036  000000                   L$DTP:: .WORD   L$DISPATCH
000040  000124'                  L$PRIO::.WORD   0
000042  000000                   L$ENVI::.WORD   0
000044  000000                   L$EXP1::.WORD   0
000046  000000                   L$MREV::
000050                                  .BYTE   3
000050  003                             .BYTE   3
000051  003                     L$EF::  .WORD   0
000052  000000                          .WORD   0
000054  000000                   L$SPC:: .WORD   0
000056  000000                   L$DEVP::.WORD   L$DVTYP
000060  000000G                  L$REPP::.WORD   L$RPT
000062  000000G                  L$EXP4::.WORD   0
000064  000000
```

```
000066  000000          L$EXP5::.WORD   0
000070  000000G         L$AUT:: .WORD   L$AU
000072  000000G         L$DUT:: .WORD   L$DU
000074  000000          L$LUN:: .WORD   0
000076  000000G         L$DESP::.WORD   L$DESC
000100  104035          L$LOAD::.WORD   -73743
000102  000126'         L$ETP:: .WORD   L$ERRTBL
000104  000000G         L$ICP:: .WORD   L$INIT
000106  000000G         L$CCP:: .WORD   L$CLEAN
000110  000000G         L$ACP:: .WORD   L$AUTO
000112  000172'         L$PRT:: .WORD   L$PROT
000114  000000          L$TEST::.WORD   0
000116  000000          L$DLY:: .WORD   0
000120  000000          L$HIME::.WORD   0
000122  000001          D$PCNT::.WORD   1
000124  000000G         L$DISPATCH::
                                .WORD   T1
000126                  ERRTYP::.BLKW   1
000130                  ERRNBR::.BLKW   1
000132                  ERRMSG::.BLKW   1
000134                  ERRBLK::.BLKW   1
000136  000000C         L$HWLEN::
                                .WORD   <<L$NDHW-L$HWLEN>/2>
000140  000000          A.HWTBL::
                                .WORD   0
000142  000000          B.HWTBL::
                                .WORD   0
000144  000000          C.HWTBL::
                                .WORD   0
000146  000000          D.HWTBL::
                                .WORD   0
000150  000000          E.HWTBL::
                                .WORD   0
000152  176400          F.HWTBL::
                                .WORD   -1400
000154  000000          G.HWTBL::
                                .WORD   0
000156  000001          H.HWTBL::
                                .WORD   1
000160  000001          I.HWTBL::
                                .WORD   1
000162                  L$NDHW::.BLKW   1
000164  000000C         L$SWLEN::
                                .WORD   <<L$NDSW-L$SWLEN>/2>
000166  000000          SWTBL$.RET::
                                .WORD   0
000170                  L$NDSW::.BLKW   1
000172  177777          L$PROT::.WORD   -1
000174  177777                  .WORD   -1
000176  177777                  .WORD   -1
```

```
                        .GLOBL  L$SOFT, T$PTHV, L$RPT, L$INIT
                        .GLOBL  L$CLEAN, L$LAST, L$HARD, L$DVTYP
                        .GLOBL  L$DESC, L$DU, L$AU, L$AUTO, T1

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
001000                  BIT9==          1000
000400                  BIT8==          400
000200                  BIT7==          200
000100                  BIT6==          100
000040                  BIT5==          40
000020                  BIT4 =          20
000010                  BIT3==          10
000004                  BIT2==          4
000002                  BIT1==          2
000001                  BIT0==          1
000040                  EF.START==      40
000037                  EF.RESTART==    37
000036                  EF.CONTINUE=    36
000035                  EF.NEW==        35
000034                  EF.PWR==        34
000340                  PRI07==         340
000300                  PRI06==         300
000240                  PRI05==         240
000200                  PRI04 =         200
000140                  PRI03 =         140
000100                  PRI02 =         100
000040                  PRI01==         40
000000                  PRI00==         0
000004                  EVL==           4
000010                  LOT =           10
000020                  ADR==           20
000040                  IDJ==           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
        000126'                 L$ERRTBL==      ERRTYP
        000166'                 L$SW==          L$SWLEN+2
        000140'                 L$HW==          L$HWLEN+2
        000011'                 L$DEPO==        L$REV+1
        000140'                 DFPTBL==        L$HWLEN+2
        000166'                 SFPTBL==        L$SWLEN+2



; Size:          0 code + 64 data words
; Run Time:      00:04.2
; Elapsed Time: 00:08.4
; Memory Used:  29 pages
; Compilation Complete
```

```
          0001  MODULE BSKEL3 (IDENT = 'REV A PATCH 00') =
          0002  BEGIN
          0003  REQUIRE 'BLSMAC.REQ';
          1493  EQUALS;
          1494  %SBTTL 'TYPE AND DESCRIPTION'
          1495
          1496
          1497
          1498
          1499  DEVTYP  (%ASCIZ'ML-11 BLOCK MODE MEMORY SYS');
          1500
          1501
          1502  DESCRIPT (%ASCIZ'ML-11 ARRAY MAINT PROG');
          1503
          1504
          1505
          1506  %SBTTL 'HARDWARE PARAMETER CODING SECTION'
          1507
        C 1508  %(
        C 1509  ;++
        C 1510  ; THE HARDWARE PARAMETER CODING SECTION CONTAINS MACROS
        C 1511  ; THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES.  THE
        C 1512  ; MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
        C 1513  ; INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES.  THE
        C 1514  ; MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
        C 1515  ; WITH THE OPERATOR.
        C 1516  ;--
          1517  )%
          1518
          1519  BGNHRD:
          1520      BIND
          1521
          1522              ! DEFINE HARDWARE MESSAGES
          1523              !
          1524              M_SYS = UPLIT(%ASCIZ'IS ENTIRE ML-11 SYSTEM TO BE MASKED'),
          1525              M_ARR = UPLIT(%ASCIZ'IS A SINGLE ARRAY TO BE MASKED'),
          1526              M_BNK = UPLIT(%ASCIZ'IS A SINGLE BANK TO BE MASKED'),
          1527              M_BNK_NO = UPLIT(%ASCIZ'ENTER BANK NO. TO BE MASKED'),
          1528              M_BRD_NO = UPLIT(%ASCIZ'ENTER BOARD NO. TO BE MASKED'),
          1529              M_RH_BASE = UPLIT(%ASCIZ'STARTING RH BASE REGISTER ADDRESS'),
          1530              M_DUT = UPLIT(%ASCIZ'DUT DRIVE NUMBER'),
          1531              M_OPTION = UPLIT(%ASCIZ'IS DRIVE OPTION AN ML11A'),
          1532              M_CORRECT = UPLIT(%ASCIZ'ARE YOUR INPUTED PARAMETERS CORRECT');
          1533          !
          1534          ! SELECT WHICH OPTION THE PROGRAM IS
          1535          ! TO RUN UNDER.
          1536          !       OPTION 1. PROM MAINT ENTIRE ML11 SYSTEM
          1537          !       OPTION 2. PROM MAINT A SINGLE ARRAY
          1538          !       OPTION 3. PROM MAINT A SINGLE BANK
          1539          !
          1540          ! ASK AM I TO MASK AN ENTIRE SYSTEM ?
          1541          !
```

```
;          1542          GPRML(M_SYS,%O'0',1,NO,1);
;          1543          XFERT(DRV_PRAM);
;          1544          !
;          1545          ! ASK AM I TO MASK AN ENTIRE ARRAY ?
;          1546          !
;          1547          GPRML(M_ARR,%O'2',1,NO,1);
;          1548          XFERT(ENTER_BOARD);
;          1549          !
;          1550          ! ASK AM I TO MASK A SINGLE BANK ?
;          1551          !
;          1552          GPRML(M_BNK,%O'4',1,NO,1);
;          1553          !
;          1554          ! INPUT TO THE PROGRAM THE SELECTED
;          1555          ! PROGRAM RUN TIME PARAMETERS.
;          1556          !
;          1557          XFERF(DONE_HRD);
;          1558          !
;          1559          ! ENTIRE BANK TO BE PROM MAINT
;          1560          !
;          1561          GPRMD(M_BNK_NO,%O'6',D,%O'7',0,%DECIMAL'3',NO,1);
;          1562      $L(ENTER_BOARD);
;          1563          !
;          1564          ! ENTIRE BOARD TO BE PROM MAINT
;          1565          !
;          1566          GPRMD(M_BRD_NO,%O'10',D,%O'77',0,%DECIMAL'15',NO,1);
;          1567      $L(DRV_PRAM);
;          1568          !
;          1569          ! ENTIRE RH BASE ADDRESS
;          1570          ! ENTIRE DRIVE UNDER TEST
;          1571          ! ENTIRE DRIVE OPTION 16K PARTS OR 64K PARTS
;          1572          !
;          1573          GPRMA(M_RH_BASE,%O'12',0,0,%O'177777',YES,1);
;          1574          GPRMD(M_DUT,%O'14',0,%O'7',0,%O'77',NO,1);
;          1575          GPRML(M_OPTION,%O'16',1,NO,1);
;          1576          !
;          1577          ! FORCE THE OPERATOR TO REVIEW HIS INPUTED
;          1578          ! PARAMETERS FOR CORRECTNESS BY ASKING THE
;          1579          ! NEXT QUESTION.  THE INIT CODE WILL ABORT
;          1580          ! THE PROGRAM EXECUTION IF HE RETURNS A NO
;          1581          ! RESPONCE
;          1582          !
;          1583      $L(DONE_HRD);
;          1584          GPRML(M_CORRECT,%O'20',1,YES,1);
;          1585  ENDHRD;
;          1586
;          1587  %SBTTL 'SOFTWARE PARAMETER CODING SECTION'
;          1588
;        C 1589  %(
;        C 1590  ;++
;        C 1591  ;  THE SOFTWARE PARAMETER CODING SECTION CONTAINS MACROS
;        C 1592  ;  THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES.  THE
;        C 1593  ;  MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
```

```
;       C 1594  ;  INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES.  THE
;       C 1595  ;  MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
;       C 1596  ;  WITH THE OPERATOR.  HOWEVER THIS PROGRAM WILL NOT USE THIS SECTION.
;       C 1597  ;--
;         1598  )%
;         1599
;         1600  BGNSFT;
;         1601      BIND
;         1602
;         1603              ! DEFINE THE SOFTWARE MESSAGES
;         1604              !
;         1605              !
;         1606              !
;         1607              M_RET = UPLIT(%ASCIZ' NOT USED TYPE <CR>');
;         1608              !
;         1609              ! INPUT THE SOFTWARE QUESTION TO THE PROGRAM
;         1610              !
;         1611              GPRML(M_RET,0,1,YES,');
;         1612
;         1613  ENDSFT;
;         1614
;         1615  %SBTTL 'REPORT CODING SECTION'
;         1616
;         1617
;         1618
;       C 1619  %(
;       C 1620  !+
;       C 1621  ! THE REPORT CODING SECTION CONTAINS THE
;       C 1622  ! 'PRINTS' CALLS THAT GENERATE STATISTICAL REPORTS.
;       C 1623  !-
;         1624  )%
;         1625  ! THIS SECTION 'AUTO DROP' IS NOT USED
;         1626  ! DURING THIS PROGRAM
;         1627  !
;         1628  BGNRPT;
;         1629  RETURN;
;         1630  ENDRPT;


                                                .TITLE  BSKEL3
                                                .IDENT  /REV A /

000000                                           PSECT  $CODE$
000000      115     114     055         L$DVTYP::
                                                .ASCII  /ML-/
000003      061     061     040                 .ASCII  /11 /
000006      102     114     117                 .ASCII  /BLO/
000011      103     113     040                 .ASCII  /CK /
000014      115     117     104                 .ASCII  /MOD/
000017      105     040     115                 .ASCII  /E M/
000022      105     115     117                 .ASCII  /EMO/
```

```
000025      122     131     040                 .ASCII  /RY /
000030      123     131     123                 .ASCII  /SYS/
000033      000                                 .ASCII  <00>
000034                                          .BLKB   2
000036      115     114     055     L$DESC::    .ASCII  /ML-/
000041      061     061     040                 .ASCII  /11 /
000044      101     122     122                 .ASCII  /ARR/
000047      101     131     040                 .ASCII  /AY /
000052      115     101     111                 .ASCII  /MAI/
000055      116     124     040                 .ASCII  /NT /
000060      120     122     117                 .ASCII  /PRO/
000063      107     000     000                 .ASCII  /G/<00><00>
000066  000000C                     L$HRDLN::
                                                .WORD   <<<L$NDHRD-L$HRDLN>/2>-1>
000070  000120                      GP$1::      .WORD   120
000072  000000'                                .WORD   M.SYS
000074  000001                                  .WORD   1
000076  000000C                     $DRV.PRAM:
                                                .WORD   <<<<$LDRV.PRAM-$DRV.PRAM>*400>+4>+20>
000100  001120                      GP$2::      .WORD   1120
000102  000044'                                .WORD   M.ARR
000104  000001                                  .WORD   1
000106  000000C                     $ENTER.BOARD:
                                                .WORD   <<<<$LENTER.BOARD-$ENTER.BOARD>*400>+4>+20>
000110  002120                      GP$3::      .WORD   2120
000112  000104'                                .WORD   M.BNK
000114  000001                                  .WORD   1
000116  000000C                     $DONE.HRD:
                                                .WORD   <<<<$LDONE.HRD-$DONE.HRD>*400>+4>+40>
000120  003042                      GP$4::      .WORD   3042
000122  000142'                                .WORD   M.BNK.NO
000124  000007                                  .WORD   7
000126  000000                                  .WORD   0
000130  000003                                  .WORD   3
000132  001004                      $LENTER.BOARD:
                                                .WORD   1004
000134  004042                      GP$5::      .WORD   4042
000136  000176'                                .WORD   M.BRD.NO
000140  000077                                  .WORD   77
000142  000000                                  .WORD   0
000144  000017                                  .WORD   17
000146  001004                      $LDRV.PRAM:
                                                .WORD   1004
000150  005031                      GP$6::      .WORD   5031
000152  000234'                                .WORD   M.RH.BASE
000154  000000                                  .WORD   0
000156  177777                                  .WORD   -1
000160  006022                      GP$7::      .WORD   6022
000162  000276'                                .WORD   M.DUT
000164  000007                                  .WORD   7
000166  000000                                  .WORD   0
000170  000077                                  .WORD   77
```

```
000172  007120                          GP$8::  .WORD   7120
000174  000320'                                 .WORD   M.OPTION
000176  000001                                  .WORD   1
000200  001004                          $LDONE.HRD:
                                                .WORD   1004
000202  010130                          GP$9::  .WORD   10130
000204  000352'                                 .WORD   M.CORRECT
000206  000001                                  .WORD   1
000210                                  L$NDHRD::
                                                .BLKW   1
000212  000000C                         L$SFTLN::
                                                .WORD   <<<L$NDSFT-L$SFTLN>/2>-1>
000214  000130                          GP$10:: .WORD   130
000216  000416'                                 .WORD   M.RET
000220  000001                                  .WORD   1
000222                                  L$NDSFT::
                                                .BLKW   1



000000                                          .PSECT  $PLIT$,  D
000000      111     123     040         P.AAA:  .ASCII  /IS /
000003      105     116     124                 .ASCII  /ENT/
000006      111     122     105                 .ASCII  /IRE/
000011      040     115     114                 .ASCII  / ML/
000014      055     061     061                 .ASCII  /-11/
000017      040     123     131                 .ASCII  / SY/
000022      123     124     105                 .ASCII  /STE/
000025      115     040     124                 .ASCII  /M T/
000030      117     040     102                 .ASCII  /O B/
000033      105     040     115                 .ASCII  /E M/
000036      101     123     113                 .ASCII  /ASK/
000041      105     104     000                 .ASCII  /ED/<00>
000044      111     123     040         P.AAB:  .ASCII  /IS /
000047      101     040     123                 .ASCII  /A S/
000052      111     116     107                 .ASCII  /ING/
000055      114     105     040                 .ASCII  /LE /
000060      101     122     122                 .ASCII  /ARR/
000063      101     131     040                 .ASCII  /AY /
000066      124     117     040                 .ASCII  /TO /
000071      102     105     040                 .ASCII  /BE /
000074      115     101     123                 .ASCII  'MAS/
000077      113     105     104                 .ASCII  /KED/
000102      000     000                         .ASCII  <00><00>
000104      111     123     040         P.AAC:  .ASCII  /IS /
000107      101     040     123                 .ASCII  /A S/
000112      111     116     107                 .ASCII  /ING/
000115      114     105     040                 .ASCII  /LE /
000120      102     101     116                 .ASCII  /BAN/
000123      113     040     124                 .ASCII  /K T/
000126      117     040     102                 .ASCII  /O B/
000131      105     040     115                 .ASCII  /E M/
```

```
000134  101  123  113                .ASCII  /ASK/
000137  105  104  000                .ASCII  /ED/<00>
000142  105  116  124        P.AAD:  .ASCII  /ENT/
000145  105  122  040                .ASCII  /ER /
000150  102  101  116                .ASCII  /BAN/
000153  113  040  116                .ASCII  /K N/
000156  117  056  040                .ASCII  /O. /
000161  124  117  040                .ASCII  /TO /
000164  102  105  040                .ASCII  /BE /
000167  115  101  123                .ASCII  /MAS/
000172  113  105  104                .ASCII  /KED/
000175  000                         .ASCII  <00>
000176  105  116  124        P.AAE:  .ASCII  /ENT/
000201  105  122  040                .ASCII  /ER /
000204  102  117  101                .ASCII  /BOA/
000207  122  104  040                .ASCII  /RD /
000212  116  117  056                .ASCII  /NO./
000215  040  124  117                .ASCII  / TO/
000220  040  102  105                .ASCII  / BE/
000223  040  115  101                .ASCII  / MA/
000226  123  113  105                .ASCII  /SKE/
000231  104  000  000                .ASCII  /D/<00><00>
000234  123  124  101        P.AAF:  .ASCII  /STA/
000237  122  124  111                .ASCII  /RTI/
000242  116  107  040                .ASCII  /NG /
000245  122  110  040                .ASCII  /RH /
000250  102  101  123                .ASCII  /BAS/
000253  105  040  122                .ASCII  /E R/
000256  105  107  111                .ASCII  /EGI/
000261  123  124  105                .ASCII  /STE/
000264  122  040  101                .ASCII  /R A/
000267  104  104  122                .ASCII  /DDR/
000272  105  123  123                .ASCII  /ESS/
000275  000                         .ASCII  <00>
000276  104  125  124        P.AAG:  .ASCII  /DUT/
000301  040  104  122                .ASCII  / DR/
000304  111  126  105                .ASCII  /IVE/
000307  040  116  125                .ASCII  / NU/
000312  115  102  105                .ASCII  /MBE/
000315  122  000  000                .ASCII  /R/<00><00>
000320  111  123  040        P.AAH:  .ASCII  /IS /
000323  104  122  111                .ASCII  /DRI/
000326  126  105  040                .ASCII  /VE /
000331  117  120  124                .ASCII  /OPT/
000334  111  117  116                .ASCII  /ION/
000337  040  101  116                .ASCII  / AN/
000342  040  115  114                .ASCII  / ML/
000345  061  061  101                .ASCII  /11A/
000350  000  000                     .ASCII  .00><00>
000352  101  122  105        P.AAI:  .ASCII  /ARE/
000355  040  131  117                .ASCII  / YO/
000360  125  122  040                .ASCII  /UR /
```

```
000363  111  116  120                .ASCII  /INP/
000366  125  124  105                .ASCII  /UTE/
000371  104  040  120                .ASCII  /D P/
000374  101  122  101                .ASCII  /ARA/
000377  115  105  124                .ASCII  /MET/
000402  105  122  123                .ASCII  /ERS/
000405  040  103  117                .ASCII  / CO/
000410  122  122  105                .ASCII  /RRE/
000413  103  124  000                .ASCII  /CT/<00>
000416  040  116  117      P.AAJ:    .ASCII  / NO/
000421  124  040  125                .ASCII  /T U/
000424  123  105  104                .ASCII  /SED/
000427  040  124  131                .ASCII  / TY/
000432  120  105  040                .ASCII  /PE /
000435  074  103  122                .ASCII  /<CR/
000440  076  000                     .ASCII  />/<00>

        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
        001000          BIT9==           1000
        000400          BIT8==           400
        000200          BIT7==           200
        000100          BIT6==           100
        000040          BIT5==           40
        000020          BIT4==           20
        000010          BIT3==           10
        000004          BIT2==           4
        000002          BIT1==           2
        000001          BIT0==           1
        000040          EF.START=        40
        000037          EF.RESTART==     37
        000036          EF.CONTINUE==    36
        000035          EF.NEW==         35
        000034          EF.PWR==         34
        000340          PRI07--          340
        000300          PRI06--          300
        000240          PRI05--          240
```

```
000200                          PRI04==         200
000140                          PRI03=-         140
000100                          PRI02==         100
000040                          PRI01=-         40
000000                          PRI00==         0
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
C00070'                         L$HARD=-        L$HRDLN+2
000000'                         M.SYS=          P.AAA
000044'                         M.ARR=          P.AAB
000104'                         M.BNK=          P.AAC
000142'                         M.BNK.NO=       P.AAD
000176'                         M.BRD.NO=       P.AAE
000234'                         M.RH.BASE=      P.AAF
000276'                         M.DUT=          P.AAG
000320'                         M.OPTION=       P.AAH
000352'                         M.CORRECT=      P.AAI
000214'                         L$SOFT-         L$SFTLN+2
000416'                         M.RET-          P.AAJ


                                .SBTTL  LRPT REPORT CODING SECTION
000224                          .PSECT  $CODE$

000224  000207          LRPT:   RTS     PC                      ;                       1613

; Routine Size:  1 word
; Maximum stack depth per invocation:  0 words



                                .SBTTL  L$RPT REPORT CODING SECTION
000226  004767  177772  L$RPT:: JSR     PC,LRPT                 ;                       1629
000232  104425                  TRAP    25
000234  000207                  RTS     PC

; Routine Size:  4 words
; Maximum stack depth per invocation:  0 words
;        1631
```

```
;          1632  %SBTTL  'AUTODROP SECTION'
;          1633
;        C 1634  %(
;        C 1635  !+
;        C 1636  ! THIS CODE IS EXECUTED IMMEDIATELY AFTER THE INITIALIZE CODE IF
;        C 1637  ! THE "ADR" FLAG WAS SET.  THE UNIT(S) UNDER TEST ARE CHECKED TO
;        C 1638  ! SEE IF THEY WILL RESPOND.  THOSE THAT DON'T ARE IMMEDIATELY
;        C 1639  ! DROPPED FROM TESTING.
;        C 1640  !-
;          1641  )%
;          1642
;          1643  ! THIS SECTION 'AUTO DROP' IS NOT USED
;          1644  ! DURING THIS PROGRAM
;          1645  !
;          1646  BGNAUTO:
;          1647  RETURN;
;          1648  ENDAUTO;


                                        .SBTTL  LAUTO AUTODPOP SECTION
000236  000207                  LAUTO:  RTS     PC                      ;                       1630

; Routine Size:  1 word
; Maximum stack depth per invocation:  0 words


                                        .SBTTL  L$AUTO AUTODROP SECTION
000240  004767  177772          L$AUTO::JSR     PC,LAUTO                ;                       1647
000244  104461                          TRAP    61
000246  000207                          RTS     PC

; Routine Size:  4 words
; Maximum stack depth per invocation:  0 words


;          1649
;          1650
;          1651  %SBTTL  'DROP UNIT SECTION'
;          1652
;        C 1653  %(
;        C 1654  !+
;        C 1655  ! THE DROP-UNIT SECTION CONTAINS THE CODING THAT CAUSES A DEVICE
;        C 1656  ! TO NO LONGER BE TESTED.
;        C 1657  !-
;          1658  )%
;          1659  . THIS SECTION 'DROP UNIT' IS NOT USED
;          1660  . DURING THIS PROGRAM
;          1661  .
;          1662  BGNDU:
;          1663  RETURN;
```

```
;         1664  ENDDU;


                                                .SBTTL  LDU DROP UNIT SECTION
000250  000207                          LDU:    RTS     PC                      ;                       1648

; Routine Size:  1 word
; Maximum stack depth per invocation:  0 words



                                                .SBTTL  L$DU DROP UNIT SECTION
000252  004767  177772                  L$DU::  JSR     PC,LDU                  ;                       1663
000256  104453                                  TRAP    53
000260  000207                                  RTS     PC

; Routine Size:  4 words
; Maximum stack depth per invocation:  0 words


;         1665
;         1666
;         1667  %SBTTL  'ADD UNIT SECTION'
;         1668
;       ( 1669  %(
;       ( 1670  !+
;       ( 1671  ! THE ADD-UNIT SECTION CONTAINS ANY CODE THE PROGRAMMER WISHES
;       ( 1672  !  TO BE EXECUTED IN CONJUNCTION WITH THE ADDING OF A UNIT BACK
;       ( 1673  !  TO THE TEST CYCLE.
;       ( 1674  !-
;         1675  )%
;         1676  ! THIS SECTION 'ADD UNIT' IS NOT USED
;         1677  ! DURING THIS PROGRAM
;         1678  !
;         1679  BGNAU;
;         1680  RETURN;
;         1681  ENDAU;


                                                .SBTTL  LAU ADD UNIT SECTION
000262  000207                          LAU:    RTS     PC                      ;                       1664

; Routine Size:  1 word
; Maximum stack depth per invocation:  0 words



                                                .SBTTL  L$AU ADD UNIT SECTION
000264  004767  177772                  L$AU::  JSR     PC,LAU                  ;                       1680
000270  104452                                  TRAP    52
```

```
000272  000207                                  RTS     PC

; Routine Size:  4 words
; Maximum stack depth per invocation:  0 words


;          1682
;          1683
;          1684
;          1685  END
;          1686  ELUDOM




; Size:          20 code + 219 data words
; Run Time:      00:05.5
; Elapsed Time:  00:09.5
; Memory Used:   30 pages
; Compilation Complete
```

```
          0001  MODULE BSKEL4 ( IDENT - 'REV A PATCH 00') =
          0002
          0003  BEGIN
          0004  !
          0005  ! PRETTY BLF COMMANDS
          0006  !
          0007  .<BLF/NOFRROR>
          0008  .<BLF/LOWERCASE_KEY>
          0009  !
          0010  ! LIBRARY FILES
          0011  .
          0012  !
          0013  ! REQUIRE FILES
          0014  .
          0015
          0016  require 'BLSMAC.REQ';
          1506
        C 1507  %(
        C 1508  DUE TO THE BLISS-16 COMPILER BLISS16 COMPILERS  RESTRICTION OF 6 CHARACTERS PER UNIQUE VARIABLE NAME
        C 1509  THE FOLLOWING ROUTINE SUBSCRIPTS ARE USED:
        C 1510
        C 1511          ! ROUTINE SUBSCPIPTS
        C 1512          !
        C 1513          I_ROUTINE NAME - INITIALIZE
        C 1514          IN_ROUTINE NAME = INTERIGATE
        C 1515          L_ROUTINE NAME = LOAD
        C 1516          DM_ROUTINE NAME = DIAGNOSTIC MODE
        C 1517          GEN_ROUTINE NAME = GENERATE
        C 1518          FB_ROUTINE NAME = FIND BAD
        C 1519          CLR_ROUTINE NAME = CLEAR
        C 1520          OFF_ROUTINE NAME = OFFSETS
        C 1521          MOV_ROUTINE NAME = MOVE
        C 1522          CS_ROUTINE NAME = CHECK SUM
        C 1523          PM_ROUTINE NAME = PROM MAINTANENCE
        C 1524          CAL_ROUTINE NAME = CALCULATE
        C 1525          BL_ROUTINE NAME - BLAST
        C 1526          VER_ROUTINE NAME = VERIFY
        C 1527          S_ROUTINE NAME - SEARCH
        C 1528          X_ROUTINE NAME - TRANSFER
        C 1529          !
        C 1530          ! MISCELLANEOUS SUBSCRIPTS
        C 1531          !
        C 1532          F_MISCELANEOUS NAME   FLAG
          1533  )%
```

```
1534  %sbttl 'CONSTANT LITERAL DECLARATIONS'
1535
1536  ! CONSTANT LITERAL DECLARATIONS
1537  !
1538
1539  Literal
1540      !
1541      ! MASS BUS TRANSFER CODES
1542      !
1543      FUNC_1 = %o'000001',                    !NOOP FUNCTION
1544      FUNC_2 = %o'000011',                    !DRIVE CLEAR FUNCTION
1545      FUNC_3 = %o'000031',                    !SEARCH FUNCTION
1546      FUNC_4 = %o'000051',                    !WRITE CHECK FUNCTION
1547      FUNC_5 = %o'000061',                    !WRITE FUNCTION
1548      FUNC_6 = %o'000071',                    !READ FUNCTION
1549      !
1550      ! DATA CONSTANTS
1551      !
1552      ONES = %o'177777',                      !ALL ONES DATA FIELD
1553      ONE = %o'000001',                       !SINGLE ONE DATA FIELD
1554      ZEROES = %o'000000',                    !ALL ZEROES DATA FIE D
1555      ZERO = %o'000000',                      !ZERO DATA FIELD
1556      !
1557      ! REGISTER ACCESS OFFSET INDEXES
1558      !
1559      MLCS1 = 0,                              !ML_ADDR + %O'0' CONTROL AND STATUS REGISTER 1
1560      MLWC = 1,                               !ML_ADDR + %O'2' WORD COUNT REGISTER
1561      MLBA = 2,                               !ML_ADDR + %O'4' UNIBUS ADDRESS REGISTER
1562      MLDA = 3,                               !ML_ADDR + %O'6' DESIRED ADDRESS REGISTER
1563      MLCS2 = 4,                              !ML_ADDR + %O'10' CONTROL AND STATUS REGISTER 2
1564      MLDS = 5,                               !ML_ADDR + %O'12' DRIVE STATUS REGISTER
1565      MLER = 6,                               !ML_ADDR + %O'14' ERROR REGISTER
1566      MLAS = 7,                               !ML_ADDR + %O'16' ATTENTION SUMMARY REGISTER
1567      MLLA = 8,                               !ML_ADDR + %O'20' LOOK AHEAD REGISTER
1568      MLPA = 8,                               !ML_ADDR + %O'20' PROM ADDRESS REGISTER
1569      MLDB = 9,                               !ML_ADDR + %O'22' DATA BUFFER REGISTER
1570      MLMR = 10,                              !ML_ADDR + %O'24' MAINTENANCE REGISTER
1571      MLDT = 11,                              !ML_ADDR + %O'26' DRIVE TYPE REGISTER
1572      MLSN = 12,                              !ML_ADDR + %O'30' SERIAL NUMBER REGISTER
1573      MLE1 = 13,                              !ML_ADDR + %O'32' ECC CRC WORD REGISTER 1
1574      MLE2 = 14,                              !ML_ADDR + %O'34' ECC CRC WORD REGISTER 2
1575      MLD1 = 15,                              !ML_ADDR + %O'36' DATA DIAGNOSTIC REGISTER 1
1576      MLD2 = 16,                              !ML_ADDR + %O'40' DATA DIAGNOSTIC REGISTER 2
1577      MLEE = 17,                              !ML_ADDR + %O'42' ECC ERROR REGISTER
1578      MLEL = 18,                              !ML_ADDR + %O'44' ECC ERROR LOCATION REGISTER
1579      MLPD = 19,                              !ML_ADDR + %O'46' PROM DATA REGISTER
1580      MLBAE = 20,                             !ML_ADDR + %O'50' BUS ADDRESS E52TENSION REGISTER
1581      MLCS3 = 21,                             !ML_ADDR + %O'52' CONTRO_ AND STATUS REGISTER 3
1582      !
1583      ! ERROR NUMBER CONSTANTS                !**** ERROR LOCATION ****
1584      !
1585
```

```
1586        ERR_1 = 1,                                  .INIT CODE SECTION
1587        ERR_2 = 2,                                  .INIT CODE SECTION
1588        ERR_3 = 3,                                  !INIT CODE SECTION
1589        ERR_4 = 4,                                  .PM_THIS_BANK ROUTINE
1590        ERR_5 = 5,                                  .IN_ERROR_MAP ROUTINE
1591        ERR_6 = 6,                                  !IN_ERROR_MAP ROUTINE
1592        ERR_7 = 7,                                  .IN_BLAST_TBL ROUTINE
1593        ERR_8 = 8,                                  !VER_BLAST ROUTINE
1594        ERR_9 = 9,                                  .INIT CODE SECTION
1595        ERR_10 = 10,                                !BL_PROMS ROUTINE
1596        ERR_11 = 11,                                .VER_ERROR_MASK ROUTINE
1597        ERR_12 = 12,                                .VER_ERROR_MASK ROUTINE
1598        ERR_13 = 13,                                !MASS BUS TRANSFER ROUTINE
1599
1600        ! DESIRED SECTOR ADDRESS FIELD SELECT CONSTANT VALUES
1601        !
1602        BNK$SEL_SIZE = 2,                           . BANK SELECT SIZE EXPRESSION
1603        ARR$SEL_SIZE = 4,                           . ARRAY SELECT SIZE EXPRESSION
1604
1605        ! FLAG REGISTER SELECTION DEFINITION
1606        !
1607        F_FLG_ERR = 0,                              !GENERAL PURPOSE ERROR FLG
1608        F_UNC_ERR_FLG = 1,                          .INDICATES AN UNCORRECTABLE ERROR WAS DETECTED
1609        F_ERR_MAP_ENTERED = 2,                      !INDICATES THAT THE ERROR MAP HAS AN ENTRY
1610        F_BLST_TBL_ENTERED = 3,                     !INDICATES THAT THE BLAST TABLE WAS ENTERED
1611        F_ALL_BAD_CHIP = 4,                 .INDICATES THAT THIS BANK HAS ONE ALL BAD CHIP >14 ROW OR COL BAD
1612        F_RAND_DATA = 5,                            .INDICATES THAT RANDOM DATA PATTERN IS PRESENTLY USED
1613        F_ABORT_ARRAY = 6,                  .INDICATES THAT FURTHER TESTING OF THIS ARRAY IS TO BE ABORTED
1614        F_D_CLK_TIMR_OUT = 7,                       !INDICATES THAT THE DATA CLOCK BIT IS HUNG HIGH
1615        !
1616        ! BOOLEAN VALUES
1617        !
1618        TRUE = 1,                                   !LOGICAL TRUE INDICATOR
1619        FALSE = 0,                                  !LOGICAL FALSE INDICATOR
1620
1621        ! CRC DATA BIT DEFINITIONS
1622        !
1623        CRC_P = 36,                                 !ECC CRC CHIP 36
1624        CRC_A = 37,                                 !ECC CRC CHIP 37
1625        CRC_B = 38,                                 .ECC CRC CHIP 38
1626        CRC_NIBBLE = 9,                             !ECC CRC NIBBLE NINE
1627        !
1628        ! DELAY MACRO DELAY VARIABLES
1629        !
1630        ONE_US = 1,                                 !DELAY FOR ONE MICRO SECOND
1631        FIFTY_MS = 100,                             !DELAY FOR 50 MILLI SECONDS
1632        TEN_MS = 10000,                             !DELAY FOR 10 MILLI SECONDS
1633        !
1634        ! MISCELLANIOUS CONSTANTS
1635        !
1636        ML11A = 1,                                  !ML11A IS A 16K MOS RAM ARRAY
1637        SET_FLG = 1,                                !CONSTANT TO SET A FLAG
```

```
;        1638      CLR_FLG = 0,                    .CONSTANT TO SET A FLAG
;        1639      enaBE = 1,                      .ENABLE MLMR REGISTER FUNCTION
;        1640      DISABE - 0;                     .DISABLE MLMR REGISTER FUNCTION
;        1641
```

```
;          1642  %sbttl 'FIELD DECLARATIONS'
;          1643  !
;          1644  ! FIELD DECLARATIONS
;          1645  !
;          1646
;          1647  field
;          1648
;          1649      ! REMAINING ERROR TABLE STRUCTURE MAP
;          1650      !
;          1651      MAP_REM_TBL =
;          1652          set
;          1653          ROW = [8, 8, 0],                    !SELECT FAILING ROW NUMBER POSITION
;          1654          COL = [0, 8, 0],                    !SELECT FAILING COLUMN POSITION
;          1655          RO_CL = [0, 16, 0]                  !SELECT BOTH ROW AND COLUMN POSITION
;          1656          tes,
;          1657
;          1658      ! FAILING CHIP AND PATTERN STRUCTURE MAP
;          1659      !
;          1660      MAP_CHIP_TBL =
;          1661          set
;          1662          FAULT = [15, 1, 0],                 !SET WHEN THIS CHIP HAS ADDITIONAL FAILING ROW/COL
;          1663          PAT_0 = [14, 1, 0],                 !THIS CHIP FAILS ZEROES PATTERN
;          1664          PAT_1 = [13, 1, 0],                 !THIS CHIP FAILS ONES PATTERN
;          1665          RAN_1 = [12, 1, 0],                 !THIS CHIP FAILS RANDOM PAT 1
;          1666          RAN_2 = [11, 1, 0],                 !THIS CHIP FAILS RANDOM PAT 2
;          1667          RAN_3 = [10, 1, 0],                 !THIS CHIP FAILS RANDOM PAT 3
;          1668          RAN_4 = [9, 1, 0],                  !THIS CHIP FAILS RANDOM PAT 4
;          1669          RAN_5 = [8, 1, 0],                  !THIS CHIP FAILS RANDOM PAT 5
;          1670          RAN_6 = [7, 1, 0],                  !THIS CHIP FAILS RANDOM DATA 6
;          1671          RAN_7 = [6, 1, 0],                  !THIS CHIP FAILS RANDOM DATA 7
;          1672          RAN_8 = [5, 1, 0],                  !THIS CHIP FAILS RANDOM DATA 8
;          1673          RAN_9 = [4, 1, 0],                  !THIS CHIP FAILS RANDOM DATA 9
;          1674          RAN_10 = [3, 1, 0],                 !THIS CHIP FAILS RANDOM DATA 10
;          1675          RAN_11 = [2, 1, 0],                 !THIS CHIP FAILS RANDOM DATA 11
;          1676          RAN_12 = [1, 1, 0],                 !THIS CHIP FAILS RANDOM DATA 12
;          1677          RAN_13 = [0, 1, 0],                 !THIS CHIP FAILS RANDOM DATA 13
;          1678          PATS = [0, 15, 0],                  !SELECT ALL FAILING PATTERNS
;          1679          ALL = [0, 16, 0]                    !SELECT ALL BITS IN BYTE
;          1680          tes,
;          1681      !
;          1682      ! WRITE BUFFER STRUCTURE MAP
;          1683      !
;          1684      MAP_WRT_BUF =
;          1685          set
;          1686          BIT_0 = [0, 1, 0],                  !DEFINES BIT 0 OF WRITE BUFFER
;          1687          BIT_1 = [1, 1, 0],                  !DEFINES BIT 1 "     "     "
;          1688          BIT_2 = [2, 1, 0],                  !DEFINES BIT 2 "     "     "
;          1689          BIT_3 = [3, 1, 0],                  !DEFINES BIT 3 "     "     "
;          1690          BIT_4 = [4, 1, 0],                  !DEFINES BIT 4 "     "     "
;          1691          BIT_5 = [5, 1, 0],                  !DEFINES BIT 5 "     "     "
;          1692          BIT_6 = [6, 1, 0],                  !DEFINES BIT 6 "     "     "
;          1693          BIT_7 = [7, 1, 0],                  !DEFINES BIT 7 "     "     "
```

```
;          1694          BIT_8 = [8, 1, 0],                    !DEFINES BIT 8 ''    ''    ''
;          1695          BIT_9 = [9, 1, 0],                    !DEFINES BIT 9 ''    ''    ''
;          1696          BIT_10 = [10, 1, 0],                  .DEFINES BIT 10 ''    ''    ''
;          1697          BIT_11 = [11, 1, 0],                  !DEFINES BIT 11 ''    ''    ''
;          1698          BIT_12 = [12, 1, 0],                  !DEFINES BIT 12 ''    ''    ''
;          1699          BIT_13 = [13, 1, 0],                  !DEFINES BIT 13 ''    ''    ''
;          1700          BIT_14 = [14, 1, 0],                  .DEFINES BIT 14 ''    ''    ''
;          1701          BIT_15 = [15, 1, 0],                  .DEFINES BIT 15 ''    ''    ''
;          1702          WRD = [0, 16, 0]                      !DEFINES WORD OF THE WRITE BUFFER
;          1703          tes,
;          1704      !
;          1705      ! TEMPORARY BLAST TABLE STRUCTURE MAP
;          1706      !
;          1707      MAP_TMP_BLST_TBL =
;          1708          set
;          1709          NIB_NO = [0, 4, 0],                   !DEFINES FAILING CHIPS NIBBLE POSITION
;          1710          R_C_NO = [4, 8, 0],                   .DEFINES CHIPS FAILING ROW OR COLUMN
;          1711          R_C = [15, 1, 0]                      .SET = ROW ADDRESS, CLR = COLUMN ADDRESS
;          1712          tes,
;          1713      !
;          1714      ! RH / ML-11 REGISTER ACCESSING STRUCTRUE MAP
;          1715      !
;          1716      MAP_ML11_REG =
;          1717          set
;          1718          !
;          1719          ! MLCS1 CONTROL STAUTS REGISTER 1
;          1720          !
;          1721          SC = [15, 1, 0],                      !SPECIAL CONDITION
;          1722          TRE = [14, 1, 0],                     !TRANSFER ERROR
;          1723          MCPE = [13, 1, 0],                    !BUS PARITY ERROR
;          1724          DVA = [11, 1, 0],                     !DRIVE AVAILABLE
;          1725          RDY = [7, 1, 0],                      .DRIVE READY
;          1726          IE = [6, 1, 0],                       .INTERRUPT ENABLE
;          1727          GO = [0, 1, 0],                       !FUNTION GO BIT
;          1728          FUNC = [0, 6, 0],                     !FUNTION CODE
;          1729          !
;          1730          ! MLWC WORD COUNT REGISTER
;          1731          !
;          1732          WC_REG = [0, 16, 0],                  .WORD COUNT REGISTER
;          1733          !
;          1734          ! MLBA UNIBUS ADDRESS REGISTER
;          1735          !
;          1736          BA_REG = [0, 16, 0],                  !UNIBUS ADDRESS REGISTER
;          1737          !
;          1738          ! MLDA DESIRED ADDRESS REGISTER
;          1739          !
;          1740          DA_REG = [0, 16, 0],                  !DESIRED SECTOR REGISTER
;          1741          !
;          1742          ! MLCS2 CONTROL AND STATUS REGISTER 2
;          1743          !
;          1744          DLT = [15, 1, 0],                     .DATA LATE
;          1745          WCE = [14, 1, 0],                     .WRITE CHECK ERROR
```

```
1746            PE = [13, 1, 0],                    .UNIBUS PARITY ERROR
1747            NED = [12, 1, 0],                   !NON-EXISTENT DRIVE
1748            NEM = [11, 1, 0],                   .NON-EXISTENT MEMORY
1749            PGE = [10, 1, 0],                   !PROGRAM ERROR
1750            MXF = [9, 1, 0],                    .MISSED TRANFER
1751            MDPE = [8, 1, 0],                   .MASSBUS DATA BUS PARITY ERROR
1752            ORDY = [7, 1, 0],                   !OUTPUT READY
1753            IRDY = [6, 1, 0],                   !INPUT READY
1754            CLR = [5, 1, 0],                    !CONTROLLER CLEAR
1755            PAT = [4, 0],                       !PARITY TEST
1756            BAI = [3, 1, 0],                    .UNIBUS ADDRESS INCREMENT INHIBIT
1757            DRV_SEL = [0, 3, 0],                .UNIT SELECT
1758            !
1759            ! MLDS DRIVE STATUS REGISTER
1760            !
1761            ATTN = [15, 1, 0],                  .ATTENTION ACTIVE
1762            COMP_ERR = [14, 1, 0],              !ERROR SUMMARY
1763            MOL = [12, 1, 0],                   .MEDIUM ON LINE
1764            LBT = [10, 1, 0],                   .LAST BLOCK
1765            DPR = [8, 1, 0],                    !DRIVE PRESENT
1766            DRY = [7, 1, 0],                    .DRIVE READY
1767            VV = [6, 1, 0],                     !VOLUME VALID
1768            !
1769            ! MLER ERROR REGISTER
1770            !
1771            DCK = [15, 1, 0],                   .DATA CHECK
1772            UNS = [14, 1, 0],                   !DRIVE UNSAFE
1773            OPI = [13, 1, 0],                   .OPERATION INCOMPLETE
1774            IAE = [10, 1, 0],                   !INVALID ADDRESS ERROR
1775            AOE = [9, 1, 0],                    !ADDRESS OVERFLOW ERROR
1776            ECH_ERR = [6, 1, 0],                !ECC HARD ERROR
1777            DPAR = [5, 1, 0],                   !DATA PARITY ERROR
1778            CPAR = [3, 1, 0],                   !CONTROL PARITY ERROR
1779            RMR = [2, 1, 0],                    !REGISTER MOD ERFUSED ERROR
1780            ILR = [1, 1, 0],                    !ILLEGAL REGISTER ERROR
1781            ILF = [0, 1, 0],                    !ILLEGAL FUNCTION
1782            !
1783            ! MLAS ATTENTION SUMMARY REGISTER
1784            !
1785            ATTN_REG = [0, 16, 0],              !ATTENTION SUMMARY REGISTER
1786            !
1787            ! MLPA PROM ADDRESS REGISTER
1788            !
1789            PA_REG = [0, 16, 0],                !PROM ADDRESS REGISTER
1790            !
1791            ! MLMR MAINTENANCE REGISTER
1792            !
1793            SIZING = [11, 5, 0],                !SYSTEM SIZING
1794            TRT = [8, 2, 0],                    .TRANSFER RATE
1795            ARR_TYP = [10, 1, 0],               !ARRAY TYPE
1796            REF_MAR = [7, 1, 0],                .REFRESH MARGIN
1797            P_RW = [6, 1, 0],                   !PROM READ/WRITE
```

```
;          1798          P_DIS = [5, 1, 0],                  .PROM DISABLE
;          1799          D_CLK = [4, 1, 0],                  !DATA CLOCK
;          1800          D_DM = [3, 1, 0],                   !DATA DIAG MODE
;          1801          D_EN = [2, 1, 0],                   .DATA CHECK ENABLE
;          1802          E_DIS = [1, 1, 0],                  .ECC DISABLE MODE
;          1803          E_DM = [0, 1, 0],                   !ECC DIAG MODE
;          1804          MR_REG = [0, 16, 0],                 MAINTENANCE REGISTER
;          1805          !
;          1806          ! MLDT DRIVE TYPE REGISTER
;          1807          !
;          1808          DRVTYP = [1, 8, 0],                 !DRIVE TYPE
;          1809          DT_REG = [0, 1, 0],                 .DRIVE TYPE REGISTER
;          1810          !
;          1811          ! MLSN SERIAL NUMBER REGISTER
;          1812          !
;          1813          SERIAL_REG = [0, 16, 0],            .SERIAL NUMBER REGISTER
;          1814          !
;          1815          ! MLE1 ECC CRC WORD REGISTER 1
;          1816          !
;          1817          PAR_CRC_WRD = [8, 6, 0],            !PARITY CRC WORD
;          1818          CRC$A = [0, 6, 0],                  !CRC WORD A-A
;          1819          E1_REG = [0, 16, 0],                !ECC CRC REGISTER 1
;          1820          !
;          1821          ! MLE2 ECC CRC WORD REGISTER 2
;          1822          !
;          1823          B_32 = [8, 1, 0],                   !DATA BIT 32
;          1824          B_33 = [9, 1, 0],                   .DATA BIT 33
;          1825          B_34 = [10, 1, 0],                  !DATA BIT 34
;          1826          B_35 = [11, 1, 0],                  !DATA BIT 35
;          1827          B_36 = [12, 1, 0],                  !DATA BIT 36
;          1828          B_37 = [13, 1, 0],                  !DATA BIT 37
;          1829          B_38 = [14, 1, 0],                  !DATA BIT 38
;          1830          CRC$B = [0, 6, 0],                  .CRC WORD B-B
;          1831          E2_REG = [0, 16, 0],                !ECC CRC REGISTER 2
;          1832          !
;          1833          ! MLD1 DATA DIAGNOSTIC REGISTER 1
;          1834          !
;          1835          B_0 = [0, 1, 0],                    !DATA BIT 0
;          1836          B_1 = [1, 1, 0],                    !DATA BIT 1
;          1837          B_2 = [2, 1, 0],                    !DATA BIT 2
;          1838          B_3 = [3, 1, 0],                    !DATA BIT 3
;          1839          B_4 = [4, 1, 0],                    !DATA BIT 4
;          1840          B_5 = [5, 1, 0],                    !DATA BIT 5
;          1841          B_6 = [6, 1, 0],                    .DATA BIT 6
;          1842          B_7 = [7, 1, 0],                    !DATA BIT 7
;          1843          B_8 = [8, 1, 0],                    !DATA BIT 8
;          1844          B_9 = [9, 1, 0],                    !DATA BIT 9
;          1845          B_10 = [10, 1, 0],                  !DATA BIT 10
;          1846          B_11 = [11, 1, 0],                  !DATA BIT 11
;          1847          B_12 = [12, 1, 0],                  !DATA BIT 12
;          1848          B_13 = [13, 1, 0],                  !DATA BIT 13
;          1849          B_14 = [14, 1, 0],                  !DATA BIT 14
```

```
1850            B_15 = [15, 1, 0],                      !DATA BIT 15
1851            DB1_REG = [0, 16, 0],                   !DATA DIAG REGISTER 1
1852            !
1853            ! MLD2 DATA DIAGNOSTIC REGISTER 2
1854            !
1855            B_16 = [0, 15, 0],                      !DATA BIT 16
1856            B_17 = [1, 1, 0],                       .DATA BIT 17
1857            B_18 = [2, 1, 0],                       !DATA BIT 18
1858            B_19 = [3, 1, 0],                       .DATA BIT 19
1859            B_20 = [4, 1, 0],                       !DATA BIT 20
1860            B_21 = [5, 1, 0],                       .DATA BIT 21
1861            B_22 = [6, 1, 0],                       !DATA BIT 22
1862            B_23 = [7, 1, 0],                       !DATA BIT 23
1863            B_24 = [8, 1, 0],                       .DATA BIT 24
1864            B_25 = [9, 1, 0],                       !DATA BIT 25
1865            B_26 = [10, 1, 0],                      .DATA BIT 26
1866            B_27 = [11, 1, 0],                      !DATA BIT 27
1867            B_28 = [12, 1, 0],                      !DATA BIT 28
1868            B_29 = [13, 1, 0],                      .DATA BIT 29
1869            B_30 = [14, 1, 0],                      !DATA BIT 30
1870            B_31 = [15, 1, 0],                      .DATA BIT 31
1871            DB2_REG = [0, 16, 0],                   !DATA DIAG REGISTER 2
1872            !
1873            ! MLEE ECC ERROR REGISTER
1874            !
1875            UNC = [15, 1, 0],                       .UNCORRECTABLE ERROR
1876            SGL = [14, 1, 0],                       !SINGLE ERROR
1877            crc = [13, 1, 0],                       !CRC ERROR
1878            CHAN = [6, 6, 0],                       !CHANNEL IN ERROR
1879            E_Bit = [0, 6, 0],                      !ERROR FUNCTION
1880            !
1881            ! MLEL ECC ERROR LOCATION REGISTER
1882            !
1883            EL_REG = [0, 16, 0],                    !ERROR LOCATION REGISTER
1884            !
1885            ! MLPD PROM DATA REGISTER
1886            !
1887            PD_REG = [0, 16, 0],                    !PROM DATA REGISTER
1888            !
1889            ! MLBAE BUS ADDRESS EXTENTION REGISTER
1890            !
1891            BAE_REG = [0, 16, 0],                   .BUS ADDRESS EXTENTION REGISTER
1892            !
1893            ! MLCS3 CONTROL AND STATUS REGISTER 3
1894            !
1895            CS3_REG = [0, 16, 0],                   !CONTROL AND STATUS REGISTER 3
1896            !
1897            ! ML_ALL ACCESS ALL BITS IN SELECTED REGISTER
1898            !
1899            ML_ALL    [0, 16, 0]                    .ACCESS ALL REGISTER BITS
1900            tes;
1901
```

```
1902  %sbttl 'STRUCTURE DECLARATIONS'
1903  !
1904  ! STRUCTURE DECLARATIONS
1905  !
1906
1907  structure
1908      .
1909      . ML-11 REGISTER ACCESSING STRUCTURE
1910      .
1911      ! THIS STRUCTURE ALLOWS ML-11 REGISTER
1912      ! ACCESSING TO BE TRANSPORTABLE BETWEEN
1913      . THE PDP-11 AND VAX DIAGNOSTIC SUPERVISORS
1914      .
1915  !<BLF/NOFORMAT>
1916
1917
1918      RH [O, P, S, E] -
1919          begin
1920
1921          local
1922              ML_REG;
1923
1924          ML_REG = .(RH + %upval*O)<O, %bpval, O>;
1925          ML_REG
1926          end <P, S, E>;
1927
1928  !<BLF/FORMAT>
```

```
;          1929  %sbttl 'EXTERNAL DECLARATIONS'
;          1930
;          1931  ! EXTERNAL DECLARATIONS
;          1932  !
;          1933
;          1934  external routine
;          1935      !
;          1936      ! RANDOM NUMBER GENERATOR ROUTINE WRITTEN IN MACRO SOURCE CODE
;          1937      !
;          1938      RANGEN : novalue;
;          1939
;          1940  external
;          1941      !
;          1942      ! RANDOM NUMBER GENERATOR ROUTINE SEED VARIABLES
;          1943      !
;          1944      SEED1,                                  !FIRST WORD OF RANDOM NUMBER BUFFER
;          1945      SEED2,                                  !SECOND WORD OF RANDOM NUMBER BUFFER
;          1946      SEED3,                                  !THIRD WORD OF RANDOM NUMBER BUFFER
;          1947      RANDAT,                                 !LINKAGE WHICH RANDOM NUMBERS ARE RETURNED
;          1948
;          1949      ! DIAGNOSTIC SUPERVISOR GLOBAL VARIABLES
;          1950      !
;          1951      L$UNIT;                                 !SUPERVISOR STORAGE OF UNITS SELECTED
;          1952
```

```
1953  %sbttl 'OWN STORAGE DECLARATIONS'
1954
1955  ! OWN STORAGE DECLARATIONS
1956  !
1957
1958  own
1959
1960      ! ERROR MAP INTERIGATION STRUCTURES
1961      !
1962      COL_CNT_TBL : vector [256, byte] volatile,  !COLUMN COUNT TABLE
1963      REM_TBL : block [100] field (MAP_REM_TBL) volatile, !REMAINDER TABLE
1964      COPIED_REM_TBL : block [100] field (MAP_REM_TBL) volatile,  !COPIED REMAINDER TABLE
1965      COL_PTR : vector [10, byte] volatile,       !COLUMN COUNT TABLE POINTER
1966      !
1967      ! I/O BUFFERS
1968      !
1969      WRT_BUF : block [(128 + 511*2)] field (MAP_WRT_BUF) volatile,       !WRITE BUFFER
1970      RD_BUF : vector [256] volatile,             .READ BUFFER
1971      !
1972      ! FAILING ROW AND SECTOR STORAGE STRUCTURE
1973      !
1974      ERROR_MAP : blockvector [512, 8] volatile,  !ERROR MAP OF FAILING ROWS AND SECTORS
1975      !
1976      ! FAILING CHIPS AND PATTERN STORAGE STRUCTURE
1977      !
1978      CHIP_TBL : block [39] field (MAP_CHIP_TBL) volatile,        !FAILING CHIP TABLE
1979      !
1980      ! ROW & COL BLAST INFORMATION STORAGE STRUCTURE
1981      !
1982      TMP_BLST_TBL : block [10] field (MAP_TMP_BLST_TBL) volatile,        .TEMPORARY BLAST TABLE
1983      BLAST_TBL : blockvector [4, 512] volatile,  !ACTUAL BLAST TABLE
1984      COL_BASE,                                   !COLUMN ADRS BASE INDEX INTO BLAST TABLE
1985      !
1986      ! ML-11 ACCESSING STRUCTURE REFERANCE
1987      !
1988      ML_ADDR : ref RH field (MAP_ML11_REG) volatile,     .RH BASE REGISTER ADDRESS
1989      !
1990      ! INIT CODE GLOBAL VARIABLES
1991      !
1992      MAX_CHIP_COL,                               !NUMBER OF COLUMNS PER MOS RAM
1993      ML_DUT : volatile,                          !ML-11 DIVICE UNDER TEST
1994      ARRSBNK_SEL : volatile,             !FABRICATED DESIRED SECTOR ADRS WHERE ARRAY MAINT IS PERFORMED
1995      TSTED_BNK : volatile,                       !COUNT OF HOW MANY BANKS TO ARRAY MAINT
1996      INC_BNK : volatile,                         .BANK ADRS INCREMENT VALUE
1997      INC_ARR : volatile,                         .ARRAY ADRS INCREMENT VALUE
1998      ARR_SEL_POS : volatile,                     !FIELD SELECT  POSITION EXPRESSION
1999      BNK_SEL_POS : volatile,                     !FIELD SELECT POSITION EXPRESSION
2000      BNK_NUM_SEC : volatile,                     !INDICATES NUMBER OF SECTORS PER BANK. 16K - 128, 64K - 256
2001      !
2002      ! PROGRAM FLAG REGISTERS
2003      !
2004      BAD_BNK_REG : bitvector [8] volatile,       !INDICATES IF BANK HAS NEW ERRORS
```

```
;         2005      FLG_REG : bitvector [16] volatile,          .GENERAL FLAG REGISTER
;         2006      DEGRADE_MOD_REG : bitvector [8] volatile,   .DEGRADE ARRAY MODE F.AG REGISTER
;         2007
;         2008      ! MISCELLANEOUS VARIABLES
;         2009      !
;         2010      SIZE : volatile,                            !TRANSFERS WORD COUNT SIZE
;         2011      DST : volatile,                             .TRANSFERS DESTINATION ADDRESS
;         2012      SRC : volatile,                             .TRANSFERS SOURCE ADDRESS
;         2013      LST_TSTED_ARR : volatile,                   !LAST ARRAY MODULE NUMBER IN MI-11 SYSTEM
;         2014      RAND_PASS : volatile;                       .RANDOM DATA PASS VARIABLE
;         2015
;         2016  !
;         2017  ! SUPERVISOR GLOBAL EQUATES
;         2018  !
;         2019  EQUALS;
```

```
;        2020  %sbttl 'BIND DECLARATIONS'
;        2021  .
;        2022  . BIND DECLARATIONS
;        2023  .
;        2024
;        2025  bind
;        2026      !
;        2027      . ERROR AND RUN TIME MESSAGES
;        2028      .
;        2029      ILL_CMD_MSG = uplit (%asciz'ILLEGAL PROM MAINTENANCE PROGRAM COMMAND '),
;        2030      BGN_MSG = uplit (%asciz'STARTING PROM MAINTENANCE'),
;        2031      START_MSG = uplit (%asciz'TYPE START TO EXECUTE PROGRAM'),
;        2032      HQ_ERR_MSG = uplit (%asciz'HARDWARE QUESTIONS NOT ANSWERED CORRECTLY '),
;        2033      HQ_MSG = uplit (%asciz'BLASTING SYSTEM, ARRAY OR BANK NOT SELECTED '),
;        2034      END_MSG = uplit (%asciz'PROM MAINTANENCE COMPLETED '),
;        2035      SUPRES_MSG = uplit (%asciz'SUPPRESSING PROGRAM EXECUTION '),
;        2036      RET_DRS_MSG = uplit (%asciz'RETURNING TO DRS>'),
;        2037      LUN_MISS_MSG = uplit (%asciz'LUN 0 P-TABLE NOT PRESENT'),
;        2038      CON_A_MSG = uplit (%asciz'CONDITION A'),
;        2039      CON_B_MSG = uplit (%asciz'CONDITION B'),
;        2040      CON_C_MSG = uplit (%asciz'CONDITION C'),
;        2041      CON_D_MSG = uplit (%asciz'CONDITION D '),
;        2042      UNC_CHIP_MSG = uplit (%asciz'UNCONFIRMED FAILING CHIP'),
;        2043      GTR_MSG = uplit (%asciz'MORE THAN ONE UNIT SELECTED FOR PROM MAINT'),
;        2044      UNIT_SEL_MSG = uplit (%asciz'ALL UNITS EXCEPT UNIT 0 WILL BE IGNORED'),
;        2045      SW_BUG_MSG = uplit (%asciz'UNEXPECTED SOFTWARE BUG DETECTED NOTIFY DIAG ENG'),
;        2046      ABORT_MSG = uplit (%asciz'PROM MAINTANENCE ABORTED FOR THIS ARRAY'),
;        2047      NO_AD_MSG = uplit (%asciz'NO ADDITIONAL ERRORS BLASTED FOR THIS ARRAY'),
;        2048      UNC_ERR_MSG = uplit (%asciz'UNCORRECTABLE ERRORS DETECTED AFTER BLASTING'),
;        2049      TIME_OUT_MSG = uplit (%asciz'DATA CLOCK BIT FAILED TO RESET AFTER PROM WRITE'),
;        2050      REP_M7363_MSG = uplit (%asciz'REPLACE ARRAY DATA MODULE M7363'),
;        2051      MEM_ERR_MSG = uplit (%asciz'MEMORY ERRORS STILL EXIST AFTER BLASTING'),
;        2052      UNX_DRV_ERR_MSG = uplit (%asciz'UNEXPECTED DRIVE ERRORS DETECTED'),
;        2053      RD_XFER_MSG = uplit (%asciz'OCCURED DURING A MASS BUS READ TRANSFER'),
;        2054      WRT_XFER_MSG = uplit (%asciz'OCCURED DURING A MASS BUS WRITE TRANSFER'),
;        2055      WT_CHK_XFER_MSG = uplit (%asciz'OCCURED DURING A MASS BUS WRITE CHECK TRANSFER'),
;        2056      X_MSG = uplit (%asciz''),
;        2057
;        2058      ! PRINTING FORMATS
;        2059      .
;        2060      ONE_MSG = uplit (%asciz'%T%N'),                ! PRINT ONE MESSAGE
;        2061      TWO_MSG = uplit (%asciz'%T%T%N'),              .PRINT TWO MESSAGES
;        2062      THREE_MSG = uplit (%asciz'%T%T%T%N'),          . PRINT THREE MESAGES
;        2063      FOUR_MSG = uplit (%asciz'%T%T%T%T%N'),         . PRINT FOUR MESSAGES
;        2064      LF = uplit (%asciz'%N'),                       .PRINT A LINE FEED <CR>
;        2065      LFS = uplit (%asciz'%N%N'),                    ! PRINT TWO <CR>
;        2066      O_1_PRINT = uplit (%asciz'%T%O1%N'),           . PRINT OCTAL DIGIT AND A MESSAGE
;        2067      O_6_PRINT = uplit (%asciz'%T%O6%N'),           ! PRINT SIX OCTAL DIGITS AND A MESSAGE
;        2068      D_2_PRINT = uplit (%asciz'%T%D2%N'),           . PRINT A TWO DIGIT NUMBER AND A MESSAGE
;        2069      DRV_SEL_PRINT = uplit (%asciz'%ADRIVE %O1%A SELECTED%N'),
;        2070                                                     . PRINT THE DRIVE SELECTED NUMBER TO THE OPERATOR
;        2071      ARR_SEL_PRINT = uplit (%asciz'%S%S%S%S%S%S%S%S%S%ATESTING ARRAY %D2%N'),
```

```
2072                                                    ! PRINT THE ARRAY SELECTED TO THE OPERATOR
2073        BNK_SEL_PRINT = uplit (%asciz'%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%ABANK%S%S%S%D1%N'),
2074                                                    ! PRINT THE BANK PRESENTLY UNDER TEST TO THE OPERATOR
2075        A_B_C_PRINT = uplit (%asciz'%AARRAY%S%D2%S%ABANK%S%O1%S%ABIT #%S%D2%N'),
2076        A_B_N_PRINT = uplit (%asciz'%AARRAY%S%D2%S%ABANK%S%O1%S%ANIBBLE%S%D2%N'),
2077        REP_PRINT = uplit (%asciz'%AREPLACE ARRAY MODULE%S%D2%S%S%AOR RERUN PROGRAM%N'),
2078        A_B_PRINT = uplit (%asciz'%AARRAY%S%D2%S%ABANK%S%O1%N'),
2079        CS1_PRINT = uplit (%asciz'%S%S%S%S%AMLCS1%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%O6%N'),
2080        WC_PRINT = uplit (%asciz'%S%S%S%S%AMLWC%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%O6%N'),
2081        BA_PRINT = uplit (%asciz'%S%S%S%S%AMLBA%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%O6%N'),
2082        DA_PRINT = uplit (%asciz'%S%S%S%S%AMLDA%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%O6%N'),
2083        CS2_PRINT = uplit (%asciz'%S%S%S%S%AMLCS2%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%O6%N'),
2084        DS_PRINT = uplit (%asciz'%S%S%S%S%AMLDS%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%O6%N'),
2085        ER_PRINT = uplit (%asciz'%S%S%S%S%AMLER%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%O6%N'),
2086        AS_PRINT = uplit (%asciz'%S%S%S%S%AMLAS%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%O6%N'),
2087        MR_PRINT = uplit (%asciz'%S%S%S%S%AMLMR%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%O6%N'),
2088        DT_PRINT = uplit (%asciz'%S%S%S%S%AMLDT%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%O6%N'),
2089        SN_PRINT = uplit (%asciz'%S%S%S%S%AMLSN%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%O6%N'),
2090        EE_PRINT = uplit (%asciz'%S%S%S%S%AMLEE%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%O6%N'),
2091        EL_PRINT = uplit (%asciz'%S%S%S%S%AMLEL%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%S%O6%N'),
2092        HEADER_PRINT = uplit (%asciz'%S%S%S%S%S%S%S%S%S%S%AREGISTER  DUMP%N%N'),
2093        COL_DESC_PRINT   uplit (%asciz'%S%S%AREGISTER NAME%S%S%S%S%S%ACONTENTS%N'),
2094        X_PRINT   uplit (%asciz'');
2095
```

```
          2096  %sbttl 'MACRO DECLARATIONS'
          2097  !
          2098  ! MACRO DECLARATIONS
          2099  !
          2100
          2101  macro
          2102      !
          2103      ! ML-LL REGISTER ACCESSING MACRO
          2104      !
          2105      ! THIS MACRO ALLOWS ML-11 REGISTER ACCESSING TO BE
          2106      ! TRANSPORTABLE BETWEEN THE PDP-11 AND VAX DIAGNOSTIC
          2107      ! SUPERVISORS.
          2108      !
        M 2109      WRT_RH (O, FIELDNAM, IMAGE) =
        M 2110              begin
        M 2111
        M 2112              local
        M 2113                  MLREG;
        M 2114
        M 2115              MLREG = .ML_ADDR [O, ML_ALL];
        M 2116              MLREG<%fieldexpand (FIELDNAM)> = IMAGE;
        M 2117              (.ML_ADDR + %upval*O) = .MLREG;
          2118              end;%,
          2119      !
          2120      ! CLEAR RH AND MASS BUS DEVICES
          2121      !
        M 2122      CLR_MBUS =
        M 2123  WRT_RH (MLCS2, CLR,  ONE);!CLEAR THE MASS BUS
          2124  WRT_RH (MLCS2, DRV_SEL,  .ML_DUT);%,          .RESTORE THE DRIVE SELECT NUMBER
          2125      !
          2126      ! CLEAR MASS BUS DEVICES
          2127      !
        M 2128      CLR_DRIVE =
          2129  WRT_RH (MLCS1, FUNC,  FUNC_2);%,              !CLEAR THE DRIVE
          2130      !
          2131      ! MAINTANENCE REGISTER DIAG MODE SETTING MACROS
          2132      !
        M 2133      RD_PROM_MODE -
          2134  WRT_RH(MLMR,MR_REG,PM_RD_MODE);%,             .ENABLE PROM READS
        M 2135      WRT_PROM_MODE -
          2136  WRT_RH(MLMR,MR_REG,PM_WRT_MODE);%,            .ENABLE PROM WRITES
        M 2137      DAT_DM (X) =
          2138  WRT_RH (MLMR,D_DM,  X);%,                     .ENABLE/DISABLE DATA DIAG MODE
        M 2139      ECC_DIS (X) -
          2140  WRT_RH (MLMR,E_DIS,  X);%,                    !ENABLE/DISABLE ECC DISABLE MODE
        M 2141      PROM_RW (X) -
          2142  WRT_RH (MLMR,P_RW,  X);%,                     .ENABLE PROM READ WRITE
        M 2143      DAT_CLK -
          2144  WRT_RH (MLMR,D_CLK,  ONE);%,                  .DATA CLOCK
        M 2145      PROM_DIS (X) =
          2146  WRT_RH (MLMR,P_DIS,  X);%,                    !ENABLE PROM DISABLE
        M 2147      DCK_EN (X)
```

```
          2148  WRT_RH (MLMR,D_EN,  X);%,                  .ENABLE DATA CHECK ENABLE
        M 2149      ECC_DM (X)
          2150  WRT_RH (MLMR,E_DM,  X);%,                  .ENABLE ECC DISABLE
          2151      !
          2152      ! MISCELLANIOUS  MACROS DEFINITIONS
          2153      !
        M 2154      PM_RD_MODE =
          2155  %o'000040'%,                               .ENABLE PROM READ MODE
        M 2156      PM_WRT_MODE -
          2157  %o'000040'%,                               .ENABLE PROM WRITE MODE
        M 2158      FULL_WRD
          2159  0, 16, 0%;                                 .SELECTS FULL WORD OF DATA ELEMENT
          2160
```

```
;         2161  %sbttl 'EXTENDED MESSAGE PRINTING SECTION'
;         2162
;         2163  !++
;         2164  !  FUNCTIONAL DESCRIPTION: REGISTER DUMP
;         2165  !                          THIS BGNMSG WILL GET CALLED WHEN THE
;         2166  !                          DRIVE DETECTS ERRORS DURING A MASS
;         2167  !                          BUS TRANSFER.  THIS BGNMSG WILL PRINT
;         2168  !                          OUT TO THE TERMINAL THE ERROR REGISTERS.
;         2169  !--
;         2170
;         2171  BGNMSG (DUMPER);                              !PRINT THE DUMP MESSAGE

                                        .TITLE  BSKEL4
                                        .IDENT  /REV A /


000000                                  .PSECT  $PLIT$,  D
000000  111     114     114     P.AAA:  .ASCII  /ILL/
000003  105     107     101             .ASCII  /EGA/
000006  114     040     120             .ASCII  /L P/
000011  122     117     115             .ASCII  /ROM/
000014  040     115     101             .ASCII  / MA/
000017  111     116     124             .ASCII  /INT/
000022  105     116     101             .ASCII  /ENA/
000025  116     103     105             .ASCII  /NCE/
000030  040     120     122             .ASCII  / PR/
000033  117     107     122             .ASCII  /OGR/
000036  101     115     040             .ASCII  /AM /
000041  103     117     115             .ASCII  /COM/
000044  115     101     116             .ASCII  /MAN/
000047  104     040     000             .ASCII  /D /<00>
000052  123     124     101     P.AAB:  .ASCII  /STA/
000055  122     124     111             .ASCII  /RTI/
000060  116     107     040              ASCII  /NG /
000063  120     122     117             .ASCII  /PRO/
000066  115     040     115             .ASCII  /M M/
000071  101     111     116             .ASCII  /AIN/
000074  124     105     116             .ASCII  /TEN/
000077  101     116     103             .ASCII  /ANC/
000102  105     000                     .ASCII  /E/<00>
000104  124     131     120     P.AAC:  .ASCII  /TYP/
000107  105     040     123             .ASCII  /E S/
000112  124     101     122             .ASCII  /TAR/
000115  124     040     124             .ASCII  /T T/
000120  117     040     105             .ASCII  /O E/
000123  130     105     103             .ASCII  /XEC/
000126  125     124     105             .ASCII  /UTE/
000131  040     120     122             .ASCII  / PR/
000134  117     107     122             .ASCII  /OGR/
000137  101     115     000             .ASCII  /AM/<00>
000142  110     101     122     P.AAD:  .ASCII  /HAR/
```

```
000145  104  127  101                  .ASCII  /DWA/
000150  122  105  040                  .ASCII  /RE /
000153  121  125  105                  .ASCII  /QUE/
000156  123  124  111                  .ASCII  /STI/
000161  117  116  123                  .ASCII  /ONS/
000164  040  116  117                  .ASCII  / NO/
000167  124  040  101                  .ASCII  /T A/
000172  116  123  127                  .ASCII  /NSW/
000175  105  122  105                  .ASCII  /ERE/
000200  104  040  103                  .ASCII  /D C/
000203  117  122  122                  .ASCII  /ORR/
000206  105  103  124                  .ASCII  /ECT/
000211  114  131  040                  .ASCII  /LY /
000214  000  000                       .ASCII  <00><00>
000216  102  114  101          P.AAE:  .ASCII  /BLA/
000221  123  124  111                  .ASCII  /STI/
000224  116  107  040                  .ASCII  /NG /
000227  123  131  123                  .ASCII  /SYS/
000232  124  105  115                  .ASCII  /TEM/
000235  054  040  101                  .ASCII  /, A/
000240  122  122  101                  .ASCII  'RRA/
000243  131  040  117                  .ASCII  /Y O/
000246  122  040  102                  .ASCII  /R B/
000251  101  116  113                  .ASCII  /ANK/
000254  040  116  117                  .ASCII  / NO/
000257  124  040  123                  .ASCII  /T S/
000262  105  114  105                  .ASCII  /ELE/
000265  103  124  105                  .ASCII  /CTE/
000270  104  040  000                  .ASCII  /D /<00>
000273  000                           .ASCII  <00>
000274  120  122  117          P.AAF:  .ASCII  /PRO/
000277  115  040  115                  .ASCII  /M M/
000302  101  111  116                  .ASCII  /AIN/
000305  124  101  116                  .ASCII  /TAN/
000310  105  116  103                  .ASCII  /ENC/
000313  105  040  103                  .ASCII  /E C/
000316  117  115  120                  .ASCII  /OMP/
000321  114  105  124                  .ASCII  /LET/
000324  105  104  040                  .ASCII  /ED /
000327  000                           .ASCII  <00>
000330  123  125  120          P.AAG:  .ASCII  /SUP/
000333  120  122  105                  .ASCII  /PRE/
000336  123  123  111                  .ASCII  /SSI/
000341  116  107  040                  .ASCII  /NG /
000344  120  122  117                  .ASCII  /PRO/
000347  107  122  101                  .ASCII  /GRA/
000352  115  040  105                  .ASCII  /M E/
000355  130  105  103                  .ASCII  /XEC/
000360  125  124  111                  .ASCII  /UTI/
000363  117  116  040                  .ASCII  /ON /
000366  000  000                       .ASCII  <00><00>
000370  122  105  124          P.AAH:  .ASCII  /RET/
```

```
000373  125  122  116                .ASCII  /URN/
000376  111  116  107                .ASCII  /ING/
000401  040  124  117                .ASCII  / TO/
000404  040  104  122                .ASCII  / DR/
000407  123  076  000                .ASCII  /S>/<00>
000412  114  125  116          P.AAI:.ASCII  /LUN/
000415  040  060  040                .ASCII  / 0 /
000420  120  055  124                .ASCII  /P-T/
000423  101  102  114                .ASCII  /ABL/
000426  105  040  116                .ASCII  /E N/
000431  117  124  040                .ASCII  /OT /
000434  120  122  105                .ASCII  /PRE/
000437  123  105  116                .ASCII  /SEN/
000442  124  000                     .ASCII  /T/<00>
000444  103  117  116          P.AAJ:.ASCII  /CON/
000447  104  111  124                 ASCII  /DIT/
000452  111  117  116                .ASCII  /ION/
000455  040  101  000                .ASCII  / A/<00>
000460  103  117  116          P.AAK:.ASCII  /CON/
000463  104  111  124                .ASCII  /DIT/
000466  111  117  116                .ASCII  /ION/
000471  040  102  000                .ASCII  / B/<00>
000474  103  117  116          P.AAL:.ASCII  /CON/
000477  104  111  124                .ASCII  /DIT/
000502  111  117  116                .ASCII  /ION/
000505  040  103  000                .ASCII  / C/<00>
000510  103  117  116          P.AAM:.ASCII  /CON/
000513  104  111  124                .ASCII  /DIT/
000516  111  117  116                .ASCII  /ION/
000521  040  104  040                .ASCII  / D /
000524  000  000                     .ASCII  <00><00>
000526  125  116  103          P.AAN:.ASCII  /UNC/
000531  117  116  106                .ASCII  /ONF/
000534  111  122  115                .ASCII  /IRM/
000537  105  104  040                .ASCII  /ED /
000542  106  101  111                .ASCII  /FAI/
000545  114  111  116                .ASCII  /LIN/
000550  107  040  103                .ASCII  /G C/
000553  110  111  120                .ASCII  /HIP/
000556  000  000                     .ASCII  <00><00>
000560  115  117  122          P.AAO:.ASCII  /MOR/
000563  105  040  124                .ASCII  /E T/
000566  110  101  116                .ASCII  /HAN/
000571  040  117  116                .ASCII  / ON/
000574  105  040  125                .ASCII  /E U/
000577  116  111  124                .ASCII  /NIT/
000602  040  123  105                .ASCII  / SE/
000605  114  105  103                .ASCII  /LEC/
000610  124  105  104                .ASCII  /TED/
000613  040  106  117                .ASCII  / FO/
000616  122  040  120                .ASCII  /R P/
000621  122  117  115                .ASCII  /ROM/
```

```
000624  040  115  101            .ASCII  / MA/
000627  111  116  124            .ASCII  /INT/
000632  000  000                 .ASCII  <00><00>
000634  101  114  114  P.AAP:    .ASCII  /ALL/
000637  040  125  116            .ASCII  / UN/
000642  111  124  123            .ASCII  /ITS/
000645  040  105  130            .ASCII  / EX/
000650  103  105  120            .ASCII  /CEP/
000653  124  040  125            .ASCII  /T U/
000656  116  111  124            .ASCII  /NiT/
000661  040  060  040            .ASCII  / 0 /
000664  127  111  114            .ASCII  /WIL/
000667  114  040  102            .ASCII  /L B/
000672  105  040  111            .ASCII  /E I/
000675  107  116  117            .ASCII  /GNO/
000700  122  105  104            .ASCII  /RED/
000703  000                      .ASCII  <00>
000704  125  116  105  P.AAQ:    .ASCII  /UNE/
000707  130  120  105            .ASCII  /XPE/
000712  103  124  105            .ASCII  /CTE/
000715  104  040  123            .ASCII  /D S/
000720  117  106  124            .ASCII  /OFT/
000723  127  10   122            .ASCII  /WAR/
000726  105  040  102            .ASCII  /E B/
000731  125  107  040            .ASCII  /UG /
000734  104  105  124            .ASCII  /DET/
000737  105  103  124            .ASCII  /ECT/
000742  105  104  040            .ASCII  /ED /
000745  116  117  124            .ASCII  /NOT/
000750  111  106  131            .ASCII  /IFY/
000753  040  104  111            .ASCII  / DI/
000756  101  107  040            .ASCII  /AG /
000761  105  116  107            .ASCII  /ENG/
000764  000  000                 .ASCII  <00><00>
000766  120  122  117  P.AAR:    .ASCII  /PRO/
000771  115  040  115            .ASCII  /M M/
000774  101  111  116            .ASCII  /AIN/
000777  124  101  116            .ASCII  /TAN/
001002  105  116  103            .ASCII  /ENC/
001005  105  040  101            .ASCII  /E A/
001010  102  117  122            .ASCII  /BOR/
001013  124  105  104            .ASCII  /TED/
001016  040  106  117            .ASCII  / FO/
001021  122  040  124            .ASCII  /R T/
001024  110  111  123            .ASCII  /HIS/
001027  040  101  122            .ASCII  / AR/
001032  122  101  131            .ASCII  /RAY/
001035  000                      .ASCII  <00>
001036  116  117  040  P.AAS:    .ASCII  /NO /
001041  101  104  104            .ASCII  /ADD/
001044  111  124  111            .ASCII  /ITI/
001047  117  116  101            .ASCII  /ONA/
```

```
001052  114  040  105                  .ASCII  /L E/
001055  122  122  117                  .ASCII  /RRO/
001060  122  123  040                  .ASCII  /RS /
001063  102  114  101                  .ASCII  /BLA/
001066  123  124  105                  .ASCII  /STE/
001071  104  040  106                  .ASCII  /D F/
001074  117  122  040                  .ASCII  /OR /
001077  124  110  111                  .ASCII  /THI/
001102  123  040  101                  .ASCII  /S A/
001105  122  122  101                  .ASCII  /RRA/
001110  131  000                       .ASCII  /Y/<00>
001112  125  116  103          P.AAT:  .ASCII  /UNC/
001115  117  122  122                  .ASCII  /ORR/
001120  105  103  124                  .ASCII  /ECT/
001123  101  102  114                  .ASCII  /ABL/
001126  105  040  105                  .ASCII  /E E/
001131  122  122  117                  .ASCII  /RRO/
001134  122  123  040                  .ASCII  /RS /
001137  104  105  124                  .ASCII  /DET/
001142  105  103  124                  .ASCII  /ECT/
001145  105  104  040                  .ASCII  /ED /
001150  101  106  124                  .ASCII  /AFT/
001153  105  122  040                  .ASCII  /ER /
001156  102  114  101                  .ASCII  /BLA/
001161  123  124  111                  .ASCII  /STI/
001164  116  107  000                  .ASCII  /NG/<00>
001167  000                           .ASCII  <00>
001170  104  101  124          P.AAU:  .ASCII  /DAT/
001173  101  040  103                  .ASCII  /A C/
001176  114  117  103                  .ASCII  /LOC/
001201  113  040  102                  .ASCII  /K B/
001204  111  124  040                  .ASCII  /IT /
001207  106  101  111                  .ASCII  /FAI/
001212  114  105  104                  .ASCII  /LED/
001215  040  124  117                  .ASCII  / TO/
001220  040  122  105                  .ASCII  / RE/
001223  123  105  124                  .ASCII  /SET/
001226  040  101  106                  .ASCII  / AF/
001231  124  105  122                  .ASCII  /TER/
001234  040  120  122                  .ASCII  / PR/
001237  117  115  040                  .ASCII  /OM /
001242  127  122  111                  .ASCII  /WRI/
001245  124  105  000                  .ASCII  /TE/<00>
001250  122  105  120          P.AAV:  .ASCII  /REP/
001253  114  101  103                  .ASCII  /LAC/
001256  105  040  101                  .ASCII  /E A/
001261  122  122  101                  .ASCII  /RRA/
001264  131  040  104                  .ASCII  /Y D/
001267  101  124  101                  .ASCII  /ATA/
001272  040  115  117                  .ASCII  / MO/
001275  104  125  114                  .ASCII  /DUL/
001300  105  040  115                  .ASCII  /E M/
```

```
001303  067  063  066              .ASCII  /736/
001306  063  000                   .ASCII  /3/<00>
001310  115  105  115    P.AAW:    .ASCII  /MEM/
001313  117  122  131              .ASCII  /ORY/
001316  040  105  122              .ASCII  / ER/
001321  122  117  122              .ASCII  /ROR/
001324  123  040  123              .ASCII  /S S/
001327  124  111  114              .ASCII  /TIL/
001332  114  040  105              .ASCII  /L E/
001335  130  111  123              .ASCII  /XiS/
001340  124  040  101              .ASCII  /T A/
001343  106  124  105              .ASCII  /FTE/
001346  122  040  102              .ASCII  /R B/
001351  114  101  123              .ASCII  /LAS/
001354  124  111  116              .ASCII  /TIN/
001357  107  000  000              .ASCII  /G/<00><00>
001362  125  116  105    P.AAX:    .ASCII  /UNE/
001365  130  120  105              .ASCII  /XPE/
001370  103  124  105              .ASCII  /CTE/
001373  104  040  104              .ASCII  /D D/
001376  122  111  126              .ASCII  /RIV/
001401  105  040  105              .ASCII  /E E/
001404  122  122  117              .ASCII  /RRO/
001407  122  123  040              .ASCII  /RS /
001412  104  105  124              .ASCII  /DET/
001415  105  103  124              .ASCII  /ECT/
001420  105  104  000              .ASCII  /ED/<00>
001423  000                       .ASCII  <00>
001424  117  103  103    P.AAY:    .ASCII  /OCC/
001427  125  122  105              .ASCII  /URE/
001432  104  040  104              .ASCII  /D D/
001435  125  122  111              .ASCII  /URI/
001440  116  107  040              .ASCII  /NG /
001443  101  040  115              .ASCII  /A M/
001446  101  123  123              .ASCII  /ASS/
001451  040  102  125              .ASCII  / BU/
001454  123  040  122              .ASCII  /S R/
001457  105  101  104              .ASCII  /EAD/
001462  040  124  122              .ASCII  / TR/
001465  101  116  123              .ASCII  /ANS/
001470  106  105  122              .ASCII  /FER/
001473  000                       .ASCII  <00>
001474  117  103  103    P.AAZ:    .ASCII  /OCC/
001477  125  122  105              .ASCII  /URE/
001502  104  040  104              .ASCII  /D D/
001505  125  122  111              .ASCII  /URI/
001510  116  107  040              .ASCII  /NG /
001513  101  040  115              .ASCII  /A M/
001516  101  123  123              .ASCII  /ASS/
001521  040  102  125              .ASCII  / BU/
001524  123  040  127              .ASCII  /S W/
001527  122  111  124              .ASCII  /RIT/
```

```
001532  105  040  124                .ASCII  /E T/
001535  122  101  116                .ASCII  /RAN/
001540  123  106  105                .ASCII  /SFE/
001543  122  000  000                .ASCII  /R/<00><00>
001546  117  103  103        P.ABA:  .ASCII  /OCC/
001551  125  122  105                .ASCII  /URE/
001554  104  040  104                .ASCII  /D D/
001557  125  122  111                .ASCII  /URI/
001562  116  107  040                .ASCII  /NG /
001565  101  040  115                .ASCII  /A M/
001570  101  123  123                .ASCII  /ASS/
001573  040  102  125                .ASCII  / BU/
001576  123  040  127                .ASCII  /S W/
001601  122  111  124                .ASCII  /RIT/
001604  105  040  103                .ASCII  /E C/
001607  110  105  103                .ASCII  /HEC/
001612  113  040  124                .ASCII  /K T/
001615  122  101  116                .ASCII  /RAN/
001620  123  106  105                .ASCII  /SFE/
001623  122  000  000                .ASCII  /R/<00><00>
001626  000  000            P.ABB:  .ASCII  <00><00>
001630  045  124  045       P.ABC:  .ASCII  /%T%/
001633  116  000  000                .ASCII  /N/<00><00>
001636  045  124  045       P.ABD:  .ASCII  /%T%/
001641  124  045  116                .ASCII  /T%N/
001644  000  000                     .ASCII  <00><00>
001646  045  124  045       P.ABE:  .ASCII  /%T%/
001651  124  045  124                .ASCII  /T%T/
001654  045  116  000                .ASCII  /%N/<00>
001657  000                         .ASCII  <00>
001660  045  124  045       P.ABF:  .ASCII  /%T%/
001663  124  045  124                .ASCII  /T%T/
001666  045  124  045                .ASCII  /%T%/
001671  116  000  000                .ASCII  /N/<00><00>
001674  045  116  000       P.ABG:  .ASCII  /%N/<00>
001677  000                         .ASCII  <00>
001700  045  116  045       P.ABH:  .ASCII  /%N%/
001703  116  000  000                .ASCII  /N/<00><00>
001706  045  124  045       P.ABI:  .ASCII  /%T%/
001711  117  061  045                .ASCII  /01%/
001714  116  000                     .ASCII  /N/<00>
001716  045  124  045       P.ABJ:  .ASCII  /%T%/
001721  117  066  045                .ASCII  /06%/
001724  116  000                     .ASCII  /N/<00>
001726  045  124  045       P.ABK:  .ASCII  /%T%/
001731  104  062  045                .ASCII  /D2%/
001734  116  000                     .ASCII  /N/<00>
001736  045  101  104       P.ABL:  .ASCII  /%AD/
001741  122  111  126                .ASCII  /RIV/
001744  105  040  045                .ASCII  /E %/
001747  117  061  045                .ASCII  /01%/
001752  101  040  123                .ASCII  /A S/
```

```
001755  105  114  105                    .ASCII  /ELE/
001760  103  124  105                    .ASCII  /CTE/
001763  104  045  116                    .ASCII  /D%N/
001766  000  000                         .ASCII  <00><00>
001770  045  123  045          P.ABM:    .ASCII  /%S%/
001773  123  045  123                    .ASCII  /S%S/
001776  045  123  045                    .ASCII  /%S%/
002001  123  045  123                    .ASCII  /S%S/
002004  045  123  045                    .ASCII  /%S%/
002007  123  045  101                    .ASCII  /S%A/
002012  124  105  123                    .ASCII  /TES/
002015  124  111  116                    .ASCII  /TIN/
002020  107  040  101                    .ASCII  /G A/
002023  122  122  101                    .ASCII  /RRA/
002026  131  040  045                    .ASCII  /Y %/
002031  104  062  045                    .ASCII  /D2%/
002034  116  000                         .ASCII  /N/<00>
002036  045  123  045          P.ABN:    .ASCII  /%S%/
002041  123  045  123                    .ASCII  /S%S/
002044  045  123  045                    .ASCII  /%S%/
002047  123  045  123                    .ASCII  /S%S/
002052  045  123  045                    .ASCII  /%S%/
002055  123  045  123                    .ASCII  /S%S/
002060  045  123  045                    .ASCII  /%S%/
002063  123  045  123                    .ASCII  /S%S/
002066  045  123  045                    .ASCII  /%S%/
002071  123  045  123                    .ASCII  /S%S/
002074  045  123  045                    .ASCII  /%S%/
002077  101  102  101                    .ASCII  /ABA/
002102  116  113  045                    .ASCII  /NK%/
002105  123  045  123                    .ASCII  /S%S/
002110  045  123  045                    .ASCII  /%S%/
002113  104  061  045                    .ASCII  /D1%/
002116  116  000                         .ASCII  /N/<00>
002120  045  101  101          P.ABO:    .ASCII  /%AA/
002123  122  122  101                    .ASCII  /RRA/
002126  131  045  123                    .ASCII  /Y%S/
002131  045  104  062                    .ASCII  /%D2/
002134  045  123  045                    .ASCII  /%S%/
002137  101  102  101                    .ASCII  /ABA/
002142  116  113  045                    .ASCII  /NK%/
002145  123  045  117                    .ASCII  /S%O/
002150  061  045  123                    .ASCII  /1%S/
002153  045  101  102                    .ASCII  /%AB/
002156  111  124  040                    .ASCII  /IT /
002161  043  045  123                    .ASCII  /#%S/
002164  045  104  062                    .ASCII  /%D2/
002167  045  116  000                    .ASCII  /%N/<00>
002172  045  101  101          P.ABP:    .ASCII  /%AA/
002175  122  122  101                    .ASCII  /RRA/
002200  131  045  123                    .ASCII  /Y%S/
002203  045  104  062                    .ASCII  /%D2/
```

```
002206  045  123  045            .ASCII  /%S%/
002211  101  102  101            .ASCII  /ABA/
002214  116  113  045            .ASCII  /NK%/
002217  123  045  117            .ASCII  /S%O/
002222  061  045  123            .ASCII  /1%S/
002225  045  101  116            .ASCII  /%AN/
002230  111  102  102            .ASCII  /IBB/
002233  114  105  045            .ASCII  /LE%/
002236  123  045  104            .ASCII  /S%D/
002241  062  045  116            .ASCII  /2%N/
002244  000  000                 .ASCII  <00><00>
002246  045  101  122  P.ABQ:    .ASCII  /%AR/
002251  105  120  114            .ASCII  /EPL/
002254  101  103  105            .ASCII  /ACE/
002257  040  101  122            .ASCII  / AR/
002262  122  101  131            .ASCII  /RAY/
002265  040  115  117            .ASCII  / MO/
002270  104  125  114            .ASCII  /DUL/
002273  105  045  123            .ASCII  /E%S/
002276  045  104  062            .ASCII  /%D2/
002301  045  123  045            .ASCII  /%S%/
002304  123  045  101            .ASCII  /S%A/
002307  117  122  040            .ASCII  /OR /
002312  122  105  122            .ASCII  /RER/
002315  125  116  040            .ASCII  /UN /
002320  120  122  117            .ASCII  /PRO/
002323  107  122  101            .ASCII  /GRA/
002326  115  045  116            .ASCII  /M%N/
002331  000                      .ASCII  <00>
002332  045  101  101  P.ABR:    .ASCII  /%AA/
002335  122  122  101            .ASCII  /RRA/
002340  131  045  123            .ASCII  /Y%S/
002343  045  104  062            .ASCII  /%D2/
002346  045  123  045            .ASCII  /%S%/
002351  101  102  101            .ASCII  /ABA/
002354  116  113  045            .ASCII  /NK%/
002357  123  045  117            .ASCII  /S%O/
002362  061  045  116            .ASCII  /1%N/
002365  000                      .ASCII  <00>
002366  045  123  045  P.ABS:    .ASCII  /%S%/
002371  123  045  123            .ASCII  /S%S/
002374  045  123  045            .ASCII  /%S%/
002377  101  115  114            .ASCII  /AML/
002402  103  123  061            .ASCII  /CS1/
002405  045  123  045            .ASCII  /%S%/
002410  123  045  123            .ASCII  /S%S/
002413  045  123  045            .ASCII  /%S%/
002416  123  045  123            .ASCII  /S%S/
002421  045  123  045            .ASCII  /%S%/
002424  123  045  123            .ASCII  /S%S/
002427  045  123  045            .ASCII  /%S%/
002432  123  045  123            .ASCII  /S%S/
```

```
002435  045  123  045                .ASCII  /%S%/
002440  123  045  123                .ASCII  /S%S/
002443  045  123  045                .ASCII  /%S%/
002446  117  066  045                .ASCII  /06%/
002451  116  000  000                .ASCII  /N/<00><00>
002454  045  123  045       P.ABT:   .ASCII  /%S%/
002457  123  045  123                .ASCII  /S%S/
002462  045  123  045                .ASCII  /%S%/
002465  101  115  114                .ASCII  /AML/
002470  127  103  045                .ASCII  /WC%/
002473  123  045  123                .ASCII  /S%S/
002476  045  123  045                .ASCII  /%S%/
002501  123  045  123                .ASCII  /S%S/
002504  045  123  045                .ASCII  /%S%/
002507  123  045  123                .ASCII  /S%S/
002512  045  123  045                .ASCII  /%S%/
002515  123  045  123                .ASCII  /S%S/
002520  045  123  045                .ASCII  /%S%/
002523  123  045  123                .ASCII  /S%S/
002526  045  123  045                .ASCII  /%S%/
002531  123  045  117                .ASCII  /S%O/
002534  066  045  116                .ASCII  /6%N/
002537  000                         .ASCII  <00>
002540  045  123  045       P.ABU:   .ASCII  /%S%/
002543  123  045  123                .ASCII  /S%S/
002546  045  123  045                .ASCII  /%S%/
002551  101  115  114                .ASCII  /AML/
002554  102  101  045                .ASCII  /BA%/
002557  123  045  123                .ASCII  /S%S/
002562  045  123  045                .ASCII  /%S%/
002565  123  045  123                .ASCII  /S%S/
002570  045  123  045                .ASCII  /%S%/
002573  123  045  123                .ASCII  /S%S/
002576  045  123  045                .ASCII  /%S%/
002601  123  045  123                .ASCII  /S%S/
002604  045  123  045                .ASCII  /%S%/
002607  123  045  123                .ASCII  /S%S/
002612  045  123  045                .ASCII  /%S%/
002615  123  045  117                .ASCII  /S%O/
002620  066  045  116                .ASCII  /6%N/
002623  000                         .ASCII  <00>
002624  045  123  045       P.ABV:   .ASCII  /%S%/
002627  123  045  123                .ASCII  /S%S/
002632  045  123  045                .ASCII  /%S%/
002635  101  115  114                .ASCII  /AML/
002640  104  101  045                .ASCII  /DA%/
002643  123  045  123                .ASCII  /S%S/
002646  045  123  045                .ASCII  /%S%/
002651  123  045  123                .ASCII  /S%S/
002654  045  123  045                .ASCII  /%S%/
002657  123  045  123                .ASCII  /S%S/
002662  045  123  045                .ASCII  /%S%/
```

```
002665  123  045  123                .ASCII  /S%S/
002670  045  123  045                .ASCII  /%S%/
002673  123  045  123                .ASCII  /S%S/
002676  045  123  045                .ASCII  /%S%/
002701  123  045  117                .ASCII  /S%O/
002704  066  045  116                .ASCII  /6%N/
002707  000                         .ASCII  <00>
002710  045  123  045     P.ABW:     .ASCII  /%S%/
002713  123  045  123                .ASCII  /S%S/
002716  045  123  045                .ASCII  /%S%/
002721  101  115  114                .ASCII  /AML/
002724  103  123  062                .ASCII  /CS2/
002727  045  123  045                .ASCII  /%S%/
002732  123  045  123                .ASCII  /S%S/
002735  045  123  045                .ASCII  /%S%/
002740  123  045  123                .ASCII  /S%S/
002743  045  123  045                .ASCII  /%S%/
002746  123  045  123                .ASCII  /S%S/
002751  045  123  045                .ASCII  /%S%/
002754  123  045  123                .ASCII  /S%S/
002757  045  123  045                .ASCII  /%S%/
002762  123  045  123                .ASCII  /S%S/
002765  045  123  045                .ASCII  /%S%/
002770  117  066  045                .ASCII  /O6%/
002773  116  000  000                .ASCII  /N/<00><00>
002776  045  123  045     P.ABX:     .ASCII  /%S%/
003001  123  045  123                .ASCII  /S%S/
003004  045  123  045                .ASCII  /%S%/
003007  101  115  114                .ASCII  /AML/
003012  104  123  045                .ASCII  /DS%/
003015  123  045  123                .ASCII  /S%S/
003020  045  123  045                .ASCII  /%S%/
003023  123  045  123                .ASCII  /S%S/
003026  045  123  045                .ASCII  /%S%/
003031  123  045  123                .ASCII  /S%S/
003034  045  123  045                .ASCII  /%S%/
003037  123  045  123                .ASCII  /S%S/
003042  045  123  045                .ASCII  /%S%/
003045  123  045  123                 ASCII  /S%S/
003050  045  123  045                .ASCII  /%S%/
003053  123  045  117                .ASCII  /S%O/
003056  066  045  116                .ASCII  /6%N/
003061  000                         .ASCII  <00>
003062  045  123  045     P.ABY:     .ASCII  /%S%/
003065  123  045  123                .ASCII  /S%S/
003070  045  123  045                .ASCII  /%S%/
003073  101  115  114                .ASCII  /AML/
003076  105  122  045                .ASCII  /ER%/
003101  123  045  123                .ASCII  /S%S/
003104  045  123  045                .ASCII  /%S%/
003107  123  045  123                .ASCII  /S%S/
003112  045  123  045                .ASCII  /%S%/
```

```
003115  123  045  123            .ASCII  /S%S/
003120  045  123  045            .ASCII  /%S%/
003123  123  045  123            .ASCII  /S%S/
003126  045  123  045            .ASCII  /%S%/
003131  123  045  123            .ASCII  /S%S/
003134  045  123  045            .ASCII  /%S%/
003137  123  045  117            .ASCII  /S%O/
003142  066  045  116            .ASCII  /6%N/
003145  000                      .ASCII  <00>
003146  045  123  045  P.ABZ:    .ASCII  /%S%/
003151  123  045  123            .ASCII  /S%S/
003154  045  123  045            .ASCII  /%S%/
003157  101  115  114            .ASCII  /AML/
003162  101  123  045            .ASCII  /AS%/
003165  123  045  123            .ASCII  /S%S/
003170  045  123  045            .ASCII  /%S%/
003173  123  045  123            .ASCII  /S%S/
003176  045  123  045            .ASCII  /%S%/
003201  123  045  123            .ASCII  /S%S/
003204  045  123  045            .ASCII  /%S%/
003207  123  045  123            .ASCII  /S%S/
003212  045  123  045            .ASCII  /%S%/
003215  123  045  123            .ASCII  /S%S/
003220  045  123  045            .ASCII  /%S%/
003223  123  045  117            .ASCII  /S%O/
003226  066  045  116            .ASCII  /6%N/
003231  000                      .ASCII  <00>
003232  045  123  045  P.ACA:    .ASCII  /%S%/
003235  123  045  123            .ASCII  /S%S/
003240  045  123  045            .ASCII  /%S%/
003243  101  115  114            .ASCII  /AML/
003246  115  122  045            .ASCII  /MR%/
003251  123  045  123            .ASCII  /S%S/
003254  045  123  045            .ASCII  /%S%/
003257  123  045  123            .ASCII  /S%S/
003262  045  123  045            .ASCII  /%S%/
003265  123  045  123            .ASCII  /S%S/
003270  045  123  045            .ASCII  /%S%/
003273  123  045  123            .ASCII  /S%S/
003276  045  123  045            .ASCII  /%S%/
003301  123  045  123            .ASCII  /S%S/
003304  045  123  045            .ASCII  /%S%/
003307  123  045  117            .ASCII  /S%O/
003312  066  045  116            .ASCII  /6%N/
003315  000                      .ASCII  <00>
003316  045  123  045  P.ACB:    .ASCII  /%S%/
003321  123  045  123            .ASCII  /S%S/
003324  045  123  045            .ASCII  /%S%/
003327  101  115  114            .ASCII  /AML/
003332  104  124  045            .ASCII  /DT%/
003335  123  045  123            .ASCII  /S%S/
003340  045  123  045            .ASCII  /%S%/
```

```
003343  123  045  123            .ASCII  /S%S/
003346  045  123  045            .ASCII  /%S%/
003351  123  045  123            .ASCII  /S%S/
003354  045  123  045            .ASCII  /%S%/
003357  123  045  123            .ASCII  /S%S/
003362  045  123  045            .ASCII  /%S%/
003365  123  045  123            .ASCII  /S%S/
003370  045  123  045            .ASCII  /%S%/
003373  123  045  117            .ASCII  /S%O/
003376  066  045  116            .ASCII  /6%N/
003401  000                      .ASCII  <00>
003402  045  123  045  P.ACC:    .ASCII  /%S%/
003405  123  045  123            .ASCII  /S%S/
003410  045  123  045            .ASCII  /%S%/
003413  101  115  114            .ASCII  /AML/
003416  123  116  045            .ASCII  /SN%/
003421  123  045  123            .ASCII  /S%S/
003424  045  123  045            .ASCII  /%S%/
003427  123  045  123            .ASCII  /S%S/
003432  045  123  045            .ASCII  /%S%/
003435  123  045  123            .ASCII  /S%S/
003440  045  123  045            .ASCII  /%S%/
003443  123  045  123            .ASCII  /S%S/
003446  045  123  045            .ASCII  /%S%/
003451  123  045  123            .ASCII  /S%S/
003454  045  123  045            .ASCII  /%S%/
003457  123  045  117            .ASCII  /S%O/
003462  066  045  116            .ASCII  /6%N/
003465  000                      .ASCII  <00>
003466  045  123  045  P.ACD:    .ASCII  /%S%/
003471  123  045  123            .ASCII  /S%S/
003474  045  123  045            .ASCII  /%S%/
003477  101  115  114            .ASCII  /AML/
003502  105  105  045            .ASCII  /EE%/
003505  123  045  123            .ASCII  /S%S/
003510  045  123  045            .ASCII  /%S%/
003513  123  045  123            .ASCII  /S%S/
003516  045  123  045            .ASCII  /%S%/
003521  123  045  123            .ASCII  /S%S/
003524  045  123  045            .ASCII  /%S%/
003527  123  045  123            .ASCII  /S%S/
003532  045  123  045            .ASCII  /%S%/
003535  123  045  123            .ASCII  /S%S/
003540  045  123  045            .ASCII  /%S%/
003543  123  045  117            .ASCII  /S%O/
003546  066  045  116            .ASCII  /6%N/
003551  000                      .ASCII  <00>
003552  045  123  045  P.ACE:    .ASCII  /%S%/
003555  123  045  123            .ASCII  /S%S/
003560  045  123  045            .ASCII  /%S%/
003563  101  115  114            .ASCII  /AML/
003566  105  114  045            .ASCII  /EL%/
```

```
003571  123  045  123                    .ASCII  /S%S/
003574  045  123  045                    .ASCII  /%S%/
003577  123  045  123                    .ASCII  /S%S/
003602  045  123  045                    .ASCII  /%S%/
003605  123  045  123                    .ASCII  /S%S/
003610  045  123  045                    .ASCII  /%S%/
003613  123  045  123                    .ASCII  /S%S/
003616  045  123  045                    .ASCII  /%S%/
003621  123  045  123                    .ASCII  /S%S/
003624  045  123  045                    .ASCII  /%S%/
003627  123  045  117                    .ASCII  /S%O/
003632  066  045  116                    .ASCII  /6%N/
003635  000                             .ASCII  <00>
003636  045  123  045          P.ACF:    .ASCII  /%S%/
003641  123  045  123                    .ASCII  /S%S/
003644  045  123  045                    .ASCII  /%S%/
003647  123  045  123                    .ASCII  /S%S/
003652  045  123  045                    .ASCII  /%S%/
003655  123  045  123                    .ASCII  /S%S/
003660  045  123  045                    .ASCII  /%S%/
003663  101  122  105                    .ASCII  /ARE/
003666  107  111  123                    .ASCII  /GIS/
003671  124  105  122                    .ASCII  /TER/
003674  040  040  104                    .ASCII  /  D/
003677  125  115  120                    .ASCII  /UMP/
003702  045  116  045                    .ASCII  /%N%/
003705  116  000  000                    .ASCII  /N/<00><00>
003710  045  123  045          P.ACG:    .ASCII  /%S%/
003713  123  045  101                    .ASCII  /S%A/
003716  122  105  107                    .ASCII  /REG/
003721  111  123  124                    .ASCII  /IST/
003724  105  122  040                    .ASCII  /ER /
003727  116  101  115                    .ASCII  /NAM/
003732  105  045  123                    .ASCII  /E%S/
003735  045  123  045                    .ASCII  /%S%/
003740  123  045  123                    .ASCII  /S%S/
003743  045  123  045                    .ASCII  /%S%/
003746  123  045  101                    .ASCII  /S%A/
003751  103  117  116                    .ASCII  /CON/
003754  124  105  116                    .ASCII  /TEN/
003757  124  123  045                    .ASCII  /TS%/
003762  116  000                        .ASCII  /N/<00>
003764  000  000                P.ACH:  .ASCII  <00><00>


000000                                  .PSECT  $OWN$,  D
000000                          COL.CNT.TBL:
                                        .BLKW   200
000400                          REM.TBL:.BLKW   144
000710                          COPIED.REM.TBL:
                                        .BLKW   144
```

```
001220                          COL.PTR:.BLKW   5
001232                          WRT.BUF:.BLKW   2176
005626                          RD.BUF: .BLKW   400
006626                          ERROR.MAP:
                                        .BLKW   10000
026626                          CHIP.TBL:
                                        .BLKW   47
026744                          TMP.BLST.TBL:
                                        .BLKW   12
026770                          BLAST.TBL:
                                        .BLKW   4000
036770                          COL.BASE:
                                        .BLKW   1
036772                          ML.ADDR:.BLKW   1
036774                          MAX.CHIP.COL:
                                        .BLKW   1
036776                          ML.DUT: .BLKW   1
037000                          ARR$BNK.SEL:
                                        .BLKW   1
037002                          TSTED.BNK:
                                        .BLKW   1
037004                          INC.BNK:.BLKW   1
037006                          INC.ARR:.BLKW   1
037010                          ARR.SEL.POS:
                                        .BLKW   1
037012                          BNK.SEL.POS:
                                        .BLKW   1
037014                          BNK.NUM.SEC:
                                        .BLKW   1
037016                          BAD.BNK.REG:
                                        .BLKB   1
                                        .EVEN
037020                          FLG.REG:.BLKW   1
037022                          DEGRADE.MOD.REG:
                                        .BLKB   1
                                        .EVEN
037024                          SIZE:   .BLKW   1
037026                          DST:    .BLKW   1
037030                          SRC:    .BLKW   1
037032                          LST.TSTED.ARR:
                                        .BLKW   1
037034                          RAND.PASS:
                                        .BLKW   1


                                .GLOBL  RANGEN, SEED1, SEED2, SEED3, RANDAT
                                .GLOBL  L$UNIT


        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
001000                          BIT9==          1000
000400                          BIT8==          400
000200                          BIT7==          200
000100                          BIT6==          100
000040                          BIT5==          40
000020                          BIT4==          20
000010                          BIT3==          10
000004                          BIT2==          4
000002                          BIT1==          2
000001                          BIT0==          1
000040                          EF.START==      40
000037                          EF.RESTART==    37
000036                          EF.CONTINUE==   36
000035                          EF.NEW==        35
000034                          EF.PWR==        34
000340                          PRI07==         340
000300                          PRI06==         300
000240                          PRI05==         240
000200                          PRI04==         200
000140                          PRI03==         140
000100                          PRI02==         100
000040                          PRI01==         40
000000                          PRI00==         0
000004                          EVL==           4
000010                          LOT==           10
000020                          ADR==           20
000040                          IDU==           40
000100                          ISR==           100
000200                          UAM==           200
000400                          BOE==           400
001000                          PNI==           1000
002000                          PRI==           2000
004000                          IXE==           4000
010000                          IBE==           10000
020000                          IER==           20000
040000                          LOE==           40000
100000                          HOE==           -100000
000000'                         ILL.CMD.MSG=    P.AAA
000052'                         BGN.MSG=        P.AAB
```

```
000104'                 START.MSG=          P.AAC
000142'                 HQ.ERR.MSG=         P.AAD
000216'                 HQ.MSG=             P.AAE
000274'                 END.MSG=            P.AAF
000330'                 SUPRES.MSG=         P.AAG
000370'                 RET.DRS.MSG=        P.AAH
000412'                 LUN.MISS.MSG=       P.AAI
000444'                 CUN.A.MSG=          P.AAJ
000460'                 CON.B.MSG=          P.AAK
000474'                 CON.C.MSG=          P.AAL
000510'                 CON.D.MSG=          P.AAM
000526'                 UNC.CHIP.MSG=       P.AAN
000560'                 GTR.MSG=            P.AAO
000634'                 UNIT.SEL.MSG-       P.AAP
000704'                 SW.BUG.MSG=         P.AAQ
000766'                 ABORT.MSG=          P.AAR
001036'                 NO.AD.MSG=          P.AAS
001112'                 UNC.ERR.MSG=        P.AAT
001170'                 TIME.OUT.MSG=       P.AAU
001250'                 REP.M7363.MSG=      P.AAV
001310'                 MEM.ERR.MSG=        P.AAW
001362'                 UNX.DRV.ERR.MSG=    P.AAX
001424'                 RD.XFER.MSG=        P.AAY
001474'                 WRT.XFER.MSG=       P.AAZ
001546'                 WT.CHK.XFER.MSG=    P.ABA
001626'                 X.MSG=              P.ABB
001630'                 ONE.MSG=            P.ABC
001636'                 TWO.MSG=            P.ABD
001646'                 THREE.MSG=          P.ABE
001660'                 FOUR.MSG=           P.ABF
001674'                 LF=                 P.ABG
001700'                 LFS=                P.ABH
001706'                 0.1.PRINT=          P.ABI
001716'                 0.6.PRINT=          P.ABJ
001726'                 D.2.PRINT=          P.ABK
001736'                 DRV.SEL.PRINT-      P.ABL
001770'                 ARR.SEL.PRINT=      P.ABM
002036'                 BNK.SEL.PRINT=      P.ABN
002120'                 A.B.C.PRINT=        P.ABO
002172'                 A.B.N.PRINT-        P.ABP
002246'                 REP.PRINT=          P.ABQ
002332'                 A.B.PRINT=          P.ABR
002366'                 CS1.PRINT=          P.ABS
002454'                 WC.PRINT-           P.ABT
002540'                 BA.PRINT=           P.ABU
002624'                 DA.PRINT=           P.ABV
002710'                 CS2.PRINT=          P.ABW
002776'                 DS.PRINT-           P.ABX
003062'                 ER.PRINT=           P.ABY
003146'                 AS.PRINT            P.ABZ
003232'                 MR.PRINT=           P.ACA
003316'                 DT.PRINT-           P.ACB
```

```
        003402'                         SN.PRINT=       P.ACC
        003466'                         EE.PRINT=       P.ACD
        003552'                         EL.PRINT=       P.ACE
        003636'                         HEADER.PRINT-   P.ACF
        003710'                         COL.DESC.PRINT= P.ACG
        003764'                         X.PRINT=        P.ACH


                                        .SBTTL  DUMPER EXTENDED MESSAGE PRINTING SECTION
000000                                  .PSECT  $CODE$

000000  004767  000000V                DUMPER::JSR     PC,M$DUMPER             ;                       2171
000004  104423                                  TRAP    23
000006  000207                                  RTS     PC

; Routine Size:  4 words
; Maximum stack depth per invocation:  0 words

;       2172  PRINTB (HEADER_PRINT);                          .PRINT THE HEADER
;       2173  PRINTB (COL_DESC_PRINT);                       .DESCRIBE THE COLUMN HEADINGS
;       2174  PRINTB (CS1_PRINT, .ML_ADDR [MLCS1, ML_ALL]);  !PRINT MLCS1 CONTENTS
;       2175  PRINTB (WC_PRINT, .ML_ADDR [MLWC, ML_ALL]);    .PRINT MLWC CONTENTS
;       2176  PRINTB (BA_PRINT, .ML_ADDR [MLBA, ML_ALL]);    !PRINT MLBA CONTENTS
;       2177  PRINTB (DA_PRINT, .ML_ADDR [MLDA, ML_ALL]);    !PRINT MLDA CONTENTS
;       2178  PRINTB (CS2_PRINT, .ML_ADDR [MLCS2, ML_ALL]);  !PRINT MLCS2 CONTENTS
;       2179  PRINTB (DS_PRINT, .ML_ADDR [MLDS, ML_ALL]);    .PRINT MLDS CONTENTS
;       2180  PRINTB (ER_PRINT, .ML_ADDR [MLER, ML_ALL]);    !PRINT MLER CONTENTS
;       2181  PRINTB (AS_PRINT, .ML_ADDR [MLAS, ML_ALL]);    .PRINT MLAS CONTENTS
;       2182  PRINTB (MR_PRINT, .ML_ADDR [MLMR, ML_ALL]);    !PRINT MLMR CONTENTS
;       2183  PRINTB (DT_PRINT, .ML_ADDR [MLDT, ML_ALL]);    !PRINT MLDT CONTENTS
;       2184  PRINTB (SN_PRINT, .ML_ADDR [MLSN, ML_ALL]);    !PRINT MLSN CONTENTS
;       2185  PRINTB (EE_PRINT, .ML_ADDR [MLEE, ML_ALL]);    !PRINT MLEE CONTENTS
;       2186  PRINTB (EL_PRINT, .ML_ADDR [MLEL, ML_ALL]);    !PRINT MLEL CONTENTS
;       2187  ENDMSG;


                                        .SBTTL  M$DUMPER EXTENDED MESSAGE PRINTING SECTION
000010  162706  000032              M$DUMPER:
                                        SUB     #32,SP                  ;                       2171
000014  012746  003636'                 MOV     #HEADER.PRINT,-(SP)     ;                       2172
000020  012746  000001                  MOV     #1,-(SP)
000024  010600                          MOV     SP,R0                   ; SP,*
000026  104414                          TRAP    14
000030  012716  003710'                 MOV     #COL.DESC.PRINT,(SP)    ;                       2173
000034  012746  000001                  MOV     #1,-(SP)
000040  010600                          MOV     SP,R0                   ; SP,*
000042  104414                          TRAP    14
000044  017766  036772' 000036          MOV     @ML.ADDR,36(SP)         ; *,ML.REG              2174
000052  016616  000036                  MOV     36(SP),(SP)             ; ML.REG,*
000056  012746  002366'                 MOV     #CS1.PRINT,-(SP)
000062  012746  000002                  MOV     #2,-(SP)
```

```
000066  010600                          MOV     SP,R0                           ; SP,*
000070  104414                          TRAP    14
                                        MOV     ML.ADDR,R0                      ;                       2175
000072  016700  036772'                 MOV     2(R0),40(SP)                    ; *,ML.REG
000076  016066  000002  000040          MOV     40(SP),(SP)                     ; ML.REG,*
000104  016616  000040                  MOV     #WC.PRINT,-(SP)
000110  012746  002454'                 MOV     #2,-(SP)
000114  012746  000002                  MOV     SP,R0                           ; SP,*
000120  010600                          TRAP    14
000122  104414                          MOV     ML.ADDR,R0                      ;                       2176
000124  016700  036772'                 MOV     4(R0),42(SP)                    ; *,ML.REG
000130  016066  000004  000042          MOV     42(SP),(SP)                     ; ML.REG,*
000136  016616  000042                  MOV     #BA.PRINT,-(SP)
000142  012746  002540'                 MOV     #2,-(SP)
000146  012746  000002                  MOV     SP,R0                           ; SP,*
000152  010600                          TRAP    14
000154  104414                          MOV     ML.ADDR,R0                      ;                       2177
000156  016700  036772'                 MOV     6(R0),44(SP)                    ; *,ML.REG
000162  016066  000006  000044          MOV     44(SP),(SP)                     ; ML.REG,*
000170  016616  000044                  MOV     #DA.PRINT,-(SP)
000174  012746  002624'                 MOV     #2,-(SP)
000200  012746  000002                  MOV     SP,R0                           ; SP,*
000204  010600                          TRAP    14
000206  104414                          MOV     ML.ADDR,R0                      ;                       2178
000210  016700  036772'                 MOV     10(R0),46(SP)                   ; *,ML.REG
000214  016066  000010  000046          MOV     46(SP),(SP)                     ; ML.REG,*
000222  016616  000046                  MOV     #CS2.PRINT,-(SP)
000226  012746  002710'                 MOV     #2,-(SP)
000232  012746  000002                  MOV     SP,R0                           ; SP,*
000236  010600                          TRAP    14
000240  104414                          MOV     ML.ADDR,R0                      ;                       2179
000242  016700  036772'                 MOV     12(R0),50(SP)                   ; *,ML.REG
000246  016066  000012  000050          MOV     50(SP),(SP)                     ; ML.REG,*
000254  016616  000050                  MOV     #DS.PRINT,-(SP)
000260  012746  002776'                 MOV     #2,-(SP)
000264  012746  000002                  MOV     SP,R0                           ; SP,*
000270  010600                          TRAP    14
000272  104414                          MOV     ML.ADDR,R0                      ;                       2180
000274  016700  036772'                 MOV     14(R0),52(SP)                   ; *,ML.REG
000300  016066  000014  000052          MOV     52(SP),(SP)                     ; ML.REG,*
000306  016616  000052                  MOV     #ER.PRINT,-(SP)
000312  012746  003062'                 MOV     #2,-(SP)
000316  012746  000002                  MOV     SP,R0                           ; SP,*
000322  010600                          TRAP    14
000324  104414                          MOV     ML.ADDR,R0                      ;                       2181
000326  016700  036772'                 MOV     16(R0),54(SP)                   ; *,ML.REG
000332  016066  000016  000054          MOV     54(SP),(SP)                     ; ML.REG,*
000340  016616  000054                  MOV     #AS.PRINT,-(SP)
000344  012746  003146'                 MOV     #2,-(SP)
000350  012746  000002                  MOV     SP,R0                           ; SP,*
000354  010600                          TRAP    14
000356  104414                          MOV     ML.ADDR,R0                      ;                       2182
000360  016700  036772'
```

```
000364  016066  000024  000056          MOV     24(R0),56(SP)                   ; *,ML.REG
000372  016616  000056                  MOV     56(SP),(SP)                     ; ML.REG,*
000376  012746  003232'                 MOV     #MR.PRINT,-(SP)
000402  012746  000002                  MOV     #2,-(SP)
000406  010600                          MOV     SP,R0                           ; SP,*
000410  104414                          TRAP    14
000412  016700  036772'                 MOV     ML.ADDR,R0                      ;                                               2183
000416  016066  000026  000060          MOV     26(R0),60(SP)                   ; *,ML.REG
000424  016616  000060                  MOV     60(SP),(SP)                     ; ML.RFG,*
000430  012746  003316'                 MOV     #DT.PRINT,-(SP)
000434  012746  000002                  MOV     #2,-(SP)
000440  010600                          MOV     SP,R0                           ; SP,*
000442  104414                          TRAP    14
000444  016700  036772'                 MOV     ML.ADDR,R0                      ;                                               2184
000450  016066  000030  000062          MOV     30(R0),62(SP)                   ; *,ML.REG
000456  016616  000062                  MOV     62(SP),(SP)                     ; ML.REG,*
000462  012746  003402'                 MOV     #SN.PRINT,-(SP)
000466  012746  000002                  MOV     #2,-(SP)
000472  010600                          MOV     SP,R0                           ; SP,*
000474  104414                          TRAP    14
000476  016700  036772'                 MOV     ML.ADDR,R0                      ;                                               2185
000502  016066  000042  000064          MOV     42(R0),64(SP)                   ; *,ML.REG
000510  016616  000064                  MOV     64(SP),(SP)                     ; ML.REG,*
000514  012746  003466'                 MOV     #EE.PRINT,-(SP)
000520  012746  000002                  MOV     #2,-(SP)
000524  010600                          MOV     SP,R0                           ; SP,*
000526  104414                          TRAP    14
000530  016700  036772'                 MOV     ML.ADDR,R0                      ;                                               2186
000534  016066  000044  000066          MOV     44(R0),66(SP)                   ; *,ML.REG
000542  016616  000066                  MOV     66(SP),(SP)                     ; ML.REG,*
000546  012746  003552'                 MOV     #EL.PRINT,-(SP)
000552  012746  000002                  MOV     #2,-(SP)
000556  010600                          MOV     SP,R0                           ; SP,*
000560  104414                          TRAP    14
000562  062706  000124                  ADD     #124,SP                         ;                                               2171
000566  000207                          RTS     PC

; Routine Size:  184 words
; Maximum stack depth per invocation:  42 words
```

```
;         2188  %sbttl 'ROUTINE DECLARATIONS'
;         2189  routine GEN_RANDOM_DATA : novalue =
;         2190      begin
;         2191
;         2192  !++
;         2193  !  FUNCTIONAL DESCRIPTION: GENERATE RANDOM DATA
;         2194  !
;         2195  !                          THIS ROUTINE WILL GENERATE A WRITE BUFFER
;         2196  !                          OF RANDOM DATA.  THE ROUTINE TO ACTUALLY
;         2197  !                          GENERATE THE RANDOM NUMBER IS A GLOBAL MACRO
;         2198  !                          ROUTINE LINKED AT LINK TIME.
;         2199  !
;         2200  !--
;         2201
;         2202      local
;         2203          OFFSET;                                 .OFFSET VARIABLE INTO WRITE BUFFER
;         2204
;         2205      WRT_BUF + %o'0' = .SEED1;                   .INITIAL SEED1 TO WRITE BUFFER + 0
;         2206      WRT_BUF + %o'2' = .SEED2;                   !INITIAL SEED2 TO WRITE BUFFER + 2
;         2207      WRT_BUF + %o'4' = .SEED3;                   .INITIAL SEED3 TO WRITE BUFFER + 4
;         2208      OFFSET = %o'6';                             .ADJUST OFFSET BY 6
;         2209
;         2210      incr WRD_CNT from 4 to (128 + 511*2) do     !LOAD 768 WORDS WITH RANDOM DATA
;         2211          begin
;         2212          RANGEN ();                              !GENERATE THE RANDOM NUMBER
;         2213          BREAK;
;         2214          WRT_BUF + .OFFSET = .RANDAT;            .WRITE BUFFER + OFFSET GETS THE RANDOM NUMBER
;         2215          OFFSET = .OFFSET + 2;                   .BUMP THE OFFSET BY 2
;         2216          end;
;         2217
;         2218      end;


                                        .SBTTL  GEN.RANDOM.DATA ROUTINE DECLARATIONS
000570  004167  000000G           GEN.RANDOM.DATA:
                                    JSR     R1,$SAVE2                       ;                       2189
000574  016767  000000G 001232'     MOV     SEED1,WRT.BUF                   ;                       2205
000602  016767  000000G 001234'     MOV     SEED2,WRT.BUF+2                 ;                       2206
000610  016767  000000G 001236'     MOV     SEED3,WRT.BUF+4                 ;                       2207
000616  012701  000006              MOV     #6,R1                           ; *,OFFSET              2208
000622  012702  000004              MOV     #4,R2                           ; *,WRD.CNT             2210
000626  004767  000000G           1$: JSR     PC,RANGEN                       ;                       2212
000632  104422                      TRAP    22
000634  016761  000000G 001232'     MOV     RANDAT,WRT.BUF(R1)              ; *,*(OFFSET)           2214
000642  062701  000002              ADD     #2,R1                           ; *,OFFSET              2215
000646  005202                      INC     R2                              ; WRD.CNT               2210
000650  020227  002176              CMP     R2,#2176                        ; WRD.CNT,*
000654  003764                      BLE     1$
000656  000207                      RTS     PC                              ;                       2189

; Routine Size:  28 words
; Maximum stack depth per invocation:  3 words
```

```
;          2219  routine DM_RD_TRANSFER (SIZE, DST, SRC) : novalue =
;          2220      begin
;          2221
;          2222  !++
;          2223  !  FUNCTIONAL DESCRIPTION: DIAGNOSTIC READ TRANSFER
;          2224  !
;          2225  !                         THIS ROUTINE WILL PERFROM A DIAGNOSTIC
;          2226  !                         MODE READ TRANSFER AT THE DESIGNATED
;          2227  !                         SOURCE AND DESTINATION ADDRESSES.
;          2228  !
;          2229  !  FORMAL PARAMETERS:
;          2230  !                  SIZE:
;          2231  !                         SPECIFIES THE SIZE OF THE TRANSFER
;          2232  !                  DST:
;          2233  !                         DISTINATION ADDRESS OF TRANSFER
;          2234  !                  SRC:
;          2235  !                         SOURCE ADDRESS OF TRANSFER
;          2236  !--
;          2237
;          2238      CLR_MBUS;                                   !CLEAR THE MASS BUS
;          2239      DAT_DM (ENABE);                             !ENABLE DATA DIAG MODE
;          2240      WRT_RH (MLCS2, DRV_SEL, .ML_DUT);           !SELECT THE DIVICE UNDER TEST
;          2241      WRT_RH (MLWC, WC_REG, .SIZE);               !LOAD THE TRANSFER SIZE
;          2242      WRT_RH (MLDA, DA_REG, .SRC);                !LOAD THE TRANSFER SOURCE ADDRESS
;          2243      WRT_RH (MLBA, BA_REG, .DST);                !LOAD THE TRANSFER DESTINATION ADDRESS
;          2244      WRT_RH (MLCS1, FUNC, FUNC_4);               !LOAD A READ FUNCTION WITH GO SET
;          2245      end;

                                .SBTTL  DM.RD.TRANSFER ROUTINE DECLARATIONS
000660  010146          DM.RD.TRANSFER: MOV     R1,-(SP)                ;                       2219
000662  016700  036772'                  MOV     ML.ADDR,R0              ;                       2220
000666  016046  0000'0                  MOV     10(R0),-(SP)            ; *,ML.REG
000672  011601                          MOV     (SP),R1                 ; ML.REG,MLREG
000674  152701  000040                  BISB    #40,R1                  ; *,MLREG
000700  016700  036772'                  MOV     ML.ADDR,R0
000704  010160  000010                  MOV     R1,10(R0)               ; MLREG,*
000710  016700  036772'                  MOV     ML.ADDR,R0
000714  016046  000010                  MOV     10(R0),-(SP)            ; *,ML.REG
000720  011601                          MOV     (SP),R1                 ; ML.REG,MLREG
000722  016700  036776'                  MOV     ML.DUT,R0
000726  042700  177770                  BIC     #177770,R0
000732  142701  000007                  BICB    #7,R1                   ; *,MLREG
000736  050001                          BIS     R0,R1                   ; *,MLREG
000740  016700  036772'                  MOV     ML.ADDR,R0
000744  010160  000010                  MOV     R1,10(R0)               ; MLREG,*
000750  016700  036772'                  MOV     ML.ADDR,R0              ;                       2239
000754  016046  000024                  MOV     24(R0),-(SP)            ; *,ML.REG
000760  011601                          MOV     (SP),R1                 ; ML.REG,MLREG
000762  152701  000010                  BISB    #10,R1                  ; *,MLREG
000766  016700  036772'                  MOV     ML.ADDR,R0
```

```
000772  010160  000024          MOV     R1,24(R0)                       ; MLREG,*
000776  016700  036772'         MOV     ML.ADDR,R0                      ;                       2240
001002  016046  000010          MOV     10(R0),-(SP)                    ; *,ML.REG
001006  011601                  MOV     (SP),R1                         ; ML.REG,MLREG
001010  016700  036776'         MOV     ML.DUT,R0
001014  042700  177770          BIC     #177770,R0
001020  142701  000007          BICB    #7,R1                           ; *,MLREG
001024  050001                  BIS     R0,R1                           ; *,MLREG
001026  016700  036772'         MOV     ML.ADDR,R0
001032  010160  000010          MOV     R1,10(R0)                       ; MLREG,*
001036  016700  036772'         MOV     ML.ADDR,R0                      ;                       2241
001042  016046  000002          MOV     2(R0),-(SP)                     ; *,ML.REG
001046  016601  000022          MOV     22(SP),R1                       ; SIZE,M REG
001052  016700  036772'         MOV     ML.ADDR,R0
001056  010160  000002          MOV     R1,2(R0)                        ; MLREG,*
001062  016700  036772'         MOV     ML.ADDR,R0                      ;                       2242
001066  016046  000006          MOV     6(R0),-(SP)                     ; *,ML.REG
001072  016601  000020          MOV     20(SP),R1                       ; SRC,MLREG
001076  016700  036772'         MOV     ML.ADDR,R0
001102  010160  000006          MOV     R1,6(R0)                        ; MLREG,*
001106  016700  036772'         MOV     ML.ADDR,R0                      ;                       2243
001112  016046  000004          MOV     4(R0),-(SP)                     ; *,ML.REG
001116  016601  000024          MOV     24(SP),R1                       ; DST,MLREG
001122  016700  036772'         MOV     ML.ADDR,R0
001126  010160  000004          MOV     R1,4(R0)                        ; MLREG,*
001132  011746  036772'         MOV     @ML.ADDR,-(SP)                  ; *,ML.REG              2244
001136  012600                  MOV     (SP)+,R0                        ; ML.REG,MLREG
001140  142700  000077          BICB    #77,R0                          ; *,MLREG
001144  152700  000051          BISB    #51,R0                          ; *,MLREG
001150  010077  036772'         MOV     R0,@ML.ADDR                     ; MLREG,*
001154  062706  000016          ADD     #16,SP                          ;                       2219
001160  012601                  MOV     (SP)+,R1
001162  000207                  RTS     PC

; Routine Size:  98 words
; Maximum stack depth per invocation:  10 words
```

```
2246  routine DM_WRT_TRANSFER (SIZE, DST, SRC) : novalue =
2247      begin
2248
2249  !++
2250  !  FUNCTIONAL DESCRIPTION: DIAGNOSTIC MODE WRITE TRANSFER
2251  !
2252  !                          THIS ROUTINE WILL PERFORM A DIAGNOSTIC
2253  !                          MODE WRITE TRANSFER AT THE DESIGNATED
2254  !                          SOURCE AND DESTINATION ADDRESSES.
2255  !
2256  !  FORMAL PARAMETERS:
2257  !                  SIZE:
2258  !                          SIZE OF THE TRANSFER
2259  !                  DST:
2260  !                          DESTINATION ADDRESS OF THE TRANSFER
2261  !                  SRC:
2262  !                          SOURCE ADDRESS OF THE TRANSFER
2263  !--
2264
2265      CLR_MBUS;                                   !CLEAR THE MASS BUS
2266      DAT_DM (ENABE);                             !ENABLE DATA DIAG MODE
2267      WRT_RH (MLCS2, DRV_SEL, .ML_DUT);           !SELECT THE DIVICE UNDER TEST
2268      WRT_RH (MLWC, WC_REG, .SIZE);               !LOAD THE TRANSFER SIZE
2269      WRT_RH (MLDA, DA_REG, .DST);                !LOAD THE DESTINATION ADDRESS
2270      WRT_RH (MLBA, BA_REG, .SRC);                !LOAD THE SOURCE ADDRESS
2271      WRT_RH (MLCS1, FUNC, FUNC_5);               !LOAD A WRITE FUNCTION WITH GO SET
2272      end;


                        .SBTTL  DM.WRT.TRANSFER ROUTINE DECLARATIONS
001164  010146          DM.WRT.TRANSFER:
                        MOV     R1,-(SP)            ;                       2246
                        MOV     ML.ADDR,R0          ;                       2247
001166  016700  036772' MOV     ML.ADDR,R0          ;
001172  016046  000010  MOV     10(R0),-(SP)        ; *,ML.REG
001176  011601          MOV     (SP),R1             ; ML.REG,MLREG
001200  152701  000040  BISB    #40,R1              ; *,MLREG
001204  016700  036772' MOV     ML.ADDR,R0
001210  010160  000010  MOV     R1,10(R0)           ; MLREG,*
001214  016700  036772' MOV     ML.ADDR,R0
001220  016046  000010  MOV     10(R0),-(SP)        ; *,ML.REG
001224  011601          MOV     (SP),R1             ; ML.REG,MLREG
001226  016700  036776' MOV     ML.DUT,R0
001232  042700  177770  BIC     #177770,R0
001236  142701  000007  BICB    #7,R1               ; *,MLREG
001242  050001          BIS     R0,R1               ; *,MLREG
001244  016700  036772' MOV     ML.ADDR,R0
001250  010160  000010  MOV     R1,10(R0)           ; MLREG,*
001254  016700  036772' MOV     ML.ADDR,R0          ;                       <266
001260  016046  000024  MOV     24(R0),-(SP)        ; *,ML.REG
001264  011601          MOV     (SP),R1             ; ML.REG,MLREG
001266  152701  000010  BISB    #10,R1              ; *,MLREG
001272  016700  036772' MOV     ML.ADDR,R0
```

```
001276  010160  000024          MOV     R1,24(R0)               ; MLREG,*
001302  016700  036772'         MOV     ML.ADDR,R0              ;                       2267
001306  016046  000010          MOV     10(R0),-(SP)            ; *,ML.REG
001312  011601                  MOV     (SP),R1                 ; ML.REG,MLREG
001314  016700  036776'         MOV     ML.DUT,R0
001320  042700  177770          BIC     #177770,R0
001324  142701  000007          BICB    #7,R1                   ; *,MLREG
001330  050001                  BIS     R0,R1                   ; *,MLREG
001332  016700  036772'         MOV     ML.ADDR,R0
001336  010160  000010          MOV     R1,10(R0)               ; MLREG,*
001342  016700  036772'         MOV     ML.ADDR,R0              ;                       2268
001346  016046  000002          MOV     2(R0),-(SP)             ; *,ML.REG
001352  016601  000022          MOV     22(SP),R1               ; SIZE,MLREG
001356  016700  036772'         MOV     ML.ADDR,R0
001362  010160  000002          MOV     R1,2(R0)                ; MLREG,*
001366  016700  036772'         MOV     ML.ADDR,R0              ;                       2269
001372  016046  000006          MOV     6(R0),-(SP)             ; *,ML.REG
001376  016601  000022          MOV     22(SP),R1               ; DST,MLREG
001402  016700  036772'         MOV     ML.ADDR,R0
001406  010160  000006          MOV     R1,6(R0)                ; MLREG,*
001412  016700  036772'         MOV     ML.ADDR,R0              ;                       2270
001416  016046  000004          MOV     4(R0),-(SP)             ; *,ML.REG
001422  016601  000022          MOV     22(SP),R1               ; SRC,MLREG
001426  016700  036772'         MOV     ML.ADDR,R0
001432  010160  000004          MOV     R1,4(R0)                ; MLREG,*
001436  017746  036772'         MOV     @ML.ADDR,-(SP)          ; *,ML.REG              2271
001442  012600                  MOV     (SP)+,R0                ; ML.REG,MLREG
001444  142700  000077          BICB    #77,R0                  ; *,MLREG
001450  152700  000061          BISB    #61,R0                  ; *,MLREG
001454  010077  036772'         MOV     R0,@ML.ADDR             ; MLREG,*
001460  062706  0000'6           ADD     #16,SP                  ;                       2246
001464  012601                  MOV     (SP)+,R1
001466  000207                  RTS     PC

; Routine Size:  98 words
; Maximum stack depth per invocation:  10 words
```

```
;          2273  routine DM_RAND_LOAD (TEMP_ARR$BNK_SEL) : novalue =
;          2274      begin
;          2275
;          2276  .++
;          2277  !   FUNCTIONAL DESCRIPTION: DIAGNOSTIC MODE RANDOM DATA LOAD
;          2278  !
;          2279  !                           THIS ROUTINE WILL LOAD VIA DIAGNOSTIC MODE
;          2280  !                           RANDOM DATA TO THE DESIGNATED ARRYA AND BANK
;          2281  !                           SELECTED BY THE TEMP ARRAY AND BANK SELECT
;          2282  !                           ARGUMENT.
;          2283  !
;          2284  !   FORMAL PARAMETERS:
;          2285  !                   TEMP_ARR$BNK_SEL:
;          2286  !                           THIS ARGUMENT STORES THE SELECTED ARRAYS
;          2287  !                           ARRAY SELECT NUMBER AND THE SELECTED ARRAYS
;          2288  !                           BANK SELECT NUMBER.
;          2289  .--
;          2290
;          2291      local
;          2292          S_BUF_INDEX,                                !START BUFFER INDEX POINTER
;          2293          BUF_INDEX;                                  .BUFFER INDEX POINTER
;          2294
;          2295      GEN_RANDOM_DATA ();                             !GENERATE A WRITE BUFFER WITH RANDOM DATA
;          2296      S_BUF_INDEX = ZERO;                             !THE FIRST SECTOR STARTS AT WRITE BUF WORD ZERO
;          2297      BUF_INDEX = .S_BUF_INDEX;                       !BUF INDEX WILL GET 128 RANDOM DATA WORDS
;          2298      DM_WRT_TRANSFER (SIZE = -256, DST = .TEMP_ARR$BNK_SEL, SRC = WRT_BUF);
;          2299
;          2300      incr SEC_CNT from 0 to .BNK_NUM_SEC do          !LOAD ALL SECTORS IN BANK WITH RANDOM DATA
;          2301          begin
;          2302          BREAK;
;          2303
;          2304          incr WRD_CNT from 0 to 127 do               !LOAD ALL WORDS IN SECTOR WITH RANDOM DATA
;          2305              begin
;          2306              WRT_RH (MLD1, DB1_REG, .WRT_BUF [.BUF_INDEX, WRD]); !LOAD DATA REG 1
;          2307              WRT_RH (MLD2, DB2_REG, .WRT_BUF [.BUF_INDEX, WRD]); !LOAD DATA REG 2
;          2308              WRT_RH (MLE2, E2_REG, .WRT_BUF [.BUF_INDEX, WRD]);  !LOAD DATA REG 3
;          2309              DAT_CLK;                                !CLOCK THE RANDOM DATA INTO THE ML-11 LONG WORD
;          2310              BUF_INDEX   .BUF_INDEX + ONE;           !GET THE NEXT RANDOM DATA WORD
;          2311              end;
;          2312
;          2313          S_BUF_INDEX = .S_BUF_INDEX + 2;             .START THE NEXT SECTOR 2 WORD DEEPER
;          2314          BUF_INDEX   .S_BUF_INDEX;                   !BUF INDEX WILL GET 128 MORE RANDOM DATA WORDS
;          2315          end;
;          2316
;          2317      CLR_MBUS;                                       .CLEAR THE SINGLE STEP DMA MODE
;          2318      end;


                                        .SBTTL  DM.RAND.LOAD ROUTINE DECLARATIONS
001470  004167  000000G          DM.RAND.LOAD:
                                        JSR     R1,$SAVE5                       ;                                   2273
001474  162706  000020                  SUB     #20,SP
```

```
001500  004767  177064                      JSR     PC,GEN.RANDOM.DATA              ;                               2295
001504  005005                              CLR     R5                              ; S.BUF.INDEX                   2296
001506  005004                              CLR     R4                              ; BUF.INDEX                     2297
001510  012746  177400                      MOV     #-400,-(SP)                     ;                               2298
001514  011667  037024'                     MOV     (SP),SIZE
001520  016646  000040                      MOV     40(SP),-(SP)                    ; TEMP.ARR$BNK.SE,*
001524  011667  037026'                     MOV     (SP),DST
001530  012746  001232'                     MOV     #WRT.BUF,-(SP)
001534  011667  037030'                     MOV     (SP),SRC
001540  004767  177420                      JSR     PC,DM.WRT.TRANSFER
001544  016766  037014'  000010             MOV     BNK.NUM.SEC,10(SP)              ;                               2300
001552  005066  000006                      CLR     6(SP)                           ; SEC.CNT
001556  000474                              BR      3$
001560  104422                  1$:         TRAP    22                              ;                               2301
001562  005000                              CLR     R0                              ; WRD.CNT                       2304
001564  016703  036772'          2$:         MOV     ML.ADDR,R3                      ;                               2306
001570  016366  000036  000024              MOV     36(R3),24(SP)                   ; *,ML.REG
001576  010403                              MOV     R4,R3                           ; BUF.INDEX,*
001600  006303                              ASL     R3
001602  012701  001232'                     MOV     #WRT.BUF,R1
001606  060301                              ADD     R3,R1
001610  011102                              MOV     (R1),R2                         ; *,MLREG
001612  016703  036772'                     MOV     ML.ADDR,R3
001616  010263  000036                      MOV     R2,36(R3)                       ; MLREG,*                       2307
001622  016703  036772'                     MOV     ML.ADDR,R3                      ;
001626  016366  000040  000022              MOV     40(R3),22(SP)                   ; *,ML.REG
001634  011102                              MOV     (R1),R2                         ; *,MLREG
001636  016703  036772'                     MOV     ML.ADDR,R3
001642  010263  000040                      MOV     R2,40(R3)                       ; MLREG,*
001646  016703  036772'                     MOV     ML.ADDR,R3                      ;                               2308
001652  016366  000034  000020              MOV     34(R3),20(SP)                   ; *,ML.REG
001660  011102                              MOV     (R1),R2                         ; *,MLREG
001662  016703  036772'                     MOV     ML.ADDR,R3
001666  010263  000034                      MOV     R2,34(R3)                       ; MLREG,*
001672  016703  036772'                     MOV     ML.ADDR,R3
001676  016366  000024  000016              MOV     24(R3),16(SP)                   ; *,ML.REG
001704  016602  000016                      MOV     16(SP),R2                       ; ML.REG,MLREG
001710  152702  000020                      BISB    #20,R2                          ; *,MLREG
001714  016703  036772'                     MOV     ML.ADDR,R3
001720  010263  000024                      MOV     R2,24(R3)                       ; MLREG,*
001724  005204                              INC     R4                              ; BUF.INDEX                     2310
001726  005200                              INC     R0                              ; WRD.CNT                       2304
001730  020027  000177                      CMP     R0,#177                         ; WRD.CNT,*
001734  003713                              BLE     2$
001736  062705  000002                      ADD     #2,R5                           ; *,S.BUF.INDEX                 2313
001742  010504                              MOV     R5,R4                           ; S.BUF.INDEX,BUF.INDEX         2314
001744  005266  000006                      INC     6(SP)                           ; SEC.CNT                       2300
001750  026666  000006  000010      3$:     CMP     6(SP),10(SP)                    ; SEC.CNT,*
001756  003700                              BLE     1$
001760  016703  036772'                     MOV     ML.ADDR,R3                      ;                               2315
001764  016366  000010  000014              MOV     10(R3),14(SP)                   ; *,ML.REG
001772  016602  000014                      MOV     14(SP),R2                       ; ML.REG,MLREG
```

```
001776  152702  000040                  BISB    #40,R2                          ; *,MLREG
002002  016703  036772'                 MOV     ML.ADDR,R3
002006  010263  000010                  MOV     R2,10(R3)                       ; MLREG,*
002012  016703  036772'                 MOV     ML.ADDR,R3
002016  016366  000010  000012          MOV     10(R3),12(SP)                   ; *,ML.REG
002024  016602  000012                  MOV     12(SP),R2                       ; ML.REG,MLREG
002030  016705  036776'                 MOV     ML.DUT,R5
002034  042705  177770                  BIC     #177770,R5
002040  142702  000007                  BICB    #7,R2                           ; *,MLREG
002044  050502                          BIS     R5,R2                           ; *,MLREG
002046  016703  036772'                 MOV     ML.ADDR,R3
002052  010263  000010                  MOV     R2,10(R3)                       ; MLREG,*
002056  062706  000026                  ADD     #26,SP                          ;                                       2273
002062  000207                          RTS     PC

; Routine Size:  126 words
; Maximum stack depth per invocation:  17 words
```

```
;          2319  routine DM_1_0_LOAD (TEMP_ARR$BNK_SEL, DATA) : novalue =
;          2320      begin
;          2321
;          2322  ;**
;          2323  ;   FUNCTIONAL DESCRIPTION: DIAGNOSTIC MODE LOAD ONES AND ZEROES DATA
;          2324  ;
;          2325  ;                           THIS ROUTINE WILL LOAD VIA DIAGNOSTIC MODE
;          2326  ;                           ONES OR ZEROES " THE CONTENTS OF DATA " TO
;          2327  ;                           THE DESIGNATED ARRAY AND BANK SELECTED BY
;          2328  ;                           TEMP ARRAY BANK SELECT ARGUMENT.
;          2329  ;
;          2330  ;   FORMAL PARAMETERS:
;          2331  ;                   TEMP_ARR$BNK_SEL:
;          2332  ;                           THIS ARGUMENT STORES THE SELECTED ARRAYS ARRAY
;          2333  ;                           SELECT NUMBER AND THE SELECTED ARRAYS BANK SEL-
;          2334  ;                           ECT NUMBER.
;          2335  ;                   DATA:
;          2336  ;                           THIS ARGUMENT CONTAINS THE SELECTED DATA PATTERN
;          2337  ;                           TO BE WRITTEN THE THE ARRAY AND BANK SELECTED BY
;          2338  ;                           ARRAY BANK SELECT.
;          2339  ;--
;          2340
;          2341                                                  !SINGLE STEP DMA WRITE MODE
;          2342      DM_WRT_TRANSFER (SIZE = -256, DST = .TEMP_ARR$BNK_SEL, SRC = WRT_BUF);
;          2343      WRT_RH (MLD1, DB1_REG, .DATA);              .LOAD DATA REG 1 WITH DATA
;          2344      WRT_RH (MLD2, DB2_REG, .DATA);              !LOAD DATA REG 2 WITH DATA
;          2345      WRT_RH (MLE2, E2_REG, .DATA);               .LOAD DATA REG 3 WITH DATA
;          2346
;          2347      incr SEC_CNT from 0 to .BNK_NUM_SEC do      !LOAD ALL SECTORS IN BANK WITH DATA
;          2348
;          2349          incr WRD_CNT from 0 to 127 do           !LOAD ALL WORDS IN SECTOR WITH DATA
;          2350              DAT_CLK;                            !CLOCK THE DATA INTO THE ML-11
;          2351
;          2352      incr WRD_CNT from 0 to (127 + 511*2) do     !FILL THE WRITE BUFFER UP WITH DATA
;          2353          WRT_BUF [.WRD_CNT, WRD] = .DATA;
;          2354
;          2355      CLR_MBUS;                                   !CLEAR THE SINGLE STEP DMA MODE
;          2356      end;


                                        .SBTTL  DM.1.0.LOAD ROUTINE DECLARATIONS
002064  004167  000000G          DM.1.0.LOAD:
                                        JSR     R1,$SAVE5                       ;                       2319
002070  162706  000014                  SUB     #14,SP
002074  012746  177400                  MOV     #-400,-(SP)                     ;                       2342
002100  011667  037024'                 MOV     (SP),SIZE
002104  016646  000036                  MOV     36(SP),-(SP)                    ; TEMP.ARR$BNK.SE,*
002110  011667  037026'                 MOV     (SP),DST
002114  012746  001232'                 MOV     #WRT.BUF,-(SP)
002120  011667  037030'                 MOV     (SP),SRC
002124  004767  177034                  JSR     PC,DM.WRT.TRANSFER
002130  016701  036772'                 MOV     ML.ADDR,R1                      ;                       2343
```

```
002134  016166  000036  000020          MOV     36(R1),20(SP)                   ; *,ML.REG
002142  016605  000040                  MOV     40(SP),R5                       ; DATA,*
002146  010500                          MOV     R5,R0                           ; *,MLREG
002150  016701  036772'                 MOV     ML.ADDR,R1
002154  010061  000036                  MOV     R0,36(R1)                       ; MLREG,*
002160  016701  036772'                 MOV     ML.ADDR,R1                      ;                                       2344
002164  016166  000040  000016          MOV     40(R1),16(SP)                   ; *,ML.REG
002172  016701  036772'                 MOV     ML.ADDR,R1
002176  010061  000040                  MOV     R0,40(R1)                       ; MLREG,*
002202  016701  036772'                 MOV     ML.ADDR,R1                      ;                                       2345
002206  016166  000034  000014          MOV     34(R1),14(SP)                   ; *,ML.REG
002214  016701  036772'                 MOV     ML.ADDR,R1
002220  010061  000034                  MOV     R0,34(R1)                       ; MLREG,*
002224  016704  037014'                 MOV     BNK.NUM.SEC,R4                  ;                                       2347
002230  005003                          CLR     R3                              ; SEC.CNT
002232  000423                          BR      3$
002234  005002                  1$:     CLR     R2                              ; WRD.CNT                               2349
002236  016701  036772'         2$:     MOV     ML.ADDR,R1
002242  016166  000024  000012          MOV     24(R1),12(SP)                   ; *,ML.REG
002250  016600  000012                  MOV     12(SP),R0                       ; ML.REG,MLREG
002254  152700  000020                  BISB    #20,R0                          ; *,MLREG
002260  016701  036772'                 MOV     ML.ADDR,R1
002264  010061  000024                  MOV     R0,24(R1)                       ; MLREG,*
002270  005202                          INC     R2                              ; WRD.CNT
002272  020227  000177                  CMP     R2,#177                         ; WRD.CNT,*
002276  003757                          BLE     2$
002300  005203                          INC     R3                              ; SEC.CNT                               2347
002302  020304                  3$:     CMP     R3,R4                           ; SEC.CNT,*
002304  003753                          BLE     1$
002306  005000                          CLR     R0                              ; WRD.CNT                               2352
002310  010001                  4$:     MOV     R0,R1                           ; WRD.CNT,*                             2353
002312  006301                          ASL     R1
002314  010561  001232'                 MOV     R5,WRT.BUF(R1)
002320  005200                          INC     R0                              ; WRD.CNT                               2352
002322  020027  002175                  CMP     R0,#2175                        ; WRD.CNT,*
002326  003770                          BLE     4$
002330  016701  036772'                 MOV     ML.ADDR,R1                      ;                                       2353
002334  016166  000010  000010          MOV     10(R1),10(SP)                   ; *,ML.REG
002342  016600  000010                  MOV     10(SP),R0                       ; ML.REG,MLREG
002346  152700  000040                  BISB    #40,R0                          ; *,MLREG
002352  016701  036772'                 MOV     ML.ADDR,R1
002356  010061  000010                  MOV     R0,10(R1)                       ; MLREG,*
002362  016701  036772'                 MOV     ML.ADDR,R1
002366  016166  000010  000006          MOV     10(R1),6(SP)                    ; *,ML.REG
002374  016600  000006                  MOV     6(SP),R0                        ; ML.REG,MLREG
002400  016705  036776'                 MOV     ML.DUT,R5
002404  042705  177770                  BIC     #177770,R5
002410  142700  000007                  BICB    #7,R0                           ; *,MLREG
002414  050500                          BIS     R5,R0                           ; *,MLREG
002416  016701  036772'                 MOV     ML.ADDR,R1
002422  010061  000010                  MOV     R0,10(R1)                       ; MLREG,*
00242(  062706  000022                  ADD     #22,SP                          ;                                       2319
```

```
002432  000207                                  RTS     PC

; Routine Size:  116 words
; Maximum stack depth per invocation:  15 words
```

```
2357  routine WRT_CHK_TRANSFER (SIZE, DST, SRC) : novalue
2358      begin
2359
2360  !++
2361  !  FUNCTIONAL DESCRIPTION: WRITE CHECK TRANSFER
2362  !
2363  !                          THIS ROUTINE WILL PERFORM A WRITE CHECK TRANSFER
2364  !                          AT THE DESIGNATED SOURCE AND DESTINATION ADDRESS.
2365  !
2366  !  FORMAL PARAMETERS:
2367  !                  SIZE:
2368  !                          STORES THE SIZE OF THE TRANSFER
2369  !                  DST:
2370  !                          STORES THE DESTINATION ADDRESS OF THE TRANSFER
2371  !                  SRC:
2372  !                          STORES THE SOURCE ADDRESS OF THE TRANSFER
2373  !--
2374
2375      CLR_MBUS;                                   !CLEAR THE MASS BUS
2376      ECC_DIS (ENABE);                            .DISABLE ERROR CORRECTION
2377      WRT_RH (MLCS2, DRV_SEL, .ML_DUT);           !SELECT THE DIVICE UNDER TEST
2378      WRT_RH (MLWC, WC_REG, .SIZE);               .LOAD THE TRANSFER SIZE
2379      WRT_RH (MLDA, DA_REG, .DST);                !LOAD THE DESTINATION ADDRESS
2380      WRT_RH (MLBA, BA_REG, .SRC);                .LOAD THE SOURCE ADDRESS
2381      WRT_RH (MLCS1, FUNC, FUNC_4);               !LOAD A WRITE CHECK FUNCTION WITH GO SET
2382
2383      if .ML_ADDR [MLCS1, SC]                     !DOES THE TRANSFER CAUSE A SC ERROR
2384      then
2385          begin
2386
2387          if not .ML_ADDR [MLCS2, WCE]            !WE EXPECT THE WRITE CHECK BIT TO BE SET
2388          then
2389              begin
2390              ERRSF (ERR_13, UNX_DRV_ERR_MSG, DUMPER);    !REPORT THE UNEXPECTED ERROR
2391              PRINTB (ONE_MSG, WT_CHK_XFER_MSG);  !PRINT WHAT TRANSFER CAUSED THE ERROR
2392              DODU (.L$UNIT);                     !DROP THIS UNIT
2393              DOCLN;                              !JUMP TO THE CLEAN UP CODE
2394              end;
2395
2396          end;
2397
2398      do                                          .DELAY FOR TRANSFER TO COMPLETE
2399          0
2400      until .ML_ADDR [MLDS, DRY];
2401
2402      if .ML_ADDR [MLCS1, SC]                     !DID THE TRANSFER CAUSE A SC ERROR
2403      then
2404          begin
2405
2406          if not .ML_ADDR [MLCS2, WCE]            !WE EXPECT THE WRITE CHECK BIT TO BE SET
2407          then
2408              begin
```

```
;         2409              ERRSF (ERR_13, UNX_DRV_ERR_MSG, DUMPER);    .REPORT THE UNEXPECTED ERROR
;         2410              PRINTB (ONE_MSG, WT_CHR_XFER_MSG);  .PRINT WHICH FUNCTION CAUSED THE ERROR
;         2411              DODU (.L$UNIT);                     .DROP THIS UNIT
;         2412              DOCLN;                              !JUMP TO THE CLEAN UP CODE
;         2413              end;
;         2414
;         2415          end;
;         2416
;         2417      end;


                                        .SBTTL  WRT.CHK.TRANSFE ROUTINE DECLARATIONS
002434  010146                  WRT.CHK.TRANSFE:
                                MOV     R1,-(SP)                        ;                                       2357
002436  162706  000032          SUB     #32,SP
002442  016700  036772'         MOV     ML.ADDR,R0                      ;                                       2358
002446  016066  000010  000030  MOV     10(R0),30(SP)                   ; *,ML.REG
002454  016601  000030          MOV     30(SP),R1                       ; ML.REG,MLREG
002460  152701  000040          BISB    #40,R1                          ; *,MLREG
002464  016700  036772'         MOV     ML.ADDR,R0
002470  010160  000010          MOV     R1,10(R0)                       ; MLREG,*
002474  016700  036772'         MOV     ML.ADDR,R0
002500  016066  000010  000026  MOV     10(R0),26(SP)                   ; *,ML.REG
002506  016601  000026          MOV     26(SP),R1                       ; ML.REG,MLREG
002512  016700  036776'         MOV     ML.DUT,R0
002516  042700  177770          BIC     #177770,R0
002522  142701  000007          BICB    #7,R1                           ; *,MLREG
002526  050001                  BIS     R0,R1                           ; *,MLREG
002530  016700  036772'         MOV     ML.ADDR,R0
002534  010160  000010          MOV     R1,10(R0)                       ; MLREG,*
002540  016700  036772'         MOV     ML.ADDR,R0                      ;                                       2376
002544  016066  000024  000024  MOV     24(R0),24(SP)                   ; *,ML.REG
002552  016601  000024          MOV     24(SP),R1                       ; ML.REG,MLREG
002556  152701  000002          BISB    #2,R1                           ; *,MLREG
002562  016700  036772'         MOV     ML.ADDR,R0
002566  010160  000024          MOV     R1,24(R0)                       ; MLREG,*
002572  016700  036772'         MOV     ML.ADDR,R0                      ;                                       2377
002576  016066  000010  000022  MOV     10(R0),22(SP)                   ; *,ML.REG
002604  016601  000022          MOV     22(SP),R1                       ; ML.REG,MLREG
002610  016700  036776'         MOV     ML.DUT,R0
002614  042700  177770          BIC     #177770,R0
002620  142701  000007          BICB    #7,R1                           ; *,MLREG
002624  050001                  BIS     R0,R1                           ; *,MLREG
002626  016700  036772'         MOV     ML.ADDR,R0
002632  010160  000010          MOV     R1,10(R0)                       ; MLREG,*
002636  016700  036772'         MOV     ML.ADDR,R0                      ;                                       2378
002642  016066  000002  000020  MOV     2(R0),20(SP)                    ; *,ML.REG
002650  016601  000042          MOV     42(SP),R1                       ; SIZE,MLREG
002654  016700  036772'         MOV     ML.ADDR,R0
002660  010160  000002          MOV     R1,2(R0)                        ; MLREG,*
002664  016700  036772'         MOV     ML.ADDR,R0                      ;                                       2379
002670  016066  000006  000016  MOV     6(R0),16(SP)                    ; *,ML.REG
```

```
002676  016601  000040
002702  016700  036772'
002706  010160  000006
002712  016700  036772'
002716  016066  000004  000014
002724  016601  000036
002730  016700  036772'
002734  010160  000004
002740  017766  036772' 000012
002746  016600  000012
002752  142700  000077
002756  152700  000051
002762  010077  036772'
002766  017766  036772' 000010
002774  100033
002776  016700  036772'
003002  016066  000010  000006
003010  032766  040000  000006
003016  001022
003020  104454
003022  000015
003024  001362'
003026  000000'
003030  012746  001546'
003034  012746  001630'
003040  012746  000002
003044  010600
003046  104414
003050  016700  000000G
003054  104451
003056  104444
003060  062706  000006
003064  016700  036772'          1$:  MOV     ML.ADDR,R0                      ;                                       2400
003070  016066  000012  000004        MOV     12(R0),4(SP)                    ; *,ML.REG
003076  105766  000004                TSTB    4(SP)                           ; ML.REG
003102  100370                       BPL     1$
003104  017766  036772' 000002        MOV     @ML.ADDR,2(SP)                  ; *,ML.REG                              2402
003112  100031                       BPL     2$
003114  016700  036772'              MOV     ML.ADDR,R0                      ;                                       2406
003120  016016  000010              MOV     10(R0),(SP)                     ; *,ML.REG
003124  032716  040000              BIT     #40000,(SP)                     ; *,ML.REG
003130  001022                      BNE     2$
003132  104454                      TRAP    54                              ;                                       2409
003134  000015                      .WORD   15
003136  001362'                     .WORD   UNX.DRV.ERR.MSG
003140  000000'                     .WORD   DUMPER
003142  012746  001546'              MOV     #WT.CHK.XFER.MSG,-(SP)          ;                                       2410
003146  012746  001630'              MOV     #ONE.MSG,-(SP)
003152  012746  000002              MOV     #2,-(SP)
003156  010600                      MOV     SP,R0                           ; SP,*
003160  104414                      TRAP    14
003162  016700  000000G             MOV     L$UNIT,R0                       ;                                       2411
```

```
MOV     40(SP),R1                       ; DST,MLREG
MOV     ML.ADDR,R0
MOV     R1,6(R0)                        ; MLREG,*
MOV     ML.ADDR,R0                      ;                                       2380
MOV     4(R0),14(SP)                    ; *,ML.REG
MOV     36(SP),R1                       ; SRC,MLREG
MOV     ML.ADDR,R0
MOV     R1,4(R0)                        ; MLREG,*
MOV     @ML.ADDR,12(SP)                 ; *,ML.REG                              2381
MOV     12(SP),R0                       ; ML.REG,MLREG
BICB    #77,R0                          ; *,MLREG
BISB    #51,R0                          ; *,MLREG
MOV     R0,@ML.ADDR                     ; MLREG,*
MOV     @ML.ADDR,10(SP)                 ; *,ML.REG                              2383
BPL     1$
MOV     ML.ADDR,R0                      ;                                       2387
MOV     10(R0),6(SP)                    ; *,ML.REG
BIT     #40000,6(SP)                    ; *,ML.REG
BNE     1$
TRAP    54                              ;                                       2390
.WORD   15
.WORD   UNX.DRV.ERR.MSG
.WORD   DUMPER
MOV     #WT.CHK.XFER.MSG,-(SP)          ;                                       2391
MOV     #ONE.MSG,-(SP)
MOV     #2,-(SP)
MOV     SP,R0                           ; SP,*
TRAP    14
MOV     L$UNIT,R0                       ;                                       2392
TRAP    51
TRAP    44
ADD     #6,SP                           ;                                       2389
```

```
003166  104451                          TRAP    51
003170  104444                          TRAP    44
003172  062706  000006                  ADD     #6,SP                   ;                       2408
003176  062706  000032          2$:     ADD     #32,SP                  ;                       2357
003202  012601                          MOV     (SP)+,R1
003204  000207                          RTS     PC

; Routine Size:  181 words
; Maximum stack depth per invocation:  17 words
```

```
2418  routine WRT_TRANSFER (SIZE, DST, SRC) : novalue =
2419      begin
2420
2421  !++
2422  !  FUNCTIONAL DESCRIPTION: WRITE TRANSFER
2423  !
2424  !                          THIS ROUTINE WILL PERFORM A WRITE TRANSFER
2425  !                          AT THE DESIGNATED SOURCE AND DESTINATIONS
2426  !                          ADDRESS.
2427  !
2428  !  FORMAL PARAMETERS:
2429  !                  SIZE:
2430  !                          STORES THE SIZE OF THE TRANSFER
2431  !                  DST:
2432  !                          STORES THE DESTINATION ADDRESS OF THE TRANSFER
2433  !                  SRC:
2434  !                          STORES THE SOURCE ADDRESS OF THE TRANSFER
2435  !--
2436
2437      CLR_MBUS;                                       .CLEAR THE MASS BUS
2438      WRT_RH (MLCS2, DRV_SEL, .ML_DUT);               .SELECT THE DIVICE UNDER TEST
2439      WRT_RH (MLWC, WC_REG, .SIZE);                   .LOAD THE TRANSFER SIZE
2440      WRT_RH (MLDA, DA_REG, .DST);                    .LOAD THE DESTINATION ADDRESS
2441      WRT_RH (MLBA, BA_REG, .SRC);                    .LOAD THE SOURCE ADDRESS
2442      WRT_RH (MLCS1, FUNC, FUNC_5);                   !LOAD A WRITE FUNCTION WITH GO SET
2443
2444      if .ML_ADDR [MLCS1, SC]                         .DOES THE TRANSFER CAUSE A SC ERROR
2445      then
2446          begin
2447          ERRSF (ERR_13, UNX_DRV_ERR_MSG, DUMPER);        .REPORT THE UNEXPECTED ERROR
2448          PRINTB (ONE_MSG, WRT_XFER_MSG);             !PRINT WHICH FUNCTION CAUSED THE ERROR
2449          DODU (.L$UNIT);                             .DROP THIS UNIT
2450          DOCLN;                                      !!JUMP TO THE CLEAN UP CODE
2451          end;
2452
2453      do                                              .DELAY FOR TRANSFER TO COMPLETE
2454          0
2455      until .ML_ADDR [MLDS, DRY];
2456
2457      if .ML_ADDR [MLCS1, SC]                         !DID THE TRANSFER CAUSE A SC ERROR
2458      then
2459          begin
2460          ERRSF (ERR_13, UNX_DRV_ERR_MSG, DUMPER);        !REPORT THE UNEXPECTED ERROR
2461          PRINTB (ONE_MSG, WRT_XFER_MSG);             .REPORT WHICH FUNCTION CAUSED THE ERROR
2462          DODU (.L$UNIT);                             .DROP THIS UNIT
2463          DOCLN;                                      !JUMP TO THE CLEAN UP CODE
2464          end;
2465
2466      end;

                                  .SBTTL  WRT.TRANSFER ROUTINE DECLARATIONS
```

```
003206  010146                          WRT.TRANSFER:
                                                MOV     R1,-(SP)                ;                               2418
003210  162706  000024                          SUB     #24,SP
003214  016700  036772'                        MOV     ML.ADDR,R0              ;                               2419
003220  016066  000010  000022                  MOV     10(R0),22(SP)           ; *,ML.REG
003226  016601  000022                          MOV     22(SP),R1               ; ML.REG,MLREG
003232  152701  000040                          BISB    #40,R1                  ; *,MLREG
003236  016700  036772'                        MOV     ML.ADDR,R0
003242  010160  000010                          MOV     R1,10(R0)               ; MLREG,*
003246  016700  036772'                        MOV     ML.ADDR,R0
003252  016066  000010  000020                  MOV     10(R0),20(SP)           ; *,ML.REG
003260  016601  000020                          MOV     20(SP),R1               ; ML.REG,MLREG
003264  016700  036776'                        MOV     ML.DUT,R0
003270  042700  177770                          BIC     #177770,R0
003274  142701  000007                          BICB    #7,R1                   ; *,MLREG
003300  050001                                  BIS     R0,R1                   ; *,MLREG
003302  016700  036772'                        MOV     ML.ADDR,R0
003306  010160  000010                          MOV     R1,10(R0)               ; MLREG,*
003312  016700  036772'                        MOV     ML.ADDR,R0              ;                               2438
003316  016066  000010  000016                  MOV     10(R0),16(SP)           ; *,ML.REG
003324  016601  000016                          MOV     16(SP),R1               ; ML.REG,MLREG
003330  016700  036776'                        MOV     ML.DUT,R0
003334  042700  177770                          BIC     #177770,R0
003340  142701  000007                          BICB    #7,R1                   ; *,MLREG
003344  050001                                  BIS     R0,R1                   ; *,MLREG
003346  016700  036772'                        MOV     ML.ADDR,R0
003352  010160  000010                          MOV     R1,10(R0)               ; MLREG,*
003356  016700  036772'                        MOV     ML.ADDR,R0              ;                               2439
003362  016066  000002  000014                  MOV     2(R0),14(SP)            ; *,ML.REG
003370  016601  000034                          MOV     34(SP),R1               ; SIZE,MLREG
003374  016700  036772'                        MOV     ML.ADDR,R0
003400  010160  000002                          MOV     R1,2(R0)                ; MLREG,*
003404  016700  036772'                        MOV     ML.ADDR,R0              ;                               2440
003410  016066  000006  000012                  MOV     6(R0),12(SP)            ; *,ML.REG
003416  016601  000032                          MOV     32(SP),R1               ; DST,MLREG
003422  016700  036772'                        MOV     ML.ADDR,R0
003426  010160  000006                          MOV     R1,6(R0)                ; MLREG,*
003432  016700  036772'                        MOV     ML.ADDR,R0              ;                               2441
003436  016066  000004  000010                  MOV     4(R0),10(SP)            ; *,ML.REG
003444  016601  000030                          MOV     30(SP),R1               ; SRC,MLREG
003450  016700  036772'                        MOV     ML.ADDR,R0
003454  010160  000004                          MOV     R1,4(R0)                ; MLREG,*
003460  017766  036772'  000006                MOV     @ML.ADDR,6(SP)          ; *,ML.REG                      2442
003466  016600  000006                          MOV     6(SP),R0                ; ML.REG,MLREG
003472  142700  000077                          BICB    #77,R0                  ; *,MLREG
003476  152700  000061                          BISB    #61,R0                  ; *,MLREG
003502  010077  036772'                        MOV     R0,@ML.ADDR             ; MLREG,*
003506  017766  036772'  000004                MOV     @ML.ADDR,4(SP)          ; *,ML.REG                      2444
003514  100022                                  BPL     1$
003516  104454                                  TRAP    54                      ;                               2447
003520  000015                                  .WORD   15
003522  001362'                                .WORD   UNX.DRV.ERR.MSG
```

```
003524  000000'                         .WORD   DUMPER
003526  012746  001474'                 MOV     #WRT.XFER.MSG,-(SP)             ;                       2448
003532  012746  001630'                 MOV     #ONE.MSG,-(SP)
003536  012746  000002                  MOV     #2,-(SP)
003542  010600                          MOV     SP,R0                           ; SP,*
003544  104414                          TRAP    14
003546  016700  000000G                 MOV     L$UNIT,R0                       ;                       2449
003552  104451                          TRAP    51
003554  104444                          TRAP    44
003556  062706  000006                  ADD     #6,SP                           ;                       2446
003562  016700  036772'          1$:     MOV     ML.ADDR,R0                      ;                       2455
003566  016066  000012  000002         MOV     12(R0),2(SP)                    ; *,ML.REG
003574  105766  000002                  TSTB    2(SP)                           ; ML.REG
003600  100370                          BPL     1$
003602  017716  036772'                 MOV     @ML.ADDR,(SP)                   ; *,ML.REG              2457
003606  100022                          BPL     2$
003610  104454                          TRAP    54                              ;                       2460
003612  000015                          .WORD   15
003614  001362'                         .WORD   UNX.DRV.ERR.MSG
003616  000000'                         .WORD   DUMPER
003620  012746  001474'                 MOV     #WRT.XFER.MSG,-(SP)             ;                       2461
003624  012746  001630'                 MOV     #ONE.MSG,-(SP)
003630  012746  000002                  MOV     #2,-(SP)
003634  010600                          MOV     SP,R0                           ; SP,*
003636  104414                          TRAP    14
003640  016700  000000G                 MOV     L$UNIT,R0                       ;                       2462
003644  104451                          TRAP    51
003646  104444                          TRAP    44
003650  062706  000006                  ADD     #6,SP                           ;                       2459
003654  062706  000024          2$:     ADD     #24,SP                          ;                       2418
003660  012601                          MOV     (SP)+,R1
003662  000207                          RTS     PC

; Routine Size:  151 words
; Maximum stack depth per invocation:  14 words
```

```
;          2467  routine RD_TRANSFER (SIZE, DST, SRC) : novalue =
;          2468      begin
;          2469
;          2470  !++
;          2471  !  FUNCTIONAL DESCRIPTION: READ TRANSFER
;          2472  !
;          2473  !                          THIS ROUTINE PERFORMS A READ TRANSFER AT THE
;          2474  !                          DESIGNATED SOURCE AND DESTIONATION ADDRESSES.
;          2475  !
;          2476  !  FORMAL PARAMETERS:
;          2477  !                  SIZE:
;          2478  !                          STORES THE SIZE OF THE TRANSFER
;          2479  !                  DST:
;          2480  !                          STORES THE DESTINATION ADDRESS OF TRANSFER
;          2481  !                  SRC:
;          2482  !                          STORES THE SOURCE ADDRESS OF TRANSFER
;          2483  !--
;          2484
;          2485      CLR_MBUS;                                   !CLEAR THE MASS BUS
;          2486      ECC_DIS (ENABE);                            .DISABLE ERROR CORRECTION
;          2487      WRT_RH (MLCS2, DRV_SEL, .ML_DUT);           !SELECT THE DIVICE UNDER TEST
;          2488      WRT_RH (MLWC, WC_REG, .SIZE);               .LOAD THE TRANSFER SIZE
;          2489      WRT_RH (MLDA, DA_REG, .SRC);                !LOAD THE SOURCE ADDRESS
;          2490      WRT_RH (MLBA, BA_REG, .DST);                !LOAD THE DESTINATION ADDRESS
;          2491      WRT_RH (MLCS1, FUNC, FUNC_6);               .LOAD A READ FUNCTION WITH GO SET
;          2492
;          2493      if .ML_ADDR [MLCS1, SC]                     .DOES THIS FUNCTION CAUSE A SC ERROR
;          2494      then
;          2495          begin
;          2496          ERRSF (ERR_13, UNX_DRV_ERR_MSG, DUMPER);        !REPORT THE UNEXPECTED ERROR
;          2497          PRINTB (ONE_MSG, RD_XFER_MSG);          !REPORT WHICH FUNCTION CAUSE THE ERROR
;          2498          DODU (.L$UNIT);                         .DROP THIS UNIT
;          2499          DOCLN;                                  .JUMP TO THE CLEAN UP CODE
;          2500          end;
;          2501
;          2502      do                                          !DELAY FOR THE TRANSFER TO COMPLETE
;          2503          0
;          2504      until .ML_ADDR [MLDS, DRY];
;          2505
;          2506      if .ML_ADDR [MLCS1, SC]                     .DID THE TRANSFER CAUSE A SC ERROR
;          2507      then
;          2508          begin
;          2509          ERRSF (ERR_13, UNX_DRV_ERR_MSG, DUMPER);        !REPORT THE UNEXPECTED ERROR
;          2510          PRINTB (ONE_MSG, RD_XFER_MSG);          !PRINT WHICH FUNCTION CAUSED THE ERROR
;          2511          DODU (.L$UNIT);                         !DROP THIS UNIT
;          2512          DOCLN;                                  !JUMP TO THE CLEAN UP CODE
;          2513          end;
;          2514
;          2515      end;

                                  .SBTTL  RD.TRANSFER ROUTINE DECLARATIONS
```

```
003664  010146                          RD.TRANSFER:
                                                MOV     R1,-(SP)                ;                               2467
003666  162706  000026                          SUB     #26,SP
003672  016700  036772'                        MOV     ML.ADDR,R0              ;                               2468
003676  016066  000010  000024                 MOV     10(R0),24(SP)           ; *,ML.REG
003704  016601  000024                         MOV     24(SP),R1               ; ML.REG,MLREG
003710  152701  000040                         BISB    #40,R1                  ; *,MLREG
003714  016700  036772'                        MOV     ML.ADDR,R0
003720  010160  000010                         MOV     R1,10(R0)               ; MLREG,*
003724  016700  036772'                        MOV     ML.ADDR,R0
003730  016066  000010  000022                 MOV     10(R0),22(SP)           ; *,ML.REG
003736  016601  000022                         MOV     22(SP),R1               ; ML.REG,MLREG
003742  016700  036776'                        MOV     ML.DUT,R0
003746  042700  177770                         BIC     #177770,R0
003752  142701  000007                         BICB    #7,R1                   ; *,MLREG
003756  050001                                BIS     R0,R1                   ; *,MLREG
003760  016700  036772'                        MOV     ML.ADDR,R0
003764  010160  000010                         MOV     R1,10(R0)               ; MLREG,*
003770  016700  036772'                        MOV     ML.ADDR,R0              ;                               2486
003774  016066  000024  000020                 MOV     24(R0),20(SP)           ; *,ML.REG
004002  016601  000020                         MOV     20(SP),R1               ; ML.REG,MLREG
004006  152701  000002                         BISB    #2,R1                   ; *,MLREG
004012  016700  036772'                        MOV     ML.ADDR,R0
004016  010160  000024                         MOV     R1,24(R0)               ; MLREG,*
004022  016700  036772'                        MOV     ML.ADDR,R0              ;                               2487
004026  016066  000010  000016                 MOV     10(R0),16(SP)           ; *,ML.REG
004034  016601  000016                         MOV     16(SP),R1               ; ML.REG,MLREG
004040  016700  036776'                        MOV     ML.DUT,R0
004044  042700  177770                         BIC     #177770,R0
004050  142701  000007                         BICB    #7,R1                   ; *,MLREG
004054  050001                                BIS     R0,R1                   ; *,MLREG
004056  016700  036772'                        MOV     ML.ADDR,R0
004062  010160  000010                         MOV     R1,10(R0)               ; MLREG,*
004066  016700  036772'                        MOV     ML.ADDR,R0              ;                               2488
004072  016066  000002  000014                 MOV     2(R0),14(SP)            ; *,ML.REG
004100  016601  000036                         MOV     36(SP),R1               ; SIZE,MLREG
004104  016700  036772'                        MOV     ML.ADDR,R0
004110  010160  000002                         MOV     R1,2(R0)                ; MLREG,*
004114  016700  036772'                        MOV     ML.ADDR,R0              ;                               2489
004120  016066  000006  000012                 MOV     6(R0),12(SP)            ; *,ML.REG
004126  016601  000032                         MOV     32(SP),R1               ; SRC,MLREG
004132  016700  036772'                        MOV     ML.ADDR,R0
004136  010160  000006                         MOV     R1,6(R0)                ; MLREG,*
004142  016700  036772'                        MOV     ML.ADDR,R0              ;                               2490
004146  016066  000004  000010                 MOV     4(R0),10(SP)            ; *,ML.REG
004154  016601  000034                         MOV     34(SP),R1               ; DST,MLREG
004160  016700  036772'                        MOV     ML.ADDR,R0
004164  010160  000004                         MOV     R1,4(R0)                ; MLREG,*
004170  017766  036772'  000006                MOV     @ML.ADDR,6(SP)          ; *,ML.REG                      2491
004176  016600  000006                         MOV     6(SP),R0                ; ML.REG,MLREG
004202  142700  000077                         BICB    #77,R0                  ; *,MLREG
004206  152700  000071                         BISB    #71,R0                  ; *,MLREG
```

```
004212  010077  036772'                      MOV     R0,@ML.ADDR                   ; MLREG,*
004216  017766  036772' 000004               MOV     @ML.ADDR,4(SP)                ; *,ML.REG                 2493
004224  100022                               BPL     1$
004226  104454                               TRAP    54                            ;                          2496
004230  000015                               .WORD   15
004232  001362'                              .WORD   UNX.DRV.ERR.MSG
004234  000000'                              .WORD   DUMPER
004236  012746  001424'                      MOV     #RD.XFER.MSG,-(SP)            ;                          2497
004242  012746  001630'                      MOV     #ONE.MSG,-(SP)
004246  012746  000002                       MOV     #2,-(SP)
004252  010600                               MOV     SP,R0                         ; SP,*
004254  104414                               TRAP    14
004256  016700  000000G                      MOV     L$UNIT,R0                     ;                          2498
004262  104451                               TRAP    51
004264  104444                               TRAP    44
004266  062706  000006                       ADD     #6,SP                         ;                          2495
004272  016700  036772'              1$:     MOV     ML.ADDR,R0                    ;                          2504
004276  016066  000012  000002               MOV     12(R0),2(SP)                  ; *,ML.REG
004304  105766  000002                       TSTB    2(SP)                         ; ML.REG
004310  100370                               BPL     1$
004312  017716  036772'                      MOV     @ML.ADDR,(SP)                 ; *,ML.REG                 2506
004316  100022                               BPL     2$
004320  104454                               TRAP    54                            ;                          2509
004322  000015                               .WORD   15
004324  001362'                              .WORD   UNX.DRV.ERR.MSG
004326  000000'                              .WORD   DUMPER
004330  012746  001424'                      MOV     #RD.XFER.MSG,-(SP)            ;                          2510
004334  012746  001630'                      MOV     #ONE.MSG,-(SP)
004340  012746  000002                       MOV     #2,-(SP)
004344  010600                               MOV     SP,R0                         ; SP,*
004346  104414                               TRAP    14
004350  016700  000000G                      MOV     L$UNIT,R0                     ;                          2511
004354  104451                               TRAP    51
004356  104444                               TRAP    44
004360  062706  000006                       ADD     #6,SP                         ;                          2508
004364  062706  000026               2$:     ADD     #26,SP                        ;                          2467
004370  012601                               MOV     (SP)+,R1
004372  000207                               RTS     PC

; Routine Size:  164 words
; Maximum stack depth per invocation:  15 words
```

```
;          2516  routine I_CHIP_TBL : novalue =
;          2517      begin
;          2518
;          2519  !++
;          2520  !   FUNCTIONAL DESCRIPTION: INITIALIZE THE BAD CHIP TABLE
;          2521  !
;          2522  !                           THIS ROUTINE WILL CLEAR A L 39 TABLE
;          2523  !                           ENTRIES TO ZERO.
;          2524  !--
;          2525
;          2526      incr TBL_INDEX from 0 to 38 do              .INITIALIZE THE BAD CHIP TABLE
;          2527          CHIP_TBL [.TBL_INDEX, ALL] = ZEROES;
;          2528
;          2529      end;


                                        .SBTTL  I.CHIP.TBL ROUTINE DECLARATIONS
004374  010146                    I.CHIP.TBL:
                                            MOV     R1,-(SP)                        ;                       2516
004376  005000                            CLR     R0                              ; TBL.INDEX             2526
004400  010001                    1$:     MOV     R0,R1                           ; TBL.INDEX,*           2527
004402  006301                            ASL     R1
004404  005061  026626'                    CLR     CHIP.TBL(R1)
004410  005200                            INC     R0                              ; TBL.INDEX             2526
004412  020027  000046                    CMP     R0,#46                          ; TBL.INDEX,*
004416  003770                            BLE     1$
004420  012601                            MOV     (SP)+,R1                        ;                       2516
004422  000207                            RTS     PC

; Routine Size:  12 words
; Maximum stack depth per invocation:  2 words
```

```
;          2530  routine I_TMP_BLST_TBL : novalue =
;          2531      begin
;          2532
;          2533  .++
;          2534  !  FUNCTIONAL DESCRIPTION: INITIALIZE THE TEMP BLAST TABLE
;          2535  !
;          2536  !                          THIS ROUTINE WILL CLEAR ALL 100 ENTRIES
;          2537  !                          OF THE TEMP BLAST TABLE TO ZEROES.
;          2538  !--
;          2539
;          2540      incr TBL_INDEX from 0 to 9 do                !INITIALIZE THE TEMPORARY BLAST TABLE
;          2541          TMP_BLST_TBL [.TBL_INDEX, FULL_WRD] = ZEROFS;
;          2542
;          2543      end;


                                        .SBTTL  I.TMP.BLST.TBL ROUTINE DECLARATIONS
004424  010146                  I.TMP.BLST.TBL:
                                        MOV     R1,-(SP)                ;                                       2530
004426  005000                          CLR     R0                      ; TBL.INDEX                             2540
004430  010001                  1$:     MOV     R0,R1                   ; TBL.INDEX,*                           2541
004432  006301                          ASL     R1
004434  005061  026744'                 CLR     TMP.BLST.TBL(R1)
004440  005200                          INC     R0                      ; TBL.INDEX                             2540
004442  020027  000011                  CMP     R0,#11                  ; TBL.INDEX,*
004446  003770                          BLE     1$
004450  012601                          MOV     (SP)+,R1                ;                                       2530
004452  000207                          RTS     PC

; Routine Size:  12 words
; Maximum stack depth per invocation:  2 words
```

```
;          2544  routine I_COL_CNT_TBL : novalue
;          2545      begin
;          2546
;          2547  !++
;          2548  !  FUNCTIONAL DESCRIPTION: INITIALIZE  THE COLUMN COUNT TABLE
;          2549  !
;          2550  !                          THIS ROUTINE WILL CLEAR THE COLUMN COUNT
;          2551  !                          TABLES 256 ENTRIES TO ZEROES.
;          2552  !--
;          2553
;          2554      incr TBL_INDEX from 0 to 255 do              .INITIALIZE THE COLUMN COUNT TABLE
;          2555          COL_CNT_TBL [.TBL_INDEX]   ZEROES;
;          2556
;          2557      end;


                                        .SBTTL  I.COL.CNT.TBL ROUTINE DECLARATIONS
004454  005000                          I.COL.CNT.TBL:
                                        CLR     R0                      ; TBL.INDEX                     2554
004456  105060  000000'                1$:     CLRB    COL.CNT.TBL(R0)         ; *(TBL.INDEX)                  2555
004462  005200                                  INC     R0                      ; TBL.INDEX                     2554
004464  020027  000377                        CMP     R0,#377                 ; TBL.INDEX,*
004470  003772                                BLE     1$
004472  000207                                RTS     PC                      ;                               2544

; Routine Size:  8 words
; Maximum stack depth per invocation:  0 words
```

```
;          2558  routine I_REM_TBL : novalue =
;          2559      begin
;          2560
;          2561  !++
;          2562  !   FUNCTIONAL DESCRIPTION: INITIALIZE THE REMAINDER TABLE
;          2563  !
;          2564  !                           THIS ROUTINE WILL CLEAR THE REMAINDER
;          2565  !                           TABLES 100 ENTRIES TO ZEROES.
;          2566  !--
;          2567
;          2568      incr TBL_INDEX from 0 to 99 do              !INITIALIZE THE REMAINDER AND COPIED REM TABLE
;          2569          begin
;          2570          REM_TBL [.TBL_INDEX, RO_CL] = ZEROES;
;          2571          COPIED_REM_TBL [.TBL_INDEX, RO_CL] = ZEROES;
;          2572          end;
;          2573
;          2574      end;


                                        .SBTTL  I.REM.TBL ROUTINE DECLARATIONS
004474  010146                  I.REM.TBL:
                                        MOV     R1,-(SP)                        ;                                       2558
004476  005000                          CLR     R0                              ; TBL.INDEX                             2568
004500  010001                  1$:     MOV     R0,R1                           ; TBL.INDEX,*                           2570
004502  006301                          ASL     R1
004504  005061  000400'                  CLR     REM.TBL(R1)
004510  005061  000710'                  CLR     COPIED.REM.TBL(R1)              ;                                       2571
004514  005200                          INC     R0                              ; TBL.INDEX                             2568
004516  020027  000143                  CMP     R0,#143                         ; TBL.INDEX,*
004522  003766                          BLE     1$
004524  012601                          MOV     (SP)+,R1                        ;                                       2558
004526  000207                          RTS     PC

; Routine Size:  14 words
; Maximum stack depth per invocation:  2 words
```

```
;          2575  routine I_COL_PTR : novalue =
;          2576      begin
;          2577
;          2578  !++
;          2579  !  FUNCTIONAL DESCRIPTION: INITIALIZE THE COLUMN POINTER TABLE
;          2580  !
;          2581  !                          THIS ROUTINE WILL CLEAR THE COLUMN POINTER
;          2582  !                          TABLES 10 ENTRIES TO ZEROES.
;          2583  !--
;          2584
;          2585      incr TBL_INDEX from 0 to 9 do                !INITIALIZE THE COLUMN POINTER TABLE
;          2586          COL_PTR [.TBL_INDEX] = ZEROES;
;          2587
;          2588      end;


                                        .SBTTL  I.COL.PTR ROUTINE DECLARATIONS
004530  005000                    I.COL.PTR:
                                    CLR     R0                      ; TBL.INDEX                     2585
004532  105060  001220'            1$:     CLRB    COL.PTR(R0)         ; *(TBL.INDEX)                  2586
004536  005200                            INC     R0                  ; TBL.INDEX                     2585
004540  020027  000011                    CMP     R0,#11              ; TBL.INDEX,*
004544  003772                            BLE     1$
004546  000207                            RTS     PC                  ;                               2575

; Routine Size:  8 words
; Maximum stack depth per invocation:  0 words
```

```
;          2589  routine I_ERROR_MAP : novalue =
;          2590      begin
;          2591
;          2592  !++
;          2593  !  FUNCTIONAL DESCRIPTION: INITIALIZE THE ERROR MAP
;          2594  !
;          2595  !                          INITIALIZE THE ERROR MAPS 512 BLOCKS
;          2596  !                          TO ZEROES.
;          2597  !--
;          2598
;          2599      incr TBL_INDEX from 0 to %o'17776' by %o'2' do      !INTIALIZE THE ERROR MAP
;          2600          (ERROR_MAP + .TBL_INDEX) = ZEROES;
;          2601
;          2602      end;


                                        .SBTTL  I.ERROR.MAP ROUTINE DECLARATIONS
004550  005000                  I.ERROR.MAP:
                                CLR     R0                      ; TBL.INDEX                 2599
004552  005060  006626'        1$:     CLR     ERROR.MAP(R0)           ; *(TBL.INDEX)          2600
004556  062700  000002                  ADD     #2,R0                   ; *,TBL.INDEX           2599
004562  020027  017776                  CMP     R0,#17776               ; TBL.INDEX,*
004566  003771                          BLE     1$
004570  000207                          RTS     PC                      ;                       2589

; Routine Size:  9 words
; Maximum stack depth per invocation:  0 words
```

```
;          2603  routine I_BLAST_TBL : novalue =
;          2604      begin
;          2605
;          2606  !++
;          2607  !  FUNCTIONAL DESCRIPTION: INITIALIZE THE BLAST TABLE
;          2608  !
;          2609  !                         CLEAR THE BLAST TABLES 4 BLOCKS TO
;          2610  !                         TO ZEROES.
;          2611  !--
;          2612
;          2613      incr TBL_INDEX from 0 to %o'7776' by %o'2' do       !INITIALIZE THE BLAST TABLE
;          2614          (BLAST_TBL + .TBL_INDEX) = ZEROES;
;          2615
;          2616      end;


                                        .SBTTL  I.BLAST.TBL ROUTINE DECLARATIONS
004572  005000                  I.BLAST.TBL:
                                        CLR     R0                      ; TBL.INDEX                     2613
004574  005060  026770'          1$:     CLR     BLAST.TBL(R0)           ; *(TBL.INDEX)                  2614
004600  062700  000002                  ADD     #2,R0                   ; *,TBL.INDEX                   2613
004604  020027  007776                  CMP     R0,#7776                ; TBL.INDEX,*
004610  003771                          BLE     1$
004612  000207                          RTS     PC                      ;                               2603

; Routine Size:  9 words
; Maximum stack depth per invocation:  0 words
```

```
;        2617  routine L_RANDOM_DATA (TEMP_ARR$BNK_SEL) : novalue =
;        2618      begin
;        2619
;        2620  !++
;        2621  !  FUNCTIONAL DESCRIPTION: LOAD RANDOM DATA
;        2622  !
;        2623  !                          THIS ROUTINE WILL LOAD VIA MASS BUS
;        2624  !                          TRANSFERS RANDOM DATA TO THE SELECTED
;        2625  !                          ARRAY AND BANK SELECTED BY ARRAY BANK
;        2626  !                          SELECT.
;        2627  !
;        2628  !  FORMAL PARAMETERS:
;        2629  !                  TEMP_ARR$BNK_SEL:
;        2630  !                          THIS ARGUMENT STORES THE SELECTED ARRAY
;        2631  !                          AND BANKS ARRAY AND BANK SELECT NUMBERS.
;        2632  !--
;        2633
;        2634      local
;        2635          RANDOM_INDEX;                           !WRITE BUFFER RANDOM DATA INDEX
;        2636
;        2637      RANDOM_INDEX    ZERO;                       !THE FIRST TRANSFER STARTS AT WRITE BUF WORD ZERO
;        2638      GEN_RANDOM_DATA ();                         !GENERATE A WRITE BUF OF RANDOM DATA
;        2639
;        2640      incr SEC_CNT from 0 to .BNK_NUM_SEC do      !LOAD ALL SECTORS IN BANK WITH RANDOM DATA
;        2641          begin
;        2642          WRT_TRANSFER (SIZE = -256, DST = .TEMP_ARR$BNK_SEL + .SEC_CNT, SRC = WRT_BUF [.RANDOM_INDEX, WRD]);
;        2643          RANDOM_INDEX = .RANDOM_INDEX + 2;       !NEXT SECTOR STARTS 2 WORD DEEPER IN WRITE BUFFER
;        2644          end;
;        2645
;        2646      end;


                        .SBTTL  L.RANDOM.DATA ROUTINE DECLARATIONS
004614  004167  000000G  L.RANDOM.DATA:
                        JSR     R1,$SAVE4                       ;                                   2617
004620  005003          CLR     R3                              ; RANDOM.INDEX                      2637
004622  004767  173742  JSR     PC,GEN.RANDOM.DATA              ;                                   2638
004626  016704  037014' MOV     BNK.NUM.SEC,R4                  ;                                   2640
004632  005001          CLR     R1                              ; SEC.CNT
004634  000430          BR      2$
004636  012746  177400  1$:     MOV     #-400,-(SP)             ;                                   2642
004642  011667  037024' MOV     (SP),SIZE
004646  010102          MOV     R1,R2                           ; SEC.CNT,*
004650  066602  000016  ADD     16(SP),R2                       ; TEMP.ARR$BNK.SE,*
004654  010267  037026' MOV     R2,DST
004660  010246          MOV     R2,-(SP)
004662  010302          MOV     R3,R2                           ; RANDOM.INDEX,*
004664  006302          ASL     R2
004666  062702  001232' ADD     #WRT.BUF,R2
004672  010267  037030' MOV     R2,SRC
004676  010246          MOV     R2,-(SP)
004700  004767  176302  JSR     PC,WRT.TRANSFER
```

```
004704  062703  000002                    ADD     #2,R3           ; *,RANDOM.INDEX                2643
004710  062706  000006                    ADD     #6,SP           ;                               2641
004714  005201                            INC     R1              ; SEC.CNT                       2640
004716  020104                    2$:     CMP     R1,R4           ; SEC.CNT,*
004720  003746                            BLE     1$
004722  000207                            RTS     PC              ;                               2617

; Routine Size:  36 words
; Maximum stack depth per invocation:  8 words
```

```
;          2647  routine L_1_0_DATA (TEMP_ARR$BNK_SEL, DATA) : novalue =
;          2648      begin
;          2649
;          2650  .++
;          2651  !   FUNCTIONAL DESCRIPTION: LOAD ONES AND ZEROES DATA
;          2652  !                           THIS ROUTINE WILL LOAD THE SELECTED ARRAY AND BANK WITH ONES OR ZEROES DATA.
;          2653  !   FORMAL PARAMETERS:
;          2654  !                       TEMP_ARR$BNK_SEL:
;          2655  !                           THIS ARGUMENT STORES THE SELECTED ARRAYS ARRAY AND BANK SELECT NUMBER.
;          2656  .--
;          2657
;          2658      incr WRD_CNT from 0 to (127 + 511*2) do      .LOAD THE WRITE BUF WITH DATA
;          2659          WRT_BUF [.WRD_CNT, WRD] = .DATA;
;          2660
;          2661      incr SEC_CNT from 0 to .BNK_NUM_SEC do       .LOAD ALL SECTORS IN BANK WITH DATA
;          2662          WRT_TRANSFER (SIZE = -256, DST = .TEMP_ARR$BNK_SEL + .SEC_CNT, SRC = WRT_BUF);
;          2663
;          2664      end;


                                        .SBTTL  L.1.0.DATA ROUTINE DECLARATIONS
004724  004167  000000G          L.1.0.DATA:
                                        JSR     R1,$SAVE3                       ;                                       2647
004730  005000                          CLR     R0                              ; WRD.CNT                               2658
004732  010001                  1$:     MOV     R0,R1                           ; WRD.CNT,*                             2659
004734  006301                          ASL     R1
004736  016661  000012  001232'         MOV     12(SP),WRT.BUF(R1)              ; DATA,*
004744  005200                          INC     R0                              ; WRD.CNT                               2658
004746  020027  002175                  CMP     R0,#2175                        ; WRD.CNT,*
004752  003767                          BLE     1$
004754  016703  037014'                 MOV     BNK.NUM.SEC,R3                  ;                                       2661
004760  005002                          CLR     R2                              ; SEC.CNT
004762  000423                          BR      3$
004764  012746  177400          2$:     MOV     #-400,-(SP)                     ;                                       2662
004770  011667  037024'                 MOV     (SP),SIZE
004774  010201                          MOV     R2,R1                           ; SEC.CNT,*
004776  066601  000016                  ADD     16(SP),R1                       ; TEMP.ARR$BNK.SE,*
005002  010167  037026'                 MOV     R1,DST
005006  010146                          MOV     R1,-(SP)
005010  012746  001232'                 MOV     #WRT.BUF,-(SP)
005014  011667  037030'                 MOV     (SP),SRC
005020  004767  176162                  JSR     PC,WRT.TRANSFER
005024  062706  000006                  ADD     #6,SP
005030  005202                          INC     R2                              ; SEC.CNT                               2661
005032  020203                  3$:     CMP     R2,R3                           ; SEC.CNT,*
005034  003753                          BLE     2$
005036  000207                          RTS     PC                              ;                                       2647

; Routine Size:  38 words
; Maximum stack depth per invocation:  7 words
```

```
2665  routine L_CHIP_TBL (FAIL_CHIP, PAT_SEL) : novalue =
2666      begin
2667
2668  !++
2669  !  FUNCTIONAL DESCRIPTION: LOAD BAD CHIP TABLE
2670  !
2671  !                          THE BAD CHIP TABLE STORES FOR A BANK
2672  !                          ALL THE FAILING MOS RAMS AND THIER
2673  !                          FAILING DATA PATTERNS.  THE POSSIBLE
2674  !                          DATA PATTERNS THAT COULD FAIL ARE:
2675  !                              1.  ALL ONES DATA
2676  !                              2.  ALL ZEROES DATA
2677  !                              3.  THIRTEEN RANDOM DATA PATTERNS
2678  !
2679  !                          THIS ROUTINE WILL SET THE FAULT FLAG
2680  !                          AND FAILING DATA PATTERN FLAG FOR EACH
2681  !                          MOS RAM FOUND TO BE BAD.
2682  !
2683  !  FORMAL PARAMETERS:
2684  !                  FAIL_CHIP:
2685  !                          POINTS TO THE FAILING CHIPS TABLE POSITION
2686  !                  PAT_SEL:
2687  !                          POINTS TO THE CURRENT FAILING PATTERN FOR THIS
2688  !                          CHIP
2689  !--
2690
2691      CHIP_TBL [.FAIL_CHIP, FAULT] = SET_FLG;      !SET THIS CHIPS FAULT FLAG
2692
2693      case .PAT_SEL from 0 to 14 of                !SELECT THE CHIPS FAILING DATA PATTERN
2694          set
2695
2696          [0] :
2697              CHIP_TBL [.FAIL_CHIP, PAT_0] = SET_FLG;      !SET ZEROES DATA FLAG
2698
2699          [1] :
2700              CHIP_TBL [.FAIL_CHIP, PAT_1] = SET_FLG;      !SET ONES FLAG
2701
2702          [2] :
2703              CHIP_TBL [.FAIL_CHIP, RAN_1] = SET_FLG;      !SET RANDOM DATA 1 FLAG
2704
2705          [3] :
2706              CHIP_TBL [.FAIL_CHIP, RAN_2] = SET_FLG;      !SET RANDOM DATA 2 FLAG
2707
2708          [4] :
2709              CHIP_TBL [.FAIL_CHIP, RAN_3] = SET_FLG;      !SET RANDOM DATA 3 FLAG
2710
2711          [5] :
2712              CHIP_TBL [.FAIL_CHIP, RAN_4] = SET_FLG;      !SET RANDOM DATA 4 FLAG
2713
2714          [6] :
2715              CHIP_TBL [.FAIL_CHIP, RAN_5] = SET_FLG;      .SET RANDOM DATA 5 FLAG
2716
```

```
;        2717            [7] :
;        2718                CHIP_TBL [.FAIL_CHIP, RAN_6] = SET_FLG;     .SET RANDOM DATA 6 FLAG
;        2719
;        2720            [8] :
;        2721                CHIP_TBL [.FAIL_CHIP, RAN_7] = SET_FLG;     !SET RANDOM DATA 7 FLAG
;        2722
;        2723            [9] :
;        2724                CHIP_TBL [.FAIL_CHIP, RAN_8] = SET_FLG;     !SET RANDOM DATA 8 FLAG
;        2725
;        2726            [10] :
;        2727                CHIP_TBL [.FAIL_CHIP, RAN_9] = SET_FLG;     .SET RANDOM DATA 9 FLAG
;        2728
;        2729            [11] :
;        2730                CHIP_TBL [.FAIL_CHIP, RAN_10] = SET_FLG;    !SET RANDOM DATA 10 FLAG
;        2731
;        2732            [12] :
;        2733                CHIP_TBL [.FAIL_CHIP, RAN_11] = SET_FLG;    !SET RANDOM DATA 11 FLAG
;        2734
;        2735            [13] :
;        2736                CHIP_TBL [.FAIL_CHIP, RAN_12] = SET_FLG;    .SET RANDOM DATA 12 FLAG
;        2737
;        2738            [14] :
;        2739                CHIP_TBL [.FAIL_CHIP, RAN_13] = SET_FLG;    !SET RANDOM DATA 13 FLAG
;        2740            tes;
;        2741
;        2742        end;


                                        .SBTTL  L.CHIP.TBL ROUTINE DECLARATIONS
005040  010146                    L.CHIP.TBL:
                                        MOV     R1,-(SP)                    ;                       2665
005042  016600  000006                  MOV     6(SP),R0                    ; FAIL.CHIP,*           2691
005046  006300                          ASL     R0
005050  062700  026626'                  ADD     #CHIP.TBL,R0
005054  052710  100000                  BIS     #100000,(R0)
005060  016601  000004                  MOV     4(SP),R1                    ; PAT.SEL,*             2693
005064  006301                          ASL     R1
005066  066107  005072'                  ADD     1$(R1),PC
005072  000036              1$:         .WORD   2$-1$
005074  000044                          .WORD   3$-1$
005076  000052                          .WORD   4$-1$
005100  000060                          .WORD   5$-1$
005102  000066                          .WORD   6$-1$
005104  000074                          .WORD   7$-1$
005106  000102                          .WORD   8$-1$
005110  000110                          .WORD   9$-1$
005112  000116                          .WORD   10$-1$
005114  000124                          .WORD   11$-1$
005116  000132                          .WORD   12$-1$
005120  000140                          .WORD   13$-1$
005122  000146                          .WORD   14$-1$
005124  000154                          .WORD   15$-1$
```

```
005126  000162                          .WORD   16$-1$                                  ;2697
005130  052710  040000        2$:       BIS     #40000,(R0)                             ;2693
005134  000451                          BR      17$                                     ;2700
005136  052710  020000        3$:       BIS     #20000,(R0)                             ;2693
005142  000446                          BR      17$                                     ;2703
005144  052710  010000        4$:       BIS     #10000,(R0)                             ;2693
005150  000443                          BR      17$                                     ;2706
005152  052710  004000        5$:       BIS     #4000,(R0)                              ;2693
005156  000440                          BR      17$                                     ;2709
005160  052710  002000        6$:       BIS     #2000,(R0)                              ;2693
005164  000435                          BR      17$                                     ;2712
005166  052710  001000        7$:       BIS     #1000,(R0)                              ;2693
005172  000432                          BR      17$                                     ;2715
005174  052710  000400        8$:       BIS     #400,(R0)                               ;2693
005200  000427                          BR      17$                                     ;2718
005202  152710  000200        9$:       BISB    #200,(R0)                               ;2693
005206  000424                          BR      17$                                     ;2721
005210  152710  000100        10$:      BISB    #100,(R0)                               ;2693
005214  000421                          BR      17$                                     ;2724
005216  152710  000040        11$:      BISB    #40,(R0)                                ;2693
005222  000416                          BR      17$                                     ;2727
005224  152710  000020        12$:      BISB    #20,(R0)                                ;2693
005230  000413                          BR      17$                                     ;2730
005232  152710  000010        13$:      BISB    #10,(R0)                                ;2693
005236  000410                          BR      17$                                     ;2733
005240  152710  000004        14$:      BISB    #4,(R0)                                 ;2693
005244  000405                          BR      17$                                     ;2736
005246  152710  000002        15$:      BISB    #2,(R0)                                 ;2693
005252  000402                          BR      17$                                     ;2739
005254  152710  000001        16$:      BISB    #1,(R0)                                 ;2665
005260  012601                17$:      MOV     (SP)+,R1                                ;
005262  000207                          RTS     PC

; Routine Size:  74 words
; Maximum stack depth per invocation:  2 words
```

```
;        2743  routine L_FAILING_CHIP (TEMP_ARR$BNK_SEL, BAD_CHIP, LST_PAT) =
;        2744      begin
;        2745
;        2746  !++
;        2747  !  FUNCTIONAL DESCRIPTION: LOAD FAILING CHIP
;        2748  !
;        2749  !                          THE PROGRAM WILL FIRST SEARCH THRU THE
;        2750  !                          SELECTED ARRAYS AND BANKS AND FIND ALL
;        2751  !                          NEWLY FAILING MOS RAMS.
;        2752  !
;        2753  !                          ONCE THEY ARE FOUND THEN THEY ARE LOADED
;        2754  !                          WITH THE DATA PATTERN WHICH FAILED THE
;        2755  !                          MASS BUS WRITE CHECKS AND THE FAILING
;        2756  !                          ROW COLUMN PAIR ADDRESS IS SEARCHED FOR.
;        2757  !
;        2758  !                          THIS ROUTINE WILL LOAD A FAILING CHIP
;        2759  !                          WITH ONE OF THE DATA PATTERNS WHICH FAILED
;        2760  !                          DURING THE MASS BUS TRANSFERS.
;        2761  !
;        2762  !                          THE CHIP_TBL STORES WHICH CHIPS HAVE FAILED
;        2763  !                          AND THIER FAILING PATTERNS.
;        2764  !
;        2765  !  FORMAL PARAMETERS:
;        2766  !                  TEMP_ARR$BNK_SEL:
;        2767  !                          THIS ARGUMENT STORES THE SELECTED ARRAY AND
;        2768  !                          BANK SELCECT NUMBER.
;        2769  !                  BAD_CHIP:
;        2770  !                          THIS POINTS TO A FAILING CHIP WHICH IS
;        2771  !                          CURRENTLY BEING PM'ED.
;        2772  !                  LST_PAT:
;        2773  !                          THIS IS A POINTER INTO THE FAILING PATTERN
;        2774  !                          PORTION OF THE TABLE.  THE FAILING PATTERN
;        2775  !                          SEARCH STARTS AT THE LAST FOUND FAIL PATTERN
;        2776  !                          AND WILL RETURN WITH FIRST FAILING PATTERN
;        2777  !                          FOUND.
;        2778  !--
;        2779
;        2780      incr PAT_SEL from .LST_PAT to 14 do          !SEARCH TH CHIP TABLE FOR A FAILING PATTERN
;        2781          begin
;        2782
;        2783          case .PAT_SEL from 0 to 14 of            !SELECT A PATTERN
;        2784              set
;        2785
;        2786              [0] :
;        2787
;        2788                  if .CHIP_TBL [.BAD_CHIP, PAT_0] !DID THIS PATTERN FAIL
;        2789                  then
;        2790                      begin
;        2791                      DM_1_0_LOAD (.TEMP_ARR$BNK_SEL, ZEROES);     !LOAD CHIP WITH ZEROES
;        2792                      CHIP_TBL [.BAD_CHIP, PAT_0] = CLR_FLG;       !CLEAR THIS PAT FLAG
;        2793                      FLG_REG [F_RAND_DATA] = CLR_FLG;     !CLEAR THE RANDOM FLAG
;        2794                      return .PAT_SEL;                 !RETURN WHICH PATTERN WAS SELECTED
```

```
2795                        end;
2796
2797                [1] :
2798
2799                    if .CHIP_TBL [.BAD_CHIP, PAT_1] .DID THIS PATTERN FAIL
2800                    then
2801                        begin
2802                        DM_1_0_LOAD (.TEMP_ARR$BNK_SEL, ONES);      !LOAD CHIP WITH ONES DATA
2803                        CHIP_TBL [.BAD_CHIP, PAT_1] = CLR_FLG;      !CLEAR THIS PAT FLAG
2804                        FLG_REG [F_RAND_DATA] - CLR_FLG;    !CLEAR THE RANDOM FLAG
2805                        return .PAT_SEL;                .RETURN WHICH PATTERN WAS SELECTED
2806                        end;
2807
2808                [2] :
2809
2810                    if not .CHIP_TBL [.BAD_CHIP, RAN_1]     !DID THIS PATTERN FAIL
2811                    then
2812                        begin
2813                        GEN_RANDOM_DATA ();         !GEN THE RANDOM DATA BUT DONT LOAD IT
2814                        end
2815                    else
2816                        begin
2817                        DM_RAND_LOAD (.TEMP_ARR$BNK_SEL);   !LOAD THE CHIP WITH RANDOM DATA
2818                        CHIP_TBL [.BAD_CHIP, RAN_1] = CLR_FLG;      !CLEAR THIS PAT FLAG
2819                        FLG_REG [F_RAND_DATA] = SET_FLG;    !SET THE RANDOM FLAG
2820                        return .PAT_SEL;                .RETURN WHICH PATTERN WAS SELECTED
2821                        end;
2822
2823                [3] :
2824
2825                    if not .CHIP_TBL [.BAD_CHIP, RAN_2]     !DID THIS PATTERN FAIL
2826                    then
2827                        begin
2828                        GEN_RANDOM_DATA ();         .GEN THE RANDOM DATA BUT DONT LOAD IT
2829                        end
2830                    else
2831                        begin
2832                        DM_RAND_LOAD (.TEMP_ARR$BNK_SEL);   !LOAD CHIP WITH RANDOM DATA
2833                        CHIP_TBL [.BAD_CHIP, RAN_2] = CLR_FLG;      !CLEAR THIS PAT FLAG
2834                        FLG_REG [F_RAND_DATA] = SET_FLG;    !SET THE RANDOM FLAG
2835                        return .PAT_SEL;                !RETURN WHICH PATTERN WAS SECECTED
2836                        end;
2837
2838                [4] :
2839
2840                    if not .CHIP_TBL [.BAD_CHIP, RAN_3]     !DID THIS PATTERN FAIL
2841                    then
2842                        begin
2843                        GEN_RANDOM_DATA ();         .GEN THE RANDOM DATA BUT DONT LOAD IT
2844                        end
2845                    else
2846                        begin
```

```
2847                    DM_RAND_LOAD (.TEMP_ARR$BNK_SEL);   !LOAD CHIP WITH RANDOM DATA
2848                    CHIP_TBL [.BAD_CHIP, RAN_3] = CLR_FLG;      !CLEAR THIS PAT FLAG
2849                    FLG_REG [F_RAND_DATA] = SET_FLG;    !SET THE RANDOM DATA FLAG
2850                    return .PAT_SEL;                    !RETURN WHICH PATTERN WAS SECECTED
2851                    end;
2852
2853                [5] :
2854
2855                    if not .CHIP_TBL [.BAD_CHIP, RAN_4]     !DID THIS PATTERN FAIL
2856                    then
2857                        begin
2858                        GEN_RANDOM_DATA ();             .GEN THE RANDOM DATA BUT DONT LOAD IT
2859                        end
2860                    else
2861                        begin
2862                        DM_RAND_LOAD (.TEMP_ARR$BNK_SEL);   !LOAD CHIP WITH RANDOM DATA
2863                        CHIP_TBL [.BAD_CHIP, RAN_4] = CLR_FLG;      !CLEAR THE PAT FLAG
2864                        FLG_REG [F_RAND_DATA] = SET_FLG;    .SET THE RANDOM DATA FLAG
2865                        return .PAT_SEL;                    .RETURN WHICH PATTERN WAS SELECTED
2866                        end;
2867
2868                [6] :
2869
2870                    if not .CHIP_TBL [.BAD_CHIP, RAN_5]     .DID THIS PATTERN FALIL
2871                    then
2872                        begin
2873                        GEN_RANDOM_DATA ();
2874                        end
2875                    else
2876                        begin
2877                        DM_RAND_LOAD (.TEMP_ARR$BNK_SEL);   !LOAD CHIP WITH RANDOM DATA
2878                        CHIP_TBL [.BAD_CHIP, RAN_5] = CLR_FLG;      .CLEAR THE PAT FLAG
2879                        FLG_REG [F_RAND_DATA] = SET_FLG;    !SET THE RANDOM DATA FLAG
2880                        return .PAT_SEL;                    !RETURN WHICH PATTERN WAS SELECTED
2881                        end;
2882
2883                [7] :
2884
2885                    if not .CHIP_TBL [.BAD_CHIP, RAN_6]     .DID THIS PATTERN FAIL
2886                    then
2887                        begin
2888                        GEN_RANDOM_DATA ();             !GEN THE RANDOM DATA BUT DONT LOAD IT
2889                        end
2890                    else
2891                        begin
2892                        DM_RAND_LOAD (.TEMP_ARR$BNK_SEL);   !LOAD CHIP WITH RANDOM DATA
2893                        CHIP_TBL [.BAD_CHIP, RAN_6] = CLR_FLG;      .CLEAR THE PAT FLAG
2894                        FLG_REG [F_RAND_DATA] = SET_FLG;    .SET THE RANDOM DATA FLAG
2895                        return .PAT_SEL;                    !RETURN WHICH PATTERN WAS SELECTED
2896                        end;
2897
2898                [8] :
```

```
2899
2900                    if not .CHIP_TBL [.BAD_CHIP, RAN_7]     .DID THIS PATTERN FAIL
2901                    then
2902                        begin
2903                        GEN_RANDOM_DATA ();             .GEN THE RANDOM DATA BUT DONT LOAD IT
2904                        end
2905                    else
2906                        begin
2907                        DM_RAND_LOAD (.TEMP_ARR$BNK_SEL);   !LOAD CHIP WITH RANDOM DATA
2908                        CHIP_TBL [.BAD_CHIP, RAN_7] = CLR_FLG;      !CLEAR THE PAT FLAG
2909                        FLG_REG [F_RAND_DATA] = SET_FLG;    !SET THE RANDOM DATA FLAG
2910                        return .PAT_SEL;                .RETURN WHICH PATTERN WAS SELECTED
2911                        end;
2912
2913                [9] :
2914
2915                    if not .CHIP_TBL [.BAD_CHIP, RAN_8]     !DID THIS PATTERN FAIL
2916                    then
2917                        begin
2918                        GEN_RANDOM_DATA ();             .GEN THE RANDOM DATA BUT DONT LOAD IT
2919                        end
2920                    else
2921                        begin
2922                        DM_RAND_LOAD (.TEMP_ARR$BNK_SEL);   .LOAD CHIP WITH RANDOM DATA
2923                        CHIP_TBL [.BAD_CHIP, RAN_8] = CLR_FLG;      !CLEAR THE PAT FLAG
2924                        FLG_REG [F_RAND_DATA] = SET_FLG;    !SET THE RANDOM DATA FLAG
2925                        return .PAT_SEL;                !RETURN WHICH PATTERN WAS SELECTED
2926                        end;
2927
2928                [10] :
2929
2930                    if not .CHIP_TBL [.BAD_CHIP, RAN_9]     .DID THIS PATTERN FAIL
2931                    then
2932                        begin
2933                        GEN_RANDOM_DATA ();             .GEN THE RANDOM DATA BUT DONT LOAD IT
2934                        end
2935                    else
2936                        begin
2937                        DM_RAND_LOAD (.TEMP_ARR$BNK_SEL);   .LOAD CHIP WITH RANDOM DATA
2938                        CHIP_TBL [.BAD_CHIP, RAN_9] = CLR_FLG;      .CLEAR THE PAT FLAG
2939                        FLG_REG [F_RAND_DATA] = SET_FLG;    !SET THE RANDOM DATA FLAG
2940                        return .PAT_SEL;                !RETURN WHICH PATTERN WAS SELECTED
2941                        end;
2942
2943                [11] :
2944
2945                    if not .CHIP_TBL [.BAD_CHIP, RAN_10]    !DID THIS PATTERN FAIL
2946                    then
2947                        begin
2948                        GEN_RANDOM_DATA ();             .GEN THE RANDOM DATA BUT DONT LOAD IT
2949                        end
2950                    else
```

```
2951                    begin
2952                    DM_RAND_LOAD (.TEMP_ARR$BNK_SEL);   .LOAD CHIP WITH RANDOM DATA
2953                    CHIP_TBL [.BAD_CHIP, RAN_10] = CLR_FLG;     !CLEAR THE PAT FLAG
2954                    FLG_REG [F_RAND_DATA] = SET_FLG;    !SET THE RANDOM DATA FLAG
2955                    return .PAT_SEL;                .RETURN WHICH PATTERN WAS SELECTED
2956                    end;
2957
2958                [12] :
2959
2960                    if not .CHIP_TBL [.BAD_CHIP, RAN_11]    !DID THIS PATTERN FAIL
2961                    then
2962                        begin
2963                        GEN_RANDOM_DATA ();             .GEN THE RANDOM DATA BUT DONT LOAD IT
2964                        end
2965                    else
2966                        begin
2967                        DM_RAND_LOAD (.TEMP_ARR$BNK_SEL);   !LOAD CHIP WITH RANDOM DATA
2968                        CHIP_TBL [.BAD_CHIP, RAN_11] = CLR_FLG;     !CLEAR THE PAT FLAG
2969                        FLG_REG [F_RAND_DATA] = SET_FLG;    !SET THE RANDOM DATA FLAG
2970                        return .PAT_SEL;                !RETURN WHICH PATTERN WAS SELECTED
2971                        end;
2972
2973                [13] :
2974
2975                    if not .CHIP_TBL [.BAD_CHIP, RAN_12]    !DID THIS PATTERN FAIL
2976                    then
2977                        begin
2978                        GEN_RANDOM_DATA ();             .GEN THE RANDOM DATA BUT DONT LOAD IT
2979                        end
2980                    else
2981                        begin
2982                        DM_RAND_LOAD (.TEMP_ARR$BNK_SEL);   !LOAD CHIP WITH RANDOM DATA
2983                        CHIP_TBL [.BAD_CHIP, RAN_12] = CLR_FLG;     !CLEAR THE PAT FLAG
2984                        FLG_REG [F_RAND_DATA] = SET_FLG;    !SET THE RANDOM DATA FLAG
2985                        return .PAT_SEL;                !RETURN WHICH PATTERN WAS SELECTED
2986                        end;
2987
2988                [14] :
2989
2990                    if .CHIP_TBL [.BAD_CHIP, RAN_13]        !DID THIS PATTERN FAIL
2991                    then
2992                        begin
2993                        DM_RAND_LOAD (.TEMP_ARR$BNK_SEL);   !LOAD CHIP WITH RANDOM DATA
2994                        CHIP_TBL [.BAD_CHIP, RAN_13] = CLR_FLG;     !CLEAR THE PAT FLAG
2995                        FLG_REG [F_RAND_DATA] = SET_FLG;    !SET THE RANDOM DATA FLAG
2996                        return .PAT_SEL;                .RETURN WHICH PATTERN WAS SELECTED
2997                        end;
2998
2999                    tes;
3000
3001            end;
3002
```

```
;         3003      return -1;
;         3004      end;

                                .RETURN NEGITIVE ONE IF ALL PATTERN FLAGS ARE CLEARED

                                .SBTTL  L.FAILING.CHIP ROUTINE DECLARATIONS
005264  004167  000000G   L.FAILING.CHIP:
                          JSR     R1,$SAVE2                       ;                       2743
005270  016600  000012              MOV     12(SP),R0               ; BAD.CHIP,*            2788
005274  006300                      ASL     R0
005276  012702  026626'              MOV     #CHIP.TBL,R2
005302  060002                      ADD     R0,R2
005304  016601  000010              MOV     10(SP),R1               ; LST.PAT,PAT.SEL       2780
005310  005301                      DEC     R1                      ; PAT.SEL
005312  000167  000560      1$:     JMP     24$
005316  010100      2$:             MOV     R1,R0                   ; PAT.SEL,*             2783
005320  006300                      ASL     R0
005322  066007  005326'              ADD     3$(R0),PC
005326  000036      3$:             .WORD   4$-3$
005330  000064                      .WORD   5$-3$
005332  000126                      .WORD   7$-3$
005334  000152                      .WORD   8$-3$
005336  000176                      .WORD   9$-3$
005340  000222                      .WORD   10$-3$
005342  000246                      .WORD   11$-3$
005344  000272                      .WORD   12$-3$
005346  000314                      .WORD   13$-3$
005350  000340                      .WORD   14$-3$
005352  000364                      .WORD   15$-3$
005354  000410                      .WORD   16$-3$
005356  000434                      .WORD   17$-3$
005360  000460                      .WORD   18$-3$
005362  000512                      .WORD   21$-3$
005364  032712  040000      4$:     BIT     #40000,(R2)             ;                       2788
005370  001750                      BEQ     1$
005372  016646  000014              MOV     14(SP),-(SP)            ; TEMP.ARR$BNK.SE,*     2791
005376  005046                      CLR     -(SP)
005400  004767  174460              JSR     PC,DM.1.0.LOAD
005404  042712  040000              BIC     #40000,(R2)             ;                       2792
005410  000413                      BR      6$                      ;                       2793
005412  032712  020000      5$:     BIT     #20000,(R2)             ;                       2799
005416  001735                      BEQ     1$
005420  016646  000014              MOV     14(SP),-(SP)            ; TEMP.ARR$BNK.SE,*     2802
005424  012746  177777              MOV     #-1,-(SP)
005430  004767  174430              JSR     PC,DM.1.0.LOAD
005434  042712  020000              BIC     #20000,(R2)             ;                       2803
005440  142767  000040  037020'  6$: BICB    #40,FLG.REG             ;                       2804
005446  022626                      CMP     (SP)+,(SP)+             ;                       2799
005450  000167  000416              JMP     23$                     ;                       2801
005454  032712  010000      7$:     BIT     #10000,(R2)             ;                       2810
005460  001555                      BEQ     19$                     ;                       2813
005462  016646  000014              MOV     14(SP),-(SP)            ; TEMP.ARR$BNK.SE,*     2817
005466  004767  173776              JSR     PC,DM.RAND.LOAD
```

```
005472  042712  010000                  BIC     #10000,(R2)             ;                               2818
005476  000571                          BR      22$                     ;                               2819
005500  032712  004000          8$:     BIT     #4000,(R2)              ;                               2825
005504  001543                          BEQ     19$                     ;                               2828
005506  016646  000014                  MOV     14(SP),-(SP)            ; TEMP.ARR$BNK.SE,*             2832
005512  004767  173752                  JSR     PC,DM.RAND.LOAD
005516  042712  004000                  BIC     #4000,(R2)              ;                               2833
005522  000557                          BR      22$                     ;                               2834
005524  032712  002000          9$:     BIT     #2000,(R2)              ;                               2840
005530  001531                          BEQ     19$                     ;                               2843
005532  016646  000014                  MOV     14(SP),-(SP)            ; TEMP.ARR$BNK.SE,*             2847
005536  004767  173726                  JSR     PC,DM.RAND.LOAD
005542  042712  002000                  BIC     #2000,(R2)              ;                               2848
005546  000545                          BR      22$                     ;                               2849
005550  032712  001000          10$:    BIT     #1000,(R2)              ;                               2855
005554  001517                          BEQ     19$                     ;                               2858
005556  016646  000014                  MOV     14(SP),-(SP)            ; TEMP.ARR$BNK.SE,*             2862
005562  004767  173702                  JSR     PC,DM.RAND.LOAD
005566  042712  001000                  BIC     #1000,(R2)              ;                               2863
005572  000533                          BR      22$                     ;                               2864
005574  032712  000400          11$:    BIT     #400,(R2)               ;                               2870
005600  001505                          BEQ     19$                     ;                               2873
005602  016646  000014                  MOV     14(SP),-(SP)            ; TEMP.ARR$BNK.SE,*             2877
005606  004767  173656                  JSR     PC,DM.RAND.LOAD
005612  042712  000400                  BIC     #400,(R2)               ;                               2878
005616  000521                          BR      22$                     ;                               2879
005620  105712          12$:    TSTB    (R2)                    ;                               2885
005622  100074                          BPL     19$                     ;                               2888
005624  016646  000014                  MOV     14(SP),-(SP)            ; TEMP.ARR$BNK.SE,*             2892
005630  004767  173634                  JSR     PC,DM.RAND.LOAD
005634  142712  000200                  BICB    #200,(R2)               ;                               2893
005640  000510                          BR      22$                     ;                               2894
005642  132712  000100          13$:    BITB    #100,(R2)               ;                               2900
005646  001462                          BEQ     19$                     ;                               2903
005650  016646  000014                  MOV     14(SP),-(SP)            ; TEMP.ARR$BNK.SE,*             2907
005654  004767  173610                  JSR     PC,DM.RAND.LOAD
005660  142712  000100                  BICB    #100,(R2)               ;                               2908
005664  000476                          BR      22$                     ;                               2909
005666  132712  000040          14$:    BITB    #40,(R2)                ;                               2915
005672  001450                          BEQ     19$                     ;                               2918
005674  016646  000014                  MOV     14(SP),-(SP)            ; TEMP.ARR$BNK.SE,*             2922
005700  004767  173564                  JSR     PC,DM.RAND.LOAD
005704  142712  000040                  BICB    #40,(R2)                ;                               2923
005710  000464                          BR      22$                     ;                               2924
005712  132712  000020          15$:    BITB    #20,(R2)                ;                               2930
005716  001436                          BEQ     19$                     ;                               2933
005720  016646  000014                  MOV     14(SP),-(SP)            ; TEMP.ARR$BNK.SE,*             2937
005724  004767  173540                  JSR     PC,DM.RAND.LOAD
005730  142712  000020                  BICB    #20,(R2)                ;                               2938
005734  000452                          BR      22$                     ;                               2939
005736  132712  000010          16$:    BITB    #10,(R2)                ;                               2945
005742  001424                          BEQ     19$                     ;                               2948
```

```
005744  016646  000014                  MOV     14(SP),-(SP)                    ; TEMP.ARR$BNK.SE,*             2952
005750  004767  173514                  JSR     PC,DM.RAND.LOAD
005754  142712  000010                  BICB    #10,(R2)                        ;                               2953
005760  000440                          BR      22$                             ;                               2954
005762  132712  000004          17$:    BITB    #4,(R2)                         ;                               2960
005766  001412                          BEQ     19$                             ;                               2963
005770  016646  000014                  MOV     14(SP),-(SP)                    ; TEMP.ARR$BNK.SE,*             2967
005774  004767  173470                  JSR     PC,DM.RAND.LOAD
006000  142712  000004                  BICB    #4,(R2)                         ;                               2968
006004  000426                          BR      22$                             ;                               2969
006006  132712  000002          18$:    BITB    #2,(R2)                         ;                               2975
006012  001003                          BNE     20$
006014  004767  172550          19$:    JSR     PC,GEN.RANDOM.DATA              ;                               2978
006020  000426                          BR      24$                             ;                               2975
006022  016646  000014          20$:    MOV     14(SP),-(SP)                    ; TEMP.ARR$BNK.SE,*             2982
006026  004767  173436                  JSR     PC,DM.RAND.LOAD
006032  142712  000002                  BICB    #2,(R2)                         ;                               2983
006036  000411                          BR      22$                             ;                               2984
006040  132712  000001          21$:    BITB    #1,(R2)                         ;                               2990
006044  001414                          BEQ     24$
006046  016646  000014                  MOV     14(SP),-(SP)                    ; TEMP.ARR$BNK.SE,*             2993
006052  004767  173412                  JSR     PC,DM.RAND.LOAD
006056  142712  000001                  BICB    #1,(R2)                         ;                               2994
006062  152767  000040  037020' 22$:    BISB    #40,FLG.REG                     ;                               2995
006070  005726                          TST     (SP)+                           ;                               2990
006072  010100          23$:    MOV     R1,R0                           ; PAT.SEL,*                     2992
006074  000207                          RTS     PC
006076  005201          24$:    INC     R1                              ; PAT.SEL                       2780
006100  020127  000016                  CMP     R1,#16                          ; PAT.SEL,*
006104  003002                          BGT     25$
006106  000167  177204                  JMP     2$
006112  012700  177777          25$:    MOV     #-1,R0                          ;                               2744
006116  000207                          RTS     PC                              ;                               2743

; Routine Size:  206 words
; Maximum stack depth per invocation:  5 words
```

```
3005  routine L_ERROR_MAP (TEMP_ARR$BNK_SEL, BAD_CHIP) : novalue =
3006      begin
3007
3008  !++
3009  !  FUNCTIONAL DESCRIPTION: LOAD THE ERROR MAP
3010  !
3011  !                          THE ERROR MAP STORES BY SECTOR ALL THE
3012  !                          NEWLY FAILING ROWS.  THE FAILING COLUMN
3013  !                          ADDRESS CAN BE CALCULATED BY ADDING THE
3014  !                          ROW NUMBER TO THE SECTOR NUMBER.
3015  !
3016  !                          THIS ROUTINE SEARCHES THE TESTED BANKS
3017  !                          LOOKING FOR BAD ROWS AND ALL SECTORS.
3018  !                          IF A BAD ROW IS DETECTED THEN THAT ROWS
3019  !                          BIT POSITION IN THE ERROR MAP AT THIS
3020  !                          SECTOR IS SET INDICATING THAT A BAD ROW
3021  !                          WAS FOUND AT THIS SECTOR.
3022  !
3023  !  FORMAL PARAMETERS:
3024  !                  TEMP_ARR$BNK_SEL:
3025  !                          THIS ARGUMENT STORES THE ARRAY AND BANK SELECT NUMBER
3026  !                  BAD_CHIP:
3027  !                          THIS ARGUMENT POINTS TO A FAILING WHICH IS PRESENTLY
3028  !                          BEING TESTED.  DIVIDING THIS ARG BY 4 GIVES THE NIBBLE
3029  !                          POSITION IN THE ML-11 WORD AND THIS ARG MOD 4 GIVES THE
3030  !                          CHIPS BIT POSITION WITHIN ITS NIBBLE.
3031  !--
3032
3033      local
3034          WRD_INDEX,                              !POINTS TO A WORD IN THE ERROR MAP
3035          BIT_INDEX,                              !POINTS TO A BIT IN THE ERROR MAP
3036          RAND_INDEX,                             !SECTOR RANDOM STARTING POSITION
3037          DIAG_REG_SEL,                           !SELECT DATA DIAG REG WHERE FAILING CHIP RESIDES
3038          BIT_SEL,                                !SELECT THE FAILING CHIP POSITION
3039          OFFSET;                         !FIRST SELECTS DATA DIAG REG; SECOND SELECTS RANDOM DATA OFFSET
3040
3041      !
3042      ! BAD_CHIP / 16 AND BAD_CHIP MOD 16 WILL
3043      ! POINT TO WHICH DIAGNOSTIC REGISTER THE BAD
3044      ! CHIP CAME FROM AND THE BITS
3045      ! POSITION WITHIN THE REGISTER.
3046      !
3047      OFFSET = .BAD_CHIP/16;                      !CALCULATE WHICH DATA DIAG REG TO READ
3048      BIT_SEL = .BAD_CHIP mod 16;                 !CALCULATE FAILING CHIPS BIT POSITION
3049      !
3050      !
3051      ! OFFSET VALUES 2 = MLE2 - 14;
3052      !               0 = MLD1 - 15;
3053      !               1 = MLD2 = 16;
3054      !
3055      !
3056      if .OFFSET eql 2                            !IS BAD CHIP DIVIDED BY 16 EQUAL TO 2
```

```
3057        then
3058            begin
3059            DIAG_REG_SEL = 14;                  !SELECT DATA DIAG REG MLE2
3060            BIT_SEL = .BIT_SEL + 8;             !THIS REGISTERS DATA BITS STARTS IN HIGH BYTE
3061            end
3062        else
3063            DIAG_REG_SEL = .OFFSET + 15;        !0 + 15 = MLD1; 1 + 15 = MLD2
3064
3065        if .FLG_REG [F_RAND_DATA] then OFFSET = 2 else OFFSET = ZERO;       .OFFSET = 2 IF RANDOM DATA
3066
3067        DM_RD_TRANSFER (SIZE = -256, DST = RD_BUF, SRC = .TEMP_ARR$BNK_SEL);
3068        RAND_INDEX = ZERO;                      .THE FIRST SECTORS DATA STARTS AT WORD ZERO
3069
3070        incr SEC_NUM from 0 to .BNK_NUM_SEC do  !READ ALL SECTORS IN FAILING CHIP
3071            begin
3072            BREAK;
3073
3074            incr ROW_NUM from 0 to '27 do       !READ ALL WORDS IN FAILING CHIP
3075                begin
3076                DAT_CLK;                        !CLOCK DATA INTO THE DATA DIAG REGISTERS
3077
3078                if .WRT_BUF [.RAND_INDEX + .ROW_NUM, .BIT_SEL, ONE, ZERO] neq .ML_ADDR [.DIAG_REG_SEL, .BIT_SEL,
3079                        ONE, ZERO]              !IS WRITE BUF DATA EQL DATA DIAG REGISTERS DATA
3080                then
3081                    begin
3082                    WRD_INDEX = .ROW_NUM/16;    !CALCULATE ERROR MAP WORD OF FAILING ROW
3083                    BIT_INDEX = .ROW_NUM mod 16;    !CAOCULATE ERROR MAP BIT POS OF FAILING ROW
3084                    ERROR_MAP [.SEC_NUM, .WRD_INDEX, .BIT_INDEX, ONE, ZERO] = SET_FLG;
3085                    FLG_REG [F_ERR_MAP_ENTERED] = SET_FLG;  .FLAG THAT ERROR MAP WAS ENTERED
3086                    end;
3087
3088                end;
3089
3090            RAND_INDEX   .RAND_INDEX + .OFFSET;     .THE NEXT SECTOR STARTS 2 WORDS DEEPER IF RANDOM DATA
3091            end;
3092
3093        CLR_MBUS;                               !CLEAR THE SINGLE STEP DMA MODE
3094        end;


                                .SBTTL  L.ERROR.MAP ROUTINE DECLARATIONS
006120  004167  000000G    L.ERROR.MAP:
                            JSR     R1,$SAVE5                   ;                       3005
006124  162706  000022              SUB     #22,SP
006130  016646  000040              MOV     40(SP),-(SP)        ; BAD.CHIP,*            3047
006134  012746  000020              MOV     #20,-(SP)
006140  004767  000000G             JSR     PC,BL$DIV
006144  010004                      MOV     R0,R4               ; *,OFFSET
006146  016616  000044              MOV     44(SP),(SP)         ; BAD.CHIP,*            3048
006152  012746  000020              MOV     #20,-(SP)
006156  004767  000000G             JSR     PC,BL$MOD
006162  010066  000006              MOV     R0,6(SP)            ; *,BIT.SEL
```

```
006166  020427  000002                    CMP     R4,#2                           ; OFFSET,*                      3056
006172  001006                            BNE     1$
006174  012702  000016                    MOV     #16,R2                          ; *,DIAG.REG.SEL                3059
006200  062766  000010  000006            ADD     #10,6(SP)                       ; *,BIT.SEL                     3060
006206  000403                            BR      2$                              ;                               3056
006210  010402                    1$:     MOV     R4,R2                           ; OFFSET,DIAG.REG.SEL           3063
006212  062702  000017                    ADD     #17,R2                          ; *,DIAG.REG.SEL
006216  032767  000040  037020'    2$:     BIT     #40,FLG.REG                     ;                               3065
006224  001403                            BEQ     3$
006226  012704  000002                    MOV     #2,R4                           ; *,OFFSET
006232  000401                            BR      4$
006234  005004                    3$:     CLR     R4                              ; OFFSET
006236  012716  177400            4$:     MOV     #-400,(SP)                      ;                               3067
006242  011667  037024'                    MOV     (SP),SIZE
006246  012746  005626'                    MOV     #RD.BUF,-(SP)
006252  011667  037026'                    MOV     (SP),DST
006256  016646  000052                    MOV     52(SP),-(SP)                    ; TEMP.ARR$BNK.SE,*
006262  011667  037030'                    MOV     (SP),SRC
006266  004767  172366                    JSR     PC,DM.RD.TRANSFER
006272  005066  000014                    CLR     14(SP)                          ; RAND.INDEX                    3068
006276  016766  037014'  000016            MOV     BNK.NUM.SEC,16(SP)              ;                               3070
006304  006302                            ASL     R2                              ;                               3079
006306  005005                            CLR     R5                              ; SEC.NUM                       3070
006310  000536                            BR      8$
006312  104422                    5$:     TRAP    22                              ;                               3071
006314  005001                            CLR     R1                              ; ROW.NUM                       3074
006316  016703  036772'            6$:     MOV     ML.ADDR,R3                      ;                               3075
006322  016366  000024  000032            MOV     24(R3),32(SP)                   ; *,ML.REG
006330  016600  000032                    MOV     32(SP),R0                       ; ML.REG,MLREG
006334  152700  000020                    BISB    #20,R0                          ; *,MLREG
006340  016703  036772'                    MOV     ML.ADDR,R3
006344  010063  000024                    MOV     R0,24(R3)                       ; MLREG,*
006350  010103                            MOV     R1,R3                           ; ROW.NUM,*                     3078
006352  066603  000014                    ADD     14(SP),R3                       ; RAND.INDEX,*
006356  006303                            ASL     R3
006360  062703  001232'                    ADD     #WRT.BUF,R3
006364  010346                            MOV     R3,-(SP)
006366  016646  000014                    MOV     14(SP),-(SP)                    ; BIT.SEL,*
006372  012746  000001                    MOV     #1,-(SP)
006376  005046                            CLR     -(SP)
006400  004767  000000G                   JSR     PC,BL$GT2
006404  062706  000006                    ADD     #6,SP
006410  010003                            MOV     R0,R3
006412  010200                            MOV     R2,R0                           ;                               3079
006414  066700  036772'                    ADD     ML.ADDR,R0
006420  011066  000032                    MOV     (R0),32(SP)                     ; *,ML.REG
006424  012716  000032                    MOV     #32,(SP)
006430  060616                            ADD     SP,(SP)                         ; ML.REG,*
006432  016646  000014                    MOV     14(SP),-(SP)                    ; BIT.SEL,*                     3078
006436  012746  000001                    MOV     #1,-(SP)
006442  005046                            CLR     -(SP)
006444  004767  000000G                   JSR     PC,BL$GT2
```

```
006450  062706  0000'0                          ADD     #10,SP
006454  020300                                   CMP     R3,R0
006456  001444                                   BEQ     7$
006460  010146                                   MOV     R1,-(SP)          ; ROW.NUM,*                    3082
006462  012746  000020                           MOV     #20,-(SP)
006466  004767  000000G                         JSR     PC,BL$DIV
006472  010066  000026                           MOV     R0,26(SP)         ; *,WRD.INDEX
006476  010116                                   MOV     R1,(SP)           ; ROW.NUM,*                    3083
006500  012746  000020                           MOV     #20,-(SP)
006504  004767  000000G                         JSR     PC,BL$MOD
006510  010066  000026                           MOV     R0,26(SP)         ; *,BIT.INDEX
006514  010500                                   MOV     R5,R0             ; SEC.NUM,*                    3084
006516  006300                                   ASL     R0
006520  006300                                   ASL     R0
006522  006300                                   ASL     R0
006524  066600  000030                           ADD     30(SP),R0         ; WRD.INDEX,*
006530  006300                                   ASL     R0
006532  062700  006626'                         ADD     #ERROR.MAP,R0
006536  010016                                   MOV     R0,(SP)
006540  016646  000026                           MOV     26(SP),-(SP)      ; BIT.INDEX,*
006544  012746  000001                           MOV     #1,-(SP)
006550  011646                                   MOV     (SP),-(SP)
006552  004767  000000G                         JSR     PC,BL$PU2
006556  152767  000004  037020'                 BISB    #4,FLG.REG        ;                              3085
006564  062706  000014                           ADD     #14,SP            ;                              3081
006570  005201                          7$:      INC     R1                ; ROW.NUM                      3074
006572  020127  000177                           CMP     R1,#177           ; ROW.NUM,*
006576  003647                                   BLE     6$
006600  060466  000014                           ADD     R4,14(SP)         ; OFFSET,RAND.INDEX            3090
006604  005205                                   INC     R5                ; SEC.NUM                      3070
006606  020566  000016                  8$:      CMP     R5,16(SP)         ; SEC.NUM,*
006612  003637                                   BLE     5$
006614  016701  036772'                         MOV     ML.ADDR,R1        ;                              3091
006620  016166  000010  000026                  MOV     10(R1),26(SP)     ; *,ML.RFG
006626  016600  000026                           MOV     26(SP),R0         ; ML.REG,MLREG
006632  152700  000040                           BISB    #40,R0            ; *,MLREG
006636  016701  036772'                         MOV     ML.ADDR,R1
006642  010061  000010                           MOV     R0,10(R1)         ; MLREG,*
006646  016701  036772'                         MOV     ML.ADDR,R1
006652  016166  000010  000024                  MOV     10(R1),24(SP)     ; *,ML.REG
006660  016600  000024                           MOV     24(SP),R0         ; ML.REG,MLREG
006664  016705  036776'                         MOV     ML.DUT,R5
006670  042705  177770                           BIC     #177770,R5
006674  142700  000007                           BICB    #7,R0             ; *,MLREG
006700  050500                                   BIS     R5,R0             ; *,MLREG
006702  016701  036772'                         MOV     ML.ADDR,R1
006706  010061  000010                           MOV     R0,10(R1)         ; MLREG,*
006712  062706  000034                           ADD     #34,SP            ;                              3005
006716  000207                                   RTS     PC

; Routine Size:  192 words
; Maximum stack depth per invocation:  26 words
```

```
;         3095  routine X_TMP_BLST_TBL (TEMP_ARR$BNK_SEL, ALL_BAD_CNT) : novalue -
;         3096      begin
;         3097
;         3098  !++
;         3099  !   FUNCTIONAL DESCRIPTION: TRANSFER TEMP BLAST TABLE TO MAIN BLAST TABLE
;         3100  !
;         3101  !                           THE TEMP BLAST TABLE STORES THE NIBBLE NUMBER,
;         3102  !                           ROW/COLUMN NUMBER AND A R/C SELECT FLAG FOR A
;         3103  !                           ROW/COLUMN WHICH HAS BEEN SELECTED FOR BLASTING.
;         3104  !
;         3105  !                           ONCE IT IS DETERMINED THAT THIS CHIP IS NOT ALL
;         3106  !                           BAD (ie. > 10 ALL BAD ROW/COL) THEN THIS TEMP BLAST
;         3107  !                           TABLE IS XFERED INTO THE MAIN BLAST TABLE.
;         3108  !
;         3109  !                           THIS TABLE IS NEEDED BECAUSE ONCE THIS INFORMATION
;         3110  !                           IS XFERED INTO THE MAIN BLAST TABLE THERE IS NO WAY
;         3111  !                           OF KNOWING WHICH CHIP THIS BAD ROW/COL CAME FROM.
;         3112  !                           THERFOR MAKING IT IMPOSSIBLE TO DELETE A ROW/COL
;         3113  !                           FROM THE BLAST TABLE ONCE IT HAS BEEN LOADED INTO IT.
;         3114  !
;         3115  !   FORMAL PARAMETERS:
;         3116  !                   TEM_ARR$BNK_SEL:
;         3117  !                           THIS ARGUMENT STORES THE ARRAY AND BANK SELECT
;         3118  !                           NUMBER.
;         3119  !                   ALL_BAD_CNT:
;         3120  !                           THIS ARGUMENT TELLS THIS ROUTINE HOW MANY
;         3121  !                           REMAINDER TABLE ENTRIES TO XFER INTO THE
;         3122  !                           THE MAIN BLAST TABLE.
;         3123  !--
;         3124
;         3125      local
;         3126          BNK_NUM;                                    !POINTS TO FAILING BANK
;         3127
;         3128      BNK_NUM   .TEMP_ARR$BNK_SEL<.BNK_SEL_POS, BNK$SEL_SIZE>;    !CALCULATE THE FAILING BANK
;         3129
;         3130      incr CNT from 0 to .ALL_BAD_CNT do          !TRANSFER TEMP BLAST TABLE TO BLAST TABLE
;         3131
;         3132          if .TMP_BLST_TBL [.CNT, R_C] eql 1      !IS THIS A FAILING ROW
;         3133          then
;         3134              BLAST_TBL [.BNK_NUM, .TMP_BLST_TBL [.CNT, R_C_NO], .TMP_BLST_TBL [.CNT, NIB_NO], ONE, ZERO]
;         3135              SET_FLG                             !LOAD THE BLAST TABLE WITH THE FAILING ROW NO. AT THIS NIBBLE
;         3136          else
;         3137              BLAST_TBL [.BNK_NUM, .TMP_BLST_TBL [.CNT, R_C_NO] + .COL_BASE, .TMP_BLST_TBL [.CNT, NIB_NO], ONE,
;         3138                  ZERO] - SET_FLG;                !LOAD THE BLAST WITH THE FAILING COLUMN AT HIIS NIBBLE NO.
;         3139
;         3140      end;


                                        .SBTTL  X.TMP.BLST.TBL ROUTINE DECLARATIONS
006720  004167  000000G          X.TMP.BLST.TBL:
                                        JSR     R1,$SAVE3                       ;                                    3095
006724  012746  000016                  MOV     #16.,-(SP)                      ;                                    2198
```

```
006730  060616                          ADD     SP,(SP)                         ; TEMP.ARR$BNK.SE,*
006732  016746  037012'                  MOV     BNK.SEL.POS,-(SP)
006736  012746  000002                  MOV     #2,-(SP)
006742  005046                          CLR     -(SP)
006744  004767  000000G                 JSR     PC,BL$GT2                                                       3134
006750  000300                          SWAB    R0                              ;
006752  105000                          CLRB    R0
006754  006300                          ASL     R0
006756  010003                          MOV     R0,R3
006760  005002                          CLR     R2                              ; CNT                           3130
006762  000450                          BR      4$                                                              3132
006764  010201                  1$:     MOV     R2,R1                           ; CNT,*
006766  006301                          ASL     R1
006770  012700  026744'                  MOV     #TMP.BLST.TBL,R0
006774  060100                          ADD     R1,R0
006776  005710                          TST     (R0)
007000  100010                          BPL     2$
007002  011001                          MOV     (R0),R1                         ;                               3134
007004  006201                          ASR     R1
007006  006201                          ASR     R1
007010  006201                          ASR     R1
007012  006201                          ASR     R1
007014  042701  177400                  BIC     #177400,R1
007020  000411                          BR      3$                                                              3137
007022  011001                  2$:     MOV     (R0),R1                         ;
007024  006201                          ASR     R1
007026  006201                          ASR     R1
007030  006201                          ASR     R1
007032  006201                          ASR     R1
007034  042701  177400                  BIC     #177400,R1
007040  066701  036770'                  ADD     COL.BASE,R1
007044  060301                  3$:     ADD     R3,R1                           ;                               3138
007046  006301                          ASL     R1
007050  062701  026770'                  ADD     #BLAST.TBL,R1
007054  010146                          MOV     R1,-(SP)
007056  111046                          MOVB    (R0),-(SP)
007060  042716  177760                  BIC     #177760,(SP)
007064  012746  000001                  MOV     #1,-(SP)
007070  011646                          MOV     (SP),-(SP)
007072  004767  000000G                 JSR     PC,BL$PU2                                                       3132
007076  062706  000010                  ADD     #10,SP                          ;                               3130
007102  005202                          INC     R2                              ; CNT
007104  020266  000022          4$:     CMP     R2,22(SP)                       ; CNT,ALL.BAD.CNT
007110  003725                          BLE     1$                                                              3096
007112  062706  000010                  ADD     #10,SP                          ;                               3095
007116  000207                          RTS     PC                              ;

; Routine Size:  64 words
; Maximum stack depth per invocation:  12 words
```

```
3141  routine X_TO_REM_TBL (REM_PTR) =
3142      begin
3143
3144  !++
3145  !   FUNCTIONAL DESCRIPTION: TRANSFER COLUMN COUNT TABLE TO T     MAINDER TABLE
3146  !
3147  !                           ONCE THE ERROR MAP AND THE COLUMN COUNT TABLES
3148  !                           HAVE SEARCHED FOR ALL BAD ROWS AND COLUMNS ALL
3149  !                           THAT REMAINS ARE SINGLE CELL FAILURES A SCATTERED
3150  !                           THROUGH OUT THE MOS RAM.
3151  !
3152  !                           THIS ROUTINE XFERS ALL THE REMAINING ROW COLUMN
3153  !                           ADDRESS PAIRS OF THE REMAINING FAILURES INTO
3154  !                           THE REMAINDER TABLE WHERE THEY ARE INTERIGATED
3155  !                           AND THE BEST BLAST SELECTION IS DETERMINED.
3156  !
3157  !   FORMAL PARAMETERS:
3158  !                   REM_PTR
3159  !                           THIS ARGUMENT WILL FIRST POINT TO THE
3160  !                           STARTING TABLE POSITION WHERE THE FIRST
3161  !                           XFERED ROW/COLUMN PAIR IS TO GO.
3162  !
3163  !                           ONCE THE ROUTINE IS COMPLETED THIS ARGUMENTS
3164  !                           VALUE IS RETURNED TO INDICATE HOW MANY ROW/COL
3165  !                           PAIRS WERE XFERED INTO THE REMAINDER TABLE.
3166  !
3167  !--
3168
3169      local
3170          WRD_INDEX,                          .STORES THE ADJACENT ROW ADDRESSES WORD POSITION
3171          BIT_INDEX,                          .STORES THE ADJACENT ROW ADDRESSES WORD BIT POSITION
3172          ROW_NUM,                            .STORES A COLUMNS ADJACENT ROW NUMBER
3173          ADJACENT_CNT,                       .STORES THE NUMBER OF ADJACENT ROW ADDRESSES FOUND
3174          START_SEC,                          .VARIABLE SECTOR STARTING ADDRESS
3175          END_SEC;                            !VARIABLE SECTOR ENDING ADDRESS
3176
3177      incr COL_NUM from 0 to .MAX_CHIP_COL - 1 do         !SEARCH COLUMN COUNT TABLE
3178          begin
3179
3180          if .COL_CNT_TBL [.COL_NUM] neq ZERO     .DOES THIS LOCATION HAVE A COUNT IN I
3181          then
3182              begin
3183
3184              if .COL_NUM gtr 127                 !IS THIS A 64K CHIP
3185              then
3186                  begin
3187                  ROW_NUM = .COL_NUM - 128;       !THE FIRST COLUMN ADDRESS STARTS AT 128
3188                  START_SEC = 256;                !THE FIRST SECTOR STARTS AT 256
3189                  END_SEC = 511;                  .THE END SECTOR ENDS AT 511
3190                  end
3191              else
3192                  begin
```

```
3193                    START_SEC - ZERO;                   .THE START SECTOR STARTS AT ZERO
3194                    ROW_NUM = .COL_NUM;                 !AT THE FIRST SECTOR THE COLUMN = ROW
3195
3196                    if .BNK_NUM_SEC eql 127 then END_SEC = 127 else END_SEC = 255;  !IS THIS A 16K CHIP
3197
3198                    end;
3199
3200
3201                ; NOW SEARCH THRU THE ERROR MAP AND FIND THIS COLUMNS
3202                ; ADJACENT FAILING ROWS USING ROW_NUM AS A POINTER TO THE
3203                ; ADJACENT ROW.
3204                ;
3205                ADJACENT_CNT = ZERO;                    .CLEAR THE ADJACENT COUNT
3206
3207                incr SEC_NUM from .START_SEC to .END_SEC do         !SEARCH ALL THESE SECTORS
3208                    begin
3209                    WRD_INDEX = .ROW_NUM/16;            !CALCULATE WHICH WORD THIS ROW IS IN
3210
3211                    if .ERROR_MAP [.SEC_NUM, .WRD_INDEX, FULL_WRD] neq ZERO !IS THIS ROW BIT SET
3212                    then
3213                        begin
3214                        BIT_INDEX = .ROW_NUM mod 16;        !CALCULATE THE ROWS WORD BIT POSITION
3215
3216                        if .ERROR_MAP [.SEC_NUM, .WRD_INDEX, .BIT_INDEX, ONE, ZERO] eql 1
3217                        then
3218                            begin
3219                            ADJACENT_CNT = .ADJACENT_CNT + 1;       !UP THE ADJACENT COUNT
3220                            REM_TBL [.REM_PTR, COL] = .COL_NUM;     !LOAD THE REM_TBL WITH THIS COLUMN NO.
3221                            COPIED_REM_TBL [.REM_PTR, COL] = .COL_NUM;      .LOAD THE COPIED TABLE TOO
3222
3223                            selectone .SEC_NUM of   !FIND WHICH PHYSICAL ROW NUMBER THIS IS
3224                                set
3225
3226                                [0 to 127, 256 to 383] :    .ARE WE IN QUAD 0 OR 2
3227                                    begin
3228                                    REM_TBL [.REM_PTR, ROW] = .WRD_INDEX*16 + .BIT_INDEX;
3229                                    COPIED_REM_TBL [.REM_PTR, ROW] = .WRD_INDEX*16 + .BIT_INDEX;
3230                                    end;
3231
3232                                [128 to 255, 384 to 511] :  !ARE WE IN QUAD 1 OR 3
3233                                    begin
3234                                    REM_TBL [.REM_PTR, ROW] = .WRD_INDEX*16 + .BIT_INDEX + 128;
3235                                    COPIED_REM_TBL [.REM_PTR, ROW] = .WRD_INDEX*16 + .BIT_INDEX + 128;
3236                                    end;
3237                                tes;
3238
3239                            REM_PTR = .REM_PTR + 1; !UP THE REM_PTR POINTER
3240
3241                            if .ADJACENT_CNT eql .COL_CNT_TBL [.COL_NUM] then exitloop;
3242
3243                            end;
3244
```

```
;          3245                    end;
;          3246
;          3247                ROW_NUM = (.ROW_NUM - 1) and %o'177';   !DECREMENT THE ROW COUNT AND AVOIDE OVER RUN
;          3248                end;
;          3249
;          3250            end;
;          3251
;          3252        end;
;          3253
;          3254    return .REM_PTR;                        !RETURN HOW MANY ROW COLUMN PAIRS WERE XFERED
;          3255    end;
```

```
                                        .SBTTL  X.TO.REM.TBL ROUTINE DECLARATIONS
007120  004167  000000G          X.TO.REM.TBL:
                                        JSR     R1,$SAVE5                       ;                               3141
007124  162706  000014                  SUB     #14,SP
007130  016766  036774' 000012          MOV     MAX.CHIP.COL,12(SP)             ;                               3177
007136  005046                          CLR     -(SP)                           ; COL.NUM
007140  000167  000520                  JMP     17$
007144  011604                  1$:     MOV     (SP),R4                         ; COL.NUM,*                     3180
007146  105764  000000'                TSTB    COL.CNT.TBL(R4)
007152  001002                          BNE     2$
007154  000167  000502                  JMP     16$
007160  021627  000177          2$:     CMP     (SP),#177                       ; COL.NUM,*                     3184
007164  003412                          BLE     3$
007166  011601                          MOV     (SP),R1                         ; COL.NUM,ROW.NUM               3187
007170  162701  000200                  SUB     #200,R1                         ; *,ROW.NUM
007174  012766  000400  000010          MOV     #400,10(SP)                     ; *,START.SEC                   3188
007202  012766  000777  000002          MOV     #777,2(SP)                      ; *,END.SEC                     3189
007210  000416                          BR      5$                              ;                               3184
007212  005066  000010          3$:     CLR     10(SP)                          ; START.SEC                     3193
007216  011601                          MOV     (SP),R1                         ; COL.NUM,ROW.NUM               3194
007220  026727  037014' 000177          CMP     BNK.NUM.SEC,#177                ;                               3196
007226  001004                          BNE     4$
007230  012766  000177  000002          MOV     #177,2(SP)                      ; *,END.SEC
007236  000403                          BR      5$
007240  012766  000377  000002  4$:     MOV     #377,2(SP)                      ; *,END.SEC
007246  005066  000012          5$:     CLR     12(SP)                          ; ADJACENT.CNT                  3205
007252  016604  000010                  MOV     10(SP),R4                       ; START.SEC,SEC.NUM             3207
007256  005304                          DEC     R4                              ; SEC.NUM
007260  000574                          BR      15$
007262  010146                  6$:     MOV     R1,-(SP)                        ; ROW.NUM,*                     3209
007264  012746  000020                  MOV     #20,-(SP)
007270  004767  000000G                 JSR     PC,BL$DIV
007274  010066  000012                  MOV     R0,12(SP)                       ; *,WRD.INDEX
007300  010400                          MOV     R4,R0                           ; SEC.NUM,*                     3211
007302  006300                          ASL     R0
007304  006300                          ASL     R0
007306  006300                          ASL     R0
007310  010005                          MOV     R0,R5
007312  066605  000012                  ADD     12(SP),R5                       ; WRD.INDEX,*
```

```
007316  006305                          ASL     R5
007320  005765  006626'                TST     ERROR.MAP(R5)
007324  001544                          BEQ     14$
007326  010146                          MOV     R1,-(SP)                ; ROW.NUM,*                     3214
007330  012746  000020                  MOV     #20,-(SP)
007334  004767  000000G                 JSR     PC,BL$MOD
007340  010066  000014                  MOV     R0,14(SP)               ; *,BIT.INDEX
007344  012746  006626'                MOV     #ERROR.MAP,-(SP)        ;                               3216
007350  060516                          ADD     R5,(SP)
007352  010046                          MOV     R0,-(SP)                ; BIT.INDEX,*
007354  012746  000001                  MOV     #1,-(SP)
007360  005046                          CLR     -(SP)
007362  004767  000000G                 JSR     PC,BL$GT2
007366  062706  000010                  ADD     #10,SP
007372  005300                          DEC     R0
007374  001117                          BNE     13$
007376  005266  000022                  INC     22(SP)                  ; ADJACENT.CNT                  3219
007402  016605  000044                  MOV     44(SP),R5               ; REM.PTR,*                     3220
007406  006305                          ASL     R5
007410  012702  000400'                MOV     #REM.TBL,R2
007414  060502                          ADD     R5,R2
007416  116612  000010                  MOVB    10(SP),(R2)             ; COL.NUM,*
007422  012703  000710'                MOV     #COPIED.REM.TBL,R3      ;                               3221
007426  060503                          ADD     R5,R3
007430  116613  000010                  MOVB    10(SP),(R3)             ; COL.NUM,*
007434  005704                          TST     R4                      ; SEC.NUM                       3223
007436  002403                          BLT     7$
007440  020427  000177                  CMP     R4,#177                 ; SEC.NUM,*
007444  003406                          BLE     8$
007446  020427  000400          7$:     CMP     R4,#400                 ; SEC.NUM,*
007452  002420                          BLT     9$
007454  020427  000577                  CMP     R4,#577                 ; SEC.NUM,*
007460  003015                          BGT     9$
007462  016600  000016          8$:     MOV     16(SP),R0               ; WRD.INDEX,*                   3228
007466  006300                          ASL     R0
007470  006300                          ASL     R0
007472  006300                          ASL     R0
007474  006300                          ASL     R0
007476  066600  000014                  ADD     14(SP),R0               ; BIT.INDEX,*
007502  110062  000001                  MOVB    R0,1(R2)
007506  110063  000001                  MOVB    R0,1(R3)                ;                               3229
007512  000433                          BR      12$                     ;                               3223
007514  020427  000200          9$:     CMP     R4,#200                 ; SEC.NUM,*
007520  002403                          BLT     10$
007522  020427  000377                  CMP     R4,#377                 ; SEC.NUM,*
007526  003406                          BLE     11$
007530  020427  000600          10$:    CMP     R4,#600                 ; SEC.NUM,*
007534  002422                          BLT     12$
007536  020427  000777                  CMP     R4,#777                 ; SEC.NUM,*
007542  003017                          BGT     12$
007544  016600  000016          11$:    MOV     16(SP),R0               ; WRD.INDEX,*                   3234
007550  006300                          ASL     R0
```

```
007552  006300                          ASL     R0
007554  006300                          ASL     R0
007556  006300                          ASL     R0
007560  066600  000014                  ADD     14(SP),R0               ; BIT.INDEX,*
007564  010005                          MOV     R0,R5
007566  062705  000200                  ADD     #200,R5
007572  110562  000001                  MOVB    R5,1(R2)
007576  110563  000001                  MOVB    R5,1(R3)                ;                       3235
007602  005266  000044          12$:    INC     44(SP)                  ; REM.PTR               3239
007606  005005                          CLR     R5                      ;                       3241
007610  016603  000010                  MOV     10(SP),R3               ; COL.NUM,*
007614  156305  000000'                BISB    COL.CNT.TBL(R3),R5
007620  026605  000022                  CMP     22(SP),R5               ; ADJACENT.CNT,*
007624  001003                          BNE     13$
007626  062706  000010                  ADD     #10,SP
007632  000413                          BR      16$
007634  022626                  13$:    CMP     (SP)+,(SP)+             ;                       3213
007636  010105                  14$:    MOV     R1,R5                   ; ROW.NUM,*             3247
007640  005305                          DEC     R5
007642  010501                          MOV     R5,R1                   ; *,ROW.NUM
007644  042701  177600                  BIC     #177600,R1              ; *,ROW.NUM
007650  022626                          CMP     (SP)+,(SP)+             ;                       3208
007652  005204                  15$:    INC     R4                      ; SEC.NUM               3207
007654  020466  000002                  CMP     R4,2(SP)                ; SEC.NUM,END.SEC
007660  003600                          BLE     6$
007662  005216                  16$:    INC     (SP)                    ; COL.NUM               3177
007664  021666  000014          17$:    CMP     (SP),14(SP)             ; COL.NUM,*
007670  002002                          BGE     18$
007672  000167  177246                  JMP     1$
007676  016600  000034          18$:    MOV     34(SP),R0               ; REM.PTR,*             3142
007702  062706  000016                  ADD     #16,SP                  ;                       3141
007706  000207                          RTS     PC

; Routine Size:  188 words
; Maximum stack depth per invocation:  21 words
```

```
3256  routine COL_PURGE (COL_SEL, QUANTITY, LST_REM_ENTRY) : novalue =
3257      begin
3258
3259  !++
3260  !  FUNCTIONAL DESCRIPTION: COLUMN PURGE
3261  !
3262  !                          THE REMAINDER TABLE IS LOADED WITH ALL THE
3263  !                          THE REMAINING SINGLE CELL FAILURES SCATTERED
3264  !                          THROUGH OUT THE CHIP.  THIS REMAINDER TABLE
3265  !                          IS INTERIGATED FOR THE BEST POSSIBLE CHOICE
3266  !                          FOR ROW OR COLUMN SELECTION FOR BLASTING.
3267  !
3268  !                          ONCE A ROW OR COLUMN IS SELECTED FOR BLASTING
3269  !                          IT OCCURANCE IN THE REMAINDER TABLE MUST BE
3270  !                          TAKEN OUT.
3271  !
3272  !                          THIS ROUTINE WILL SEARCH THE REMAINDER TABLE
3273  !                          OF A SELECTED COLUMN FOR BLASTING AND PURGE
3274  !                          THE TABLE OF ITS OCCURANCE AND ALSO ITS
3275  !                          ROW PAIR ADDRESS.
3276  !
3277  !
3278  !  FORMAL PARAMETERS:
3279  !                  COL_SEL:
3280  !                          THIS ARGUMENT POINTS TO THE COLUMN SELECTED
3281  !                          FOR BLASTING AND THE COLUMN TO BE PURGED FROM
3282  !                          THE TABLE.
3283  !                  QUANTITY:
3284  !                          INDICATES HOW MANY OCCURANCES OF THIS COLUMN
3285  !                          TO EXPECT TO FIND IN THE TABLE.
3286  !                  LST_REM_ENTRY:
3287  !                          POINTS TO THE LAST TABLE ENTRY + 1
3288  !--
3289
3290      if .LST_REM_ENTRY gtr ONE                       !PURGE THE COLUMNS IF REMAINDER TABLE ENTRIES > 1
3291      then
3292          begin
3293
3294          incr PUR_LOC from 0 to .LST_REM_ENTRY - 1 do    !PURGE THE REMAINDER TABLE OF THIS COLUMN
3295              begin
3296
3297              if .REM_TBL [.PUR_LOC, COL] eql .COL_SEL    !IS THIS COLUMN TO BE PURGED
3298              then
3299                  begin
3300
3301                  if (.PUR_LOC + .QUANTITY) neq .LST_REM_ENTRY    !IS THIS COLUMN LAST IN TABLE
3302                  then
3303                      begin
3304
3305                      incr PURGE from .PUR_LOC to (.LST_REM_ENTRY - 1) - .QUANTITY do
3306                          REM_TBL [.PURGE, RO_CL] = .REM_TBL [.PURGE + .QUANTITY, RO_CL];
3307
```

```
;          3308                    incr COPY from 0 to .LST_REM_ENTRY - .QUANTITY do    .COPY REM TBLL TO COPIED TBL
;          3309                        COPIED_REM_TBL [.COPY, RO_CL] = .REM_TBL [.COPY, RO_CL];
;          3310
;          3311                    end;
;          3312
;          3313                exitloop;                           !EXIT THE LOOP WHEN PURGE IS COMPLETE .
;          3314                end;
;          3315
;          3316            end;
;          3317
;          3318        end;
;          3319
;          3320    end;


                                        .SBTTL  COL.PURGE ROUTINE DECLARATIONS
007710  004167  000000G          COL.PURGE:
                                        JSR     R1,$SAVE5                       ;                                       3256
007714  005746                          TST     -(SP)
007716  016605  000020                  MOV     20(SP),R5                       ; LST.REM.ENTRY,*                       3290
007722  020527  000001                  CMP     R5,#1
007726  003460                          BLE     8$
007730  005016                          CLR     (SP)                            ; PUR.LOC                               3294
007732  000454                          BR      7$
007734  011601                  1$:     MOV     (SP),R1                         ; PUR.LOC,*                             3297
007736  006301                          ASL     R1
007740  005004                          CLR     R4
007742  156104  000400'                 BISB    REM.TBL(R1),R4
007746  020466  000024                  CMP     R4,24(SP)                       ; *,COL.SEL
007752  001043                          BNE     6$
007754  016603  000022                  MOV     22(SP),R3                       ; QUANTITY,*                            3301
007760  010300                          MOV     R3,R0
007762  061600                          ADD     (SP),R0                         ; PUR.LOC,*
007764  020005                          CMP     R0,R5
007766  001440                          BEQ     8$
007770  010502                          MOV     R5,R2                           ;                                       3305
007772  160302                          SUB     R3,R2
007774  011604                          MOV     (SP),R4                         ; PUR.LOC,PURGE
007776  005304                          DEC     R4                              ; PURGE
010000  000410                          BR      3$
010002  010400                  2$:     MOV     R4,R0                           ; PURGE,*                               3306
010004  006300                          ASL     R0
010006  010301                          MOV     R3,R1
010010  060401                          ADD     R4,R1                           ; PURGE,*
010012  006301                          ASL     R1
010014  016160  000400' 000400'         MOV     REM.TBL(R1),REM.TBL(R0)
010022  005204                  3$:     INC     R4                              ; PURGE                                 3305
010024  020402                          CMP     R4,R2                           ; PURGE,*
010026  002765                          BLT     2$
010030  010502                          MOV     R5,R2                           ;                                       3308
010032  160302                          SUB     R3,R2
010034  005000                          CLR     R0                              ; COPY
```

```
010036  000406                          BR      5$
010040  010001                  4$:     MOV     R0,R1                           ; COPY,*                        3309
010042  006301                          ASL     R1
010044  016161  000400' 000710'         MOV     REM.TBL(R1),COPIED.REM.TBL(R1)
010052  005200                          INC     R0                              ; COPY                          3308
010054  020002                  5$:     CMP     R0,R2                           ; COPY,*
010056  003770                          BLE     4$
010060  000403                          BR      8$                              ;                               3313
010062  005216                  6$:     INC     (SP)                            ; PUR.LOC                       3294
010064  021605                  7$:     CMP     (SP),R5                         ; PUR.LOC,*
010066  002722                          BLT     1$
010070  005726                  8$:     TST     (SP)+                           ;                               3256
010072  000207                          RTS     PC

; Routine Size:  58 words
; Maximum stack depth per invocation:  7 words
```

```
;          3321  routine ROW_PURGE (ROW_SEL, QUANTITY, LST_REM_ENTRY) : novalue =
;          3322      begin
;          3323
;          3324  !++
;          3325  !  FUNCTIONAL DESCRIPTION: ROW PURGE
;          3326  !
;          3327  !                          THIS ROUTINE IS EXACTLY THE SAME AS COLUMN
;          3328  !                          PURGE EXCEPT THAT THE COPIED REMAINDER
;          3329  !                          TABEL IS SEARCH AND PURGED OF ROW NUMBERS.
;          3330  !--
;          3331
;          3332      if .LST_REM_ENTRY gtr ONE                       .PURGE THE ROW IF COPIED TABLE ENTRIES  > 1
;          3333      then
;          3334          begin
;          3335
;          3336          incr PUR_LOC from 0 to .LST_REM_ENTRY - 1 do     .PURGE THE COPIED REM TABLE
;          3337              begin
;          3338
;          3339              if .COPIED_REM_TBL [.PUR_LOC, ROW] eql .ROW_SEL      !IS THIS THE ROW TO BE PURGED
;          3340              then
;          3341                  begin
;          3342
;          3343                  if (.PUR_LOC + .QUANTITY) neq .LST_REM_ENTRY     !IS THIS THE LAST TABLE ENTRY
;          3344                  then
;          3345                      begin
;          3346
;          3347                      incr PURGE from .PUR_LOC to (.LST_REM_ENTRY - 1) - .QUANTITY do
;          3348                          COPIED_REM_TBL [.PURGE, RO_CL] = .COPIED_REM_TBL [.PURGE + .QUANTITY, RO_CL];
;          3349
;          3350                      incr COPY from 0 to .LST_REM_ENTRY - .QUANTITY do   !COPY COPIED TO REM
;          3351                          REM_TBL [.COPY, RO_CL] = .COPIED_REM_TBL [.COPY, RO_CL];
;          3352
;          3353                      end;
;          3354
;          3355                  exitloop;                           .EXIT LOOP WHEN PURGE IS COMPLETED
;          3356                  end;
;          3357
;          3358              end;
;          3359
;          3360          end;
;          3361
;          3362      end;


                                        .SBTTL  ROW.PURGE ROUTINE DECLARATIONS
010074  004167  000000G          ROW.PURGE:
                                        JSR     R1,$SAVE5                       ;                       3321
010100  005746                          TST     -(SP)
010102  016605  000020                  MOV     20(SP),R5                       ; LST.REM.ENTRY,*       3332
010106  020527  000001                  CMP     R5,#1
010112  003460                          BLE     8$
010114  005016                          CLR     (SP)                            ; PUR.LOC               3336
```

```
010116  000454                          BR      7$
010120  011601                  '$:     MOV     (SP),R1                         ; PUR.LOC,*                     3339
010122  006301                          ASL     R1
010124  005004                          CLR     R4
010126  156104  000711'                 BISB    COPIED.REM.TBL+1(R1),R4
010132  020466  000024                  CMP     R4,24(SP)                       ; *,ROW.SEL
010136  001043                          BNE     6$
010140  016603  000022                  MOV     22(SP),R3                       ; QUANTITY,*                    3343
010144  010300                          MOV     R3,R0
010146  061600                          ADD     (SP),R0                         ; PUR.LOC,*
010150  020005                          CMP     R0,R5
010152  001440                          BEQ     8$
010154  010502                          MOV     R5,R2                           ;                               3347
010156  160302                          SUB     R3,R2
010160  011604                          MOV     (SP),R4                         ; PUR.LOC,PURGE
010162  005304                          DEC     R4                              ; PURGE
010164  000410                          BR      3$
010166  010400                  2$:     MOV     R4,R0                           ; PURGE,*                       3348
010170  006300                          ASL     R0
010172  010301                          MOV     R3,R1
010174  060401                          ADD     R4,R1                           ; PURGE,*
010176  006301                          ASL     R1
010200  016160  000710' 000710'         MOV     COPIED.REM.TBL(R1),COPIED.REM.TBL(R0) ;
010206  005204                  3$:     INC     R4                              ; PURGE                         3347
010210  020402                          CMP     R4,R2                           ; PURGE,*
010212  002765                          BLT     2$
010214  010502                          MOV     R5,R2                           ;                               3350
010216  160302                          SUB     R3,R2
010220  005000                          CLR     R0                              ; COPY
010222  000406                          BR      5$
010224  010001                  4$:     MOV     R0,R1                           ; COPY,*                        3351
010226  006301                          ASL     R1
010230  016161  000710' 000400'         MOV     COPIED.REM.TBL(R1),REM.TBL(R1)
010236  005200                          INC     R0                              ; COPY                          3350
010240  020002                  5$:     CMP     R0,R2                           ; COPY,*
010242  003770                          BLE     4$
010244  000403                          BR      8$                              ;                               3355
010246  005216                  6$:     INC     (SP)                            ; PUR.LOC                       3336
010250  021605                  7$:     CMP     (SP),R5                         ; PUR.LOC,*
010252  002722                          BLT     1$
010254  005726                  8$:     TST     (SP)+                           ;                               3321
010256  000207                          RTS     PC

; Routine Size:  58 words
; Maximum stack depth per invocation:  7 words
```

```
;         3363  routine COL_SORT (LST_REM_ENTRY) : novalue
;         3364      begin
;         3365
;         3366  ;++
;         3367  !  FUNCTIONAL DESCRIPTION: COLUMN SORT
;         3368  !
;         3369  !                          BEFORE THE REMAINDER TABLE IS INTERIGATED FOR COLUMNS
;         3370  !                          THE COLUMN SIDE OF THE TABLE MUST BE SORTED.
;         3371  !
;         3372  !                          THIS ROUTINE WILL SORT THE COLUMN SIDE OF THE
;         3373  !                          REMAINDER TABLE AND THE ROW PAIR GETS MOVED
;         3374  !                          AROUND WITH THE COLUMN PAIR.
;         3375  !
;         3376  !  FORMAL PARAMETERS:
;         3377  !                  LST_REM_ENTRY:
;         3378  !                          POINTS TO THE LAST REMAINDER TABLE ENTRY
;         3379  !--
;         3380
;         3381      local
;         3382          TEMP;                                       .TEMPORARY STORAGE FOR BUBBLED ENTRY
;         3383
;         3384      if .LST_REM_ENTRY gtr ONE                       !SORT THE TABLE IF ENTRIES ARE > 1
;         3385      then
;         3386          begin
;         3387
;         3388          incr PASS from 1 to .LST_REM_ENTRY - 1 do       .BUBBLE SORT TABLE UNTIL IN ASENDING ORDER
;         3389
;         3390              incr index from 0 to ((.LST_REM_ENTRY - 2) - (.PASS - 1)) do        .BUBBLE UP THE BIGGEST NUMBER
;         3391
;         3392                  if .REM_TBL [.index, COL] gtr .REM_TBL [.index + 1, COL]        !IS THE NEXT ENTRY > THIS ONE
;         3393                  then
;         3394                      begin
;         3395                      TEMP = .REM_TBL [.index, RO_CL];    !COPY THE BIGGER ENTRY
;         3396                      REM_TBL [.index, RO_CL] = .REM_TBL [.index + 1, RO_CL];     !PUT THE SMALLER ENTRY HERE
;         3397                      REM_TBL [.index + 1, RO_CL] = .TEMP;        .PUT THE BIGGER ENTRY IN NEXT ENTRY
;         3398                      end;
;         3399
;         3400              end;
;         3401
;         3402      end;


                                        .SBTTL  COL.SORT ROUTINE DECLARATIONS
010260  004167  000000G          COL.SORT:
                                        JSR     R1,$SAVE5                       ;                               3363
010264  005746                          TST     -(SP)
010266  016646  000020                  MOV     20(SP),-(SP)                    ; LST.REM.ENTRY,*               3384
010272  021627  000001                  CMP     (SP),#1
010276  003441                          BLE     6$
010300  011600                          MOV     (SP),R0                         ;                               3390
010302  162700  000002                  SUB     #2,R0
010306  005003                          CLR     R3                              ; PASS                          3388
```

```
010310  000431                          BR      5$
010312  010005          1$:             MOV     R0,R5                   ;                       3390
010314  160305                          SUB     R3,R5                   ; PASS,*
010316  010501                          MOV     R5,R1
010320  005201                          INC     R1
010322  005004                          CLR     R4                      ; INDEX
010324  000421                          BR      4$
010326  010405          2$:             MOV     R4,R5                   ; INDEX,*               3392
010330  006305                          ASL     R5
010332  012702  000400'                 MOV     #REM.TBL,R2
010336  060502                          ADD     R5,R2
010340  010405                          MOV     R4,R5                   ; INDEX,*
010342  006305                          ASL     R5
010344  062705  000402'                 ADD     #REM.TBL+2,R5
010350  121215                          CMPB    (R2),(R5)
010352  101405                          BLOS    3$
010354  011266  000002                  MOV     (R2),2(SP)              ; *,TEMP                3395
010360  011512                          MOV     (R5),(R2)               ;                       3396
010362  016615  000002                  MOV     2(SP),(R5)              ; TEMP,*                3397
010366  005204          3$:             INC     R4                      ; INDEX                 3390
010370  020401          4$:             CMP     R4,R1                   ; INDEX,*
010372  003755                          BLE     2$
010374  005203          5$:             INC     R3                      ; PASS                  3388
010376  020316                          CMP     R3,(SP)                 ; PASS,*
010400  002744                          BLT     1$
010402  022626          6$:             CMP     (SP)+,(SP)+             ;                       3363
010404  000207                          RTS     PC

; Routine Size:  43 words
; Maximum stack depth per invocation:  8 words
```

```
;          3403  routine ROW_SORT (LST_REM_ENTRY) : novalue =
;          3404      begin
;          3405
;          3406  !++
;          3407  !  FUNCTIONAL DESCRIPTION: ROW SORT
;          3408  !
;          3409  !                          THIS ROUTINE SERVES THE SAME PURPOSE AS
;          3410  !                          COLUMN SORT EXCEPT THE COPIED REM TABLE
;          3411  !                          ROWS ARE SORTED.
;          3412  !--
;          3413
;          3414      local
;          3415          TEMP;                                   !TEMPORY STORAGE FOR BUBBLED ENTRY
;          3416
;          3417      if .LST_REM_ENTRY gtr ONE                   !SORT THE TABLE IF ENTRIES > 1
;          3418      then
;          3419          begin
;          3420
;          3421          incr PASS from 1 to .LST_REM_ENTRY - 1 do       !BUBBLE SORT TABLE UNTIL IN ASEENDING ORDER
;          3422
;          3423              incr index from 0 to ((.LST_REM_ENTRY - 2) - (.PASS - 1)) do        .BUBBLE UP THE BIGGEST NUMBER
;          3424
;          3425                  if .COPIED_REM_TBL [.index, ROW] gtr .COPIED_REM_TBL [.index + 1, ROW]
;          3426                  then
;          3427                      begin
;          3428                      TEMP   .COPIED_REM_TBL [.index, RO_CL];      .COPY THE BIGGER ENTRY
;          3429                      COPIED_REM_TBL [.index, RO_CL] = .COPIED_REM_TBL [.index + 1, RO_CL];
;          3430                      COPIED_REM_TBL [.index + 1, RO_CL] = .TEMP; !PUT THE BIGGER ENTYR IN NEXT ENTRY
;          3431                      end;
;          3432
;          3433              end;
;          3434
;          3435      end;


                                        .SBTTL  ROW.SORT ROUTINE DECLARATIONS
010406  004167  000000G          ROW.SORT:
                                        JSR     R1,$SAVE5                       ;                               3403
010412  005746                          TST     -(SP)
010414  016646  000020                  MOV     20(SP),-(SP)                    ; LST.REM.ENTRY,*               3417
010420  021627  000001                  CMP     (SP),#1
010424  003443                          BLE     6$
010426  011600                          MOV     (SP),R0                         ;                               3423
010430  162700  000002                  SUB     #2,R0
010434  005003                          CLR     R3                              ; PASS                          3421
010436  000433                          BR      5$
010440  010005                  1$:     MOV     R0,R5                           ;                               3423
010442  160305                          SUB     R3,R5                           ; PASS,*
010444  010501                          MOV     R5,R1
010446  005201                          INC     R1
010450  005004                          CLR     R4                              ; INDEX
010452  000423                          BR      4$
```

```
010454  010405                          2$:     MOV     R4,R5                           ; INDEX,*                       3425
010456  006305                                  ASL     R5
010460  012702  000710'                        MOV     #COPIED.REM.TBL,R2
010464  060502                                  ADD     R5,R2
010466  010405                                  MOV     R4,R5                           ; INDEX,*
010470  006305                                  ASL     R5
010472  062705  000712'                        ADD     #COPIED.REM.TBL+2,R5
010476  126265  000001  000001                 CMPB    1(R2),1(R5)
010504  101405                                  BLOS    3$
010506  011266  000002                        MOV     (R2),2(SP)                      ; *,TEMP                        3428
010512  011512                                  MOV     (R5),(R2)                       ;                               3429
010514  016615  000002                        MOV     2(SP),(R5)                      ; TEMP,*                        3430
010520  005204                          3$:     INC     R4                              ; INDEX                         3423
010522  020401                          4$:     CMP     R4,R1                           ; INDEX,*
010524  003753                                  BLE     2$
010526  005203                          5$:     INC     R3                              ; PASS                          3421
010530  020316                                  CMP     R3,(SP)                         ; PASS,*
010532  002742                                  BLT     1$
010534  022626                          6$:     CMP     (SP)+,(SP)+                     ;                               3403
010536  000207                                  RTS     PC

; Routine Size:  45 words
; Maximum stack depth per invocation:  8 words
```

```
;          3436  routine S_BLAST_TBL (BNK_NUM) =
;          3437      begin
;          3438
;          3439  !++
;          3440  !  FUNCTIONAL DESCRIPTION: SEARCH THE BLAST TABLE
;          3441  !
;          3442  !                          TO DETERMINE IF A BANK HAS ANY ADDITIONAL
;          3443  !                          FAILING ROW OR COLUMNS THE BLAST TABLE IS
;          3444  !                          SEARCHED FOR ANY BITS SET.
;          3445  !
;          3446  !                          IF BITS ARE SET THEN THIS BANK MUST BE
;          3447  !                          PM'ED.  IF NO BITS ARE SET THEN THIS BANK
;          3448  !                          CAN BE SKIPPED AND NOT PM'ED
;          3449  !
;          3450  !                          THIS ROUTINE SEARCHES THE BLAST TABLE AT
;          3451  !                          THE SELECTED BANK AND SEES IF ANY BITS ARE
;          3452  !                          ARE SET INDICATING NEW ERRORS FOR THIS BANK.
;          3453  !                          A TRUE INDICATOR IS RETURNED IF NEW ERRORS
;          3454  !                          EXIST AND A FALSE INDICATOR IS RETURNED IF
;          3455  !                          NO NEW ERRORS ARE FOUND.
;          3456  !
;          3457  !  FORMAL PARAMETERS:
;          3458  !                  BNK_NUM:
;          3459  !                          POINTS TO THE BANK TO BE SEARCHED
;          3460  !--
;          3461
;          3462      incr SEARCH from 0 to .MAX_CHIP_COL*2 - 1 do        !LOOK AT ALL ROWS AND COLS FOR THIS BANK
;          3463          begin
;          3464
;          3465          if .BLAST_TBL [.BNK_NUM, .SEARCH, FULL_WRD] neq ZERO then return TRUE;  !RETURN TRU IF ANY BITS SET
;          3466
;          3467          end;
;          3468
;          3469      return FALSE;                                   !RETURN FALSE IF NO BITS ARE SET
;          3470      end;


                                        .SBTTL  S.BLAST.TBL ROUTINE DECLARATIONS
010540  004167  000000G          S.BLAST.TBL:
                                        JSR     R1,$SAVE3                       ;                               3436
010544  016703  036774'                 MOV     MAX.CHIP.COL,R3                 ;                               3462
010550  006303                          ASL     R3
010552  016600  000012                  MOV     12(SP),R0                       ; BNK.NUM,*                     3465
010556  000300                          SWAB    R0
010560  105000                          CLRB    R0
010562  006300                          ASL     R0
010564  005001                          CLR     R1                              ; SEARCH                        3462
010566  000412                          BR      3$
010570  010002                  1$:     MOV     R0,R2                           ;                               3465
010572  060102                          ADD     R1,R2                           ; SEARCH,*
010574  006302                          ASL     R2
010576  005762  026770'                 TST     BLAST.TBL(R2)
```

```
010602  001403                          BEQ     2$
010604  012700  000001                  MOV     #1,R0
010610  000207                          RTS     PC
010612  005201                  2$:     INC     R1                      ; SEARCH                3462
010614  020103                  3$:     CMP     R1,R3                   ; SEARCH,*
010616  002764                          BLT     1$
010620  005000                          CLR     R0                      ;                       3437
010622  000207                          RTS     PC                      ;                       3436

; Routine Size:  26 words
; Maximum stack depth per invocation:  4 words
```

```
3471  routine S_COLUMNS (MOST_OFTEN, SUM_MOST, LST_REM_ENTRY) =
3472      begin
3473
3474  !++
3475  !  FUNCTIONAL DESCRIPTION: SEARCH THE COLUMN TABLE
3476  !
3477  !                          THE REMAINDER TABLE IS SEARCHED FOR THE
3478  !                          BEST POSSIBLE CHOICE FOR BLASTING ie.
3479  !                          WHETHER TO SELECT ROWS OR COLUMNS  FOR
3480  !                          BLASTING.  THIS IS DONE BY FIRST FINDING WHETHER
3481  !                          ROW NUMBERS OR COLUMN NUMBERS APPEAR MOST
3482  !                          OFTEN IN THE REMAINDER TABLE.  THEN IN THIS
3483  !                          GREATER COUNT WHICH MEMBER APPEARS MOST OFTEN.
3484  !
3485  !                          THIS ROUTINE SEARCHES THE REMAINDER TABLE FOR
3486  !                          THE NUMBER OF DIFFERENT NUMBER OF FAILING
3487  !                          COLUMNS FOUND AND WHICH FAILING COLUMN NUMBER
3488  !                          HAS THE GREATEST NUMBER OF OCCURANCES.
3489  !  FORMAL PARAMETERS:
3490  !                  MOST_OFTEN:
3491  !                          PASSED BY REFERENCE AND STORES THE MOST
3492  !                          OFTEN FOUND COLUMN.
3493  !                  SUM_OFTEN:
3494  !                          PASSED BY REFERENCE AND STORES THE SUM
3495  !                          OF THE MOST OFTEN FOUND COLUMN.
3496  !                  LST_REM_ENTRY:
3497  !                          POINTS TO THE LAST REMAINDER ENTRY
3498  !--
3499
3500      local
3501          CUR_SUM,                        !STORES CURRENT SUM OF MOST OFTEN FOUND COLUMN
3502          PRE_SUM,                        !STORES PREVIOUS SUM OF MOST OFTEN FOUND COLUMN
3503          CUR_MOST,                       !STORES CURRENT MOST OFTEN FOUND COLUMN
3504          PRE_MOST,                       !STORES PREVIOUS MOST OFTEN FOUND COLUMN
3505          DIF;                            !STORES HOW MANY DIFFERENT COLUMNS WERE FOUND
3506
3507      CUR_SUM = 1;                        !CURRENT SUM STARTS AT ONE
3508      PRE_SUM = 1;                        !PREVIOUS SUM STARTS AT ONE
3509      PRE_MOST = .REM_TBL [ZERO, COL];    !PREVIOUS MOST OFTEN STARTS AT FIRST COLUMN IN TABLE
3510      DIF = 1;                            !DIFFERENT COLUMN COUNT STARTS AT ONE
3511
3512      if .LST_REM_ENTRY gtr ONE           !FIRST TABLE ENTRY IS MOST OFTEM IF TABLE ENTRY < 1
3513      then
3514          begin
3515          COL_SORT (.LST_REM_ENTRY);      !SORT THE COLUMN SIDE OF REMAINDER TABLE
3516
3517          incr index from 0 to .LST_REM_ENTRY - 2 do      !FIND THE SUM, MOST OFTEN AND DIFFERENT
3518              begin
3519
3520              if .REM_TBL [.index, COL] eql .REM_TBL [.index + 1, COL]    .IS THIS THE SAME AS THE NEXT
3521              then
3522                  begin
```

```
;          3523                    CUR_MOST = .REM_TBL [.index, COL];      !MOST OFTEN IS THIS ENTRY
;          3524                    CUR_SUM = .CUR_SUM + 1;                 .INCREMENT THE SUM OF THE MOST
;          3525                    end
;          3526                ELSE
;          3527                    begin
;          3528                    DIF = .DIF + 1;                         .INCREMENT THE DIFFERENT COLUMN COUNT
;          3529
;          3530                    if .CUR_SUM gtr .PRE_SUM                .IS THE CURRENT SUM > PREVIOUS SUM
;          3531                    then
;          3532                        begin
;          3533                        PRE_SUM = .CUR_SUM;                 .SAVE THE SUM OF THE CURRENT MOST OFTEN
;          3534                        PRE_MOST = .CUR_MOST;               .SAVE THE CURRENT MOST OFTEN
;          3535                        end;
;          3536
;          3537                    CUR_SUM = 1;                            .START CURRENT SUM BACK TO ONE
;          3538                    end;
;          3539
;          3540                end;
;          3541
;          3542            end;
;          3543
;          3544
;          3545        ! THIS TEST TO SEE IF THE LAST SET OF
;          3546        ! UNIQUE COLUMN NUMBERS IN THE TABLE IS > THE PREVIOUS SUM
;          3547        !
;          3548
;          3549        if .CUR_SUM gtr .PRE_SUM                            .IS THE CURRENT SUM > THE PREVIOUS SUM
;          3550        then
;          3551            begin
;          3552            PRE_SUM = .CUR_SUM;                             !SAVE THE SUM OF THE CURRENT MOST OFTEN
;          3553            PRE_MOST = .CUR_MOST;                           .SAVE THE CURRENT MOST OFTEN
;          3554            end;
;          3555
;          3556        .MOST_OFTEN = .PRE_MOST;                            !RETURN THE MOST OFTEN FOUND COLUMN
;          3557        .SUM_MOST   .PRE_SUM;                               .RETURN THE SUM OF THE MOST OFTEN COLUMN
;          3558        return .DIF;                                        .RETURN THE DIFFERENT NUMBER OF COLUMNS FOUND
;          3559        end;


                                                .SBTTL  S.COLUMNS ROUTINE DECLARATIONS
010624  004167  000000G                 S.COLUMNS:
                                        JSR     R1,$SAVE5                   ;                                   3471
010630  024646                          CMP     -(SP),-(SP)
010632  012701  000001                  MOV     #1,R1                       ; *,CUR.SUM                         3507
010636  010105                          MOV     R1,R5                       ; *,PRE.SUM                         3508
010640  005046                          CLR     -(SP)                       ; PRE.MOST                          3509
010642  116716  000400'                MOVB    REM.TBL,(SP)                ; *,PRE.MOST
010646  012766  000001  000002          MOV     #1,2(SP)                    ; *,DIF                             3510
010654  016604  000024                  MOV     24(SP),R4                   ; LST.REM.ENTRY,*                   3512
010660  020427  000001                  CMP     R4,#1
010664  003444                          BLE     6$
010666  010446                          MOV     R4,-(SP)                    ;                                   3515
```

```
010670  004767  177364                  JSR     PC,COL.SORT
010674  162704  000002                  SUB     #2,R4                   ;                               3517
010700  005002                          CLR     R2                      ; INDEX
010702  000432                          BR      5$
010704  010203                  1$:     MOV     R2,R3                   ; INDEX,*                       3520
010706  006303                          ASL     R3
010710  010200                          MOV     R2,R0                   ; INDEX,*
010712  006300                          ASL     R0
010714  126360  000400' 000402'         CMPB    REM.TBL(R3),REM.TBL+2(R0)
010722  001007                          BNE     2$
010724  116366  000400' 000006          MOVB    REM.TBL(R3),6(SP)       ; *,CUR.MOST                    3523
010732  105066  000007                  CLRB    7(SP)                   ; CUR.MOST
010736  005201                          INC     R1                      ; CUR.SUM                       3524
010740  000412                          BR      4$                      ;                               3520
010742  005266  000004          2$:     INC     4(SP)                   ; DIF                           3528
010746  020105                          CMP     R1,R5                   ; CUR.SUM,PRE.SUM               3530
010750  003404                          BLE     3$
010752  010105                          MOV     R1,R5                   ; CUR.SUM,PRE.SUM               3533
010754  016666  000006  000002          MOV     6(SP),2(SP)             ; CUR.MOST,PRE.MOST             3534
010762  012701  000001          3$:     MOV     #1,R1                   ; *,CUR.SUM                     3537
010766  005202                  4$:     INC     R2                      ; INDEX                         3517
010770  020204                  5$:     CMP     R2,R4                   ; INDEX,*
010772  003744                          BLE     1$
010774  005726                          TST     (SP)+                   ;                               3514
010776  020105                  6$:     CMP     R1,R5                   ; CUR.SUM,PRE.SUM               3549
011000  003403                          BLE     7$
011002  010105                          MOV     R1,R5                   ; CUR.SUM,PRE.SUM               3552
011004  016616  000004                  MOV     4(SP),(SP)              ; CUR.MOST,PRE.MOST             3553
011010  012676  000026          7$:     MOV     (SP)+,@26(SP)           ; PRE.MOST,MOST.OFTEN           3556
011014  010576  000024                  MOV     R5,@24(SP)              ; PRE.SUM,SUM.MOST              3557
011020  012600                          MOV     (SP)+,R0                ; DIF,*                         3472
011022  005726                          TST     (SP)+                   ;                               3471
011024  000207                          RTS     PC

; Routine Size:  65 words
; Maximum stack depth per invocation:  10 words
```

```
3560  routine S_ROWS (MOST_OFTEN, SUM_MOST, LST_REM_ENTRY) =
3561      begin
3562
3563  !++
3564  ! FUNCTIONAL DESCRIPTION: SEARCH ROWS
3565  !
3566  !                       THIS ROUTINE HAS THE SAME FUNCTIONALITY
3567  !                       AS SEARCH COLUMNS EXCEPT THAT THE COPIED
3568  !                       REMAINDER TABLE IS SEARCHED FOR ROW NUMBERS.
3569  !--
3570
3571      local
3572          CUR_SUM,                                    !STORES CURRENT SUM OF MOST OFTEN FOUND ROW
3573          PRE_SUM,                                    .STORES PREVIOUS SUM OF MOST OFTEN FOUND ROW
3574          CUR_MOST,                                   .STORES CURRENT MOST OFTEN ROW FOUND
3575          PRE_MOST,                                   .STORES PREVIOUS MOST OFTEN ROW FOUND
3576          DIF;
3577
3578      CUR_SUM = 1;                                    .CURRENT SUM STARTS AT ONE
3579      PRE_SUM = 1;                                    .PREVIOUS SUM STARTS AT ONE
3580      PRE_MOST    .COPIED_REM_TBL [ZERO, ROW];        !PREVIOUS MOST OFTEN STARTS AT FIRST TABLE ENRRY
3581      DIF = 1;                                        .DIFFERENT ROW COUNT STARTS AT ONE
3582
3583      if .LST_REM_ENTRY gtr ONE                       !FIRST TABLE ENTRY IS MOST OFTEN IF TABLE ENTRY < 1
3584      then
3585          begin
3586          ROW_SORT (.LST_REM_ENTRY);                  !SORT THE ROW SIDE OF COPIED REM TABLE
3587
3588          incr index from 0 to .LST_REM_ENTRY - 2 do      .FIND THE SUM, MOST OFTEN AND DIFFERENT
3589              begin
3590
3591              if .COPIED_REM_TBL [.index, ROW] eql .COPIED_REM_TBL [.index + 1, ROW]
3592              then
3593                  begin
3594                  CUR_MOST = .COPIED_REM_TBL [.index, ROW];       !MOST OFTEN IS THIS ENTRY
3595                  CUR_SUM = .CUR_SUM + 1;             .INCREMENT THE SUM OF MOST
3596                  end
3597              else
3598                  begin
3599                  DIF = .DIF + 1;                     !INCREMENT THE DIFFFRENT ROW COUNT
3600
3601                  if .CUR_SUM gtr .PRE_SUM            !IS THE CURRENT SUM > PREVIOUS SUM
3602                  then
3603                      begin
3604                      PRE_SUM = .CUR_SUM;             !SAVE THE SUM OF THE CURRENT MOST OFTEN
3605                      PRE_MOST = .CUR_MOST;           !SAVE THE CURRENT MOST OFTEN ROW
3606                      end;
3607
3608                  CUR_SUM = 1;                        .START CURRENT SUM BACK TO ONE
3609                  end;
3610
3611              end;
```

```
;         3612
;         3613          end;
;         3614
;         3615
;         3616      ! THIS TEST TO SEE IF THE LAST SET OF
;         3617      ! UNIQUE ROW NUMBERS IN TABLE IS > THE PREVIOUS
;         3618      !
;         3619
;         3620      if .CUR_SUM gtr .PRE_SUM                    .IS THE CURRENT SUM > PREVIOUS SUM
;         3621      then
;         3622          begin
;         3623          PRE_SUM = .CUR_SUM;                     .SAVE THE SUM OF THE CURRENT MOST OFTEN
;         3624          PRE_MOST = .CUR_MOST;                   .SAVE THE CURRENT MOST OFTEN ROW
;         3625          end;
;         3626
;         3627      .MOST_OFTEN = .PRE_MOST;                    .RETURN THE MOST OFTEN FOUND ROW
;         3628      .SUM_MOST = .PRE_SUM;                       .RETURN THE SUM OF THE MOST OFTEN FOUND ROW
;         3629      return .DIF;                                !RETURN THE NUMBER OF DIFFERENT FOUND ROWS
;         3630      end;


                                        .SBTTL  S.ROWS ROUTINE DECLARATIONS
011026  004167  000000G          S.ROWS: JSR     R1,$SAVE5                       ;                       3560
011032  024646                          CMP     -(SP),-(SP)
011034  012701  000001                  MOV     #1,R1                           ; *,CUR.SUM             3578
011040  010105                          MOV     R1,R5                           ; *,PRE.SUM             3579
011042  005046                          CLR     -(SP)                           ; PRE.MOST              3580
011044  116716  000711'                MOVB    COPIED.REM.TBL+1,(SP)           ; *,PRE.MOST
011050  012766  000001  000002         MOV     #1,2(SP)                        ; *,DIF                 3581
011056  016604  000024                  MOV     24(SP),R4                       ; LST.REM.ENTRY,*       3583
011062  020427  000001                  CMP     R4,#1
011066  003444                          BLE     6$
011070  010446                          MOV     R4,-(SP)                        ;                       3586
011072  004767  177310                  JSR     PC,ROW.SORT
011076  162704  000002                  SUB     #2,R4                           ;                       3588
011102  005002                          CLR     R2                              ; INDEX
011104  000432                          BR      5$
011106  010203                  1$:     MOV     R2,R3                           ; INDEX,*               3591
011110  006303                          ASL     R3
011112  010200                          MOV     R2,R0                           ; INDEX,*
011114  006300                          ASL     R0
011116  126360  000711' 000713'         CMPB    COPIED.REM.TBL+1(R3),COPIED.REM.TBL+3(R0) ;
011124  001007                          BNE     2$
011126  116366  000711' 000006         MOVB    COPIED.REM.TBL+1(R3),6(SP)      ; *,CUR.MOST            3594
011134  105066  000007                  CLRB    7(SP)                           ; CUR.MOST
011140  005201                          INC     R1                              ; CUR.SUM               3595
011142  000412                          BR      4$                              ;                       3591
011144  005266  000004          2$:     INC     4(SP)                           ; DIF                   3599
011150  020105                          CMP     R1,R5                           ; CUR.SUM,PRE.SUM       3601
011152  003404                          BLE     3$
011154  010105                          MOV     R1,R5                           ; CUR.SUM,PRE.SUM       3604
011156  016666  000006  000002         MOV     6(SP),2(SP)                     ; CUR.MOST,PRE.MOST     3605
```

```
011164  012701  000001          3$:     MOV     #1,R1                           ; *,CUR.SUM                     3608
011170  005202                  4$:     INC     R2                              ; INDEX                         3588
011172  020204                  5$:     CMP     R2,R4                           ; INDEX,*
011174  003744                          BLE     1$
011176  005726                          TST     (SP)+                           ;                               3585
011200  020105                  6$:     CMP     R1,R5                           ; CUR.SUM,PRE.SUM               3620
011202  003403                          BLE     7$
011204  010105                          MOV     R1,R5                           ; CUR.SUM,PRE.SUM               3623
011206  016616  000004                  MOV     4(SP),(SP)                      ; CUR.MOST,PRE.MOST             3624
011212  012676  000026          7$:     MOV     (SP)+,@26(SP)                   ; PRE.MOST,MOST.OFTEN           3627
011216  010576  000024                  MOV     R5,@24(SP)                      ; PRE.SUM,SUM.MOST              3628
011222  012600                          MOV     (SP)+,R0                        ; DIF,*                         3561
011224  005726                          TST     (SP)+                           ;                               3560
011226  000207                          RTS     PC

; Routine Size:  65 words
; Maximum stack depth per invocation:  10 words
```

```
3631  routine S_REM_TBL (BAD_CHIP, ALL_BAD) =
3632      begin
3633
3634  !++
3635  !  FUNCTIONAL DESCRIPTION: SEARCH THE REMAINDER TABLE
3636  !
3637  !                          THE REMAINDER TABLE IS LOADED WITH ALL THE
3638  !                          REMAINING SINGLE CELL FAILURES SCATTERED
3639  !                          THROUGH OUT THE CHIP.
3640  !
3641  !                          THESE SCATTERED REMAINING ROW COLUMN PAIRS
3642  !                          ARE THEN TRANSFERED INTO THE REMAINDER TABLE
3643  !                          WERE THEY ARE INTERIGATED FOR THE BEST POSSIBLE
3644  !                          BLASTING SELECTION.
3645  !
3646  !                          THIS ROUTINE SEARCHES THE REMAINDER TABLE OF
3647  !                          THE REMAINING ROW COLUMN PAIRS AND DETERMINES
3648  !                          THE BEST POSSIBLE CHOICE FOR BLASTING.
3649  !
3650  !  FORMAL PARAMETERS:
3651  !                  BAD_CHIP:
3652  !                          POINTS THE THE FAILING CHIP PRESENTLY
3653  !                          BEING PM'ED.
3654  !                  ALL_BAD:
3655  !                          COUNTS HOW MANY ALL BAD ROW AND COLUMNS
3656  !                          HAVE BEEN FOUND THUS FAR.  IF THIS COUNT
3657  !                          EXCEEDS A COUNT OF 10 FOR ANY ONE CHIP
3658  !                          THEN THAT CHIP IS CALLED A ALL BAD AND
3659  !                          A CONDITION A MESSAGE IS PRINTED.  IF
3660  !                          TWO ALL BAD CHIPS ARE DISCOVERED IN THE
3661  !                          THE SAME BANK THEN A CONDITION B MESSAGE
3662  !                          IS PRINTED AND FURTHER TESTING OF THIS
3663  !                          ARRAY IS ABORTED.
3664  !--
3665
3666      local
3667          LST_REM_ENTRY,                          !POINTS TO LAS ENTRY INTO REMAINDER TABLE
3668          DIF_COLS,                               !DIFFERENT NO. OF COLS IN REMAINDER TABLE
3669          DIF_ROWS,                               !DIFFERNET NO. OF ROWS IN REMAINDER TABLE
3670          ROW_MOST_OFTEN,                         !MOST OFTEN FOUND ROW IN REMAINDER TABLE
3671          COL_MOST_OFTEN,                         !MOST OFTEN FOUND COLUMN IN REMAINDER TABLE
3672          ROW_SUM,                                !SUM OF THE MOST OFTEN FOUND ROW IN REMAINDER TABLE
3673          COL_SUM;                                .SUM OF THE MOST OFTEN FOUND COLUMN IN REMAINDER TABLE
3674
3675      I_REM_TBL ();                               !INIT THE REMAINDER AND COPIED REM TABLES
3676      LST_REM_ENTRY = ZERO;                       .LSAT REM ENTRY STARTS AT ZERO
3677      LST_REM_ENTRY = X_TO_REM_TBL (.LST_REM_ENTRY);
3678                                                  !XFER COLUMN COUNT BABLE TO REMAINDER AND COPIED TABLES
3679
3680      while .LST_REM_ENTRY gtr ZERO do            .SEARCH THE REMAINDER AND COPIED TABLE UNTIL EMPTY
3681          begin
3682          ALL_BAD = .ALL_BAD + 1;                 !INCREMENT THE ALL BAD ROW / COLUMN COUNT
```

```
;          3683
;          3684          if .ALL_BAD gtr 9                          !IS THE ALL BAD COUNT > 9
;          3685          then
;          3686              return .ALL_BAD                        !EXIT AND RETURN THE COUNT OF ALL BAD ROW/COL
;          3687          else
;          3688              begin
;          3689              DIF_COLS = S_COLUMNS (COL_MOST_OFTEN, COL_SUM, .LST_REM_ENTRY);
;          3690                                                     .FIND THE COLUMNS DIFFERENT, MOST LFTEN AND SUM
;          3691              DIF_ROWS = S_ROWS (ROW_MOST_OFTEN, ROW_SUM, .LST_REM_ENTRY);
;          3692                                                     .FIND THE ROWS DIFFERENT, MOST FOTEN AND SUM
;          3693
;          3694              if .DIF_ROWS leq .DIF_COLS             .ARE DIFFERENT ROWS < DIFFERENT COLUMNS
;          3695              then
;          3696                  begin
;          3697                  TMP_BLST_TBL [.ALL_BAD, R_C] = SET_FLG; !INDICATE THIS IS A BAD ROW
;          3698                  TMP_BLST_TBL [.ALL_BAD, R_C_NO] = .ROW_MOST_OFTEN;      !LOAD THE FAILING ROW NO.
;          3699                  TMP_BLST_TBL [.ALL_BAD, NIB_NO] = .BAD_CHIP74;  !LOAD THE FAILING NIBBLE  POSITION
;          3700                  ROW_PURGE (.ROW_MOST_OFTEN, .ROW_SUM, .LST_REM_ENTRY);  !PURGE THIS ROW FROM TABLE
;          3701                  LST_REM_ENTRY = .LST_REM_ENTRY - .ROW_SUM;      !REALIGN THE POINTER
;          3702                  end
;          3703              else
;          3704                  begin
;          3705                  TMP_BLST_TBL [.ALL_BAD, R_C] = CLR_FLG; !INDICATE THIS IS A BAD COLUMS
;          3706                  TMP_BLST_TBL [.ALL_BAD, R_C_NO] = .COL_MOST_OFTEN;      !LOAD THE FAILING COLUMN NO.
;          3707                  TMP_BLST_TBL [.ALL_BAD, NIB_NO] = .BAD_CHIP74;  .LOAD THE FAILING NIBBLE POSITION
;          3708                  COL_PURGE (.COL_MOST_OFTEN, .COL_SUM, LST_REM_ENTRY);   !PURGE COLUMN FORM TALBE
;          3709                  LST_REM_ENTRY = .LST_REM_ENTRY - .COL_SUM;      .REALIGN POINTER
;          3710                  end;
;          3711
;          3712                  end;
;          3713
;          3714              end;
;          3715
;          3716          return .ALL_BAD;                           !RETURN THE NUMBER OF ALL BAD ROWS/COLUMNS FOUND
;          3717          end;


                                .SBTTL  S.REM.TBL ROUTINE DECLARATIONS
011230  004167  000000G    S.REM.TBL:
                            JSR     R1,$SAVE5                   ;                                   3631
011234  162706  000012      SUB     #12,SP
011240  004767  173230      JSR     PC,I.REM.TBL                ;                                   3675
011244  005016              CLR     (SP)                        ; LST.REM.ENTRY                     3676
011246  005046              CLR     -(SP)                       ;                                   3677
011250  004767  175644      JSR     PC,X.TO.REM.TBL
011254  010066  000002      MOV     R0,2(SP)                    ; *,LST.REM.ENTRY
011260  016602  000002  1$: MOV     2(SP),R2                    ; LST.REM.ENTRY,*                   3680
011264  003547              BLE     5$
011266  005266  000032      INC     32(SP)                      ; ALL.BAD                           3682
011272  016603  000032      MOV     32(SP),R3                   ; ALL.BAD,*                         3684
011276  020327  000011      CMP     R3,#11
011302  003403              BLE     2$
```

```
011304  005726                          TST     (SP)+                           ;                                       3686
011306  010300                          MOV     R3,R0
011310  000540                          BR      6$
011312  012746  000014        2$:     MOV     #14,-(SP)                       ;                                       3689
011316  060616                          ADD     SP,(SP)                         ; COL.MOST.OFTEN,*
011320  012746  000014                  MOV     #14,-(SP)
011324  060616                          ADD     SP,(SP)                         ; COL.SUM,*
011326  010246                          MOV     R2,-(SP)
011330  004767  177270                  JSR     PC,S.COLUMNS
011334  010004                          MOV     R0,R4                           ; *,DIF.COLS
011336  012716  000014                  MOV     #14,(SP)                        ;                                       3691
011342  060616                          ADD     SP,(SP)                         ; ROW.MOST.OFTEN,*
011344  012746  000014                  MOV     #14,-(SP)
011350  060616                          ADD     SP,(SP)                         ; ROW.SUM,*
011352  010246                          MOV     R2,-(SP)
011354  004767  177446                  JSR     PC,S.ROWS
011360  010001                          MOV     R0,R1                           ; *,DIF.ROWS
011362  016616  000046                  MOV     46(SP),(SP)                     ; BAD.CHIP,*                            3699
011366  012746  000004                  MOV     #4,-(SP)
011372  004767  000000G                 JSR     PC,BL$DIV
011376  006303                          ASL     R3                              ;                                       3697
011400  062703  026744'                 ADD     #TMP.BLST.TBL,R3
011404  020104                          CMP     R1,R4                           ; DIF.ROWS,DIF.COLS                     3694
011406  003035                          BGT     3$
011410  052713  100000                  BIS     #100000,(R3)                    ;                                       3697
011414  016605  000022                  MOV     22(SP),R5                       ; ROW.MOST.OFTEN,*                      3698
011420  006305                          ASL     R5
011422  006305                          ASL     R5
011424  006305                          ASL     R5
011426  006305                          ASL     R5
011430  042705  170017                  BIC     #170017,R5
011434  042713  007760                  BIC     #7760,(R3)
011440  050513                          BIS     R5,(R3)
011442  042700  177760                  BIC     #177760,R0                      ;                                       3699
011446  142713  000017                  BICB    #17,(R3)
011452  150013                          BISB    R0,(R3)
011454  016646  000022                  MOV     22(SP),-(SP)                    ; ROW.MOST.OFTEN,*                      3700
011460  016646  000022                  MOV     22(SP),-(SP)                    ; ROW.SUM,*
011464  010246                          MOV     R2,-(SP)
011466  004767  176402                  JSR     PC,ROW.PURGE
011472  166666  000026  000024          SUB     26(SP),24(SP)                   ; ROW.SUM,LST.REM.ENTRY                 3701
011500  000436                          BR      4$                              ;                                       3694
011502  042713  100000        3$:     BIC     #100000,(R3)                    ;                                       3705
011506  016605  000026                  MOV     26(SP),R5                       ; COL.MOST.OFTEN,*                      3706
011512  006305                          ASL     R5
011514  006305                          ASL     R5
011516  006305                          ASL     R5
011520  006305                          ASL     R5
011522  042705  170017                  BIC     #170017,R5
011526  042713  007760                  BIC     #7760,(R3)
011532  050513                          BIS     R5,(R3)
011534  042700  177760                  BIC     #177760,R0                      ;                                       3707
```

```
011540  142713  000017                          BICB    #17,(R3)
011544  150013                                  BISB    R0,(R3)
011546  016646  000026                          MOV     26(SP),-(SP)            ; COL.MOST.O-TEN,*                      3708
011552  016646  000026                          MOV     26(SP),-(SP)            ; COL.SUM,*
011556  012746  000024                          MOV     #24,-(SP)
011562  060616                                  ADD     SP,(SP)                 ; LST.REM.ENTRY,*
011564  004767  176120                          JSR     PC,COL.PURGE
011570  166666  000032  000024                  SUB     32(SP),24(SP)           ; COL.SUM,LST.REM.ENTRY                 3709
011576  062706  000022                  4$:     ADD     #22,SP                  ;                                       3688
011602  000626                                  BR      1$                      ;                                       3680
011604  005726                          5$:     TST     (SP)+                   ;                                       3631
011606  016600  000030                          MOV     30(SP),R0               ; ALL.BAD,*                             3632
011612  062706  000012                  6$:     ADD     #12,SP                  ;                                       3631
011616  000207                                  RTS     PC

; Routine Size:  124 words
; Maximum stack depth per invocation:  21 words
```

```
3718  routine S_COL_CNT_TBL (BAD_CHIP, ALL_BAD) =
3719      begin
3720
3721  !++
3722  !  FUNCTIONAL DESCRIPTION: SEARCH THE COLUMN COUNT TABLE
3723  !
3724  !                          THE COLUMN COUNT TABLE STORES A COUNT OF
3725  !                          ADJACENT FAILING COLUMNS WHEN THE ERROR
3726  !                          MAP IS SEARCHED FOR BAD ROWS.
3727  !
3728  !                          THIS ROUTINE SEARCHES THE COLUMN COUNT
3729  !                          TABLE FOR COLUMN COUNTS FOR  COLUMNS
3730  !                          COUNTS GREATER THAN 10.  IF A COUNT OF
3731  !                          GREATER THAN 10 IS FOUND THEN THIS IS
3732  !                          CALLED A ALL BAD COLUMN AND THE TEMP
3733  !                          BLAST TABLE IS LOADED WITH THIS COLUMN
3734  !                          NUMBER.  THE COLUMN COUNT TABLE AT THIS
3735  !                          ALL BAD COLUMN NUMBER IS CLEARED.
3736  !
3737  !  FORMAL PARAMETERS:
3738  !                  BAD_CHIP:
3739  !                          POINTS A FAILING CHIP WHICH IS PRESENTLY BEING
3740  !                          PM'ED.
3741  !                  ALL_BAD:
3742  !                          KEEPS COUNT OF HOW MANY ALL BAD ROW AND OR COLUMNS
3743  !                          ARE FOUND FOR THIS CHIP.  IF THIS COUNT COUNT
3744  !                          EXCEEDS 10 THEN THIS CHIP IS CALLED A ALL BAD
3745  !                          CHIP.
3746  !
3747  !--
3748
3749      incr COL_NUM from 0 to .MAX_CHIP_COL - 1 do          !SEARCH THRU THE COLUMN TABLE
3750          begin
3751
3752          if .COL_CNT_TBL [.COL_NUM] gtr 10        .IS THIS LOCATIONS COUNT > 10
3753          then
3754              begin
3755              ALL_BAD - .ALL_BAD + 1;              !INCREMENT THE ALL BAD ROW/COL COUNT
3756
3757              if .ALL_BAD gtr 9                    !IS THE ALL BAD COUNT > 9
3758              then
3759                  return .ALL_BAD                  !EXIT AND RETURN ALL BAD COUNT
3760              else
3761                  begin
3762                  TMP_BLST_TBL [.ALL_BAD, R_C] = CLR_FLG; !INDICATE THAT THIS IS A BAD COLUMN
3763                  TMP_BLST_TBL [.ALL_BAD, R_C_NO] - .COL_NUM;     !LOAD THE BAD COLUMN NUMBER
3764                  TMP_BLST_TBL [.ALL_BAD, NIB_NO] - .BAD_CHIP/4;  .LOAD THE FAILING NIBBLE POSITION
3765                  COL_CNT_TBL [.COL_NUM] - ZEROES;         .CLEAR THE TABLE OF THIS BAD COLUMN
3766                  end;
3767
3768              end;
3769
```

```
;         3770          end;                                                                    :
;         3771
;         3772      return .ALL_BAD;                .EXIT AND RETURN THE NUMBER OF ALL BAD ROWS/COLS
;         3773      end;


                                        .SBTTL  S.COL.CNT.TBL ROUTINE DECLARATIONS
011620  004167  000000G           S.COL.CNT.TBL:
                                        JSR     R1,$SAVE2                       ;                               3718
011624  005002                            CLR     R2                              ; COL.NUM                       3749
011626  000453                            BR      4$
011630  126227  000000' 000012    1$:     CMPB    COL.CNT.TBL(R2),#12             ; *(COL.NUM),*                  3752
011636  101446                            BLOS    3$
011640  005266  000010                    INC     10(SP)                          ; ALL.BAD                       3755
011644  016601  000010                    MOV     10(SP),R1                       ; ALL.BAD,*                     3757
011650  020127  000011                    CMP     R1,#11
011654  003402                            BLE     2$
011656  010100                            MOV     R1,R0                           ;                               3759
011660  000207                            RTS     PC
011662  006301                    2$:     ASL     R1                              ;                               3762
011664  062701  026744'                   ADD     #TMP.BLST.TBL,R1
011670  042711  100000                    BIC     #100000,(R1)
011674  010200                            MOV     R2,R0                           ; COL.NUM,*                     3763
011676  006300                            ASL     R0
011700  006300                            ASL     R0
011702  006300                            ASL     R0
011704  006300                            ASL     R0
011706  042700  170017                    BIC     #170017,R0
011712  042711  007760                    BIC     #7760,(R1)
011716  050011                            BIS     R0,(R1)
011720  016646  000012                    MOV     12(SP),-(SP)                    ; BAD.CHIP,*                    3764
011724  012746  000004                    MOV     #4,-(SP)
011730  004767  000000G                   JSR     PC,BL$DIV
011734  042700  177760                    BIC     #177760,R0
011740  142711  000017                    BICB    #17,(R1)
011744  150011                            BISB    R0,(R1)
011746  105062  000000'                   CLRB    COL.CNT.TBL(R2)                 ; *(COL.NUM)                    3765
011752  022626                            CMP     (SP)+,(SP)+                     ;                               3761
011754  005202                    3$:     INC     R2                              ; COL.NUM                       3749
011756  020267  036774'           4$:     CMP     R2,MAX.CHIP.COL                 ; COL.NUM,*
011762  002722                            BLT     1$
011764  016600  000010                    MOV     10(SP),R0                       ; ALL.BAD,*                     3719
011770  000207                            RTS     PC                              ;                               3718

; Routine Size:  53 words
; Maximum stack depth per invocation:  5 words
```

```
;        3774  routine S_ERROR_MAP (BAD_CHIP) -
;        3775      begin
;        3776
;        3777  !++
;        3778  !  FUNCTIONAL DESCRIPTION : SEARCH THE ERROR MAP
;        3779  !
;        3780  !                           THE ERROR MAP IS LOADED WITH ALL THE
;        3781  !                           FAILING ROW NUMBERS AT THIER FAILING
;        3782  !                           SECTOR NUMBERS.
;        3783  !
;        3784  !                           THIS ROUTINE SEARCHES THE ERROR MAP FOR
;        3785  !                           OCCURANCES OF BAD ROWS.
;        3786  !
;        3787  !                           WHEN A BAD ROW IS FOUND A BAD ROW COUNT
;        3788  !                           IS INCREMENTED AND A BAD COLUMN COUNT
;        3789  !                           AT THE ADJACENT FAILING COLUMN IS ALSO
;        3790  !                           INCREMENTED.
;        3791  !
;        3792  !                           IF THE ROW COUNT EXCEEDS 10 THEN THIS IS
;        3793  !                           CALLED A ALL BAD ROW AND THE ROW NUMBER IS
;        3794  !                           LOADED INTO THE TEMP BLAST TABLE.  THE ADJACENT
;        3795  !                           FAILING COLUMN COUNTS IS DECREMENTED BY ONE
;        3796  !
;        3797  !                           IF AFTER SEARCHING THE ERROR MAP THE BAD ROW
;        3798  !                           DOES NOT EXCEED 10 THEN THE PROGRAM GOES ON TO
;        3799  !                           THE NEXT ROW AND THE FAILING COLUMN COUNT ARE
;        3800  !                           LEFT ALONE.
;        3801  !
;        3802  !  FORMAL PARAMETERS:
;        3803  !                   BAD_CHIP:
;        3804  !                           POINTS TO A FAILING CHIP PRESENTLY BEING
;        3805  !                           PM'ED.
;        3806  !--
;        3807
;        3808      label
;        3809          A;                                          !LEAVE LOOP LABEL
;        3810
;        3811      local
;        3812          WRD_INDEX,                                  !STORES THE ERROR MAP WORD WHERE THIS ROW RESIDES
;        3813          BIT_INDEX,                                  !STORES THE BIT IN THE ERROR MAP WORD WHERE THIS BIT RESIDES
;        3814          ROW_CNT,                                    !COUNT OF HOW MANY BAD ROW FOUND
;        3815          PTR,                                        !POINTER INTO TO THE COLUMN POINTER TABLE
;        3816          COL_INDEX,                                  !STORES THE CALCULATED ADJACENT FAILING COLUMN NUMBER
;        3817          QD_PAIR_LOOP,                               !SELECTS HOW MANY QUADRANTS TO TEST
;        3818          QD_NUM_SEARCH,                              !SELECTS HOW MANY QUADRANTS TO TEST
;        3819          QD_OFFSET,                                  !OFF SET USE IN CALCULATING WHICH QUADRANT TO TEST
;        3820          START_SEC,                                  !STARTING SECTOR VARIABLE
;        3821          END_SEC,                                    !ENDING SECTOR VARIABLE
;        3822          ALL_BAD;                                    !STORES HOW MANY ALL BAD CHIPS ARE FOUND
;        3823
;        3824      ALL_BAD = -1;                                   !START ALL BAD AT -1 THIS VALUE GETS RETURNED
;        3825      I_COL_CNT_TBL ();                               !INIT THE COLUMN COUNT TABLE
```

```
3826
3827        if .BNK_NUM_SEC eql 127                     .IS THIS A 16K CHIP
3828        then
3829            begin
3830            QD_PAIR_LOOP = 0;                       .16K CHIPS ONLY HAVE ONE QUADRANT TO TEST
3831            QD_NUM_SEARCH = 0;
3832            end
3833        else
3834            begin
3835            QD_PAIR_LOOP = 1;                       .64K CHIPS HAVE FOUR QUADRANTS TO TEST
3836            QD_NUM_SEARCH = 1;
3837            end;
3838
3839        QD_OFFSET = -2;                             .START THE OFFSET OFF AT -2
3840
3841        incr QD_LOOP from 0 to .QD_PAIR_LOOP do     .LOOP ON TWO QUADRANT PAIRS
3842            begin
3843            QD_OFFSET = .QD_OFFSET + 2;             .UP THE OFFSET BY TWO
3844
3845            incr ROW_NUM from 0 to 127 do           .SEARCH ALL WORDS IN THIS SECTOR
3846                begin
3847                WRD_INDEX = .ROW_NUM/16;            !CALCULATE WHICH WORD THIS ROW IS IN
3848                BIT_INDEX = .ROW_NUM mod 16;        .CALCULATE WHICH BIT IN WORD THIS ROW IS IN
3849    A :
3850                begin
3851
3852                incr QD_SEARCH from 0 + .QD_OFFSET to .QD_NUM_SEARCH + .QD_OFFSET do
3853                                                    .SEARCH BOTH QUADRANTS IN THIS PAIR
3854                    begin
3855
3856                    case .QD_SEARCH from 0 to 3 of  .WHICH QUADRANT ARE WE IN
3857                        set
3858
3859                        [0] :                       .WE ARE IN THE FIRST QUADRANT OR JUST A 16K PART
3860                            begin
3861                            ROW_CNT = ZERO;         .ROW STARTS AT 0 FOR THE FIRST QUAD SEARCH
3862                            PTR = -1;               .THE POINTER STARTS AT -1
3863                            I_COL_PTR ();           !INIT THE COLUMN POINTER TABLE
3864                            START_SEC = 0;          .QUAD 0 SECTORS START AT 0
3865                            END_SEC = 127;          !QUAD 0 END SECTOR ENDS AT 127
3866                            end;
3867
3868                        [1] :                       .WE ARE IN QUADRANT 1
3869                            begin
3870                            START_SEC = 256;        .QUAD 1 SECTORS START AT 256
3871                            END_SEC = 383;          .QUAD 1 SECTORS END AT 383
3872                            end;
3873
3874                        [2] :                       .WE ARE IN QUADRANT 2
3875                            begin
3876                            ROW_CNT = ZERO;         !START THE ROW COUNT AT ZERO
3877                            PTR = -1;               .RESET THE POINTER
```

```
3878                        I_COL_PTR ();               .INIT THE COLUMN POINTER TABLE
3879                        START_SEC = 128;            !QUAD 2 SECTORS START AT 128
3880                        END_SEC = 255;              .QUAD 2 SECTORS END AT 255
3881                        end;
3882
3883                    [3] :                           .WE ARE IN QUAD 3
3884                        begin
3885                        START_SEC = 384;            !QUAD 3 SECTORS START AT 384
3886                        END_SEC = 511;              .QUAD 3 SECTORS END AT 511
3887                        end;
3888                    tes;
3889
3890                incr SEC_NUM from .START_SEC to .END_SEC do     !SEARCH THRU ALL SECTORS FOR THIS QUAD
3891                    begin
3892
3893                    if .ERROR_MAP [.SEC_NUM, .WRD_INDEX, .BIT_INDEX, ONE, ZERO] eql 1   !IS THIS ROW BIT SET
3894                    then
3895                        begin
3896                        ROW_CNT = .ROW_CNT + 1; .UP THE BAD ROW COUNT
3897
3898                        if .ROW_CNT gtr 9        .IS THERE > THEN 10 BAD ROWS
3899                        then
3900                            begin
3901                            ALL_BAD = .ALL_BAD + 1;      !UP THE ALL BAD COUNT
3902
3903                            if .ALL_BAD gtr 9    .IS THERE > 10 ALL BAD ROWS/COLUMNS
3904                            then
3905                                return .ALL_BAD .EXIT AND RETURN THE THE COUNT
3906                            else
3907                                begin
3908                                TMP_BLST_TBL [.ALL_BAD, R_C] = SET_FLG; !THIS IS A BAD ROW
3909
3910                            selectone .SEC_NUM of    .FIND WHICH ROW NUMBER THIS IS
3911                                set
3912
3913                                [0 to 127, 256 to 383] :
3914                                    TMP_BLST_TBL [.ALL_BAD, R_C_NO] = .ROW_NUM;
3915
3916                                [128 to 255, 384 to 511] :
3917                                    TMP_BLST_TBL [.ALL_BAD, R_C_NO] = .ROW_NUM + 128;
3918                                tes;
3919
3920                            TMP_BLST_TBL [.ALL_BAD, NIB_NO] = .BAD_CHIP/4;  !FIND WHICH NIBBLE ITS IN
3921
3922                            incr index from 0 to .PTR do    !DECREMENT THE COLUMN COUNTS
3923                                COL_CNT_TBL [.COL_PTR [.index]] = .COL_CNT_TBL [.COL_PTR [.index]] - 1;
3924
3925                            if .QD_PAIR_LOOP eql 0  .CLEAR THE QUAD OF THIS ROWW
3926                            then
3927
3928                                incr CLR_SEC_ROW from 0 to 127 do
3929                                    ERROR_MAP [.CLR_SEC_ROW, .WRD_INDEX, .BIT_INDEX, ONE, ZERO] = CLR_FLG
```

```
3930
3931                                        else
3932
3933                                            if .QD_SEARCH leq 1
3934                                            then
3935                                                begin
3936
3937                                                incr CLR_SEC_ROW from 0 to 127 do
3938                                                    ERROR_MAP [.CLR_SEC_ROW, .WRD_INDEX, .BIT_INDEX, ONE, ZERO] -
3939                                                    CLR_FLG;
3940
3941                                                incr CLR_SEC_ROW from 256 to 383 do
3942                                                    ERROR_MAP [.CLR_SEC_ROW, .WRD_INDEX, .BIT_INDEX, ONE, ZERO]
3943                                                    CLR_FLG;
3944
3945                                                end
3946                                            else
3947                                                begin
3948
3949                                                incr CLR_SEC_ROW from 128 to 255 do
3950                                                    ERROR_MAP [.CLR_SEC_ROW, .WRD_INDEX, .BIT_INDEX, ONE, ZERO] =
3951                                                    CLR_FLG;
3952
3953                                                incr CLR_SEC_ROW from 384 to 511 do
3954                                                    ERROR_MAP [.CLR_SEC_ROW, .WRD_INDEX, .BIT_INDEX, ONE, ZERO] =
3955                                                    CLR_FLG;
3956
3957                                                end;
3958
3959                                         eave A;        !LEAVE THE LOOP AND DO THE NEXT ROW NUMBER
3960                                        end;
3961
3962                                            end
3963                                    else
3964                                        begin
3965
3966                                        selectone .SEC_NUM of       .INCREMENT THE ADJACENT COLUMN COUNT TABLE
3967                                            set
3968
3969                                            [0 to 127] :
3970                                                begin
3971                                                COL_INDEX = .ROW_NUM + .SEC_NUM;
3972
3973                                                if .COL_INDEX geq 128 then COL_INDEX = .COL_INDEX - 128;
3974
3975                                                end;
3976
3977                                            [128 to 255] :
3978                                                begin
3979                                                COL_INDEX = .ROW_NUM + (.SEC_NUM - 128);
3980
3981                                                if .COL_INDEX geq 128 then COL_INDEX = .COL_INDEX - 128;
```

```
;          3982
;          3983                                        end;
;          3984
;          3985                                    [256 to 383] :
;          3986                                        begin
;          3987                                        COL_INDEX = .ROW_NUM + (.SEC_NUM - 128);
;          3988
;          3989                                        if .COL_INDEX geq 256 then COL_INDEX = .COL_INDEX - 128;
;          3990
;          3991                                        end;
;          3992
;          3993                                    [384 to 511] :
;          3994                                        begin
;          3995                                        COL_INDEX = .ROW_NUM + (.SEC_NUM - 256);
;          3996
;          3997                                        if .COL_INDEX geq 256 then COL_INDEX = .COL_INDEX - 128;
;          3998
;          3999                                        end;
;          4000                                    tes;
;          4001
;          4002                                COL_CNT_TBL [.COL_INDEX] = .COL_CNT_TBL [.COL_INDEX] + 1;
;          4003                                PTR = .PTR + 1;
;          4004                                COL_PTR [.PTR] = .COL_INDEX;
;          4005                                end;
;          4006
;          4007                            end;
;          4008
;          4009                        end;
;          4010
;          4011                    end;
;          4012
;          4013                end;
;          4014                end;
;          4015
;          4016            end;
;          4017
;          4018        return .ALL_BAD;
;          4019        end;


                                        .SBTTL  S.ERROR.MAP ROUTINE DECLARATIONS
011772  004167  000000G           S.ERROR.MAP:
                                    JSR     R1,$SAVE5                   ;                                   3774
011776  162706  000032                SUB     #32,SP
012002  012766  177777  000010        MOV     #-1,10(SP)                  ; *,ALL.BAD                     3824
012010  004767  172440                JSR     PC,I.COL.CNT.TBL            ;                               3825
012014  026727  037014' 000177        CMP     BNK.NUM.SEC,#177            ;                               3827
012022  001005                        BNE     1$
012024  005066  000022                CLR     22(SP)                      ; QD.PAIR.LOOP                  3830
012030  005066  000030                CLR     30(SP)                      ; QD.NUM.SEARCH                 3831
012034  000406                        BR      2$                          ;                               3827
012036  012766  000001  000022    1$: MOV     #1,22(SP)                   ; *,QD.PAIR.LOOP                3835
```

```
012044  012766  000001  000030          MOV     #1,30(SP)                       ; *,QD.NUM.SEARCH                       3836
012052  012766  177776  000020  2$:     MOV     #-2,20(SP)                      ; *,QD.OFFSET                           3839
012060  005066  000026                  CLR     26(SP)                          ; QD.LOOP                               3841
012064  000167  001502                  JMP     43$
012070  062766  000002  000020  3$:     ADD     #2,20(SP)                       ; *,QD.OFFSET                           3843
012076  016666  000030  000016          MOV     30(SP),16(SP)                   ; QD.NUM.SEARCH,*                       3842
012104  066666  000020  000016          ADD     20(SP),16(SP)                   ; QD.OFFSET,*
012112  005003                          CLR     R3                              ; ROW.NUM                               3845
012114  010346                  4$:     MOV     R3,-(SP)                        ; ROW.NUM,*                             3847
012116  012746  000020                  MOV     #20,-(SP)
012122  004767  000000G                 JSR     PC,BL$DIV
012126  010066  000012                  MOV     R0,12(SP)                       ; *,WRD.INDEX
012132  010316                          MOV     R3,(SP)                         ; ROW.NUM,*                             3848
012134  012746  000020                  MOV     #20,-(SP)
012140  004767  000000G                 JSR     PC,BL$MOD
012144  010066  000012                  MOV     R0,12(SP)                       ; *,BIT.INDEX
012150  016666  000026  000006          MOV     26(SP),6(SP)                    ; QD.OFFSET,QD.SEARCH                   3852
012156  005366  000006                  DEC     6(SP)                           ; QD.SEARCH
012162  000167  001340                  JMP     40$
012166  016605  000006          5$:     MOV     6(SP),R5                        ; QD.SEARCH,*                           3856
012172  006305                          ASL     R5
012174  066507  012200'                 ADD     6$(R5),PC
012200  000010          6$:     .WORD   7$-6$
012202  000042                  .WORD   8$-6$
012204  000060                  .WORD   9$-6$
012206  000114                  .WORD   10$-6$
012210  005066  000032          7$:     CLR     32(SP)                          ; ROW.CNT                               3861
012214  012766  177777  000010          MOV     #-1,10(SP)                      ; *,PTR                                 3862
012222  004767  172302                  JSR     PC,I.COL.PTR                    ;                                       3863
012226  005066  000022                  CLR     22(SP)                          ; START.SEC                             3864
012232  012766  000177  000020          MOV     #177,20(SP)                     ; *,END.SEC                             3865
012240  000433                          BR      11$                             ;                                       3856
012242  012766  000400  000022  8$:     MOV     #400,22(SP)                     ; *,START.SEC                           3870
012250  012766  000577  000020          MOV     #577,20(SP)                     ; *,END.SEC                             3871
012256  000424                          BR      11$                             ;                                       3856
012260  005066  000032          9$:     CLR     32(SP)                          ; ROW.CNT                               3876
012264  012766  177777  000010          MOV     #-1,10(SP)                      ; *,PTR                                 3877
012272  004767  172232                  JSR     PC,I.COL.PTR                    ;                                       3878
012276  012766  000200  000022          MOV     #200,22(SP)                     ; *,START.SEC                           3879
012304  012766  000377  000020          MOV     #377,20(SP)                     ; *,END.SEC                             3880
012312  000406                          BR      11$                             ;                                       3856
012314  012766  000600  000022  10$:    MOV     #600,22(SP)                     ; *,START.SEC                           3885
012322  012766  000777  000020          MOV     #777,20(SP)                     ; *,END.SEC                             3886
012330  016605  000022          11$:    MOV     22(SP),R5                       ; START.SEC,SEC.NUM                     3890
012334  005305                          DEC     R5                              ; SEC.NUM
012336  000167  001150          12$:    JMP     39$
012342  010500          13$:    MOV     R5,R0                           ; SEC.NUM,*                             3893
012344  006300                          ASL     R0
012346  006300                          ASL     R0
012350  006300                          ASL     R0
012352  066600  000014                  ADD     14(SP),R0                       ; WRD.INDEX,*
012356  006300                          ASL     R0
```

```
012360  062700  006626'                    ADD     #ERROR.MAP,R0
012364  010046                             MOV     R0,-(SP)
012366  016646  000014                     MOV     14(SP),-(SP)        ; BIT.INDEX,*
012372  012746  000001                     MOV     #1,-(SP)
012376  005046                             CLR     -(SP)
012400  004767  000000G                    JSR     PC,BL$GT2
012404  062706  000010                     ADD     #10,SP
012410  005300                             DEC     R0
012412  001351                             BNE     12$
012414  005266  000032                     INC     32(SP)              ; ROW.CNT                   3896
012420  026627  000032  000011             CMP     32(SP),#11          ; ROW.CNT,*                 3898
012426  003002                             BGT     14$
012430  000167  000654                     JMP     33$
012434  005266  000016          14$:       INC     16(SP)              ; ALL.BAD                   3901
012440  026627  000016  000011             CMP     16(SP),#11          ; ALL.BAD,*                 3903
012446  003404                             BLE     15$
012450  062706  000006                     ADD     #6,SP               ;                           3905
012454  000167  001126                     JMP     44$
012460  016604  000016          15$:       MOV     16(SP),R4           ; ALL.BAD,*                 3908
012464  006304                             ASL     R4
012466  062704  026744'                    ADD     #TMP.BLST.TBL,R4
012472  052714  100000                     BIS     #100000,(R4)
012476  005705                             TST     R5                  ; SEC.NUM                   3910
012500  002403                             BLT     16$
012502  020527  000177                     CMP     R5,#177             ; SEC.NUM,*
012506  003406                             BLE     17$
012510  020527  000400          16$:       CMP     R5,#400             ; SEC.NUM,*
012514  002405                             BLT     18$
012516  020527  000577                     CMP     R5,#577             ; SEC.NUM,*
012522  003002                             BGT     18$
012524  010302                  17$:       MOV     R3,R2               ; ROW.NUM,*                 3914
012526  000417                             BR      21$
012530  020527  000200          18$:       CMP     R5,#200             ; SEC.NUM,*                 3910
012534  002403                             BLT     19$
012536  020527  000377                     CMP     R5,#377             ; SEC.NUM,*
012542  003406                             BLE     20$
012544  020527  000600          19$:       CMP     R5,#600             ; SEC.NUM,*
012550  002417                             BLT     22$
012552  020527  000777                     CMP     R5,#777             ; SEC.NUM,*
012556  003014                             BGT     22$
012560  010302                  20$:       MOV     R3,R2               ; ROW.NUM,*                 3917
012562  062702  000200                     ADD     #200,R2
012566  006302                  21$:       ASL     R2
012570  006302                             ASL     R2
012572  006302                             ASL     R2
012574  006302                             ASL     R2
012576  042702  170017                     BIC     #170017,R2
012602  042714  007760                     BIC     #7760,(R4)
012606  050214                             BIS     R2,(R4)
012610  016646  000056          22$:       MOV     56(SP),-(SP)        ; BAD.CHIP,*                3920
012614  012746  000004                     MOV     #4,-(SP)
012620  004767  000000G                    JSR     PC,BL$DIV
```

```
012624  042700  177760                  BIC     #177760,R0
012630  142714  000017                  BICB    #17,(R4)
012634  150014                          BISB    R0,(R4)
012636  005002                          CLR     R2                      ; INDEX                 3922
012640  000412                          BR      24$                                             3923
012642  005004                  23$:    CLR     R4                      ;
012644  156204  001220'                BISB    COL.PTR(R2),R4          ; *(INDEX),*
012650  005000                          CLR     R0
012652  156400  000000'                BISB    COL.CNT.TBL(R4),R0
012656  005300                          DEC     R0
012660  110064  000000'                MOVB    R0,COL.CNT.TBL(R4)
012664  005202                          INC     R2                      ; INDEX                 3922
012666  020266  000014          24$:    CMP     R2,14(SP)               ; INDEX,PTR
012672  003763                          BLE     23$
012674  005766  000034                  TST     34(SP)                  ; QD.PAIR.LOOP          3925
012700  001031                          BNE     26$
012702  005004                          CLR     R4                      ; CLR.SEC.ROW           3928
012704  010400                  25$:    MOV     R4,R0                   ; CLR.SEC.ROW,*         3929
012706  006300                          ASL     R0
012710  006300                          ASL     R0
012712  006300                          ASL     R0
012714  066600  000020                  ADD     20(SP),R0               ; WRD.INDEX,*
012720  006300                          ASL     R0
012722  062700  006626'                ADD     #ERROR.MAP,R0
012726  010046                          MOV     R0,-(SP)
012730  016646  000020                  MOV     20(SP),-(SP)            ; BIT.INDEX,*
012734  012746  000001                  MOV     #1,-(SP)
012740  005046                          CLR     -(SP)
012742  004767  000000G                 JSR     PC,BL$PU2
012746  062706  000010                  ADD     #10,SP
012752  005204                          INC     R4                      ; CLR.SEC.ROW           3928
012754  020427  000177                  CMP     R4,#177                 ; CLR.SEC.ROW,*
012760  003751                          BLE     25$
012762  000550                          BR      32$                     ;                       3925
012764  026627  000012  000001  26$:    CMP     12(SP),#1               ; QD.SEARCH,*           3933
012772  003062                          BGT     29$
012774  005004                          CLR     R4                      ; CLR.SEC.ROW           3937
012776  010400                  27$:    MOV     R4,R0                   ; CLR.SEC.ROW,*         3938
013000  006300                          ASL     R0
013002  006300                          ASL     R0
013004  006300                          ASL     R0
013006  066600  000020                  ADD     20(SP),R0               ; WRD.INDEX,*
013012  006300                          ASL     R0
013014  062700  006626'                ADD     #ERROR.MAP,R0
013020  010046                          MOV     R0,-(SP)
013022  016646  000020                  MOV     20(SP),-(SP)            ; BIT.INDEX,*
013026  012746  000001                  MOV     #1,-(SP)
013032  005046                          CLR     -(SP)
013034  004767  000000G                 JSR     PC,BL$PU2
013040  062706  000010                  ADD     #10,SP
013044  005204                          INC     R4                      ; CLR.SEC.ROW           3937
013046  020427  000177                  CMP     R4,#177                 ; CLR.SEC.ROW,*
```

```
013052  003751                          BLE     27$
013054  012704  000400                  MOV     #400,R4                 ; *,CLR.SEC.ROW                 3941
013060  010400                  28$:    MOV     R4,R0                   ; CLR.SEC.ROW,*                 3942
013062  006300                          ASL     R0
013064  006300                          ASL     R0
013066  006300                          ASL     R0
013070  066600  000020                  ADD     20(SP),R0               ; WRD.INDEX,*
013074  006300                          ASL     R0
013076  062700  006626'                ADD     #ERROR.MAP,R0
013102  010046                          MOV     R0,-(SP)
013104  016646  000020                  MOV     20(SP),-(SP)            ; BIT.INDEX,*
013110  012746  000001                  MOV     #1,-(SP)
013114  005046                          CLR     -(SP)
013116  004767  000000G                 JSR     PC,BL$PU2
013122  062706  000010                  ADD     #10,SP
013126  005204                          INC     R4                      ; CLR.SEC.ROW                   3941
013130  020427  000577                  CMP     R4,#577                 ; CLR.SEC.ROW,*
013134  003751                          BLE     28$
013136  000462                          BR      32$                     ;                               3933
013140  012704  000200          29$:    MOV     #200,R4                 ; *,CLR.SEC.ROW                 3949
013144  010400                  30$:    MOV     R4,R0                   ; CLR.SEC.ROW,*                 3950
013146  006300                          ASL     R0
013150  006300                          ASL     R0
013152  006300                          ASL     R0
013154  066600  000020                  ADD     20(SP),R0               ; WRD.INDEX,*
013160  006300                          ASL     R0
013162  062700  006626'                ADD     #ERROR.MAP,R0
013166  010046                          MOV     R0,-(SP)
013170  016646  000020                  MOV     20(SP),-(SP)            ; BIT.INDEX,*
013174  012746  000001                  MOV     #1,-(SP)
013200  005046                          CLR     -(SP)
013202  004767  000000G                 JSR     PC,BL$PU2
013206  062706  000010                  ADD     #10,SP
013212  005204                          INC     R4                      ; CLR.SEC.ROW                   3949
013214  020427  000377                  CMP     R4,#377                 ; CLR.SEC.ROW,*
013220  003751                          BLE     30$
013222  012704  000600                  MOV     #600,R4                 ; *,CLR.SFC.ROW                 3953
013226  010400                  31$:    MOV     R4,R0                   ; CLR.SEC.ROW,*                 3954
013230  006300                          ASL     R0
013232  006300                          ASL     R0
013234  006300                          ASL     R0
013236  066600  000020                  ADD     20(SP),R0               ; WRD.INDEX,*
013242  006300                          ASL     R0
013244  062700  006626'                ADD     #ERROR.MAP,R0
013250  010046                          MOV     R0,-(SP)
013252  016646  000020                  MOV     20(SP),-(SP)            ; BIT.INDEX,*
013256  012746  000001                  MOV     #1,-(SP)
013262  005046                          CLR     -(SP)
013264  004767  000000G                 JSR     PC,BL$PU2
013270  062706  000010                  ADD     #10,SP
013274  005204                          INC     R4                      ; CLR.SEC.ROW                   3953
013276  020427  000777                  CMP     R4,#777                 ; CLR.SEC.ROW,*
```

```
013302  003751                          BLE     31$
013304  022626                  32$:    CMP     (SP)+,(SP)+                     ;                               3959
013306  000517                          BR      41$
013310  005705                  33$:    TST     R5                              ; SEC.NUM                       3966
013312  002411                          BLT     34$
013314  020527  000177                  CMP     R5,#177                         ; SEC.NUM,*
013320  003006                          BGT     34$
013322  010501                          MOV     R5,R1                           ; SEC.NUM,COL.INDEX             3971
013324  060301                          ADD     R3,R1                           ; ROW.NUM,COL.INDEX
013326  020127  000200                  CMP     R1,#200                         ; COL.INDEX,*                   3973
013332  002457                          BLT     38$
013334  000454                          BR      37$
013336  020527  000200          34$:    CMP     R5,#200                         ; SEC.NUM,*                     3966
013342  002414                          BLT     35$
013344  020527  000377                  CMP     R5,#377                         ; SEC.NUM,*
013350  003011                          BGT     35$
013352  010504                          MOV     R5,R4                           ; SEC.NUM,*                     3979
013354  060304                          ADD     R3,R4                           ; ROW.NUM,*
013356  010401                          MOV     R4,R1                           ; *,COL.INDEX
013360  162701  000200                  SUB     #200,R1                         ; *,COL.INDEX
013364  020127  000200                  CMP     R1,#200                         ; COL.INDEX,*                   3981
013370  002440                          BLT     38$
013372  000435                          BR      37$
013374  020527  000400          35$:    CMP     R5,#400                         ; SEC.NUM,*                     3966
013400  002414                          BLT     36$
013402  020527  000577                  CMP     R5,#577                         ; SEC.NUM,*
013406  003011                          BGT     36$
013410  010504                          MOV     R5,R4                           ; SEC.NUM,*                     3987
013412  060304                          ADD     R3,R4                           ; ROW.NUM,*
013414  010401                          MOV     R4,R1                           ; *,COL.INDEX
013416  162701  000200                  SUB     #200,R1                         ; *,COL.INDEX
013422  020127  000400                  CMP     R1,#400                         ; COL.INDEX,*                   3989
013426  002421                          BLT     38$
013430  000416                          BR      37$
013432  020527  000600          36$:    CMP     R5,#600                         ; SEC.NUM,*                     3966
013436  002415                          BLT     38$
013440  020527  000777                  CMP     R5,#777                         ; SEC.NUM,*
013444  003012                          BGT     38$
013446  010504                          MOV     R5,R4                           ; SEC.NUM,*                     3995
013450  060304                          ADD     R3,R4                           ; ROW.NUM,*
013452  010401                          MOV     R4,R1                           ; *,COL.INDEX
013454  162701  000400                  SUB     #400,R1                         ; *,COL.INDEX
013460  020127  000400                  CMP     R1,#400                         ; COL.INDEX,*                   3997
013464  002402                          BLT     38$
013466  162701  000200          37$:    SUB     #200,R1                         ; *,COL.INDEX
013472  105261  000000'        38$:    INCB    COL.CNT.TBL(R1)                 ; *(COL.INDEX)                  4002
013476  005266  000010                  INC     10(SP)                          ; PTR                           4003
013502  016604  000010                  MOV     10(SP),R4                       ; PTR,*                         4004
013506  110164  001220'                  MOVB    R1,COL.PTR(R4)                  ; COL.INDEX,*
013512  005205                  39$:    INC     R5                              ; SEC.NUM                       3890
013514  020566  000020                  CMP     R5,20(SP)                       ; SEC.NUM,END.SEC
013520  003002                          BGT     40$
```

```
013522  000167  176614                      JMP     13$
013526  005266  000006                40$:  INC     6(SP)                   ; QD.SEARCH                     3852
013532  026666  000006  000024            CMP     6(SP),24(SP)            ; QD.SEARCH,*
013540  003002                            BGT     41$
013542  000167  176420                      JMP     5$
013546  062706  000006                41$:  ADD     #6,SP                   ;                               3846
013552  005203                            INC     R3                      ; ROW.NUM                       3845
013554  020327  000177                      CMP     R3,#177                 ; ROW.NUM,*
013560  003002                            BGT     42$
013562  000167  176326                      JMP     4$
013566  005266  000026                42$:  INC     26(SP)                  ; QD.LOOP                       3841
013572  026666  000026  000022        43$:  CMP     26(SP),22(SP)           ; QD.LOOP,QD.PAIR.LOOP
013600  003002                            BGT     44$
013602  000167  176262                      JMP     3$
013606  016600  000010                44$:  MOV     10(SP),R0               ; ALL.BAD,*                     3775
013612  062706  000032                      ADD     #32,SP                  ;                               3774
013616  000207                            RTS     PC

; Routine Size:  459 words
; Maximum stack depth per invocation:  28 words
```

```
;         4020  routine S_CHIP_TBL (START) =
;         4021      begin
;         4022
;         4023  !++
;         4024  !   FUNCTIONAL DESCRIPTION: SEARCH THE BAD CHIP TABLE
;         4025  !
;         4026  !                           THIS ROUTINE SEARCHES THE BAD CHIP TABLE
;         4027  !                           LOOKING FOR FAILING CHIPS.  WHEN A FAILING
;         4028  !                           CHIP IS FOUND THE FAILING CHIP NUMBER IS
;         4029  !                           RETURNED.  IF NO FAILING CHIP IS FOUND
;         4030  !                           THEN A -1 IS RETURNED.
;         4031  !
;         4032  !   FORMAL PARAMETERS:
;         4033  !                   STRAT:
;         4034  !                           INDICATES WHERE TO START SEARCHING
;         4035  !                           IN THE BAD CHIP TABLE
;         4036  !--
;         4037
;         4038      incr TBL_INDEX from .START to 38 do         .SEARCH THRU THE CHIP TABLE
;         4039
;         4040          if .CHIP_TBL [.TBL_INDEX, FAULT] then return .TBL_INDEX;        !EXIT AND RETURN NDEX
;         4041
;         4042      return -1;                                  .EXIT AND RETURN -1 IF NO CHIP FAULTS ARE DETECTED
;         4043      end;


                                        .SBTTL  S.CHIP.TBL ROUTINE DECLARATIONS
013620  010146                  S.CHIP.TBL:
                                        MOV     R1,-(SP)                ;                               4020
013622  016600  000004                  MOV     4(SP),R0                ; START,TBL.INDEX               4038
013626  005300                          DEC     R0                      ; TBL.INDEX
013630  000405                          BR      2$
013632  010001                  1$:     MOV     R0,R1                   ; TBL.INDEX,*                   4040
013634  006301                          ASL     R1
013636  005761  026626'                TST     CHIP.TBL(R1)
013642  100406                          BMI     3$
013644  005200                  2$:     INC     R0                      ; TBL.INDEX                     4038
013646  020027  000046                CMP     R0,#46                  ; TBL.INDEX,*
013652  003767                          BLE     1$
013654  012700  177777                MOV     #-1,R0                  ;                               4021
013660  012601                  3$:     MOV     (SP)+,R1                ;                               4020
013662  000207                          RTS     PC

; Routine Size:  18 words
; Maximum stack depth per invocation:  2 words
```

```
4044  routine RD_OLD_PROM_DATA (ARR$BNK_SEL) : novalue =
4045      begin
4046
4047  .++
4048  .  FUNCTIONAL DESCRIPTION: READ THE OLD PROM DATA INTO THE ERROR MAP
4049  .
4050  .                          THIS ROUTINE WILL READ THE OLD PROM DATA
4051  .                          STORED IN PROM ON THE TESTED ARRAY INTO
4052  !                          THE ERROR MAP.
4053  .
4054  .  FORMAL PARAMETERS:
4055  !                  ARR$BNK_SEL:
4056  .                          STORES THE SELECTED ARRAYS ARRAY AND BANK SELECT
4057  !                          NUMBERS
4058  .--
4059
4060      local
4061          PROM_ADRS,                                  !STORES THE BUILT PROM ADDRESS
4062          FINISH;                                     .INCR LOOP ENDING VARIABLE
4063
4064      map
4065          ERROR_MAP : blockvector [4, 512] volatile;      .MAP THE ERROR_MAP THE SAME AS THE BLAST TABLE
4066
4067      if .BNK_NUM_SEC eql 127 then FINISH = 127 else FINISH - 255;        !SELECT THE LOOPS ENDING
4068
4069      RD_PROM_MODE;                                   .ENABLE PROM READS
4070      PROM_ADRS = ZEROES;                             !CLEAR THE PROM ADDRESS
4071      PROM_ADRS<11, 4>    .ARR$BNK_SEL<.ARR_SEL_POS, ARR$SEL_SIZE>;       !SELECT THE ARRAY
4072
4073      incr SEL_BNK from 0 to 3 do                     !READ PROM DATA FOR ALL FOUR BANKS ON THIS ARRAY
4074          begin
4075          PROM_ADRS<10, 1> - ZERO;                    !THE FIRST LOOP READS ROW PROM DATA
4076          PROM_ADRS<8, 2> = .SEL_BNK;                 !SELECT THE PROMS BANK POSITION
4077
4078          incr ROW_NUM from 0 to .FINISH do           .READ OUT ALL THE PROM ROW DATA
4079              begin
4080              PROM_ADRS<0, 8> = .ROW_NUM;             !SELECT THE ROW ADDRESS
4081              WRT_RH (MLPA, PA_REG, .PROM_ADRS);      !WRITE THE ADDRESS TO THE MLPA REG
4082              DELAY (ONE_US);                         !WAIT FOR PROM DATA TO CLOCK INTO MLPD
4083              ERROR_MAP [.SEL_BNK, .ROW_NUM, FULL_WRD] - .ML_ADDR [MLPD, PD_REG];
4084                                                      !PUT THE PROM DATA INTO THE ERROR MAP
4085              end;
4086
4087          PROM_ADRS<10, 1> - ONE;                     !THIS LOOP WILL READ OUT THE PROM COLUMN DATA
4088
4089          incr COL_NUM from 0 to .FINISH do           !READ OUT ALL THE PROM COLUMN DATA
4090              begin
4091              PROM_ADRS<0, 8>    .COL_NUM;            .SELECT THE COLUMN ADDRESS
4092              WRT_RH (MLPA, PA_REG, .PROM_ADRS);      !WRITE THE ADDRESS TO THE MLPA REG
4093              DELAY (ONE_US);                         !WAIT FOR PROM DATA TO CLOCK INTO MLPD
4094              ERROR_MAP [.SEL_BNK, .COL_NUM + .COL_BASE, FULL_WRD] = .ML_ADDR [MLPD, PD_REG];
4095                                                      !LOAD THE ERROR MAP WITH THE PROM DATA
```

```
;         4096              end;
;         4097
;         4098          end;
;         4099
;         4100      CLR_MBUS;                          .CLEAR OUT THE PROM READ MODE
;         4101      end;


                                        .GLOBL  L$DLY


                                        .SBTTL  RD.OLD.PROM.DAT ROUTINE DECLARATIONS
013664  004167  000000G          RD.OLD.PROM.DAT:
                                        JSR     R1,$SAVE5                       ;                       4044
013670  162706  000022                  SUB     #22,SP
013674  026727  037014' 000177          CMP     BNK.NUM.SEC,#177                ;                       4067
013702  001003                          BNE     1$
013704  012716  000177                  MOV     #177,(SP)                       ; *,FINISH
013710  000402                          BR      2$
013712  012716  000377          1$:     MOV     #377,(SP)                       ; *,FINISH
013716  016701  036772'          2$:     MOV     ML.ADDR,R1
013722  016166  000024  000016          MOV     24(R1),16(SP)                   ; *,ML.REG
013730  012700  000040                  MOV     #40,R0                          ; *,MLREG
013734  016701  036772'                  MOV     ML.ADDR,R1
013740  010061  000024                  MOV     R0,24(R1)                       ; MLREG,*
013744  005002                          CLR     R2                              ; PROM.ADRS             4070
013746  012746  000042                  MOV     #42,-(SP)                       ;                       4071
013752  060616                          ADD     SP,(SP)                         ; ARR$BNK.SEL,*
013754  016746  037010'                  MOV     ARR.SEL.POS,-(SP)
013760  012746  000004                  MOV     #4,-(SP)
013764  005046                          CLR     -(SP)
013766  004767  000000G                  JSR     PC,BL$GT2
013772  000300                          SWAB    R0
013774  006300                          ASL     R0
013776  006300                          ASL     R0
014000  006300                          ASL     R0
014002  042700  103777                  BIC     #103777,R0
014006  042702  074000                  BIC     #74000,R2                       ; *,PROM.ADRS
014012  050002                          BIS     R0,R2                           ; *,PROM.ADRS
014014  005005                          CLR     R5                              ; SEL.BNK               4073
014016  010504                  3$:     MOV     R5,R4                           ; SEL.BNK,*             4076
014020  000304                          SWAB    R4
014022  042704  176377                  BIC     #176377,R4
014026  042702  003400                  BIC     #3400,R2                        ; *,PROM.ADRS
014032  050402                          BIS     R4,R2                           ; *,PROM.ADRS
014034  010500                          MOV     R5,R0                           ; SEL.BNK,*             4083
014036  000300                          SWAB    R0
014040  105000                          CLRB    R0
014042  006300                          ASL     R0
014044  005004                          CLR     R4                              ; ROW.NUM               4078
014046  000444                          BR      9$
014050  105002                  4$:     CLRB    R2                              ; PROM.ADRS             4080
```

```
014052  150402                          BISB    R4,R2                   ; ROW.NUM,PROM.ADRS
014054  016701  036772'                 MOV     ML.ADDR,R1              ;                       4081
014060  016166  000020  000024          MOV     20(R1),24(SP)           ; *,ML.REG
014066  010203                          MOV     R2,R3                   ; PROM.ADRS,MLREG
014070  016701  036772'                 MOV     ML.ADDR,R1
014074  010361  000020                  MOV     R3,20(R1)               ; MLREG,*
014100  012703  000001                  MOV     #1,R3                   ; *,$$TMP2              4082
014104  001411                  5$:     BEQ     8$
014106  016701  000000G                 MOV     L$DLY,R1                ; *,$$TMP1
014112  001404                          BEQ     7$
014114  005066  000030          6$:     CLR     30(SP)                  ; $$TMP
014120  005301                          DEC     R1                      ; $$TMP1
014122  001374                          BNE     6$
014124  005303                  7$:     DEC     R3                      ; $$TMP2
014126  000766                          BR      5$
014130  010003                  8$:     MOV     R0,R3                   ;                       4083
014132  060403                          ADD     R4,R3                   ; ROW.NUM,*
014134  006303                          ASL     R3
014136  016701  036772'                 MOV     ML.ADDR,R1
014142  016166  000046  000022          MOV     46(R1),22(SP)           ; *,ML.REG
014150  016663  000022  006626'         MOV     22(SP),ERROR.MAP(R3)    ; ML.REG,*
014156  005204                          INC     R4                      ; ROW.NUM               4078
014160  020466  000010          9$:     CMP     R4,10(SP)               ; ROW.NUM,FINISH
014164  003731                          BLE     4$
014166  052702  002000                  BIS     #2000,R2                ; *,PROM.ADRS           4087
014172  005003                          CLR     R3                      ; COL.NUM               4089
014174  000446                          BR      15$
014176  105002                  10$:    CLRB    R2                      ; PROM.ADRS             4091
014200  150302                          BISB    R3,R2                   ; COL.NUM,PROM.ADRS
014202  016701  036772'                 MOV     ML.ADDR,R1              ;                       4092
014206  016166  000020  000020          MOV     20(R1),20(SP)           ; *,ML.REG
014214  010204                          MOV     R2,R4                   ; PROM.ADRS,MLREG
014216  016701  036772'                 MOV     ML.ADDR,R1
014222  010461  000020                  MOV     R4,20(R1)               ; MLREG,*
014226  012704  000001                  MOV     #1,R4                   ; *,$$TMP2              4093
014232  001411                  11$:    BEQ     14$
014234  016701  000000G                 MOV     L$DLY,R1                ; *,$$TMP1
014240  001404                          BEQ     13$
014242  005066  000030          12$:    CLR     30(SP)                  ; $$TMP
014246  005301                          DEC     R1                      ; $$TMP1
014250  001374                          BNE     12$
014252  005304                  13$:    DEC     R4                      ; $$TMP2
014254  000766                          BR      11$
014256  010304                  14$:    MOV     R3,R4                   ; COL.NUM,*             4094
014260  066704  036770'                 ADD     COL.BASE,R4
014264  060004                          ADD     R0,R4
014266  006304                          ASL     R4
014270  016701  036772'                 MOV     ML.ADDR,R1
014274  016166  000046  000016          MOV     46(R1),16(SP)           ; *,ML.REG
014302  016664  000016  006626'         MOV     16(SP),ERROR.MAP(R4)    ; ML.REG,*
014310  005203                          INC     R3                      ; COL.NUM               4089
014312  020366  000010          15$:    CMP     R3,10(SP)               ; COL.NUM,FINISH
```

```
014316  003727                          BLE     10$
014320  005205                          INC     R5                      ; SEL.BNK                       4073
014322  020527  000003                  CMP     R5,#3                   ; SEL.BNK,*
014326  003633                          BLE     3$
014330  016701  036772'                  MOV     ML.ADDR,R1              ;                               4098
014334  016166  000010  000014          MOV     10(R1),14(SP)           ; *,ML.REG
014342  016600  000014                  MOV     14(SP),R0               ; ML.REG,MLREG
014346  152700  000040                  BISB    #40,R0                  ; *,MLREG
014352  016701  036772'                  MOV     ML.ADDR,R1
014356  010061  000010                  MOV     R0,10(R1)               ; MLREG,*
014362  016701  036772'                  MOV     ML.ADDR,R1
014366  016166  000010  000012          MOV     10(R1),12(SP)           ; *,ML.REG
014374  016600  000012                  MOV     12(SP),R0               ; ML.REG,MLREG
014400  016705  036776'                  MOV     ML.DUT,R5
014404  042705  177770                  BIC     #177770,R5
014410  142700  000007                  BICB    #7,R0                   ; *,MLREG
014414  050500                          BIS     R5,R0                   ; *,MLREG
014416  016701  036772'                  MOV     ML.ADDR,R1
014422  010061  000010                  MOV     R0,10(R1)               ; MLREG,*
014426  062706  000032                  ADD     #32,SP                  ;                               4044
014432  000207                          RTS     PC

; Routine Size:  180 words
; Maximum stack depth per invocation:  19 words
```

```
;        4102  routine OR_OLD_NEW_PD (BNK_NUM) : novalue =
;        4103      begin
;        4104
;        4105  .++
;        4106  .   FUNCTIONAL DESCRIPTION: OR THE OLD PROM DATA STORED IN THE ERROR MAP
;        4107  .                           WITH THE NEW PROM DATA STORED IN THE BLAST TABLE
;        4108  !
;        4109  .                                ONCE THE NIBBLE OFFSET HAVE COUNTED AND
;        4110  .                                THERE DOES NOT EXIST OFFSET > 14 THEN
;        4111  .                                OLD PROM DATA STORED IN THE ERROR MAP
;        4112  .                                AT THIS BANK IS OR'ED INTO THE BLAST
;        4113  .                                TABLE WHERE THEN THE CHECK SUMS CAN
;        4114  .                                BE CALCULATED.
;        4115  !   FORMAL PARAMETERS:
;        4116  !                       ARR$BNK_SEL:
;        4117  .                              STORES THE SELECTED ARRAYS ARRAY AND BANK SELECT
;        4118  .                              NUMBERS.
;        4119  .--
;        4120
;        4121      map
;        4122          ERROR_MAP : blockvector [4, 512] volatile;      .MAP THE ERROR MAP LIKE THE BLAST TASBLE
;        4123
;        4124      incr CNT from 0 to .MAX_CHIP_COL*2 - 1 do    .OR ALL NON ZERO BLAST TABLE LOCATIONS
;        4125          begin
;        4126
;        4127          if .BLAST_TBL [.BNK_NUM, .CNT, FULL_WRD] neq ZERO       .IS THIS LOCATION NOT ZERO
;        4128          then
;        4129              BLAST_TBL [.BNK_NUM, .CNT, FULL_WRD] = .BLAST_TBL [.BNK_NUM, .CNT, FULL_WRD] or .ERROR_MAP [
;        4130                  .BNK_NUM, .CNT, FULL_WRD];      !OR THIS LOCATION WITH ADJACENT ERROR MAP LOCATION
;        4131
;        4132          end;
;        4133
;        4134      end;


                                        .SBTTL  OR.OLD.NEW.PD ROUTINE DECLARATIONS
014434  004167  000000G          OR.OLD.NEW.PD:
                                        JSR     R1,$SAVE4                       ;                               4102
014440  016704  036774'                 MOV     MAX.CHIP.COL,R4                 ;                               4124
014444  006304                          ASL     R4
014446  016600  000014                  MOV     14(SP),R0                       ; BNK.NUM,*                     4127
014452  000300                          SWAB    R0
014454  105000                          CLRB    R0
014456  006300                          ASL     R0
014460  005001                          CLR     R1                              ; CNT                           4124
014462  000414                          BR      3$                                                              4127
014464  010003                  1$:     MOV     R0,R3                           ;
014466  060103                          ADD     R1,R3                           ; CNT,*
014470  006303                          ASL     R3
014472  012702  026770'                 MOV     #BLAST.TBL,R2
014476  060302                          ADD     R3,R2
014500  005712                          TST     (R2)
```

```
014502  001403                          BEQ     2$                                      4129
014504  056363  006626' C26770'         BIS     ERROR.MAP(R3),BLAST.TBL(R3)     ;
014512  005201                  2$:     INC     R1                              ; CNT   4124
014514  020104                  3$:     CMP     R1,R4                           ; CNT,*
014516  002762                          BLT     1$
014520  000207                          RTS     PC                              ;       4102

; Routine Size:  27 words
; Maximum stack depth per invocation:  5 words
```

```
4135  routine BIT_CLR_OLD_NEW (BNK_NUM) : novalue =
4136      begin
4137
4138  !++
4139  !  FUNCTIONAL DESCRIPTION: BIT CLEAR THE OLD AND NEW PROM DATA
4140  !
4141  !                          WHEN A FAILING CHIP IS SEARCHED FOR NEWLY
4142  !                          FAILING ROWS AND COLUMNS THERE IS ALWAYS
4143  !                          THE POSSIBLY THAT OLD BAD ROWS OR COLUMNS
4144  !                          MAY ALSO FAIL.  IN FACT WE HOPE THEY DO
4145  !                          SO TO GIVE US A PAST HISTORY OF HOW THE
4146  !                          THE CHIP IS PERFORMING.
4147  !
4148  !                          THESE FAILING ROWS AND COLUMNS IF ANY
4149  !                          MUST BE CLEAR OUT FROM THE BLAST TABLE
4150  !                          TO ALLOW THE COUNTING OF BAD NIBBLE OFF-
4151  !                          SETS.
4152  !
4153  !                          THIS ROUTINE BIT CLEARS THE ERROR MAP WITH
4154  !                          THE BLAST TABLE AT THE SELECTED BANK.
4155  !
4156  !  FORMAL PARAMETERS:
4157  !                  BNK_NUM:
4158  !                          SELECTED BANK NUMBER FOR BIT CLEARING.
4159  !--
4160
4161      map
4162          ERROR_MAP : blockvector [4, 512] volatile;      .MAP THE ERROR MAP LIKE THE BLAST TABLE
4163
4164      incr CNT from 0 to .MAX_CHIP_COL*2 - 1 do   !BIT CLEAR BLAST TABLE WITH ERROR MAP AT THIS BANK
4165          begin
4166          BLAST_TBL [.BNK_NUM, .CNT, FULL_WRD] = ( not .ERROR_MAP [.BNK_NUM, .CNT, FULL_WRD]) and .BLAST_TBL [
4167              .BNK_NUM, .CNT, FULL_WRD];
4168          end;
4169
4170      end;
```

```
                                .SBTTL  BIT.CLR.OLD.NEW ROUTINE DECLARATIONS
014522  004167  000000G    BIT.CLR.OLD.NEW:
                            JSR     R1,$SAVE3                       ;                       4135
014526  016703  036774'                   MOV     MAX.CHIP.COL,R3                 ;       4164
014532  006303                      ASL     R3
014534  016600  000012                    MOV     12(SP),R0                       ; BNK.NUM,*     4166
014540  000300                      SWAB    R0
014542  105000                      CLRB    R0
014544  006300                      ASL     R0
014546  005001                      CLR     R1                              ; CNT           4164
014550  000407                      BR      2$
014552  010002              1$:     MOV     R0,R2                           ;               4166
014554  060102                      ADD     R1,R2                           ; CNT,*
014556  006302                      ASL     R2
```

```
014560  046262  006626' 026770'            BIC     ERROR.MAP(R2),BLAST.TBL(R2)
014566  005201                              INC     R1                          ; CNT                   4164
014570  020103                      2$:     CMP     R1,R3                       ; CNT,*
014572  002767                              BLT     1$
014574  000207                              RTS     PC                          ;                       4135

; Routine Size:  22 words
; Maximum stack depth per invocation:  4 words
```

```
4171  routine IN_BLAST_TBL (ARR$BNK_SEL, TSTED_BNK) : novalue =
4172      begin
4173
4174  ;++
4175  !  FUNCTIONAL DESCRIPTION: INTERIGATE THE BLAST TABLE
4176  !
4177  !                          THIS ROUTINE FIRST CLEARS THE BLAST TABLE
4178  !                          OF ANY OLD BAD ROW OR COLUMN DATA THAT MIGHT
4179  !                          HAVE BEEN FOUND DURING THE SEARCHING FOR NEW
4180  !                          FAILING ROWS AND COLUMNS.
4181  !
4182  !                          THE SELECTED BANK IS THEN SEARCHED TO SEE IF
4183  !                          ANY ADDITIONAL FAILING ROWS OR COLUMNS WERE
4184  !                          FOUND.
4185  !
4186  !                          IF NO ADDITIONAL FAILING ROW COLUMNS WERE
4187  !                          FOUND THEN THE BANK IS NOT FURTHER TESTED.
4188  !
4189  !                          IF ADDITIONAL FAILING ROW OR COLUMNS WERE
4190  !                          FOUND THEN THE BLAST TABLE AND THE ERROR
4191  !                          MAP IS SEARCH FOR NIBBLE OFFSET > 14.
4192  !
4193  !
4194  !
4195  !  FORMAL PARAMETERS:
4196  !                  ARR$BNK_SEL:
4197  !                          STORES THE SELECTED ARRAYS ARRAY AND BANK SELECT
4198  !                          NUMBERS.
4199  !                  TSTED_BNK:
4200  !                          INDICATES HOW MANY BANKS PER ARRAY MODULE
4201  !                          IS TO BE PM'ED.
4202  ;--
4203
4204      local
4205          QD_0_SUM,                                   .QUADRANT 0 OFFSET SUM - SECTORS 0-127
4206          QD_1_SUM,                                   !QUADRANT 1 OFFSET SUM = SECTORS 128-255
4207          QD_2_SUM,                                   .QUADRANT 2 OFFSET SUM = SECTORS 256-383
4208          QD_3_SUM,                                   !QUADRANT 3 OFFSET SUM = SECTORS 384-511
4209          ROW_0_127_OFF,                              .OFFSETS COUNT FOR ROWS 0-127
4210          ROW_128_255_OFF,                            !OFFSET COUNT FOR ROWS 128-255
4211          COL_0_127_OFF,                              !OFFSET COUNT FOR COLUMNS 0-127
4212          COL_128_255_OFF,                            !OFFSET COUNT FOR COLUMN 128-255
4213          BNK_NUM,                                    !STORES BANK SELECTOR
4214          ARR_NUM,                                    !STORES ARRAY SELECTOR
4215          OVER_FLOW;                                  !STORES OFFSET COUNTS WHICH ARE > 14
4216
4217      map
4218          ERROR_MAP : blockvector [4, 512] volatile;      .MAP THE ERROR MAP LIKE THE BLAST TABLE
4219
4220      I_ERROR_MAP ();                                 !INIT THE ERROR MAP BEFORE WE STORE THE PROM DATA IN THERE
4221      RD_OLD_PROM_DATA (.ARR$BNK_SEL);                .READ THE OLD PROM DATA INTO THE ERROR MAP
4222      BNK_NUM = .ARR$BNK_SEL<.BNK_SEL_POS, BNK$SEL_SIZE>;  SELECT THE BANK TO INTERIGATE
```

```
;        4223      ARR_NUM - .ARR$BNK_SEL<.ARR_SEL_POS, ARR$SEL_SIZE>; !SELECT THE ARRAY OF THIS BANK
;        4224
;        4225      incr SEL_BNK from 0 to .TSTED_BNK - 1 do    .INTERIGATE THE PM'ED BANKS
;        4226          begin
;        4227          BIT_CLR_OLD_NEW (.BNK_NUM);             .CLEAR THE OLD FAILING ROWS/COLS FROM THE BLAST TABLE
;        4228
;        4229          if S_BLAST_TBL (.BNK_NUM)               .ARE THERE ANY NEW ERRORS IN THIS BANK
;        4230          then
;        4231              begin
;        4232              BAD_BNK_REG [.BNK_NUM] = SET_FLG;   .FLAG THAT THIS IS A BAD BANK
;        4233
;        4234              incr NIB_NUM from 0 to 9 do         .SEARCH THRU ALL NIBBLES IN THIS BANK
;        4235                  begin
;        4236                  ROW_0_127_OFF = ZERO;           !INIT THE ROW COUNT
;        4237                  ROW_128_255_OFF = ZERO;         .INIT THE ROW COUNT
;        4238                  COL_0_127_OFF = ZERO;           .INIT THE COLUMN COUNT
;        4239                  COL_128_255_OFF - ZERO;         .INIT THE COLUMN COUNT
;        4240
;        4241                  incr ROW_NUM from 0 to 127 do   .SUM THE NUMBER OF FAILING ROWS 0-127
;        4242                      begin
;        4243
;        4244                      if .BLAST_TBL [.BNK_NUM, .ROW_NUM, .NIB_NUM, 1, 0] then ROW_0_127_OFF = .ROW_0_127_OFF + 1
;        4245
;        4246                      ;
;        4247
;        4248                      if .ERROR_MAP [.BNK_NUM, .ROW_NUM, .NIB_NUM, 1, 0] then ROW_0_127_OFF = .ROW_0_127_OFF + 1
;        4249
;        4250                      ;
;        4251                      end;
;        4252
;        4253                  incr COL_NUM from 0 to 127 do   .SUM THE NUMBER OF FAILING COLUMNS 0-127
;        4254                      begin
;        4255
;        4256                      if .BLAST_TBL [.BNK_NUM, .COL_NUM + .COL_BASE, .NIB_NUM, 1, 0]
;        4257                      then
;        4258                          COL_0_127_OFF - .COL_0_127_OFF + 1;
;        4259
;        4260                      if .ERROR_MAP [.BNK_NUM, .COL_NUM + .COL_BASE, .NIB_NUM, 1, 0]
;        4261                      then
;        4262                          COL_0_127_OFF - .COL_0_127_OFF + 1;
;        4263
;        4264                      end;
;        4265
;        4266                  if .BNK_NUM_SEC gtr 127         !IS THIS A 64K PART
;        4267                  then
;        4268                      begin
;        4269
;        4270                      incr ROW_NUM from 128 to 255 do     !SUM NUMBER OF FAILING ROWS 128-255
;        4271                          begin
;        4272
;        4273                          if .BLAST_TBL [.BNK_NUM, ROW_NUM, .NIB_NUM, 1, 0]
;        4274                          then
```

```
;        4275                                ROW_128_255_OFF - .ROW_128_255_OFF + 1;
;        4276
;        4277                            if .ERROR_MAP [.BNK_NUM, .ROW_NUM, .NIB_NUM, 1, 0]
;        4278                            then
;        4279                                ROW_128_255_OFF - .ROW_128_255_OFF + 1;
;        4280
;        4281                            end;
;        4282
;        4283                        incr COL_NUM from 128 to 255 do     .SUM NUMBER OF FAILING COL 128-255
;        4284                            begin
;        4285
;        4286                            if .BLAST_TBL [.BNK_NUM, .COL_NUM + .COL_BASE, .NIB_NUM, 1, 0]
;        4287                            then
;        4288                                COL_128_255_OFF - .COL_128_255_OFF + 1;
;        4289
;        4290                            if .ERROR_MAP [.BNK_NUM, .COL_NUM + .COL_BASE, .NIB_NUM, 1, 0]
;        4291                            then
;        4292                                COL_128_255_OFF - .COL_128_255_OFF + 1;
;        4293
;        4294                            end;
;        4295
;        4296                        QD_1_SUM   .ROW_128_255_OFF + .COL_0_127_OFF;       !SUM NUMBER OF FAILURES IN QUAD 1
;        4297                        QD_2_SUM = .ROW_0_127_OFF + .COL_128_255_OFF;       !SUM NUMBER OF FAILURES INQUAD 2
;        4298                        QD_3_SUM = .ROW_128_255_OFF + .COL_128_255_OFF;     !SUM NUMBER OF FAILURES IN QUAD 3
;        4299                        end;
;        4300
;        4301                    QD_0_SUM = .ROW_0_127_OFF + .COL_0_127_OFF;     !SUM NUMBER OF FAILURES IN QUAD 0
;        4302
;        4303                    if .QD_0_SUM gtr 14             !IS QUAD 0 OFFSETS > 14
;        4304                    then
;        4305                        begin
;        4306                        ERRSF (ERR_7, CON_C_MSG, 0);        !REPORT THE ERROR
;        4307                        PRINTB (A_B_N_PRINT, .ARR_NUM, .BNK_NUM, .NIB_NUM); !PRINT WHERE
;        4308                        DEGRADE_MOD_REG [.BNK_NUM] = SET_FLG;       !INDICATE THIS IS A DEGRADED BANK
;        4309                        OVER_FLOW = .QD_0_SUM - 14; !CALCULATE THE DIFFERENCE
;        4310
;        4311                        incr ROW_NUM from 0 to 127 do       !DELETE ROW OFFSETS UNTIL EQL 14
;        4312
;        4313                            if .BLAST_TBL [.BNK_NUM, .ROW_NUM, .NIB_NUM, 1, 0]      !IS THIS ROW BAD
;        4314                            then
;        4315                                begin
;        4316                                BLAST_TBL [.BNK_NUM, .ROW_NUM, .NIB_NUM, 1, 0] = CLR_FLG;   !CLEAR THIS BAD ROW
;        4317                                OVER_FLOW - .OVER_FLOW - 1; .DECREMENT THE DIFFERENCE
;        4318                                QD_2_SUM - .QD_2_SUM - 1;   !DELETE OTHER SUMS USING THIS ROW
;        4319
;        4320                                if .OVER_FLOW eql ZERO then exitloop;       !EXIT IF DIFFERENCE EQL 0
;        4321
;        4322                                end;
;        4323
;        4324                        if .OVER_FLOW neq ZERO      'DID DELETING BAD ROW GET THE DIFFERENCE TO 14
;        4325                        then
;        4326                            begin
```

```
4327
4328                        incr COL_NUM from 0 to 127 do   !DELETE BAD COLUMN UNTIL OFFSETS ARE AT 14
4329
4330                            if .BLAST_TBL [.BNK_NUM, .COL_NUM + .COL_BASE, .NIB_NUM, 1, 0]
4331                            then
4332                                begin
4333                                BLAST_TBL [.BNK_NUM, .COL_NUM + .COL_BASE, .NIB_NUM, 1, 0] = CLR_FLG;
4334                                OVER_FLOW = .OVER_FLOW - 1;      !DECREMENT THE DIFFERENCE
4335                                QD_1_SUM = .QD_1_SUM - 1;        .DELETE THIS COLUMN FROM OTHER SUMS
4336
4337                                if .OVER_FLOW eql ZERO then exitloop;   .EXIT IF DIFFERENCE IS AT 0
4338
4339                                end;
4340
4341                            end;
4342
4343                        end;
4344
4345                    if .BNK_NUM_SEC gtr 127          .IS THIS A 64K PART
4346                    then
4347                        begin
4348
4349                        if .QD_1_SUM gtr 14           !IS QUAD 1 SUM > 14
4350                        then
4351                            begin
4352                            ERRSF (ERR_7, CON_C_MSG, 0);     .REPORT THE ERROR
4353                            PRINTB (A_B_N_PRINT, .ARR_NUM, .BNK_NUM, .NIB_NUM);      .TELL WHERE
4354                            DEGRADE_MOD_REG [.BNK_NUM] - SET_FLG;   !INDICATE THIS IS A DEGRADED BANK
4355                            OVER_FLOW = .QD_1_SUM - 14;      !CALCULATE THE DIFFERENCE
4356
4357                            incr ROW_NUM from 128 to 255 do          !DELETE BAD ROWS UNTIL OFFSETS ARE AT 14
4358
4359                                if .BLAST_TBL [.BNK_NUM, .ROW_NUM, .NIB_NUM, 1, 0]  !IS THIS A BAD ROW
4360                                then
4361                                    begin
4362                                    BLAST_TBL [.BNK_NUM, .ROW_NUM, .NIB_NUM, 1, 0] = CLR_FLG;
4363                                    OVER_FLOW = .OVER_FLOW - 1;      !DECREMENT THE DIFFERENCE
4364                                    QD_3_SUM = .QD_3_SUM - 1;        !DELETE OTHER SUMS OF THIS BAD ROW
4365
4366                                    if .OVER_FLOW eql ZERO then exitloop;   !EXIT WHEN DIFFERENCE IS AT 0
4367
4368                                    end;
4369
4370                            if .OVER_FLOW neq ZERO  !IS THIS QUAD OFFSET AT 14 NOW
4371                            then
4372                                begin
4373
4374                                incr COL_NUM from 0 to 127 do        .DELETE BAD COLUMNS UNTIL OFFSETS ARE AT 14
4375
4376                                    if .BLAST_TBL [.BNK_NUM, .COL_NUM + .COL_BASE, .NIB_NUM, 1, 0]
4377                                    then
4378                                        begin
```

```
                                BLAST_TBL [.BNK_NUM, .COL_NUM + .COL_BASE, .NIB_NUM, 1, 0] = CLR_FLG;
                                OVER_FLOW = .OVER_FLOW - 1; .DECREMENT DIFFERENCE

                                if .OVER_FLOW eql ZERO then exitloop;       !EXIT WHEN AT 0

                                end;

                            end;

                        end;

                    if .QD_2_SUM gtr 14         .IS THIS QUAD OFFSETS > 14
                    then
                        begin
                        ERRSF (ERR_7, CON_C_MSG, 0);     !REPORT THE ERROR
                        PRINTB (A_B_N_PRINT, .ARR_NUM, .BNK_NUM, .NIB_NUM);     !REPORT WHERE
                        DEGRADE_MOD_REG [.BNK_NUM] = SET_FLG;   !INDICATE THIS IS A DEGRADED BANK
                        OVER_FLOW = .QD_2_SUM - 14;     !CALCULATE THE DIFFERENCE

                        incr ROW_NUM from 0 to 127 do   !DELETE BAD ROWS UNTIL OFFSET ARE AT 14

                            if .BLAST_TBL [.BNK_NUM, .ROW_NUM, .NIB_NUM, 1, 0]  !IS THIS A BAD ROW
                            then
                                begin
                                BLAST_TBL [.BNK_NUM, .ROW_NUM, .NIB_NUM, 1, 0] = CLR_FLG;
                                OVER_FLOW = .OVER_FLOW - 1;     .DECREMENT THE DIFFERENCE

                                if .OVER_FLOW eql ZERO then exitloop;   !EXIT WHEN AT 0

                                end;

                        if .OVER_FLOW neq ZERO  !IS QUAD OFFSETS AT 14 YET
                        then
                            begin

                            incr COL_NUM from 128 to 255 do     !DELETE BAD COLUMNS UNTIL OFFSETS AT 14

                                if .BLAST_TBL [.BNK_NUM, .COL_NUM + .COL_BASE, .NIB_NUM, 1, 0]
                                then
                                    begin
                                    BLAST_TBL [.BNK_NUM, .COL_NUM + .COL_BASE, .NIB_NUM, 1, 0] = CLR_FLG;
                                    QD_3_SUM = .QD_3_SUM - 1;   !DELETE OTHER SUMS OF THIS COLUMN
                                    OVER_FLOW = .OVER_FLOW - 1; .DECREMENT THE DIFFERNECE

                                    if .OVER_FLOW eql ZERO then exitloop;       !EXIT WHEN AT 0

                                    end;

                            end;

                        end;
```

```
4431                    if .QD_3_SUM gtr 14          .IS THIS QUAD OFFSETS > 14
4432                    then
4433                        begin
4434                        ERRSF (ERR_7, CON_C_MSG, 0);     .REPORT THE ERROR
4435                        PRINTB (A_B_N_PRINT, .ARR_NUM, .BNK_NUM, .NIB_NUM);      !PRINT WHERE
4436                        DEGRADE_MOD_REG [.BNK_NUM] = SET_FLG;    !INDICATE THIS IS A DEGRADED BANK
4437                        OVER_FLOW = .QD_3_SUM - 14;      !CALCULATE THE DIFFERENCE
4438
4439                        incr ROW_NUM from 128 to 255 do          !DELETE BAD ROWS UNTIL OFFSET ARE AT 14
4440
4441                            if .BLAST_TBL [.BNK_NUM, .ROW_NUM, .NIB_NUM, 1, 0]  !IS THIS A BAD ROW
4442                            then
4443                                begin
4444                                BLAST_TBL [.BNK_NUM, .ROW_NUM, .NIB_NUM, 1, 0] = CLR_FLG;        !CLEAR
4445                                OVER_FLOW = .OVER_FLOW - 1;      !DECREMENT THE DIFFERENCE
4446
4447                                if .OVER_FLOW eql ZERO then exitloop;    !EXIT WHEN AT 0
4448
4449                                end;
4450
4451                        if .OVER_FLOW neq ZERO   .IS QUAD OFFSETS AT 14 YET
4452                        then
4453                            begin
4454
4455                            incr COL_NUM from 128 to 255 do      !DELETE BAD COLUMN UNTIL OFFSET AT 14
4456
4457                                if .BLAST_TBL [.BNK_NUM, .COL_NUM + .COL_BASE, .NIB_NUM, 1, 0]
4458                                then
4459                                    begin
4460                                    BLAST_TBL [.BNK_NUM, .COL_NUM + .COL_BASE, .NIB_NUM, 1, 0] = CLR_FLG;
4461                                    OVER_FLOW = .OVER_FLOW - 1; .DECREMENT DIFFERENCE
4462
4463                                    if .OVER_FLOW eql ZERO then exitloop;        !EXIT WHEN AT 0
4464
4465                                    end;
4466
4467                                end;
4468
4469                            end;
4470
4471                        end;
4472
4473                    end;
4474
4475                    OR_OLD_NEW_PD (.BNK_NUM);            .OR THE OLD PROM DATA WITH NEW PROM DATA
4476                    end;
4477
4478                BNK_NUM = .BNK_NUM + 1;                  .INCREMENT TO THE NEXT BANK
4479                end;
4480
4481        end;
```

```
                                        .SBTTL  IN.BLAST.TBL ROUTINE DECLARATIONS
014576  004167  000000G        IN.BLAST.TBL:
                                        JSR     R1,$SAVE5                       ;                               4171
014602  162706  000032                SUB     #32,SP
014606  004767  167736                JSR     PC,I.ERROR.MAP                  ;                               4220
014612  016646  000052                MOV     52(SP),-(SP)                    ; ARR$BNK.SEL,*                 4221
014616  004767  177042                JSR     PC,RD.OLD.PROM.DAT
014622  012716  000054                MOV     #54,(SP)                        ;                               4222
014626  060616                        ADD     SP,(SP)                         ; ARR$BNK.SEL,*
014630  016746  037012'               MOV     BNK.SEL.POS,-(SP)
014634  012746  000002                MOV     #2,-(SP)
014640  005046                        CLR     -(SP)
014642  004767  000000G               JSR     PC,BL$GT2
014646  010066  000010                MOV     R0,10(SP)                       ; *,BNK.NUM
014652  012716  000062                MOV     #62,(SP)                        ;                               4223
014656  060616                        ADD     SP,(SP)                         ; ARR$BNK.SEL,*
014660  016746  037010'               MOV     ARR.SEL.POS,-(SP)
014664  012746  000004                MOV     #4,-(SP)
014670  005046                        CLR     -(SP)
014672  004767  000000G               JSR     PC,BL$GT2
014676  010066  000044                MOV     R0,44(SP)                       ; *,ARR.NUM
014702  005066  000042                CLR     42(SP)                          ; SEL.BNK                       4225
014706  000167  003000                JMP     48$
014712  016646  000016        1$:     MOV     16(SP),-(SP)                    ; BNK.NUM,*                     4227
014716  004767  177600                JSR     PC,BIT.CLR.OLD.NEW
014722  016646  000020                MOV     20(SP),-(SP)                    ; BNK.NUM,*                     4229
014726  004767  173606                JSR     PC,S.BLAST.TBL
014732  005726                        TST     (SP)+
014734  006000                        ROR     R0
014736  103402                        BLO     2$
014740  000167  002734                JMP     47$
014744  016666  000020  000022  2$:   MOV     20(SP),22(SP)                   ; BNK.NUM,*                     4232
014752  006266  000022                ASR     22(SP)
014756  006266  000022                ASR     22(SP)
014762  006266  000022                ASR     22(SP)
014766  016646  000022                MOV     22(SP),-(SP)
014772  062716  037016'               ADD     #BAD.BNK.REG,(SP)
014776  016646  000022                MOV     22(SP),-(SP)                    ; BNK.NUM,*
015002  042716  177770                BIC     #177770,(SP)
015006  012746  000001                MOV     #1,-(SP)
015012  011646                        MOV     (SP),-(SP)
015014  004767  000000G               JSR     PC,BL$PU2
015020  016600  000030                MOV     30(SP),R0                       ; BNK.NUM,*                     4244
015024  000300                        SWAB    R0
015026  105000                        CLRB    R0
015030  006300                        ASL     R0
015032  010003                        MOV     R0,R3
015034  005005                        CLR     R5                              ; NIB.NUM                       4234
015036  005066  000044        3$:     CLR     44(SP)                          ; ROW.0.127.OFF                 4236
015042  005066  000046                CLR     46(SP)                          ; ROW.128.255.OFF               4237
015046  005066  000050                CLR     50(SP)                          ; COL.0.127.OFF                 4238
```

```
015052  005066  000052          CLR     52(SP)                          ; COL.128.255.OFF       4239
015056  005002                  CLR     R2                              ; ROW.NUM               4241
015060  010304          4$:     MOV     R3,R4                           ;                       4244
015062  060204                  ADD     R2,R4                           ; ROW.NUM,*
015064  006304                  ASL     R4
015066  010446                  MOV     R4,-(SP)
015070  062716  026770'          ADD     #BLAST.TBL,(SP)
015074  010546                  MOV     R5,-(SP)                        ; NIB.NUM,*
015076  012746  000001          MOV     #1,-(SP)
015102  005046                  CLR     -(SP)
015104  004767  000000G         JSR     PC,BL$GT2
015110  062706  000006          ADD     #6,SP
015114  006000                  ROR     R0
015116  005566  000046          ADC     46(SP)                          ; ROW.0.127.OFF
015122  010416                  MOV     R4,(SP)                         ;                       4248
015124  062716  006626'          ADD     #ERROR.MAP,(SP)
015130  010546                  MOV     R5,-(SP)                        ; NIB.NUM,*
015132  012746  000001          MOV     #1,-(SP)
015136  005046                  CLR     -(SP)
015140  004767  000000G         JSR     PC,BL$GT2
015144  062706  000010          ADD     #10,SP
015150  006000                  ROR     R0
015152  005566  000044          ADC     44(SP)                          ; ROW.0.127.OFF
015156  005202                  INC     R2                              ; ROW.NUM               4241
015160  020227  000177          CMP     R2,#177                         ; ROW.NUM,*
015164  003735                  BLE     4$
015166  005002                  CLR     R2                              ; COL.NUM               4253
015170  010204          5$:     MOV     R2,R4                           ; COL.NUM,*             4256
015172  066704  036770'          ADD     COL.BASE,R4
015176  060304                  ADD     R3,R4
015200  006304                  ASL     R4
015202  062704  026770'          ADD     #BLAST.TBL,R4
015206  010446                  MOV     R4,-(SP)
015210  010546                  MOV     R5,-(SP)                        ; NIB.NUM,*
015212  012746  000001          MOV     #1,-(SP)
015216  005046                  CLR     -(SP)
015220  004767  000000G         JSR     PC,BL$GT2
015224  062706  000006          ADD     #6,SP
015230  006000                  ROR     R0
015232  005566  000052          ADC     52(SP)                          ; COL.0.127.OFF         4258
015236  010204                  MOV     R2,R4                           ; COL.NUM,*             4260
015240  066704  036770'          ADD     COL.BASE,R4
015244  060304                  ADD     R3,R4
015246  006304                  ASL     R4
015250  062704  006626'          ADD     #ERROR.MAP,R4
015254  010416                  MOV     R4,(SP)
015256  010546                  MOV     R5,-(SP)                        ; NIB.NUM,*
015260  012746  000001          MOV     #1,-(SP)
015264  005046                  CLR     -(SP)
015266  004767  000000G         JSR     PC,BL$GT2
015272  062706  000010          ADD     #10,SP
015276  006000                  ROR     R0
```

```
015300  005566  000050                          ADC     50(SP)                          ; COL.0.127.OFF                 4262
015304  005202                                  INC     R2                              ; COL.NUM                       4253
015306  020227  000177                          CMP     R2,#177                         ; COL.NUM,*
015312  003726                                  BLE     5$
015314  026727  037014' 000177                  CMP     BNK.NUM.SEC,#177                ;                               4266
015322  003563                                  BLE     8$
015324  016600  000030                          MOV     30(SP),R0                       ; BNK.NUM,*                     4273
015330  000300                                  SWAB    R0
015332  105000                                  CLRB    R0
015334  006300                                  ASL     R0
015336  006300                                  ASL     R0
015340  010002                                  MOV     R0,R2
015342  012704  000060                          MOV     #60,R4
015346  060604                                  ADD     SP,R4                           ; ROW.NUM,*
015350  006304                                  ASL     R4
015352  060402                                  ADD     R4,R2
015354  062702  026770'                         ADD     #BLAST.TBL,R2
015360  012766  000200  000060                  MOV     #200,60(SP)                     ; *,ROW.NUM                     4270
015366  010246                          6$:     MOV     R2,-(SP)                        ;                               4273
015370  010546                                  MOV     R5,-(SP)                        ; NIB.NUM,*
015372  012746  000001                          MOV     #1,-(SP)
015376  005046                                  CLR     -(SP)
015400  004767  000000G                         JSR     PC,BL$GT2
015404  062706  000006                          ADD     #6,SP
015410  006000                                  ROR     R0
015412  005566  000050                          ADC     50(SP)                          ; ROW.128.255.OFF               4275
015416  010304                                  MOV     R3,R4                           ;                               4277
015420  066604  000062                          ADD     62(SP),R4                       ; ROW.NUM,*
015424  006304                                  ASL     R4
015426  062704  006626'                         ADD     #ERROR.MAP,R4
015432  010416                                  MOV     R4,(SP)
015434  010546                                  MOV     R5,-(SP)                        ; NIB.NUM,*
015436  012746  000001                          MOV     #1,-(SP)
015442  005046                                  CLR     -(SP)
015444  004767  000000G                         JSR     PC,BL$GT2
015450  062706  000010                          ADD     #10,SP
015454  006000                                  ROR     R0
015456  005566  000046                          ADC     46(SP)                          ; ROW.128.255.OFF               4279
015462  005266  000060                          INC     60(SP)                          ; ROW.NUM                       4270
015466  026627  000060  000377                  CMP     60(SP),#377                     ; ROW.NUM,*
015474  003734                                  BLE     6$
015476  012702  000200                          MOV     #200,R2                         ; *,COL.NUM                     4283
015502  010204                          7$:     MOV     R2,R4                           ; COL.NUM,*                     4286
015504  066704  036770'                         ADD     COL.BASE,R4
015510  060304                                  ADD     R3,R4
015512  006304                                  ASL     R4
015514  062704  026770'                         ADD     #BLAST.TBL,R4
015520  010446                                  MOV     R4,-(SP)
015522  010546                                  MOV     R5,-(SP)                        ; NIB.NUM,*
015524  012746  000001                          MOV     #1,-(SP)
015530  005046                                  CLR     -(SP)
015532  004767  000000G                         JSR     PC,BL$GT2
```

```
015536  062706  000006                  ADD     #6,SP
015542  006000                          ROR     R0
015544  005566  000054                  ADC     54(SP)                  ; COL.128.255.OFF                       4288
015550  010204                          MOV     R2,R4                   ; COL.NUM,*                             4290
015552  066704  036770'                 ADD     COL.BASE,R4
015556  060304                          ADD     R3,R4
015560  006304                          ASL     R4
015562  062704  006626'                 ADD     #ERROR.MAP,R4
015566  010416                          MOV     R4,(SP)
015570  010546                          MOV     R5,-(SP)                ; NIB.NUM,*
015572  012746  000001                  MOV     #1,-(SP)
015576  005046                          CLR     -(SP)
015600  004767  000000G                 JSR     PC,BL$GT2
015604  062706  000010                  ADD     #10,SP
015610  006000                          ROR     R0
015612  005566  000052                  ADC     52(SP)                  ; COL.128.255.OFF                       4292
015616  005202                          INC     R2                      ; COL.NUM                               4283
015620  020227  000377                  CMP     R2,#377                 ; COL.NUM,*
015624  003726                          BLE     7$
015626  C16666  000046  0C0036         MOV     46(SP),36(SP)           ; ROW.128.255.OFF,QD.1.SUM              4296
015634  066666  000050  000036         ADD     50(SP),36(SP)           ; COL.0.127.OFF,QD.1.SUM
015642  016666  000044  000040         MOV     44(SP),40(SP)           ; ROW.0.127.OFF,QD.2.SUM                4297
015650  066666  000052  000040         ADD     52(SP),40(SP)           ; COL.128.255.OFF,QD.2.SUM
015656  016666  000046  000042         MOV     46(SP),42(SP)           ; ROW.128.255.OFF,QD.3.SUM              4298
015664  066666  000052  000042         ADD     52(SP),42(SP)           ; COL.128.255.OFF,QD.3.SUM
015672  016666  000044  000034  8$:    MOV     44(SP),34(SP)           ; ROW.0.127.OFF,QD.0.SUM                4301
015700  066666  000050  000034         ADD     50(SP),34(SP)           ; COL.0.127.OFF,QD.0.SUM
015706  026627  000034  000016         CMP     34(SP),#16              ; QD.0.SUM,*                            4303
015714  003571                          BLE     17$
015716  104454                          TRAP    54                      ;                                       4306
015720  000007                          .WORD   7
015722  000474'                         .WORD   CON.C.MSG
015724  000000                          .WORD   0
015726  010546                          MOV     R5,-(SP)                ; NIB.NUM,*                             4307
015730  016646  000032                  MOV     32(SP),-(SP)            ; BNK.NUM,*
015734  016646  000062                  MOV     62(SP),-(SP)            ; ARR.NUM,*
015740  012746  002172'                 MOV     #A.B.N.PRINT,-(SP)
015744  012746  000004                  MOV     #4,-(SP)
015750  010600                          MOV     SP,R0                   ; SP,*
015752  104414                          TRAP    14
015754  016616  000044                  MOV     44(SP),(SP)             ;                                       4308
015760  062716  037022'                 ADD     #DEGRADE.MOD.REG,(SP)
015764  016646  000042                  MOV     42(SP),-(SP)            ; BNK.NUM,*
015770  042716  177770                  BIC     #177770,(SP)
015774  012746  000001                  MOV     #1,-(SP)
016000  011646                          MOV     (SP),-(SP)
016002  004767  000000G                 JSR     PC,BL$PU2
016006  016601  000054                  MOV     54(SP),R1               ; QD.0.SUM,OVER.FLOW                    4309
016012  162701  000016                  SUB     #16,R1                  ; *,OVER.FLOW
016016  005002                          CLR     R2                      ; ROW.NUM                               4311
016020  010304                  9$:     MOV     R3,R4                   ;                                       4313
016022  060204                          ADD     R2,R4                   ; ROW.NUM,*
```

```
016024  006304                          ASL     R4
016026  062704  026770'                 ADD     #BLAST.TBL,R4
016032  010446                          MOV     R4,-(SP)
016034  010546                          MOV     R5,-(SP)        ; NIB.NUM,*
016036  012746  000001                  MOV     #1,-(SP)
016042  005046                          CLR     -(SP)
016044  004767  000000G                 JSR     PC,BL$GT2
016050  062706  000010                  ADD     #10,SP
016054  006000                          ROR     R0
016056  103021                          BCC     11$                             4316
016060  010446                          MOV     R4,-(SP)        ;
016062  010546                          MOV     R5,-(SP)        ; NIB.NUM,*
016064  012746  000001                  MOV     #1,-(SP)
016070  005046                          CLR     -(SP)
016072  004767  000000G                 JSR     PC,BL$PU2
016076  005301                          DEC     R1              ; OVER.FLOW     4317
016100  005366  000070                  DEC     70(SP)          ; QD.2.SUM      4318
016104  005701                          TST     R1              ; OVER.FLOW     4320
016106  001003                          BNE     10$
016110  062706  000010                  ADD     #10,SP
016114  000406                          BR      12$
016116  062706  000010          10$:    ADD     #10,SP          ;               4315
016122  005202                  11$:    INC     R2              ; ROW.NUM       4311
016124  020227  000177                  CMP     R2,#177         ; ROW.NUM,*
016130  003733                          BLE     9$
016132  005701                  12$:    TST     R1              ; OVER.FLOW     4324
016134  001457                          BEQ     16$
016136  005002                          CLR     R2              ; COL.NUM       4328
016140  010204                  13$:    MOV     R2,R4           ; COL.NUM,*     4330
016142  066704  036770'                 ADD     COL.BASE,R4
016146  060304                          ADD     R3,R4
016150  006304                          ASL     R4
016152  062704  026770'                 ADD     #BLAST.TBL,R4
016156  010446                          MOV     R4,-(SP)
016160  010546                          MOV     R5,-(SP)        ; NIB.NUM,*
016162  012746  000001                  MOV     #1,-(SP)
016166  005046                          CLR     -(SP)
016170  004767  000000G                 JSR     PC,BL$GT2
016174  062706  000010                  ADD     #10,SP
016200  006000                          ROR     R0
016202  103030                          BCC     15$                             4333
016204  010204                          MOV     R2,R4           ; COL.NUM,*
016206  066704  036770'                 ADD     COL.BASE,R4
016212  060304                          ADD     R3,R4
016214  006304                          ASL     R4
016216  062704  026770'                 ADD     #BLAST.TBL,R4
016222  010446                          MOV     R4,-(SP)
016224  010546                          MOV     R5,-(SP)        ; NIB.NUM,*
016226  012746  000001                  MOV     #1,-(SP)
016232  005046                          CLR     -(SP)
016234  004767  000000G                 JSR     PC,BL$PU2
016240  005301                          DEC     R1              ; OVER.FLOW     4334
```

```
016242  005366  000066                    DEC     66(SP)                          ; QD.1.SUM                      4335
016246  005701                            TST     R1                              ; OVER.FLOW                     4337
016250  001003                            BNE     14$
016252  062706  000010                    ADD     #10,SP
016256  000406                            BR      16$
016260  062706  000010            14$:    ADD     #10,SP                          ;                               4332
016264  005202                    15$:    INC     R2                              ; COL.NUM                       4328
016266  020227  000177                    CMP     R2,#177                         ; COL.NUM,*
016272  003722                            BLE     13$
016274  062706  000020            16$:    ADD     #20,SP                          ;                               4305
016300  026727  037014' 000177    17$:    CMP     BNK.NUM.SEC,#177                ;                               4345
016306  003002                            BGT     18$
016310  000167  001334                    JMP     45$
016314  026627  000036  000016    18$:    CMP     36(SP),#16                      ; QD.1.SUM,*                    4349
016322  003567                            BLE     27$
016324  104454                            TRAP    54                              ;                               4352
016326  000007                            .WORD   7
016330  000474'                           .WORD   CON.C.MSG
016332  000000                            .WORD   0
016334  010546                            MOV     R5,-(SP)                        ; NIB.NUM,*                     4353
016336  016646  000032                    MOV     32(SP),-(SP)                    ; BNK.NUM,*
016342  016646  000062                    MOV     62(SP),-(SP)                    ; ARR.NUM,*
016346  012746  002172'                   MOV     #A.B.N.PRINT,-(SP)
016352  012746  000004                    MOV     #4,-(SP)
016356  010600                            MOV     SP,R0                           ; SP,*
016360  104414                            TRAP    14
016362  016616  000044                    MOV     44(SP),(SP)                     ;                               4354
016366  062716  037022'                   ADD     #DEGRADE.MOD.REG,(SP)
016372  016646  000042                    MOV     42(SP),-(SP)                    ; BNK.NUM,*
016376  042716  177770                    BIC     #177770,(SP)
016402  012746  000001                    MOV     #1,-(SP)
016406  011646                            MOV     (SP),-(SP)
016410  004767  000000G                   JSR     PC,BL$PU2
016414  016601  000056                    MOV     56(SP),R1                       ; QD.1.SUM,OVER.FLOW            4355
016420  162701  000016                    SUB     #16,R1                          ; *,OVER.FLOW
016424  012702  000200                    MOV     #200,R2                         ; *,ROW.NUM                     4357
016430  010304            19$:    MOV     R3,R4                           ;                               4359
016432  060204                            ADD     R2,R4                           ; ROW.NUM,*
016434  006304                            ASL     R4
016436  062704  026770'                   ADD     #BLAST.TBL,R4
016442  010446                            MOV     R4,-(SP)
016444  010546                            MOV     R5,-(SP)                        ; NIB.NUM,*
016446  012746  000001                    MOV     #1,-(SP)
016452  005046                            CLR     -(SP)
016454  004767  000000G                   JSR     PC,BL$GT2
016460  062706  000010                    ADD     #10,SP
016464  006000                            ROR     R0
016466  103021                            BCC     21$
016470  010446                            MOV     R4,-(SP)                        ;                               4362
016472  010546                            MOV     R5,-(SP)                        ; NIB.NUM,*
016474  012746  000001                    MOV     #1,-(SP)
016500  005046                            CLR     -(SP)
```

```
016502  004767  000000G             JSR     PC,BL$PU2
016506  005301                      DEC     R1                      ; OVER.FLOW                     4363
016510  005366  000072              DEC     72(SP)                  ; QD.3.SUM                      4364
016514  005701                      TST     R1                      ; OVER.FLOW                     4366
016516  001003                      BNE     20$
016520  062706  000010              ADD     #10,SP
016524  000406                      BR      22$
016526  062706  000010      20$:    ADD     #10,SP                  ;                               4361
016532  005202              21$:    INC     R2                      ; ROW.NUM                       4357
016534  020227  000377              CMP     R2,#377                 ; ROW.NUM,*
016540  003733                      BLE     19$
016542  005701              22$:    TST     R1                      ; OVER.FLOW                     4370
016544  001454                      BEQ     26$
016546  005002                      CLR     R2                      ; COL.NUM                       4374
016550  010204              23$:    MOV     R2,R4                   ; COL.NUM,*                     4376
016552  066704  036770'            ADD     COL.BASE,R4
016556  060304                      ADD     R3,R4
016560  006304                      ASL     R4
016562  062704  026770'            ADD     #BLAST.TBL,R4
016566  010446                      MOV     R4,-(SP)
016570  010546                      MOV     R5,-(SP)                ; NIB.NUM,*
016572  012746  000001              MOV     #1,-(SP)
016576  005046                      CLR     -(SP)
016600  004767  000000G             JSR     PC,BL$GT2
016604  062706  000010              ADD     #10,SP
016610  006000                      ROR     R0
016612  103025                      BCC     25$
016614  010204                      MOV     R2,R4                   ; COL.NUM,*                     4379
016616  066704  036770'            ADD     COL.BASE,R4
016622  060304                      ADD     R3,R4
016624  006304                      ASL     R4
016626  062704  026770'            ADD     #BLAST.TBL,R4
016632  010446                      MOV     R4,-(SP)
016634  010546                      MOV     R5,-(SP)                ; NIB.NUM,*
016636  012746  000001              MOV     #1,-(SP)
016642  005046                      CLR     -(SP)
016644  004767  000000G             JSR     PC,BL$PU2
016650  005301                      DEC     R1                      ; OVER.FLOW                     4380
016652  001003                      BNE     24$                     ;                               4382
016654  062706  000010              ADD     #10,SP
016660  000406                      BR      26$
016662  062706  000010      24$:    ADD     #10,SP                  ;                               4378
016666  005202              25$:    INC     R2                      ; COL.NUM                       4374
016670  020227  000177              CMP     R2,#177                 ; COL.NUM,*
016674  003725                      BLE     23$
016676  062706  000020      26$:    ADD     #20,SP                  ;                               4351
016702  026627  000040  000016  27$: CMP     40(SP),#16              ; QD.2.SUM,*                    4390
016710  003566                      BLE     36$
016712  104454                      TRAP    54                      ;                               4393
016714  000007                      .WORD   7
016716  000474'                     .WORD   CON.C.MSG
016720  000000                      .WORD   0
```

```
016722  010546                          MOV     R5,-(SP)                ; NIB.NUM,*                     4394
016724  016646  000032                  MOV     32(SP),-(SP)            ; BNK.NUM,*
016730  016646  000062                  MOV     62(SP),-(SP)            ; ARR.NUM,*
016734  012746  002172'                 MOV     #A.B.N.PRINT,-(SP)
016740  012746  000004                  MOV     #4,-(SP)
016744  010600                          MOV     SP,R0                   ; SP,*
016746  104414                          TRAP    14
016750  016616  000044                  MOV     44(SP),(SP)             ;                               4395
016754  062716  037022'                 ADD     #DEGRADE.MOD.REG,(SP)
016760  016646  000042                  MOV     42(SP),-(SP)            ; BNK.NUM,*
016764  042716  177770                  BIC     #177770,(SP)
016770  012746  000001                  MOV     #1,-(SP)
016774  011646                          MOV     (SP),-(SP)
016776  004767  000000G                 JSR     PC,BL$PU2
017002  016601  000060                  MOV     60(SP),R1               ; QD.2.SUM,OVER.FLOW            4396
017006  162701  000016                  SUB     #16,R1                  ; *,OVER.FLOW
017012  005002                          CLR     R2                      ; ROW.NUM                       4398
017014  010304                  28$:    MOV     R3,R4                   ;                               4400
017016  060204                          ADD     R2,R4                   ; ROW.NUM,*
017020  006304                          ASL     R4
017022  062704  026770'                 ADD     #BLAST.TBL,R4
017026  010446                          MOV     R4,-(SP)
017030  010546                          MOV     R5,-(SP)                ; NIB.NUM,*
017032  012746  000001                  MOV     #1,-(SP)
017036  005046                          CLR     -(SP)
017040  004767  000000G                 JSR     PC,BL$GT2
017044  062706  000010                  ADD     #10,SP
017050  006000                          ROR     R0
017052  103016                          BCC     30$
017054  010446                          MOV     R4,-(SP)                ;                               4403
017056  010546                          MOV     R5,-(SP)                ; NIB.NUM,*
017060  012746  000001                  MOV     #1,-(SP)
017064  005046                          CLR     -(SP)
017066  004767  000000G                 JSR     PC,BL$PU2
017072  005301                          DEC     R1                      ; OVER.FLOW                     4404
017074  001003                          BNE     29$                     ;                               4406
017076  062706  000010                  ADD     #10,SP
017102  000406                          BR      31$
017104  062706  000010          29$:    ADD     #10,SP                  ;                               4402
017110  005202                  30$:    INC     R2                      ; ROW.NUM                       4398
017112  020227  000177                  CMP     R2,#177                 ; ROW.NUM,*
017116  003736                          BLE     28$
017120  005701                  31$:    TST     R1                      ; OVER.FLOW                     4410
017122  001457                          BEQ     35$
017124  012702  000200                  MOV     #200,R2                 ; *,COL.NUM                     4414
017130  010204                  32$:    MOV     R2,R4                   ; COL.NUM,*                     4416
017132  066704  036770'                 ADD     COL.BASE,R4
017136  060304                          ADD     R3,R4
017140  006304                          ASL     R4
017142  062704  026770'                 ADD     #BLAST.TBL,R4
017146  010446                          MOV     R4,-(SP)
017150  010546                          MOV     R5,-(SP)                ; NIB.NUM,*
```

```
017152  012746  000001                  MOV     #1,-(SP)
017156  005046                          CLR     -(SP)
017160  004767  000000G                 JSR     PC,BL$GT2
017164  062706  000010                  ADD     #10,SP
017170  006000                          ROR     R0
017172  103027                          BCC     34$
017174  010204                          MOV     R2,R4                   ; COL.NUM,*             4419
017176  066704  036770'                 ADD     COL.BASE,R4
017202  060304                          ADD     R3,R4
017204  006304                          ASL     R4
017206  062704  026770'                 ADD     #BLAST.TBL,R4
017212  010446                          MOV     R4,-(SP)
017214  010546                          MOV     R5,-(SP)                ; NIB.NUM,*
017216  012746  000001                  MOV     #1,-(SP)
017222  005046                          CLR     -(SP)
017224  004767  000000G                 JSR     PC,BL$PU2
017230  005366  000072                  DEC     72(SP)                  ; QD.3.SUM              4420
017234  005301                          DEC     R1                      ; OVER.FLOW             4421
017236  001003                          BNE     33$                     ;                       4423
017240  062706  000010                  ADD     #10,SP
017244  000406                          BR      35$
017246  062706  000010          33$:    ADD     #10,SP                  ;                       4418
017252  005202                  34$:    INC     R2                      ; COL.NUM               4414
017254  020227  000377                  CMP     R2,#377                 ; COL.NUM,*
017260  003723                          BLE     32$
017262  062706  000020          35$:    ADD     #20,SP                  ;                       4392
017266  026627  000042  000016  36$:    CMP     42(SP),#16              ; QD.3.SUM,*            4431
017274  003565                          BLE     45$
017276  104454                          TRAP    54                      ;                       4434
017300  000007                          .WORD   7
017302  000474'                         .WORD   CON.C.MSG
017304  000000                          .WORD   0
017306  010546                          MOV     R5,-(SP)                ; NIB.NUM,*             4435
017310  016646  000032                  MOV     32(SP),-(SP)            ; BNK.NUM,*
017314  016646  000062                  MOV     62(SP),-(SP)            ; ARR.NUM,*
017320  012746  002172'                 MOV     #A.B.N.PRINT,-(SP)
017324  012746  000004                  MOV     #4,-(SP)
017330  010600                          MOV     SP,R0                   ; SP,*
017332  104414                          TRAP    14
017334  016616  000044                  MOV     44(SP),(SP)             ;                       4436
017340  062716  037022'                 ADD     #DEGRADE.MOD.REG,(SP)
017344  016646  000042                  MOV     42(SP),-(SP)            ; BNK.NUM,*
017350  042716  177770                  BIC     #177770,(SP)
017354  012746  000001                  MOV     #1,-(SP)
017360  011646                          MOV     (SP),-(SP)
017362  004767  000000G                 JSR     PC,BL$PU2
017366  016601  000062                  MOV     62(SP),R1               ; QD.3.SUM,OVER.FLOW    4437
017372  162701  000016                  SUB     #16,R1                  ; *,OVER.FLOW
017376  012702  000200                  MOV     #200,R2                 ; *,ROW.NUM             4439
017402  010304                  37$:    MOV     R3,R4                   ;                       4441
017404  060204                          ADD     R2,R4                   ; ROW.NUM,*
017406  006304                          ASL     R4
```

```
017410  062704  026770'                ADD     #BLAST.TBL,R4
017414  010446                          MOV     R4,-(SP)
017416  010546                          MOV     R5,-(SP)                ; NIB.NUM,*
017420  012746  000001                  MOV     #1,-(SP)
017424  005046                          CLR     -(SP)
017426  004767  000000G                JSR     PC,BL$GT2
017432  062706  000010                  ADD     #10,SP
017436  006000                          ROR     R0
017440  103016                          BCC     39$
017442  010446                          MOV     R4,-(SP)                ;                               4444
017444  010546                          MOV     R5,-(SP)                ; NIB.NUM,*
017446  012746  000001                  MOV     #1,-(SP)
017452  005046                          CLR     -(SP)
017454  004767  000000G                JSR     PC,BL$PU2
017460  005301                          DEC     R1                      ; OVER.FLOW             4445
017462  001003                          BNE     38$                     ;                       4447
017464  062706  000010                  ADD     #10,SP
017470  000406                          BR      40$
017472  062706  000010          38$:    ADD     #10,SP                  ;                       4443
017476  005202                  39$:    INC     R2                      ; ROW.NUM               4439
017500  020227  000377                  CMP     R2,#377                 ; ROW.NUM,*
017504  003736                          BLE     37$
017506  005701                  40$:    TST     R1                      ; OVER.FLOW             4451
017510  001455                          BEQ     44$
017512  012702  000200                  MOV     #200,R2                 ; *,COL.NUM             4455
017516  010204                  41$:    MOV     R2,R4                   ; COL.NUM,*             4457
017520  066704  036770'                ADD     COL.BASE,R4
017524  060304                          ADD     R3,R4
017526  006304                          ASL     R4
017530  062704  026770'                ADD     #BLAST.TBL,R4
017534  010446                          MOV     R4,-(SP)
017536  010546                          MOV     R5,-(SP)                ; NIB.NUM,*
017540  012746  000001                  MOV     #1,-(SP)
017544  005046                          CLR     -(SP)
017546  004767  000000G                JSR     PC,BL$GT2
017552  062706  000010                  ADD     #10,SP
017556  006000                          ROR     R0
017560  103025                          BCC     43$
017562  010204                          MOV     R2,R4                   ; COL.NUM,*                     4460
017564  066704  036770'                ADD     COL.BASE,R4
017570  060304                          ADD     R3,R4
017572  006304                          ASL     R4
017574  062704  026770'                ADD     #BLAST.TBL,R4
017600  010446                          MOV     R4,-(SP)
017602  010546                          MOV     R5,-(SP)                ; NIB.NUM,*
017604  012746  000001                  MOV     #1,-(SP)
017610  005046                          CLR     -(SP)
017612  004767  000000G                JSR     PC,BL$PU2
017616  005301                          DEC     R1                      ; OVER.FLOW             4461
017620  001003                          BNE     42$                     ;                       4463
017622  062706  000010                  ADD     #10,SP
017626  000406                          BR      44$
```

```
017630  062706  000010          42$:    ADD     #10,SP                          ;                                       4459
017634  005202                  43$:    INC     R2                              ; COL.NUM                               4455
017636  020227  000377                  CMP     R2,#377                         ; COL.NUM,*
017642  003725                          BLE     41$
017644  062706  000020          44$:    ADD     #20,SP                          ;                                       4437
017650  005205                  45$:    INC     R5                              ; NIB.NUM                               4274
017652  020527  000011                  CMP     R5,#11                          ; NIB.NUM,*
017656  003002                          BGT     46$
017660  000167  175152                  JMP     3$
017664  016616  000030          46$:    MOV     30(SP),(SP)                     ; BNK.NUM,*                             4475
017670  004767  174540                  JSR     PC,OR.OLD.NEW.PD
017674  062706  000010                  ADD     #10,SP                          ;                                       4231
017700  005266  000020          47$:    INC     20(SP)                          ; BNK.NUM                               4478
017704  005726                          TST     (SP)+                           ;                                       4226
017706  005266  000042                  INC     42(SP)                          ; SEL.BNK                               4225
017712  026666  000042  000006  48$:    CMP     42(SP),66(SP)                   ; SEL.BNK,TSTED.BNK
017720  002002                          BGE     49$
017722  000167  174764                  JMP     1$
017726  062706  000050          49$:    ADD     #50,SP                          ;                                       4171
017732  000207                          RTS     PC

; Routine Size:  815 words
; Maximum stack depth per invocation:  43 words
```

```
4482  routine IN_ERROR_MAP (TEMP_ARR$BNK_SEL, BAD_CHIP) : novalue =
4483      begin
4484
4485  !++
4486  !  FUNCTIONAL DESCRIPTION: INTERIGATE THE ERROR MAP
4487  !
4488  !                          THIS ROUTINE EXICUTES IN THE FOLLOWING
4489  !                          STEPS:
4490  !                            1. IT FIRST SEARCHES THE ERROR MAP FOR
4491  !                               ALL BAD ROWS ie. OCCURANCES OF BAD
4492  !                               ROWS > 10 TIMES.
4493  !
4494  !                            2. IT THEN SEARCHES THE COLUMN COUNT
4495  !                               TABLE FOR COLUMN COUNTS > 10>
4496  !
4497  !                            3. IT THEN XFERS TO THE REMAINDER TABLE
4498  !                               ANY REMAINING CELL FAILURES.
4499  !
4500  !                            4. IT THEN INTERIGATES THE REMAINDER TABLE
4501  !                               FOR THESE REMAINING CELL FAILURES
4502  !                               AND SELECTS THE BEST ONE FOR BLASTING.
4503  !
4504  !  FORMAL PARAMETERS:
4505  !                 TEMP_ARR$BNK_SEL:
4506  !                          STORES THE SELECTED ARRAYS ARRAY AND BANK
4507  !                          SELECT NUMBERS.
4508  !                 BAD_CHIP:
4509  !                          POINTS TO THE FAILING CHIP PERSENTLY
4510  !                          BEING PM'ED.
4511  !--
4512
4513      local
4514          ARR_SE                                  .STORES ARRAY SELECT NUMBER
4515          BNK_NUM,                                !STORES BANK SEL NUMBER
4516          ALL_BAD;                                !RECORDS TOTAL ALL BAD ROWS AND COLUMNS FOUND BAD
4517
4518      ARR_SEL = .TEMP_ARR$BNK_SEL<.ARR_SEL_POS, ARR$SEL_SIZE>;    !LOAD THE ARRAY SEL NUMBER
4519      BNK_NUM = .TEMP_ARR$BNK_SEL<.BNK_SEL_POS, BNK$SEL_SIZE>;    !LOAD THE BNK SEL NUMBER
4520      I_TMP_BLST_TBL ();                          !INIT THE TEMPORARY BLAST TABLE
4521      ALL_BAD = S_ERROR_MAP (.BAD_CHIP);          .SEARCH ERROR MAP FOR ALL BAD ROWS
4522
4523      if .ALL_BAD leq 9                           !IS THE ALL BAD COUNT <= 9
4524      then
4525          begin
4526          ALL_BAD = S_COL_CNT_TBL (.BAD_CHIP, .ALL_BAD);  !SEARCH COLUMN COUNT TABLE FOR ALL BAD COLS
4527
4528          if .ALL_BAD leq 9                       .IS THE ALL BAD COUNT <- 9
4529          then
4530              begin
4531              ALL_BAD = S_REM_TBL (.BAD_CHIP, .ALL_BAD);  .FIND THE REMAINING BAD ROWS/COLS
4532              end;
4533
```

```
;        4534          end;
;        4535
;        4536      if .ALL_BAD leq 9                                 WERE THERE <= 9 ALL BAD ROWS/COLS FOUND
;        4537      then
;        4538          begin
;        4539          X_TMP_BLST_TBL (.TEMP_ARR$BNK_SEL, .ALL_BAD);   !XFER TEMP BLAST TABLE TO BLAST TABLE
;        4540          end
;        4541      else
;        4542          begin
;        4543
;        4544          if not .FLG_REG [F_ALL_BAD_CHIP]          .IS THIS THE FIRST ALL BAD CHIP IN THIS BANK
;        4545          then
;        4546              begin
;        4547              ERRSF (ERR_5, CON_A_MSG, 0);          .REPORT A CONDITION A
;        4548              PRINTB (A_B_C_PRINT, .ARR_SEL, .BNK_NUM, .BAD_CHIP);          .REPORT WHERE
;        4549              FLG_REG [F_ALL_BAD_CHIP] = SET_FLG;  .SET THE ALL BAD CHIP FLAG
;        4550              DEGRADE_MOD_REG [.BNK_NUM] = SET_FLG;
;        4551              end
;        4552          else
;        4553              begin
;        4554              ERRSF (ERR_6, CON_B_MSG, 0);          .REPORT A CONDITION B
;        4555              PRINTB (REP_PRINT, .ARR_SEL);         !REPORT WHERE
;        4556              FLG_REG [F_ABORT_ARRAY] = SET_FLG;    .SET THE ABORT ARRAY FLAG
;        4557              end;
;        4558
;        4559          end;
;        4560
;        4561      end;


                          .SBTTL  IN.ERROR.MAP ROUTINE DECLARATIONS
017734  004167  000000G   IN.ERROR.MAP:
                          JSR     R1,$SAVE3                         ;                                    4482
017740  012746  000016    MOV     #16,-(SP)                         ;                                    4518
017744  060616            ADD     SP,(SP)                           ; TEMP.ARR$BNK.SE,*
017746  016746  037010'    MOV     ARR.SEL.POS,-(SP)
017752  012746  000004    MOV     #4,-(SP)
017756  005046            CLR     -(SP)
017760  004767  000000G   JSR     PC,BL$GT2
017764  010003            MOV     R0,R3                             ; *,ARR.SEL
017766  012716  000024    MOV     #24,(SP)                          ;                                    4519
017772  060616            ADD     SP,(SP)                           ; TEMP.ARR$BNK.SE,*
017774  016746  037012'    MOV     BNK.SEL.POS,-(SP)
020000  012746  000002    MOV     #2,-(SP)
020004  005046            CLR     -(SP)
020006  004767  000000G   JSR     PC,BL$GT2
020012  010001            MOV     R0,R1                             ; *,BNK.NUM
020014  004767  164404    JSR     PC,I.TMP.BLST.TBL                 ;                                    4520
020020  016602  000030    MOV     30(SP),R2                         ; BAD.CHIP,*                         4521
020024  010216            MOV     R2,(SP)
020026  004767  171740    JSR     PC,S.ERROR.MAP
020032  020027  000011    CMP     R0,#11                            ; ALL.BAD,*                          4523
```

```
020036  003026                          BGT     2$                                              4526
020040  010246                          MOV     R2,-(SP)        ;
020042  010046                          MOV     R0,-(SP)        ; ALL.BAD,*
020044  004767  171550                  JSR     PC,S.COL.CNT.TBL
020050  020027  000011                  CMP     R0,#11          ; ALL.BAD,*             4528
020054  003005                          BGT     1$                                      4531
020056  010246                          MOV     R2,-(SP)        ;
020060  010046                          MOV     R0,-(SP)        ; ALL.BAD,*
020062  004767  171142                  JSR     PC,S.REM.TBL
020066  022626                          CMP     (SP)+,(SP)+     ;                       4530
020070  022626                  1$:     CMP     (SP)+,(SP)+     ;                       4525
020072  020027  000011                  CMP     R0,#11          ; ALL.BAD,*             4536
020076  003006                          BGT     2$
020100  016646  000032                  MOV     32(SP),-(SP)    ; TEMP.ARR$BNK.SE,*     4539
020104  010046                          MOV     R0,-(SP)        ; ALL.BAD,*
020106  004767  166606                  JSR     PC,X.TMP.BLST.TBL
020112  000465                          BR      5$                                      4536
020114  032767  000020  037020'  2$:    BIT     #20,FLG.REG     ;                       4544
020122  001042                          BNE     3$
020124  104454                          TRAP    54                                      4547
020126  000005                          .WORD   5
020130  000444'                         .WORD   CON.A.MSG
020132  000000                          .WORD   0
020134  010246                          MOV     R2,-(SP)        ;                       4548
020136  010146                          MOV     R1,-(SP)        ; BNK.NUM,*
020140  010346                          MOV     R3,-(SP)        ; ARR.SEL,*
020142  012746  002120'                  MOV     #A.B.C.PRINT,-(SP)
020146  012746  000004                  MOV     #4,-(SP)
020152  010600                          MOV     SP,R0           ; SP,*
020154  104414                          TRAP    14                                      4549
020156  152767  000020  037020'         BISB    #20,FLG.REG     ;                       4550
020164  010100                          MOV     R1,R0           ; BNK.NUM,*
020166  006200                          ASR     R0
020170  006200                          ASR     R0
020172  006200                          ASR     R0
020174  062700  037022'                  ADD     #DEGRADE.MOD.REG,R0
020200  010016                          MOV     R0,(SP)
020202  010146                          MOV     R1,-(SP)        ; BNK.NUM,*
020204  042716  177770                  BIC     #177770,(SP)
020210  012746  000001                  MOV     #1,-(SP)
020214  011646                          MOV     (SP),-(SP)
020216  004767  000000G                 JSR     PC,BL$PU2
020222  062706  000012                  ADD     #12,SP          ;                       4546
020226  000416                          BR      4$              ;                       4544
020230  104454                  3$:     TRAP    54              ;                       4554
020232  000006                          .WORD   6
020234  000460'                         .WORD   CON.B.MSG
020236  000000                          .WORD   0
020240  010346                          MOV     R3,-(SP)        ; ARR.SEL,*             4555
020242  012746  002246'                  MOV     #REP.PRINT,-(SP)
020246  012746  000002                  MOV     #2,-(SP)
020252  010600                          MOV     SP,R0           ; SP,*
```

```
020254  104414                          TRAP    14
020256  152767  000100  C37020'         BISB    #100,FLG.REG                    ;                       4556
020264  005726                  4$:     TST     (SP)+                           ;                       4542
020266  062706  000022          5$:     ADD     #22,SP                          ;                       4483
020272  000207                          RTS     PC                              ;                       4482

; Routine Size:  112 words
; Maximum stack depth per invocation:  19 words
```

```
4562  routine FB_DATA_CHIPS (TEMP_ARR$BNK_SEL) : novalue =
4563      begin
4564
4565  !++
4566  !  FUNCTIONAL DESCRIPTION: FIND THE BAD DATA CHIPS
4567  !
4568  !                          THIS ROUTINE WILL SEARCH THE SELECTED BANKS
4569  !                          VIA MASS BUS TRANSFERS LOOKING FOR ADDITIONAL
4570  !                          FAILING CHIPS.
4571  !
4572  !                          IF A BAD CHIP IS DETECTED THEN THE ML-11 BIT
4573  !                          FROM WHICH THE FAILURE CAME FROM IS CALCULATED
4574  !                          AND THE FAILING CHIP NUMBER AND PATTERN IS
4575  !                          STORED INTO THE BAD CHIP TABLE.
4576  !
4577  !  FORMAL PARAMETERS:
4578  !                  TEMP_ARR$BNK_SEL:
4579  !                          STORES THE SELECTED ARRAYS ARRAY AND
4580  !                          BANK SELECT NUMBER.
4581  !--
4582
4583      local
4584          BITS_TSTED,                         !COUNTS NO. BITS TESTED IN READ BUFFER
4585          OFFSET,                             .RANDOM DATA OFFSET
4586          RAND_INDEX,                         !STARTS SECTOR TRANSFER AT RANDOM WRITE BUFFER WORDS
4587          BAD_BITS : bitvector [16],          .LOADED WITH FAILING READ BUFFER WORD
4588          FAIL_CHIP;                          !RECORDS THE FAILING ML-11 DATA CHIP
4589
4590      SEED1 = ZEROES;                         .INIT RANDOM DATA SEED1
4591      SEED2 = %o'1233';                       .INIT RANDOM DATA SEED2
4592      SEED3 = %o'7622';                       !INIT RANDOM DATA SEED3
4593      OFFSET = ZEROES;                        .CLEAR THE RANDOM DATA OFFSET FOR 1'S AND 0'S DATA
4594
4595      incr PAT_SEL from 0 to ONE + .RAND_PASS do  .LOOP THRU ALL PATTERNS
4596          begin
4597          RAND_INDEX - ZEROES;                .RESET RANDOM INDEX TO ZERO
4598
4599          selectone .PAT_SEL of               !SELECT A PATTERN
4600              set
4601
4602              [0] :
4603                  L_1_0_DATA (.TEMP_ARR$BNK_SEL, ZEROES); .LOAD ZEROES DATA
4604
4605              [1] :
4606                  L_1_0_DATA (.TEMP_ARR$BNK_SEL, ONES);   !LOAD ONES DATA
4607
4608              [otherwise] :
4609                  begin
4610                  L_RANDOM_DATA (.TEMP_ARR$BNK_SEL);      .LOAD RANDOM DATA
4611                  OFFSET = 2;                             .ENABLE THE RANDOM DATA OFFSET
4612                  end;
4613              tes;
```

```
4614
4615            incr SEC_CNT from 0 to .BNK_NUM_SEC do  .WRITE CHECK ALL SECTORS IN THIS BANK
4616                begin
4617                BREAK;
4618                WRT_CHK_TRANSFER (SIZE = -256, DST - (.TEMP_ARR$BNK_SEL + .SEC_CNT),
4619                    SRC = WRT_BUF [.RAND_INDEX,
4620                        WRD]);
4621
4622                if .ML_ADDR [MLEE, crc]             .WAS THERE A CRC BIT ERROR
4623                then
4624                    begin
4625                    FAIL_CHIP - .ML_ADDR [MLEE, CHAN];      .READ THE FAILING CRC BIT NUMBER
4626                    L_CHIP_TBL (.FAIL_CHIP, .PAT_SEL);      .STARE AWAY THE FAILING BIT AND PATTERN
4627                    end
4628                else
4629                    begin
4630
4631                    if .ML_ADDR [MLEE, UNC]         !WAS THERE A UNCORRECTABLE ERROR
4632                    then
4633                        begin
4634                        FLG_REG [F_UNC_ERR_FLG] = SET_FLG;  !SET THE UNC ERROR FLAG FOR FUTURE REF
4635                        end;
4636
4637                    end;
4638
4639                if .ML_ADDR [MLCS2, WCE]            .WAS THERE A WRITE CHECK ERROR
4640                then
4641                    begin
4642                    CLR_MBUS;                       .CLEAR THE MASS BUS OF THE ERROR
4643                    BITS_TSTED - ZEROES;            !CLEAR THE BITS TESTED COUNT
4644                    RD_TRANSFER (SIZE - -256, DST - RD_BUF, SRC = .TEMP_ARR$BNK_SEL + .SEC_CNT);
4645
4646                    incr BUF_INDEX from 0 to 255 do         !FIND THE FAILING READ BUFFER WORD
4647                        begin
4648
4649                        if .WRT_BUF [(.RAND_INDEX + .BUF_INDEX), WRD] neq .RD_BUF [.BUF_INDEX]
4650                        then
4651                            begin
4652                            BAD_BITS - .WRT_BUF [(.RAND_INDEX + .BUF_INDEX), WRD] xor .RD_BUF [.BUF_INDEX];
4653
4654                            incr BIT_INDEX from 0 to 15 do  .FIND THE FAILING BIT(S)
4655
4656                                if .BAD_BITS [.BIT_INDEX] eql ONE   !IS THIS A BAD BIT
4657                                then
4658                                    begin
4659                                    FAIL_CHIP = (.BITS_TSTED + .BIT_INDEX) mod 36;
4660                                                    !CALCULATE THE FAILING ML-11 BIIT
4661                                    L_CHIP_TBL (.FAIL_CHIP, .PAT_SEL);      .STORE AWAY THE BIT AND PATTERN
4662                                    end;
4663
4664                            end;
4665
```

```
;         4666                    BITS_TSTED = .BITS_TSTED + 16;      !ADD 16 BIT TO THE BIT TESTED COUNT
;         4667                    end;
;         4668
;         4669                end;
;         4670
;         4671            RAND_INDEX = .RAND_INDEX + .OFFSET; .BUMP THE RANDOM INDEX IF RANDOM DATA
;         4672            end;
;         4673
;         4674        end;
;         4675
;         4676    end;
```

```
                                    .SBTTL  FB.DATA.CHIPS ROUTINE DECLARATIONS
020274  004167  000000G       FB.DATA.CHIPS:
                                    JSR     R1,$SAVE5                       ;                               4562
020300  162706  000034              SUB     #34,SP
020304  005067  000000G             CLR     SEED1                           ;                               4590
020310  012767  001233  000000G     MOV     #1233,SEED2                     ;                               4591
020316  012767  007622  000000G     MOV     #7622,SEED3                     ;                               4592
020324  005066  000006              CLR     6(SP)                           ; OFFSET                        4593
020330  016705  037034'             MOV     RAND.PASS,R5                    ;                               4595
020334  010566  000012              MOV     R5,12(SP)
020340  005266  000012              INC     12(SP)
020344  005001                      CLR     R1                              ; PAT.SEL
020346  000167  000732              JMP     16$
020352  005066  000004      1$:     CLR     4(SP)                           ; RAND.INDEX                    4597
020356  005701                      TST     R1                              ; PAT.SEL                       4599
020360  001004                      BNE     2$
020362  016646  000052              MOV     52(SP),-(SP)                    ; TEMP.ARR$BNK.SE,*             4603
020366  005046                      CLR     -(SP)
020370  000407                      BR      3$
020372  020127  000001      2$:     CMP     R1,#1                           ; PAT.SEL,*                     4599
020376  001010                      BNE     4$
020400  016646  000052              MOV     52(SP),-(SP)                    ; TEMP.ARR$BNK.SE,*             4606
020404  012746  177777              MOV     #-1,-(SP)
020410  004767  164310      3$:     JSR     PC,L.1.0.DATA
020414  005726                      TST     (SP)+
020416  000407                      BR      5$                              ;                               4599
020420  016646  000052      4$:     MOV     52(SP),-(SP)                    ; TEMP.ARR$BNK.SE,*             4610
020424  004767  164164              JSR     PC,L.RANDOM.DATA
020430  012766  000002  000010      MOV     #2,10(SP)                       ; *,OFFSET                      4611
020436  016766  037014' 000016      MOV     BNK.NUM.SEC,16(SP)              ;                               4615
020444  005066  000002      5$:     CLR     2(SP)                           ; SEC.CNT
020450  000167  000610              JMP     14$
020454  104422              6$:     TRAP    22                              ;                               4616
020456  012746  177400              MOV     #-400,-(SP)                     ;                               4618
020462  011667  037024'             MOV     (SP),SIZE
020466  016603  000056              MOV     56(SP),R3                       ; TEMP.ARR$BNK.SE,*
020472  066603  000004              ADD     4(SP),R3                        ; SEC.CNT,*
020476  010367  037026'             MOV     R3,DST
020502  010346                      MOV     R3,-(SP)
```

```
020504  016604  000012                  MOV     12(SP),R4                       ; RAND.INDEX,*          4620
020510  006304                          ASL     R4
020512  062704  001232'                 ADD     #WRT.BUF,R4
020516  010467  037030'                 MOV     R4,SRC                          ;                       4619
020522  010446                          MOV     R4,-(SP)                        ;                       4618
020524  004767  161704                  JSR     PC,WRT.CHK.TRANSFE
020530  016705  036772'                 MOV     ML.ADDR,R5                      ;                       4622
020534  016566  000042  000042         MOV     42(R5),42(SP)                   ; *,ML.REG
020542  032766  020000  000042         BIT     #20000,42(SP)                   ; *,ML.REG
020550  001427                          BEQ     7$
020552  016705  036772'                 MOV     ML.ADDR,R5                      ;                       4625
020556  016566  000042  000040         MOV     42(R5),40(SP)                   ; *,ML.REG
020564  016605  000040                  MOV     40(SP),R5                       ; ML.REG,*
020570  006205                          ASR     R5
020572  006205                          ASR     R5
020574  006205                          ASR     R5
020576  006205                          ASR     R5
020600  006205                          ASR     R5
020602  006205                          ASR     R5
020604  042705  177700                  BIC     #177700,R5
020610  010566  000012                  MOV     R5,12(SP)                       ; *,FAIL.CHIP
020614  010546                          MOV     R5,-(SP)                        ; FAIL.CHIP,*           4626
020616  010146                          MOV     R1,-(SP)                        ; PAT.SEL,*
020620  004767  164214                  JSR     PC,L.CHIP.TBL
020624  022626                          CMP     (SP)+,(SP)+                     ;                       4624
020626  000411                          BR      8$                              ;                       4622
020630  016705  036772'          7$:    MOV     ML.ADDR,R5                      ;                       4631
020634  016566  000042  000036         MOV     42(R5),36(SP)                   ; *,ML.REG
020642  100003                          BPL     8$
020644  152767  000002  037020'         BISB    #2,FLG.REG                      ;                       4634
020652  016705  036772'          8$:    MOV     ML.ADDR,R5                      ;                       4639
020656  016566  000010  000034         MOV     10(R5),34(SP)                   ; *,ML.REG
020664  032766  040000  000034         BIT     #40000,34(SP)                   ; *,ML.REG
020672  001565                          BEQ     13$
020674  016705  036772'                 MOV     ML.ADDR,R5                      ;                       4641
020700  016566  000010  000032         MOV     10(R5),32(SP)                   ; *,ML.REG
020706  016604  000032                  MOV     32(SP),R4                       ; ML.REG,MLREG
020712  152704  000040                  BISB    #40,R4                          ; *,MLREG
020716  016705  036772'                 MOV     ML.ADDR,R5
020722  010465  000010                  MOV     R4,10(R5)                       ; MLREG,*
020726  016705  036772'                 MOV     ML.ADDR,R5
020732  016566  000010  000030         MOV     10(R5),30(SP)                   ; *,ML.REG
020740  016604  000030                  MOV     30(SP),R4                       ; ML.REG,MLREG
020744  016705  036776'                 MOV     ML.DUT,R5
020750  042705  177770                  BIC     #177770,R5
020754  142704  000007                  BICB    #7,R4                           ; *,MLREG
020760  050504                          BIS     R5,R4                           ; *,MLREG
020762  016705  036772'                 MOV     ML.ADDR,R5
020766  010465  000010                  MOV     R4,10(R5)                       ; MLREG,*
020772  005066  000020                  CLR     20(SP)                          ; BITS.TSTED            4643
020776  012746  177400                  MOV     #-400,-(SP)                     ;                       4644
021002  011667  037024'                 MOV     (SP),SIZE
```

```
021006  012746  005626'                    MOV     #RD.BUF,-(SP)
021012  011667  037026'                    MOV     (SP),DST
021016  010367  037030'                    MOV     R3,SRC
021022  010346                            MOV     R3,-(SP)
021024  004767  162634                    JSR     PC,RD.TRANSFER
021030  005002                            CLR     R2                      ; BUF.INDEX                     4646
021032  010204                    9$:     MOV     R2,R4                   ; BUF.INDEX,*                   4649
021034  066604  000022                    ADD     22(SP),R4               ; RAND.INDEX,*
021040  006304                            ASL     R4
021042  010205                            MOV     R2,R5                   ; BUF.INDEX,*
021044  006305                            ASL     R5
021046  026465  001232' 005626'           CMP     WRT.BUF(R4),RD.BUF(R5)
021054  001463                            BEQ     12$
021056  016403  001232'                    MOV     WRT.BUF(R4),R3          ;                               4652
021062  016546  005626'                    MOV     RD.BUF(R5),-(SP)
021066  040316                            BIC     R3,(SP)
021070  046503  005626'                    BIC     RD.BUF(R5),R3
021074  052603                            BIS     (SP)+,R3
021076  010366  000034                    MOV     R3,34(SP)               ; *,BAD.BITS
021102  005005                            CLR     R5                      ; BIT.INDEX                     4654
021104  010504                    10$:    MOV     R5,R4                   ; BIT.INDEX,*                   4656
021106  006204                            ASR     R4
021110  006204                            ASR     R4
021112  006204                            ASR     R4
021114  012703  000034                    MOV     #34,R3
021120  060603                            ADD     SP,R3                   ; BAD.BITS,*
021122  060304                            ADD     R3,R4
021124  010446                            MOV     R4,-(SP)
021126  010546                            MOV     R5,-(SP)                ; BIT.INDEX,*
021130  042716  177770                    BIC     #177770,(SP)
021134  012746  000001                    MOV     #1,-(SP)
021140  005046                            CLR     -(SP)
021142  004767  000000G                   JSR     PC,BL$GT2
021146  062706  000010                    ADD     #10,SP
021152  005300                            DEC     R0
021154  001017                            BNE     11$
021156  010546                            MOV     R5,-(SP)                ; BIT.INDEX,*                   4659
021160  066616  000030                    ADD     30(SP),(SP)             ; BITS.TSTED,*
021164  012746  000044                    MOV     #44,-(SP)
021170  004767  000000G                   JSR     PC,BL$MOD
021174  010066  000024                    MOV     R0,24(SP)               ; *,FAIL.CHIP
021200  010016                            MOV     R0,(SP)                 ; FAIL.CHIP,*                   4661
021202  010146                            MOV     R1,-(SP)                ; PAT.SEL,*
021204  004767  163630                    JSR     PC,L.CHIP.TBL
021210  062706  000006                    ADD     #6,SP                   ;                               4658
021214  005205                    11$:    INC     R5                      ; BIT.INDEX                     4654
021216  020527  000017                    CMP     R5,#17                  ; BIT.INDEX,*
021222  003730                            BLE     10$
021224  062766  000020  000026    12$:    ADD     #20,26(SP)              ; *,BITS.TSTED                  4666
021232  005202                            INC     R2                      ; BUF.INDEX                     4646
021234  020227  000377                    CMP     R2,#377                 ; BUF.INDEX,*
021240  003674                            BLE     9$
```

```
021242  062706  000006                          ADD     #6,SP                           ;                                       4641
021246  066666  000016  000014        13$:      ADD     16(SP),14(SP)                   ; OFFSET,RAND.INDEX                     4671
021254  062706  000006                          ADD     #6,SP                           ;                                       4616
021260  005266  000002                          INC     2(SP)                           ; SEC.CNT                               4615
021264  026666  000002  000016        14$:      CMP     2(SP),16(SP)                    ; SEC.CNT,*
021272  003002                                  BGT     15$
021274  000167  177154                          JMP     6$
021300  005726                      15$:        TST     (SP)+                           ;                                       4596
021302  005201                                  INC     R1                              ; PAT.SEL                               4595
021304  020166  000012                  16$:    CMP     R1,12(SP)                       ; PAT.SEL,*
021310  003002                                  BGT     17$
021312  000167  177034                          JMP     1$
021316  062706  000034                  17$:    ADD     #34,SP                          ;                                       4562
021322  000207                                  RTS     PC

; Routine Size:  268 words
; Maximum stack depth per invocation:  31 words
```

```
4677  routine FB_CRC_CHIPS (TEMP_ARR$BNK_SEL) : novalue =
4678      begin
4679
4680  !++
4681  !  FUNCTIONAL DESCRIPTION: FIND THE BAD CRC CHIPS
4682  !
4683  !                          THIS ROUTINE FINDS ALL THE ADDITIONAL FAILING
4684  !                          CRC CHIPS FOR A SELECTED BANK.
4685  !
4686  !                          THIS ROUTINE IS ONLY CALLED IF A UNC ERROR
4687  !                          OCCURED DURING THE FIND BAD DATA CHIP ROUTINE.
4688  !
4689  !                          IF A BAD CRC CHIP IS FOUND THEN IT IS STORED
4690  !                          INTO THE BAD CHIP TABLE ALONG WITH ITS FAILING
4691  !                          DATA PATTERN.
4692  !
4693  !  FORMAL PARAMETERS:
4694  !                  TEMP_ARR$BNK_SEL:
4695  !                          STORES THE SELECTED ARRAYS ARRAY AND BANK
4696  !                          SELECT NUMBERS.
4697  !--
4698
4699      local
4700          RAND_INDEX,                         !STARTS SECTOR TRANSFERS AT RANDOM WRITE BUFFER WORDS
4701          OFFSET,                             !RANDOM DATA OFFSET
4702          PD_SAVE : bitvector [16];           !SAVE LOCATION FOR NIBBLE PROM DATA
4703
4704      OFFSET = ZERO;                          !CLEAR THE RANDOM DATA OFFSET FOR 1'S AND 0'S DATA
4705      SEED1 = ZERO;                           !INIT THE RANDOM DATA SEED1
4706      SEED2 = %o'1233';                       !INIT THE RANDOM DATA SEED2
4707      SEED3 = %o'7622';                       !INIT THE RANDOM DATA SEED3
4708
4709      incr CRC_SEL from CRC_P to CRC_B do     !FIND BAD CRC BITS P, A, B
4710          begin
4711
4712          if not .CHIP_TBL [.CRC_SEL, FAULT]  !DID THIS CRC BIT FAIL DURING 'FB_DATA_CHIPS'
4713          then
4714              begin
4715
4716              incr PAT_SEL from 0 to ONE + .RAND_PASS do  !LOOP THRU ALL PATTERNS
4717                  begin
4718                  RAND_INDEX = ZERO;          !RESET THE RANDOM INDEX
4719
4720                  selectone .PAT_SEL of       !SELECT A PATTERN
4721                      set
4722
4723                      [0] :
4724                          DM_1_0_LOAD (.TEMP_ARR$BNK_SEL, ZEROES);        !LOAD ZEROES DATA
4725
4726                      [1] :
4727                          DM_1_0_LOAD (.TEMP_ARR$BNK_SEL, ONES);  !LOAD ONES DATA
4728
```

```
4729                        [otherwise] :
4730                            begin
4731                            DM_RAND_LOAD (.TEMP_ARR$BNK_SEL);       !LOAD RANDOM DATA
4732                            OFFSET = 2;                 .ENABLE THE RANDOM DATA OFFSET
4733                            end;
4734                        tes;
4735
4736                    DM_RD_TRANSFER (SIZE = -256, DST = RD_BUF, SRC = .TEMP_ARR$BNK_SEL);
4737
4738                    case .CRC_SEL from CRC_P to CRC_B of    !SELECT THE FAILING CRC BIT
4739                        set
4740
4741                        [36] :                          !TEST CRC BIT P
4742                            begin
4743
4744                            incr SEC_CNT from 0 to .BNK_NUM_SEC do  !READ THRU ALL SECTORS
4745                                begin
4746                                BREAK;
4747
4748                                incr WRD_CNT from 0 to 127 do       !READ THRU ALL WORDS IN SECTOR
4749                                    begin
4750                                    PD_SAVE = .ML_ADDR [MLPD, PD_REG];      !READ OUT THE PROM DATA FOR THIS WORD
4751                                    DAT_CLK;            !CLOCK THE DATA INTO THE DIAG REG'S
4752
4753                                    if not .PD_SAVE [CRC_NIBBLE]    !IS THIS A GOOD NIBBLE
4754                                    then
4755                                        begin
4756
4757                                        if .ML_ADDR [MLE2, B_36] neq .WRT_BUF [(.RAND_INDEX + .WRD_CNT), BIT_12]
4758                                        then
4759                                            begin
4760                                            L_CHIP_TBL (.CRC_SEL, .PAT_SEL);
4761                                                    !SAVE THE FAILING CHIP AND PATTERN
4762                                            end;
4763
4764                                        end;
4765
4766                                    end;
4767
4768                                RAND_INDEX = .RAND_INDEX + .OFFSET; .BUMP THE RANDOM INDEX
4769                                end;
4770
4771                            end;
4772
4773                        [37] :                          !TEST CRC BIT A
4774                            begin
4775
4776                            incr SEC_CNT from 0 to .BNK_NUM_SEC do  !READ THRU ALL SECTORS
4777                                begin
4778                                BREAK;
4779
4780                                incr WRD_CNT from 0 to 127 do       .READ THRU ALL WORDS IN THIS SECTOR
```

```
4781                                    begin
4782                                    PD_SAVE = .ML_ADDR [MLPD, PD_REG];      !GET THIS WORDS FROM DATA
4783                                    DAT_CLK;        !CLOCK OUT DATA INTO THE DIAG REG'S
4784
4785                                    if not .PD_SAVE [CRC_NIBBLE]    .IS THIS A GOOD NIBBLE
4786                                    then
4787                                        begin
4788
4789                                        if .ML_ADDR [MLE2, B_37] neq .WRT_BUF [(.RAND_INDEX + .WRD_CNT), BIT_13]
4790                                        then
4791                                            begin
4792                                            L_CHIP_TBL (.CRC_SEL, .PAT_SEL);        .SAVE THE BAD CHIP AND PATTERN
4793                                            end;
4794
4795                                        end;
4796
4797                                    end;
4798
4799                                RAND_INDEX = .RAND_INDEX + .OFFSET; .BUMP THE RANDOM INDEX
4800                                end;
4801
4802                            end;
4803
4804                        [38] :                      .TEST CRC BIT B
4805                            begin
4806
4807                            incr SEC_CNT from 0 to .BNK_NUM_SEC do  .READ ALL SECTORS
4808                                begin
4809                                BREAK;
4810
4811                                incr WRD_CNT from 0 to 127 do       .READ ALL WORDS IN THIS SECTOR
4812                                    begin
4813                                    PD_SAVE = .ML_ADDR [MLPD, PD_REG];      .GET THE PROM DATA FOR
4814                                    DAT_CLK;        !CLOCK OUT DATA INTO DIAG REG'S
4815
4816                                    if not .PD_SAVE [CRC_NIBBLE]    !IS THIS A GOOD NIBBLE
4817                                    then
4818                                        begin
4819
4820                                        if .ML_ADDR [MLE2, B_38] neq .WRT_BUF [(.RAND_INDEX + .WRD_CNT), BIT_14]
4821                                        then
4822                                            begin
4823                                            L_CHIP_TBL (.CRC_SEL, .PAT_SEL);        .SAVE THE BAD CHIP AND PATTERN
4824                                            end;
4825
4826                                        end;
4827
4828                                    end;
4829
4830                                RAND_INDEX   .RAND_INDEX + .OFFSET; .BUMP THE RANDOM INDEX
4831                                end;
4832
```

```
;          4833                        end
;          4834                    tes;
;          4835
;          4836                end;
;          4837
;          4838            end;
;          4839
;          4840        end;
;          4841
;          4842    CLR_MBUS;                       .CLEAR THE SINGLE STEP DMA MODE
;          4843    end;


                                        .SBTTL  FB.CRC.CHIPS ROUTINE DECLARATIONS
021324  004167  000000G             FB.CRC.CHIPS:
                                        JSR     R1,$SAVE5               ;                               4677
021330  162706  000042                  SUB     #42,SP
021334  005066  000002                  CLR     2(SP)                   ; OFFSET                        4704
021340  005067  000000G                 CLR     SEED1                   ;                               4705
021344  012767  001233  000000G         MOV     #1233,SEED2             ;                               4706
021352  012767  007622  000000G         MOV     #7622,SEED3             ;                               4707
021360  012701  000044                  MOV     #44,R1                  ; *,CRC.SEL                     4709
021364  010105                  1$:     MOV     R1,R5                   ; CRC.SEL,*                     4712
021366  006305                          ASL     R5
021370  005765  026626'                 TST     CHIP.TBL(R5)
021374  100002                          BPL     2$
021376  000167  001052                  JMP     27$
021402  016705  037034'          2$:     MOV     RAND.PASS,R5            ;                               4716
021406  010566  000010                  MOV     R5,10(SP)
021412  005266  000010                  INC     10(SP)
021416  005002                          CLR     R2                      ; PAT.SEL
021420  000167  001016                  JMP     26$
021424  005016                  3$:     CLR     (SP)                    ; RAND.INDEX                    4718
021426  005702                          TST     R2                      ; PAT.SEL                       4720
021430  001004                          BNE     4$
021432  016646  000060                  MOV     60(SP),-(SP,            ; TEMP.ARR$BNK.SE,*             4724
021436  005046                          CLR     -(SP)
021440  000407                          BR      5$
021442  020227  000001          4$:     CMP     R2,#1                   ; PAT.SEL,*                     4720
021446  001010                          BNE     6$
021450  016646  000060                  MOV     60(SP),-(SP)            ; TEMP.ARR$BNK.SE,*             4727
021454  012746  177777                  MOV     #-1,-(SP)
021460  004767  160400          5$:     JSR     PC,DM.1.0.LOAD
021464  005726                          TST     (SP)+
021466  000407                          BR      7$                      ;                               4720
021470  016646  000060          6$:     MOV     60(SP),-(SP)            ; TEMP.ARR$BNK.SE,*             4731
021474  004767  157770                  JSR     PC,DM.RAND.LOAD
021500  012766  000002  000004          MOV     #2,4(SP)                ; *,OFFSET                      4732
021506  012716  177400          7$:     MOV     #-400,(SP)              ;                               4736
021512  011667  037024'                 MOV     (SP),SIZE
021516  012746  005626'                 MOV     #RD.BUF,-(SP)
021522  011667  037026'                 MOV     (SP),DST
```

```
021526  016646  000064                          MOV     64(SP),-(SP)            ; TEMP.ARR$BNK.SE,*
021532  011667  037030'                        MOV     (SP),SRC
021536  004767  157116                        JSR     PC,DM.RD.TRANSFER
021542  010105                                MOV     R1,R5                   ; CRC.SEL,*                     4738
021544  162705  000044                        SUB     #44,R5
021550  006305                                ASL     R5
021552  066507  021556'                        ADD     8$(R5),PC
                                        8$:     .WORD   9$-8$
021556  000006                                .WORD   14$-8$
021560  000224                                .WORD   20$-8$
021562  000442                                                                  4744
021564  016766  037014'  000014        9$:     MOV     BNK.NUM.SEC,14(SP)      ;
021572  005066  000012                        CLR     12(SP)                  ; SEC.CNT
021576  000474                                BR      13$                             4745
021600  104422                          10$:    TRAP    22                      ;                       4748
021602  005003                                CLR     R3                      ; WRD.CNT               4750
021604  016705  036772'                11$:    MOV     ML.ADDR,R5              ;
021610  016566  000046  000046                MOV     46(R5),46(SP)           ; *,ML.REG
021616  016666  000046  000042                MOV     46(SP),42(SP)           ; ML.REG,PD.SAVE
021624  016705  036772'                        MOV     ML.ADDR,R5
021630  016566  000024  000044                MOV     24(R5),44(SP)           ; *,ML.REG
021636  016604  000044                        MOV     44(SP),R4               ; ML.REG,MLREG
021642  152704  000020                        BISB    #20,R4                  ; *,MLREG
021646  016705  036772'                        MOV     ML.ADDR,R5
021652  010465  000024                        MOV     R4,24(R5)               ; MLREG,*
021656  132766  000002  000043                BITB    #2,43(SP)               ; *,PD.SAVE+1                   4753
021664  001030                                BNE     12$                             4757
021666  016705  036772'                        MOV     ML.ADDR,R5              ;
021672  016566  000034  000040                MOV     34(R5),40(SP)           ; *,ML.REG
021700  010305                                MOV     R3,R5                   ; WRD.CNT,*
021702  066605  000006                        ADD     6(SP),R5                ; RAND.INDEX,*
021706  006305                                ASL     R5
021710  016500  001232'                        MOV     WRT.BUF(R5),R0
021714  042700  167777                        BIC     #167777,R0
021720  016604  000040                        MOV     40(SP),R4               ; ML.REG,*
021724  042704  167777                        BIC     #167777,R4
021730  020400                                CMP     R4,R0
021732  001405                                BEQ     12$
021734  010146                                MOV     R1,-(SP)                ; CRC.SEL,*                     4760
021736  010246                                MOV     R2,-(SP)                ; PAT.SEL,*
021740  004767  163074                        JSR     PC,L.CHIP.TBL
021744  022626                                CMP     (SP)+,(SP)+             ;                               4759
021746  005203                          12$:    INC     R3                      ; WRD.CNT                       4748
021750  020327  000177                        CMP     R3,#177                 ; WRD.CNT,*
021754  003713                                BLE     11$
021756  066666  000010  000006                ADD     10(SP),.6(SP)           ; OFFSET,RAND.INDEX             4768
021764  005266  000012                        INC     12(SP)                  ; SEC.CNT                       4744
021770  026666  000012  000014        13$:    CMP     12(SP),14(SP)           ; SEC.CNT,*
021776  003700                                BLE     10$
022000  000506                                BR      19$                             4739
022002  016766  037014'  000014        14$:    MOV     BNK.NUM.SEC,14(SP)      ;                       4776
022010  005066  000012                        CLR     12(SP)                  ; SEC.CNT
022014  000474                                BR      18$
```

```
022016  104422                  15$:    TRAP    22                      ;                               4777
022020  005003                          CLR     R3                      ; WRD.CNT                       4780
022022  016705  036772'          16$:    MOV     ML.ADDR,R5              ;                               4782
022026  016566  000046  000036          MOV     46(R5),36(SP)           ; *,ML.REG
022034  016666  000036  000042          MOV     36(SP),42(SP)           ; ML.REG,PD.SAVE
022042  016705  036772'                  MOV     ML.ADDR,R5
022046  016566  000024  000034          MOV     24(R5),34(SP)           ; *,ML.REG
022054  016604  000034                  MOV     34(SP),R4               ; ML.REG,MLREG
022060  152704  000020                  BISB    #20,R4                  ; *,MLREG
022064  016705  036772'                  MOV     ML.ADDR,R5
022070  010465  000024                  MOV     R4,24(R5)               ; MLREG,*
022074  132766  000002  000043          BITB    #2,43(SP)               ; *,PD.SAVE+1                   4785
022102  001030                          BNE     17$
022104  016705  036772'                  MOV     ML.ADDR,R5              ;                               4789
022110  016566  000034  000032          MOV     34(R5),32(SP)           ; *,ML.REG
022116  010305                          MOV     R3,R5                   ; WRD.CNT,*
022120  066605  000006                  ADD     6(SP),R5                ; RAND.INDEX,*
022124  006305                          ASL     R5
022126  016500  001232'                  MOV     WRT.BUF(R5),R0
022132  042700  157777                  BIC     #157777,R0
022136  016604  000032                  MOV     32(SP),R4               ; ML.RFG,*
022142  042704  157777                  BIC     #157777,R4
022146  020400                          CMP     R4,R0
022150  001405                          BEQ     17$
022152  010146                          MOV     R1,-(SP)                ; CRC.SEL,*                     4792
022154  010246                          MOV     R2,-(SP)                ; PAT.SEL,*
022156  004767  162656                  JSR     PC,L.CHIP.TBL
022162  022626                          CMP     (SP)+,(SP)+             ;                               4791
022164  005203                  17$:    INC     R3                      ; WRD.CNT                       4780
022166  020327  000177                  CMP     R3,#177                 ; WRD.CNT,*
022172  003713                          BLE     16$
022174  066666  000010  000006          ADD     10(SP),6(SP)            ; OFFSET,RAND.INDEX             4799
022202  005266  000012                  INC     12(SP)                  ; SEC.CNT                       4776
022206  026666  000012  000014  18$:    CMP     12(SP),14(SP)           ; SEC.CNT,*
022214  003700                          BLE     15$
022216  000506                  19$:    BR      25$                     ;                               4738
022220  016766  037014' 000014  20$:    MOV     BNK.NUM.SEC,14(SP)      ;                               4807
022226  005066  000012                  CLR     12(SP)                  ; SEC.CNT
022232  000474                          BR      24$
022234  104422                  21$:    TRAP    22                      ;                               4808
022236  005003                          CLR     R3                      ; WRD.CNT                       4811
022240  016705  036772'          22$:    MOV     ML.ADDR,R5              ;                               4813
022244  016566  000046  000030          MOV     46(R5),30(SP)           ; *,ML.REG
022252  016666  000030  000042          MOV     30(SP),42(SP)           ; ML.REG,PD.SAVE
022260  016705  036772'                  MOV     ML.ADDR,R5
022264  016566  000024  000026          MOV     24(R5),26(SP)           ; *,ML.REG
022272  016604  000026                  MOV     26(SP),R4               ; ML.REG,MLREG
022276  152704  000020                  BISB    #20,R4                  ; *,MLREG
022302  016705  036772'                  MOV     ML.ADDR,R5
022306  010465  000024                  MOV     R4,24(R5)               ; MLREG,*
022312  132766  000002  000043          BITB    #2,43(SP)               ; *,PD.SAVE+1                   4816
022320  001030                          BNE     23$
```

```
022322  016705  036772'                MOV     ML.ADDR,R5                      ;                               4820
022326  016566  000034  000024        MOV     34(R5),24(SP)                   ; *,ML.REG
022334  010305                        MOV     R3,R5                           ; WRD.CNT,*
022336  066605  000006                ADD     6(SP),R5                        ; RAND.INDEX,*
022342  006305                        ASL     R5
022344  016500  001232'                MOV     WRT.BUF(R5),R0
022350  042700  137777                BIC     #137777,R0
022354  016604  000024                MOV     24(SP),R4                       ; ML.REG,*
022360  042704  137777                BIC     #137777,R4
022364  020400                        CMP     R4,R0
022366  001405                        BEQ     23$
022370  010146                        MOV     R1,-(SP)                        ; CRC.SEL,*                     4823
022372  010246                        MOV     R2,-(SP)                        ; PAT.SE ,*
022374  004767  162440                JSR     PC,L.CHIP.TBL
022400  022626                        CMP     (SP)+,(SP)+                     ;                               4822
022402  005203                23$:    INC     R3                              ; WRD.CNT                       4811
022404  020327  000177                CMP     R3,#177                         ; WRD.CNT,*
022410  003713                        BLE     22$
022412  066666  000010  000006        ADD     10(SP),6(SP)                    ; OFFSET,RAND.INDEX             4830
022420  005266  000012                INC     12(SP)                          ; SEC.CNT                       4807
022424  026666  000012  000014  24$:  CMP     12(SP),14(SP)                   ; SEC.CNT,*
022432  003700                        BLE     21$
022434  062706  000006          25$:  ADD     #6,SP                           ;                               4717
022440  005202                        INC     R2                              ; PAT.SEL                       4716
022442  020266  000010          26$:  CMP     R2,10(SP)                       ; PAT.SEL,*
022446  003002                        BGT     27$
022450  000167  176750                JMP     3$
022454  005201                27$:    INC     R1                              ; CRC.SEL                       4709
022456  020127  000046                CMP     R1,#46                          ; CRC.SEL,*
022462  003002                        BGT     28$
022464  000167  176674                JMP     1$
022470  016705  036772'          28$:  MOV     ML.ADDR,R5                      ;                               4840
022474  016566  000010  000014        MOV     10(R5),14(SP)                   ; *,ML.REG
022502  016604  000014                MOV     14(SP),R4                       ; ML.REG,MLREG
022506  152704  000040                BISB    #40,R4                          ; *,MLREG
022512  016705  036772'                MOV     ML.ADDR,R5
022516  010465  000010                MOV     R4,10(R5)                       ; MLREG,*
022522  016705  036772'                MOV     ML.ADDR,R5
022526  016566  000010  000012        MOV     10(R5),12(SP)                   ; *,ML.REG
022534  016604  000012                MOV     12(SP),R4                       ; ML.REG,MLREG
022540  016705  036776'                MOV     ML.DUT,R5
022544  042705  177770                BIC     #177770,R5
022550  142704  000007                BICB    #7,R4                           ; *,MLREG
022554  050504                        BIS     R5,R4                           ; *,MLREG
022556  016705  036772'                MOV     ML.ADDR,R5
022562  010465  000010                MOV     R4,10(R5)                       ; MLREG,*
022566  062706  000042                ADD     #42,SP                          ;                               4677
022572  000207                        RTS     PC

; Routine Size:  340 words
; Maximum stack depth per invocation:  28 words
```

```
;        4844  routine FB_CHIPS (TEMP_ARR$BNK_SEL) : novalue =
;        4845      begin
;        4846
;        4847  !++
;        4848  !  FUNCTIONAL DESCRIPTION: FIND BAD CHIPS
;        4849  !
;        4850  !                          THIS ROUTINE CALLS THE FIND BAD DATA CHIP AND
;        4851  !                          FIND BAD CRC CHIP ROUTINES.
;        4852  !
;        4853  !                          IF THE FB_DATA_CHIP ROUTINE DETECTES A UNC
;        4854  !                          ERROR THEN THE FB_CRC_CHIP ROUTINE IS NOT
;        4855  !                          CALLED.
;        4856  !
;        4857  !  FORMAL PARAMETERS:
;        4858  !                  TEMP_ARR$BNK_SEL:
;        4859  !                          STORES THE SELECTED ARRAYS ARRAY AND BANK
;        4860  !                          SELECT NUMBERS.
;        4861  !--
;        4862
;        4863      FLG_REG [F_UNC_ERR_FLG] = CLR_FLG;          !CLEAR THE UNCORRECTABLE ERROR FLAG
;        4864      I_CHIP_TBL ();                              .INIT THE BAD CHIP TABLE
;        4865      FB_DATA_CHIPS (.TEMP_ARR$BNK_SEL);          .FIND ALL THE BAD DATA CHIPS 0-35 IN THIS BANK
;        4866
;        4867      if .FLG_REG [F_UNC_ERR_FLG] then FB_CRC_CHIPS (.TEMP_ARR$BNK_SEL);
;        4868
;        4869                                                  .FIND ALL THE BAD CRC CHIPS IN THIS BANK
;        4870      end;

                                        .SBTTL  FB.CHIPS ROUTINE DECLARATIONS
022574  142767  000002  037020'  FB.CHIPS:
                                        BICB    #2,FLG.REG                      ;                                       4863
022602  004767  161566                  JSR     PC,I.CHIP.TBL                   ;                                       4864
022606  016646  000002                  MOV     2(SP),-(SP)                     ; TEMP.ARR$BNK.SE,*                     4865
022612  004767  175456                  JSR     PC,FB.DATA.CHIPS
022616  032767  000002  037020'        BIT     #2,FLG.REG                      ;                                       4867
022624  001405                          BEQ     1$
022626  016646  000004                  MOV     4(SP),-(SP)                     ; TEMP.ARR$BNK.SE,*
022632  004767  176466                  JSR     PC,FB.CRC.CHIPS
022636  005726                          TST     (SP)+
022640  005726                  1$:     TST     (SP)+                           ;                                       4845
022642  000207                          RTS     PC                              ;                                       4844

; Routine Size:  20 words
; Maximum stack depth per invocation:  2 words
```

```
4871  routine PM_THIS_BANK (TEMP_ARR$BNK_SEL) : novalue =
4872      begin
4873
4874  !++
4875  !  FUNCTIONAL DESCRIPTION: PROM MAINTENANCE THIS BANK
4876  !
4877  !                          THIS LOOPS ON THE ROUTINES THAT WILL
4878  !                          FIND ALL THE NEWLY FAILING CHIPS IN
4879  !                          THIS ARRAY BANKS.
4880  !
4881  !                          THE BLAST TABLE WILL GET LOADED WITH
4882  !                          ALL THE NEWLY FAILING ROW AND COLUMN
4883  !                          ADDRESSES AS A RESULT OF THESE CALLED
4884  !                          ROUTINES.
4885  !
4886  !  FORMAL PARAMETERS:
4887  !                  TEMP_ARR$BNK_SEL:
4888  !                          STORES THE SELECTED ARRAYS ARRAY AND BANK
4889  !                          SELECT NUMBER.
4890  !--
4891
4892      local
4893          BAD_CHIP,                           .POINTS TO BAD CHIPS TO BE TESTED
4894          LST_PAT;                            .LAST PATTERN TEST FOR THIS FAILING CHIP
4895
4896      BAD_CHIP = S_CHIP_TBL (ZERO);           .FIND A BAD CHIP TO TEST
4897
4898      if .BAD_CHIP geq ZERO                   .ARE THERE ANY CHIPS TO TEST
4899      then
4900          begin
4901          FLG_REG [F_ALL_BAD_CHIP] = CLR_FLG;     .CLEAR THE ALL BAD FLAG
4902
4903          do
4904              begin
4905              SEED1 = ZERO;                   .INIT THE RANDOM DATA SEED1
4906              SEED2 = %o'1233';               .INIT THE RANDOM DATA SEED2
4907              SEED3 = %o'7622';               .INIT THE RANOMD DATA SEED3
4908              FLG_REG [F_ERR_MAP_ENTERED] = CLR_FLG;      .CLEAR THE ENTERED ERROR MAP FLAG
4909              BREAK;
4910              I_ERROR_MAP ();                 !INIT THE ERROR MAP
4911              LST_PAT = ZERO;                 .LAST PATTERN STARTS AT ZERO
4912
4913              do                              .TEST ALL FAILING PATTERNS FOR THIS BAD CHIP
4914                  begin
4915                  LST_PAT = L_FAILING_CHIP (.TEMP_ARR$BNK_SEL, .BAD_CHIP, .LST_PAT);
4916                  L_ERROR_MAP (.TEMP_ARR$BNK_SEL, .BAD_CHIP);
4917                                              !LOAD THE ERROR MAP WITH FAILING ROWS AND COLUMNS
4918                  end
4919              until .CHIP_TBL [.BAD_CHIP, PATS] eql ZEROES;       .REPEAT UNTIL ALL PATTERNS ARE TESTED
4920
4921              if .FLG_REG [F_ERR_MAP_ENTERED]     !WERE ANY FAILING ROWS/COLS DETECTED
4922              then
```

```
;        4923                    begin
;        4924                    IN_ERROR_MAP (.TEMP_ARR$BNK_SEL, .BAD_CHIP);    .INTERIGATE THE ERROR MAP
;        4925                    end
;        4926                ese
;        4927                    begin
;        4928                    ERRSF (ERR_4, UNC_CHIP_MSG, 0); !REPORT THE UNCONFIRMED ERROR
;      P 4929                    PRINTB (A_B_C_PRINT, .TEMP_ARR$BNK_SEL<.ARR_SEL_POS, ARR$SEL_SIZE>,
;        4930                        .TEMP_ARR$BNK_SEL<.BNK_SEL_POS, BNK$SEL_SIZE>, .BAD_CHIP);  .REPORT WHERE
;        4931                    end;
;        4932
;        4933                CHIP_TBL [.BAD_CHIP, FAULT] = CLR_FLG;      !CLEAR THIS CHIPS FAULT FLAG
;        4934                BAD_CHIP = S_CHIP_TBL (.BAD_CHIP);  .FIND THE NEXT FAILING CHIP
;        4935                end
;        4936            until (.BAD_CHIP lss ZERO) or (.FLG_REG [F_ABORT_ARRAY]);       .REPEAT UNTIL ALL BAD CHIPS TESTED
;        4937
;        4938            end;
;        4939
;        4940        end;


                                        .SBTTL  PM.THIS.BANK ROUTINE DECLARATIONS
022644  004167  000000G          PM.THIS.BANK:
                                        JSR     R1,$SAVE4                       ;                                       4871
022650  005046                          CLR     -(SP)                           ;                                       4896
022652  004767  170742                  JSR     PC,S.CHIP.TBL
022656  010003                          MOV     R0,R3                           ; *,BAD.CHIP
022660  002544                          BLT     5$                              ;                                       4898
022662  142767  000020  037020'        BICB    #20,FLG.REG                     ;                                       4901
022670  005067  000000G          1$:    CLR     SEED1                           ;                                       4905
022674  012767  001233  000000G        MOV     #1233,SEED2                     ;                                       4906
022702  012767  007622  000000G        MOV     #7622,SEED3                     ;                                       4907
022710  142767  000004  037020'        BICB    #4,FLG.REG                      ;                                       4908
022716  104422                          TRAP    22
022720  004767  161624                  JSR     PC,I.ERROR.MAP                  ;                                       4910
022724  005004                          CLR     R4                              ; LST.PAT                               4911
022726  010301                          MOV     R3,R1                           ; BAD.CHIP,*                            4919
022730  006301                          ASL     R1
022732  016602  000016                  MOV     16(SP),R2                       ; TEMP.ARR$BNK.SE,*                     4915
022736  010246                  2$:     MOV     R2,-(SP)
022740  010346                          MOV     R3,-(SP)                        ; BAD.CHIP,*
022742  010446                          MOV     R4,-(SP)                        ; LST.PAT,*
022744  004767  162314                  JSR     PC,L.FAILING.CHIP
022750  010004                          MOV     R0,R4                           ; *,LST.PAT
022752  010216                          MOV     R2,(SP)                         ;                                       4916
022754  010346                          MOV     R3,-(SP)                        ; BAD.CHIP,*
022756  004767  163136                  JSR     PC,L.ERROR.MAP
022762  062706  000010                  ADD     #10,SP                          ;                                       4914
022766  032761  077777  026626'        BIT     #77777,CHIP.TBL(R1)             ;                                       4919
022774  001360                          BNE     2$
022776  032767  000004  037020'        BIT     #4,FLG.REG                      ;                                       4921
023004  001405                          BEQ     3$
023006  010246                          MOV     R2,-(SP)                        ;                                       4924
```

```
023010  010346                          MOV     R3,-(SP)                ; BAD.CHIP,*
023012  004767  174716                  JSR     PC,IN.ERROR.MAP
023016  000447                          BR      4$                      ;                       4921
023020  104454                  3$:     TRAP    54                      ;                       4928
023022  000004                          .WORD   4
023024  000526'                         .WORD   UNC.CHIP.MSG
023026  000000                          .WORD   0
023030  010346                          MOV     R3,-(SP)                ; BAD.CHIP,*            4930
023032  012746  000022                  MOV     #22,-(SP)
023036  060616                          ADD     SP,(SP)                 ; TEMP.ARR$BNK.SE,*
023040  016746  037012'                 MOV     BNK.SEL.POS,-(SP)
023044  012746  000002                  MOV     #2,-(SP)
023050  005046                          CLR     -(SP)
023052  004767  000000G                 JSR     PC,BL$GT2
023056  022626                          CMP     (SP)+,(SP)+
023060  010066  000002                  MOV     R0,2(SP)
023064  012716  000024                  MOV     #24,(SP)
023070  060616                          ADD     SP,(SP)                 ; TEMP.ARR$BNK.SE,*
023072  016746  037010'                 MOV     ARR.SEL.POS,-(SP)
023076  012746  000004                  MOV     #4,-(SP)
023102  005046                          CLR     -(SP)
023104  004767  000000G                 JSR     PC,BL$GT2
023110  022626                          CMP     (SP)+,(SP)+
023112  010066  000002                  MOV     R0,2(SP)
023116  012716  002120'                 MOV     #A.B.C.PRINT,(SP)
023122  012746  000004                  MOV     #4,-(SP)
023126  010600                          MOV     SP,R0                   ; SP,*
023130  104414                          TRAP    14
023132  062706  000006                  ADD     #6,SP                   ;                       4927
023136  042761  100000  026626'  4$:    BIC     #100000,CHIP.TBL(R1)    ;                       4933
023144  010316                          MOV     R3,(SP)                 ; BAD.CHIP,*            4934
023146  004767  170446                  JSR     PC,S.CHIP.TBL
023152  010003                          MOV     R0,R3                   ; *,BAD.CHIP
023154  022626                          CMP     (SP)+,(SP)+             ;                       4904
023156  005703                          TST     R3                      ; BAD.CHIP              4936
023160  002404                          BLT     5$
023162  032767  000100  037020'         BIT     #100,FLG.REG
023170  001637                          BEQ     1$
023172  005726                  5$:     TST     (SP)+                   ;                       4872
023174  000207                          RTS     PC                      ;                       4871

; Routine Size:  109 words
; Maximum stack depth per invocation:  12 words
```

```
4941  routine CAL_CHK_SUM : novalue =
4942      begin
4943
4944  .++
4945  .  FUNCTIONAL DESCRIPTION: CALCULATE THE CHECK SUMS
4946  .
4947  .                          THIS ROUTINE WILL CALCULATE THE NEW
4948  .                          CHECK SUMS FOR THE NEW PROM DATA TO
4949  .                          BE BLASTED INTO THE SELECTED ARRAYS
4950  .                          PROMS.
4951  .--
4952
4953      local
4954          PD_SAVE : bitvector [16],              .STORES THE PROM DATA TO BE CALCULATED
4955          PD_0_9_CNT,                            .COUNTS NUMBER OF BITS 0-9 SET
4956          CHK_SUM_CNT,                           !COUNTS THE OLD CHECK SUM VALUE
4957          REMAINDER;                             .STORES THE NUMBER OF COUNT TO UPDATE THE CHECK SUMS WITH
4958
4959      label
4960          A;                                     .LEAVE LOOP LABEL
4961
4962      map
4963          ERROR_MAP : b ockvector [4, 512] volatile;      .MAKE THE ERROR MAP LOOK LIKE THE BLAST TABLE
4964
4965      incr BNK_SEL from 0 to 3 do                !LOOK AT ALL THE BANKS
4966          begin
4967
4968          if .BAD_BNK_REG [.BNK_SEL]             !IS THIS A BAD BANK
4969          then
4970              begin
4971
4972              incr PD_WRD_SEL from 0 to .MAX_CHIP_COL*2 - 1 do    !LOOK AT ALL THE ROWS AND COLUMNSNS
4973                  begin
4974
4975                  if .BLAST_TBL [.BNK_SEL, .PD_WRD_SEL, FULL_WRD] neq ZERO        !IS THIS  LOCATION ENPTY
4976                  then
4977  A :
4978                      begin
4979                      PD_0_9_CNT = ZERO;         !CLEAR THE COUNT
4980                      CHK_SUM_CNT = ZERO;        !CLEAR THE COUNT
4981                      PD_SAVE = .BLAST_TBL [.BNK_SEL, .PD_WRD_SEL, FULL_WRD];     !GET THIS PROM DTAT WORD
4982
4983                      incr CNT from 0 to 9 do    !COUNT THE # OF BITS SET IN 0 - 9
4984
4985                          if .PD_SAVE [.CNT] then PD_0_9_CNT = .PD_0_9_CNT + 1;
4986
4987                      CHK_SUM_CNT = .PD_SAVE<10, 3>;      !GET THE OLD COUNT
4988                      CHK_SUM_CNT = .CHK_SUM_CNT + .PD_SAVE<13, 1>;
4989                      CHK_SUM_CNT = .CHK_SUM_CNT + .PD_SAVE<14, 1>;
4990                      CHK_SUM_CNT = .CHK_SUM_CNT + .PD_SAVE<15, 1>;
4991                      REMAINDER = .PD_0_9_CNT - .CHK_SUM_CNT;     !HOW MANY MORE TO ENTER
4992
```

```
;       4993                    if .REMAINDER geq 4         .DO WE ADD IN MORE THAN 3
;       4994                    then
;       4995                        begin
;       4996
;       4997                        if .PD_SAVE<12, 1> eql ZERO     !IS THIS BIT ALREADY SET
;       4998                        then
;       4999                            begin
;       5000                            PD_SAVE<12, 1> = SET_FLG;   .SET THIS BIT
;       5001                            REMAINDER = .REMAINDER - 4; !SUBTRACT THE WEIGHT
;       5002
;       5003                            if .REMAINDER eql ZERO      !ARE WE ALL DONE
;       5004                            then
;       5005                                begin
;       5006                                BLAST_TBL [.BNK_SEL, .PD_WRD_SEL, FULL_WRD] = .PD_SAVE;
;       5007                                ERROR_MAP [.BNK_SEL, .PD_WRD_SEL, FULL_WRD] = .BLAST_TBL [.BNK_SEL,
;       5008                                    .PD_WRD_SEL, FULL_WRD];
;       5009                                leave A;            .LEAVE THE LOOP
;       5010                                end;
;       5011
;       5012                            end;
;       5013
;       5014                        end;
;       5015
;       5016                    if .REMAINDER geq 2         !DO WE ADD IN MORE THAN 1
;       5017                    then
;       5018                        begin
;       5019
;       5020                        if .PD_SAVE<11, 1> eql ZERO     !IS THIS BIT ALREADY SET
;       5021                        then
;       5022                            begin
;       5023                            PD_SAVE<11, 1> = SET_FLG;   !SET THIS BIT
;       5024                            REMAINDER = .REMAINDER - 2; !SUBTRACT THE WEIGHT
;       5025
;       5026                            if .REMAINDER eql ZERO      !ARE WE DONE
;       5027                            then
;       5028                                begin
;       5029                                BLAST_TBL [.BNK_SEL, .PD_WRD_SEL, FULL_WRD] = .PD_SAVE;
;       5030                                ERROR_MAP [.BNK_SEL, .PD_WRD_SEL, FULL_WRD] = .BLAST_TBL [.BNK_SEL,
;       5031                                    .PD_WRD_SEL, FULL_WRD];
;       5032                                leave A;            !LEAVE THE LOOP
;       5033                                end;
;       5034
;       5035                            end;
;       5036
;       5037                        end;
;       5038
;       5039                    if .REMAINDER geq 1         !DO WE ADD MORE THAN 0
;       5040                    then
;       5041                        begin
;       5042
;       5043                        if .PD_SAVE<10, 1> eql ZERO     !IS THIS BIT ALREADY SET
;       5044                        then
```

```
5045                                begin
5046                                PD_SAVE<10, 1> = SET_FLG;   !SET THIS BIT
5047                                REMAINDER = .REMAINDER - 1; .SUBTRACT THE WEIGHT
5048
5049                                if .REMAINDER eql ZERO      !ARE WE DONE
5050                                then
5051                                    begin
5052                                    BLAST_TBL [.BNK_SEL, .PD_WRD_SEL, FULL_WRD] = .PD_SAVE;
5053                                    ERROR_MAP [.BNK_SEL, .PD_WRD_SEL, FULL_WRD] =  BLAST_TBL [.BNK_SEL,
5054                                        .PD_WRD_SEL, FULL_WRD];
5055                                    leave A;          !LEAVE THE LOOP
5056                                    end;
5057
5058                                end;
5059
5060                            end;
5061
5062                        if .PD_SAVE<13, 1> eql ZERO .IS THIS BIT ALREADY SET
5063                        then
5064                            begin
5065                            PD_SAVE<13, 1> = SET_FLG;       .SET THIS BIT
5066                            REMAINDER = .REMAINDER - 1;
5067
5068                            if .REMAINDER eql ZERO  .ARE WE DONE
5069                            then
5070                                begin
5071                                BLAST_TBL [.BNK_SEL, .PD_WRD_SEL, FULL_WRD] = .PD_SAVE;
5072                                ERROR_MAP [.BNK_SEL, .PD_WRD_SEL, FULL_WRD] = .BLAST_TBL [.BNK_SEL, .PD_WRD_SEL,
5073                                    FULL_WRD];
5074                                leave A;              !LEAVE THE LOOP
5075                                end;
5076
5077                            end;
5078
5079                        if .PD_SAVE<14, 1> eql ZERO !IS THIS BIT ALREADY SET
5080                        then
5081                            begin
5082                            PD_SAVE<14, 1> = SET_FLG;       !SET THIS BIT
5083                            REMAINDER = .REMAINDER - 1;     !SUBTRACT THE WEIGHT
5084
5085                            if .REMAINDER eql ZERO  'ARE WE DONE
5086                            then
5087                                begin
5088                                BLAST_TBL [.BNK_SEL, .PD_WRD_SEL, FULL_WRD] = .PD_SAVE;
5089                                ERROR_MAP [.BNK_SEL, .PD_WRD_SEL, FULL_WRD] = .BLAST_TBL [.BNK_SEL, .PD_WRD_SEL,
5090                                    FULL_WRD];
5091                                leave A;              !LEAVE THE LOOP
5092                                end;
5093
5094                            end;
5095
5096                        PD_SAVE<15, 1>   SET_FLG;    .SET THE BIT
```

```
;          5097                    BLAST_TBL [.BNK_SEL, .PD_WRD_SEL, FULL_WRD] = .PD_SAVE;
;          5098                    ERROR_MAP [.BNK_SEL, .PD_WRD_SEL, FULL_WRD] = .BLAST_TBL [.BNK_SEL, .PD_WRD_SEL, FULL_WRD]
;          5099                    ;
;          5100                    end;
;          5101
;          5102                end;
;          5103
;          5104            end;
;          5105
;          5106        end;
;          5107
;          5108    end;
```

```
                                    .SBTTL  CAL.CHK.SUM ROUTINE DECLARATIONS
023176  004167  000000G      CAL.CHK.SUM:
                                    JSR     R1,$SAVE5                       ;                                       4941
023202  162706  000016              SUB     #16,SP
023206  005046                      CLR     -(SP)                           ; BNK.SEL                               4965
023210  011600              1$:     MOV     (SP),R0                         ; BNK.SEL,*                             4968
023212  006200                      ASR     R0
023214  006200                      ASR     R0
023216  006200                      ASR     R0
023220  062700  037016'              ADD     #BAD.BNK.REG,R0
023224  010046                      MOV     R0,-(SP)
023226  016646  000002              MOV     2(SP),-(SP)                     ; BNK.SEL,*
023232  042716  177770              BIC     #177770,(SP)
023236  012746  000001              MOV     #1,-(SP)
023242  005046                      CLR     -(SP)
023244  004767  000000G             JSR     PC,BL$GT2
023250  062700  000010              ADD     #10,SP
023254  006000                      ROR     R0
023256  103402                      BLO     2$
023260  000167  000530              JMP     17$
023264  016766  036774' 000014  2$: MOV     MAX.CHIP.COL,14(SP)             ;                                       4972
023272  006366  000014              ASL     14(SP)
023276  011600                      MOV     (SP),R0                         ; BNK.SEL,*                             4975
023300  000300                      SWAB    R0
023302  105000                      CLRB    R0
023304  006300                      ASL     R0
023306  010066  000012              MOV     R0,12(SP)
023312  005066  000004              CLR     4(SP)                           ; PD.WRD.SEL                            4972
023316  000167  000456              JMP     16$
023322  016605  000004      3$:     MOV     4(SP),R5                        ; PD.WRD.SEL,*                          4975
023326  066605  000012              ADD     12(SP),R5
023332  006305                      ASL     R5
023334  012766  026770' 000010      MOV     #BLAST.TBL,10(SP)
023342  060566  000010              ADD     R5,10(SP)
023346  005776  000010              TST     @10(SP)
023352  001002                      BNE     4$
023354  000167  000414              JMP     15$
023360  005066  000006      4$:     CLR     6(SP)                           ; PD.0.9.CNT                            4979
```

```
023364  005066  000002                    CLR     2(SP)                   ; CHK.SUM.CNT                   4980
023370  017666  000010  000016            MOV     @10(SP),16(SP)          ; *,PD.SAVE                     4981
023376  005004                            CLR     R4                      ; CNT                           4983
023400  010403                    5$:     MOV     R4,R3                   ; CNT,*                         4985
023402  006203                            ASR     R3
023404  006203                            ASR     R3
023406  006203                            ASR     R3
023410  012702  000016                    MOV     #16,R2
023414  060602                            ADD     SP,R2                   ; PD.SAVE,*
023416  060203                            ADD     R2,R3
023420  010346                            MOV     R3,-(SP)
023422  010446                            MOV     R4,-(SP)                ; CNT,*
023424  042716  177770                    BIC     #177770,(SP)
023430  012746  000001                    MOV     #1,-(SP)
023434  005046                            CLR     -(SP)
023436  004767  000000G                   JSR     PC,BL$GT2
023442  062706  000010                    ADD     #10,SP
023446  006000                            ROR     R0
023450  005566  000006                    ADC     6(SP)                   ; PD.0.9.CNT
023454  005204                            INC     R4                      ; CNT                           4983
023456  020427  000011                    CMP     R4,#11                  ; CNT,*
023462  003746                            BLE     5$
023464  016604  000016                    MOV     16(SP),R4               ; PD.SAVE,*                     4987
023470  006204                            ASR     R4
023472  006204                            ASR     R4
023474  000304                            SWAB    R4
023476  042704  177770                    BIC     #177770,R4
023502  010466  000002                    MOV     R4,2(SP)                ; *,CHK.SUM.CNT
023506  005004                            CLR     R4                      ;                               4988
023510  032766  020000  000016            BIT     #20000,16(SP)           ; *,PD.SAVE
023516  001401                            BEQ     6$
023520  005204                            INC     R4
023522  060466  000002            6$:     ADD     R4,2(SP)                ; *,CHK.SUM.CNT
023526  005004                            CLR     R4                      ;                               4989
023530  032766  040000  000016            BIT     #40000,16(SP)           ; *,PD.SAVE
023536  001401                            BEQ     7$
023540  005204                            INC     R4
023542  060466  000002            7$:     ADD     R4,2(SP)                ; *,CHK.SUM.CNT
023546  005004                            CLR     R4                      ;                               4990
023550  005766  000016                    TST     16(SP)                  ; PD.SAVE
023554  100001                            BPL     8$
023556  005204                            INC     R4
023560  060466  000002            8$:     ADD     R4,2(SP)                ; *,CHK.SUM.CNT
023564  016601  000006                    MOV     6(SP),R1                ; PD.0.9.CNT,REMAINDER          4991
023570  166601  000002                    SUB     2(SP),R1                ; CHK.SUM.CNT,REMAINDER
023574  020127  000004                    CMP     R1,#4                   ; REMAINDER,*                   4993
023600  002412                            BLT     9$
023602  032766  010000  000016            BIT     #10000,16(SP)           ; *,PD.SAVE                     4997
023610  001006                            BNE     9$
023612  052766  010000  000016            BIS     #10000,16(SP)           ; *,PD.SAVE                     5000
023620  162701  000004                    SUB     #4,R1                   ; *,REMAINDER                   5001
023624  001455                            BEQ     14$                     ;                               5006
```

```
023626  020127  000002          9$:     CMP     R1,#2                   ; REMAINDER,*                   5016
023632  002412                          BLT     10$
023634  032766  004000  000016          BIT     #4000,16(SP)            ; *,PD.SAVE                     5020
023642  001006                          BNE     10$
023644  052766  004000  000016          BIS     #4000,16(SP)            ; *,PD.SAVE                     5023
023652  162701  000002                  SUB     #2,R1                   ; *,REMAINDER                   5024
023656  001440                          BEQ     14$                     ;                               5029
023660  005701                  10$:    TST     R1                      ; REMAINDER                     5039
023662  003411                          BLE     11$
023664  032766  002000  000016          BIT     #2000,16(SP)            ; *,PD.SAVE                     5043
023672  001005                          BNE     11$
023674  052766  002000  000016          BIS     #2000,16(SP)            ; *,PD.SAVE                     5046
023702  005301                          DEC     R1                      ; REMAINDER                     5047
023704  001425                          BEQ     14$                     ;                               5052
023706  032766  020000  000016  11$:    BIT     #20000,16(SP)           ; *,PD.SAVE                     5062
023714  001005                          BNE     12$
023716  052766  020000  000016          BIS     #20000,16(SP)           ; *,PD.SAVE                     5065
023724  005301                          DEC     R1                      ; REMAINDER                     5066
023726  001414                          BEQ     14$                     ;                               5071
023730  032766  040000  000016  12$:    BIT     #40000,16(SP)           ; *,PD.SAVE                     5079
023736  001005                          BNE     13$
023740  052766  040000  000016          BIS     #40000,16(SP)           ; *,PD.SAVE                     5082
023746  005301                          DEC     R1                      ; REMAINDER                     5083
023750  001403                          BEQ     14$                     ;                               5088
023752  052766  100000  000016  13$:    BIS     #100000,16(SP)          ; *,PD.SAVE                     5096
023760  016676  000016  000010  14$:    MOV     16(SP),@10(SP)          ; PD.SAVE,*                     5097
023766  017665  000010  006626'        MOV     @10(SP),ERROR.MAP(R5)   ;                               5098
023774  005266  000004          15$:    INC     4(SP)                   ; PD.WRD.SEL                    4972
024000  026666  000004  000014  16$:    CMP     4(SP),14(SP)            ; PD.WRD.SEL,*
024006  002002                          BGE     17$
024010  000167  177306                  JMP     3$
024014  005216                  17$:    INC     (SP)                    ; BNK.SEL                       4965
024016  021627  000003                  CMP     (SP),#3                 ; BNK.SEL,*
024022  003002                          BGT     18$
024024  000167  177160                  JMP     1$
024030  062706  000020          18$:    ADD     #20,SP                  ;                               4941
024034  000207                          RTS     PC

; Routine Size:  208 words
; Maximum stack depth per invocation:  18 words
```

```
;         5109  routine BL_PROMS (ARR$BNK_SEL) : novalue =
;         5110      begin
;         5111
;         5112  !++
;         5113  !  FUNCTIONAL DESCRIPTION: BLAST THE PROMS
;         5114  !
;         5115  !                          THIS ROUTINE WILL SEARCH THROUGH THE BLAST
;         5116  !                          TABLE AND LOOK FOR NON-ZERO LOCATIONS.
;         5117  !                          IT WILL THEN BLAST THE PROMS AT THE SELECTED
;         5118  !                          ARRAY AND BANK WITH THE NEW PROM DATA FOUND
;         5119  !                          IN THIS LOCATION.
;         5120  !
;         5121  !  FORMAL PARAMETERS:
;         5122  !                  ARR$BNK_SEL:
;         5123  !                          STORES THE SELECTED ARRAYS ARRAY AND
;         5124  !                          BANK SELECT NUMBER.
;         5125  !--
;         5126
;         5127      label
;         5128          A;                                          .LEAVE LOOP LABEL
;         5129
;         5130      local
;         5131          PROM_ADRS,                                  .STORES THE GENERATED PROM ADDRESS
;         5132          FINISH;
;         5133
;         5134      if .BNK_NUM_SEC eql 127 then FINISH = 127 else FINISH = 255;        !IS THIS A 16K PART
;         5135
;         5136      WRT_PROM_MODE;                                  !ENABLE PROM WRITES
;         5137      PROM_ADRS = ZEROES;                             !CLEAR THE PROM ADRS
;         5138      PROM_ADRS<11, 4> = .ARR$BNK_SEL<.ARR_SEL_POS, ARR$SEL_SIZE>;        !SELECT THE ARRAY
;         5139
;         5140      incr SEL_BNK from 0 to 3 do                     !LOOP THRU ALL THE BANKS
;         5141  A :
;         5142          begin
;         5143
;         5144          if .BAD_BNK_REG [.SEL_BNK]                  !IS THIS A BAD BANK
;         5145          then
;         5146              begin
;         5147              PROM_ADRS<10, 1>   ZERO;                .LOAD THE ROW ADDRESS SELECT BIT
;         5148              PROM_ADRS<8, 2> = .SEL_BNK;             .LOAD THE BANK SELECT
;         5149
;         5150              incr ROW_NUM from 0 to .FINISH do       .BLAST ALL THE NEW ROW PROM DATA
;         5151                  begin
;         5152
;         5153                  if .BLAST_TBL [.SEL_BNK, .ROW_NUM, FULL_WRD] neq ZERO   !IS THIS TABLE FNTRY NOT ZERO
;         5154                  then
;         5155                      begin
;         5156                      PROM_ADRS<0, 8>   .ROW_NUM; !LOAD THE ROW NUMBER
;         5157                      WRT_RH (MLPA, PA_REG, .PROM_ADRS);  .SELECT THE PROM ADDRESS
;         5158                      WRT_RH (MLPD, PD_REG, .BLAST_TBL [.SEL_BNK, .ROW_NUM, FULL_WRD]);   .LOAD THE PROM DATA
;         5159                      DAT_CLK;                        !CLOCK THE NEW PROM DATA IN
;         5160                      DELAY (FIFTY_MS);               !WAIT FOR THE WRITE TO COMPLETE
```

```
5161
5162                        if .ML_ADDR [MLMR, D_CLK]    .DID THE DATA CLOCK TIME OUT
5163                        then
5164                            begin
5165                            ERRSF (ERR_10, TIME_OUT_MSG, 0);        !REPORT THE ERROR
5166                            PRINTB (ONE_MSG, REP_M7363_MSG);
5167                            PRINTB (ONE_MSG, SUPRES_MSG);
5168                            FLG_REG [F_D_CLK_TIME_OUT] = SET_FLG;
5169                            leave A;
5170                            end;
5171
5172                        end;
5173
5174                    end;
5175
5176                PROM_ADRS<10, 1> = ONE;                 .TURN ON THE COLUMN ADDRESSES
5177
5178                incr COL_NUM from 0 to .FINISH do       !BLAST ALL THE NEW COLUMNS
5179                    begin
5180
5181                    if .BLAST_TBL [.SEL_BNK, .COL_NUM + .COL_BASE, FULL_WRD] neq ZERO
5182                                                        !IS THIS TABLE entry not ZRRO
5183                    then
5184                        begin
5185                        PROM_ADRS<0, 8> = .COL_NUM; !LOAD THE COLUMN NUMBER
5186                        WRT_RH (MLPA, PA_REG, .PROM_ADRS);  .SELECT THE PROM ADDRESS
5187                        WRT_RH (MLPD, PD_REG, .BLAST_TBL [.SEL_BNK, .COL_NUM + .COL_BASE, FULL_WRD]);
5188                        DAT_CLK;                        .CLOCK THE NEW PROM DATA IN
5189                        DELAY (FIFTY_MS);               !WAIT FOR THE WRITE TO COMPLETE
5190
5191                        if .ML_ADDR [MLMR, D_CLK]    .DID THE DATA CLOCK TIME OUT
5192                        then
5193                            begin
5194                            ERRSF (ERR_10, TIME_OUT_MSG, 0);        .REPORT THE ERROR
5195                            PRINTB (ONE_MSG, REP_M7363_MSG);
5196                            PRINTB (ONE_MSG, SUPRES_MSG);
5197                            FLG_REG [F_D_CLK_TIME_OUT] = SET_FLG;
5198                            leave A;
5199                            end;
5200
5201                        end;
5202
5203                    end;
5204
5205                end;
5206
5207            end;
5208
5209        CLR_MBUS;                                       !CLEAR THE PROM WRITE ENABLE
5210        DELAY (TEN_MS);                                 .WAIT FOR THE VOLTAGE TO SETTLE
5211        end;
```

```
                        .SBTTL  BL.PROMS ROUTINE DECLARATIONS
024036  004167  000000G  BL.PROMS:
                        JSR     R1,$SAVE5                       ;                               5109
024042  162706  000034          SUB     #34,SP
024046  026727  037014' 000177  CMP     BNK.NUM.SEC,#177        ;                               5134
024054  001004                  BNE     1$
024056  012766  000177  000002  MOV     #177,2(SP)              ; *,FINISH
024064  000403                  BR      2$
024066  012766  000377  000002  1$:     MOV     #377,2(SP)      ; *,FINISH
024074  016701  036772'          2$:     MOV     ML.ADDR,R1
024100  016166  000024  000030  MOV     24(R1),30(SP)           ; *,ML.REG
024106  012702  000140          MOV     #140,R2                 ; *,MLREG
024112  016701  036772'          MOV     ML.ADDR,R1
024116  010261  000024          MOV     R2,24(R1)               ; MLREG,*
024122  005003                  CLR     R3                      ; PROM.ADRS                     5137
024124  012746  000054          MOV     #54,-(SP)               ;                               5138
024130  060616                  ADD     SP,(SP)                 ; ARR$BNK.SEL,*
024132  016746  037010'          MOV     ARR.SEL.POS,-(SP)
024136  012746  000004          MOV     #4,-(SP)
024142  005046                  CLR     -(SP)
024144  004767  000000G          JSR     PC,BL$GT2
024150  000300                  SWAB    R0
024152  006300                  ASL     R0
024154  006300                  ASL     R0
024156  006300                  ASL     R0
024160  042700  103777          BIC     #103777,R0
024164  042703  074000          BIC     #74000,R3               ; *,PROM.ADRS
024170  050003                  BIS     R0,R3                   ; *,PROM.ADRS
024172  005005                  CLR     R5                      ; SEL.BNK                       5140
024174  010500          3$:     MOV     R5,R0                   ; SEL.BNK,*                     5144
024176  006200                  ASR     R0
024200  006200                  ASR     R0
024202  006200                  ASR     R0
024204  062700  037016'          ADD     #BAD.BNK.REG,R0
024210  010046                  MOV     R0,-(SP)
024212  010546                  MOV     R5,-(SP)                ; SEL.BNK,*
024214  042716  177770          BIC     #177770,(SP)
024220  012746  000001          MOV     #1,-(SP)
024224  005046                  CLR     -(SP)
024226  004767  000000C          JSR     PC,BI$GT2
024232  062706  000010          ADD     #10,SP
024236  006000                  ROR     R0
024240  103402                  BLO     4$
024242  000167  000626          JMP     20$
024246  010504          4$:     MOV     R5,R4                   ; SEL.BNK,*                     5148
024250  000304                  SWAB    R4
024252  042704  176377          BIC     #176377,R4
024256  042703  003400          BIC     #3400,R3                ; *,PROM.ADRS
024262  050403                  BIS     R4,R3                   ; *,PROM.ADRS
024264  010500                  MOV     R5,R0                   ; SEL.BNK,*                     5153
024266  000300                  SWAB    R0
```

```
024270  105000                          CLRB    R0
024272  006300                          ASL     R0
024274  010066  000010                  MOV     R0,10(SP)
024300  005004                          CLR     R4                      ; ROW.NUM                       5150
024302  000526                          BR      11$
024304  010400                  5$:     MOV     R4,R0                   ; ROW.NUM,*                     5153
024306  066600  000010                  ADD     10(SP),R0
024312  006300                          ASL     R0
024314  005760  026770'                 TST     BLAST.TBL(R0)
024320  001516                          BEQ     10$
024322  105003                          CLRB    R3                      ; PROM.ADRS                     5156
024324  150403                          BISB    R4,R3                   ; ROW.NUM,PROM.ADRS
024326  016701  036772'                 MOV     ML.ADDR,R1              ;                               5157
024332  016166  000020  000036         MOV     20(R1),36(SP)           ; *,ML.REG
024340  010302                          MOV     R3,R2                   ; PROM.ADRS,MLREG
024342  016701  036772'                 MOV     ML.ADDR,R1
024346  010261  000020                  MOV     R2,20(R1)               ; MLREG,*
024352  016701  036772'                 MOV     ML.ADDR,R1              ;                               5158
024356  016166  000046  000034         MOV     46(R1),34(SP)           ; *,ML.REG
024364  016002  026770'                 MOV     BLAST.TBL(R0),R2        ; *,MLREG
024370  016701  036772'                 MOV     ML.ADDR,R1
024374  010261  000046                  MOV     R2,46(R1)               ; MLREG,*
024400  016701  036772'                 MOV     ML.ADDR,R1
024404  016166  000024  000032         MOV     24(R1),32(SP)           ; *,ML.REG
024412  016602  000032                  MOV     32(SP),R2               ; ML.REG,MLREG
024416  152702  000020                  BISB    #20,R2                  ; *,MLREG
024422  016701  036772'                 MOV     ML.ADDR,R1
024426  010261  000024                  MOV     R2,24(R1)               ; MLREG,*
024432  012701  000144                  MOV     #144,R1                 ; *,$$TMP2                      5160
024436  001411                  6$:     BEQ     9$
024440  016702  000000G                 MOV     L$DLY,R2                ; *,$$TMP1
024444  001404                          BEQ     8$
024446  005066  000042          7$:     CLR     42(SP)                  ; $$TMP
024452  005302                          DEC     R2                      ; $$TMP1
024454  001374                          BNE     7$
024456  005301                  8$:     DEC     R1                      ; $$TMP2
024460  000766                          BR      6$
024462  016701  036772'          9$:     MOV     ML.ADDR,R1              ;                               5162
024466  016166  000024  000030         MOV     24(R1),30(SP)           ; *,ML.REG
024474  032766  000020  000030         BIT     #20,30(SP)              ; *,ML.REG
024502  001425                          BEQ     10$
024504  104454                          TRAP    54                      ;                               5165
024506  000012                          .WORD   12
024510  001170'                         .WORD   TIME.OUT.MSG
024512  000000                          .WORD   0
024514  012746  001250'                 MOV     #REP.M7363.MSG,-(SP)    ;                               5166
024520  012746  001630'                 MOV     #ONE.MSG,-(SP)
024524  012746  000002                  MOV     #2,-(SP)
024530  010600                          MOV     SP,R0                   ; SP,*
024532  104414                          TRAP    14
024534  012716  000330'                 MOV     #SUPRES.MSG,(SP)        ;                               5167
024540  012746  001630'                 MOV     #ONE.MSG,-(SP)
```

```
024544  012746  000002                  MOV     #2,-(SP)
024550  010600                          MOV     SP,R0                   ; SP,*
024552  104414                          TRAP    14
024554  000535                          BR      17$                     ;                               5168
024556  005204                  10$:    INC     R4                      ; ROW.NUM                       5150
024560  020466  000012          11$:    CMP     R4,12(SP)               ; ROW.NUM,FINISH
024564  003647                          BLE     5$
024566  052703  002000                  BIS     #2000,R3                ; *,PROM.ADRS                   5176
024572  005004                          CLR     R4                      ; COL.NUM                       5178
024574  000534                          BR      19$
024576  010400                  12$:    MOV     R4,R0                   ; COL.NUM,*                     5181
024600  066700  036770'                ADD     COL.BASE,R0
024604  066600  000010                  ADD     10(SP),R0
024610  006300                          ASL     R0
024612  005760  026770'                TST     BLAST.TBL(R0)
024616  001522                          BEQ     18$
024620  105003                          CLRB    R3                      ; PROM.ADRS                     5185
024622  150403                          BISB    R4,R3                   ; COL.NUM,PROM.ADRS
024624  016701  036772'                MOV     ML.ADDR,R1              ;                               5186
024630  016166  000020  000026        MOV     20(R1),26(SP)           ; *,ML.REG
024636  010302                          MOV     R3,R2                   ; PROM.ADRS,MLREG
024640  016701  036772'                MOV     ML.ADDR,R1
024644  010261  000020                  MOV     R2,20(R1)               ; MLREG,*
024650  016701  036772'                MOV     ML.ADDR,R1              ;                               5187
024654  016166  000046  000024        MOV     46(R1),24(SP)           ; *,ML.REG
024662  016002  026770'                MOV     BLAST.TBL(R0),R2        ; *,MLREG
024666  016701  036772'                MOV     ML.ADDR,R1
024672  010261  000046                  MOV     R2,46(R1)               ; MLREG,*
024676  016701  036772'                MOV     ML.ADDR,R1
024702  016166  000024  000022        MOV     24(R1),22(SP)           ; *,ML.REG
024710  016600  000022                  MOV     22(SP),R0               ; ML.REG,MLREG
024714  152700  000020                  BISB    #20,R0                  ; *,MLREG
024720  016701  036772'                MOV     ML.ADDR,R1
024724  010061  000024                  MOV     R0,24(R1)               ; MLREG,*
024730  012702  000144                  MOV     #144,R2                 ; *,$$TMP2                      5189
024734  001411                  13$:    BEQ     16$
024736  016701  000000G                 MOV     L$DLY,R1                ; *,$$TMP1
024742  001404                          BEQ     15$
024744  005066  000042          14$:    CLR     42(SP)                  ; $$TMP
024750  005301                          DEC     R1                      ; $$TMP1
024752  001374                          BNE     14$
024754  005302                  15$:    DEC     R2                      ; $$TMP2
024756  000766                          BR      13$
024760  016701  036772'          16$:    MOV     ML.ADDR,R1              ;                               5191
024764  016166  000024  000020        MOV     24(R1),20(SP)           ; *,ML.REG
024772  031766  000020                  BIT     (PC),20(SP)             ; *,ML.REG
024776  001432                          BEQ     18$
025000  104454                          TRAP    54                      ;                               5194
025002  000012                          .WORD   12
025004  001170'                        .WORD   TIME.OUT.MSG
025006  000000                          .WORD   0
025010  012746  001250'                MOV     #REP.M7363.MSG,-(SP)    ;                               5195
```

```
025014  012746  001630'                        MOV     #ONE.MSG,-(SP)
025020  012746  000002                        MOV     #2,-(SP)
025024  010600                                MOV     SP,R0                   ; SP,*
025026  104414                                TRAP    14
025030  012716  000330'                        MOV     #SUPRES.MSG,(SP)        ;                               5196
025034  012746  001630'                        MOV     #ONE.MSG,-(SP)
025040  012746  000002                        MOV     #2,-(SP)
025044  010600                                MOV     SP,R0                   ; SP,*
025046  104414                                TRAP    14
025050  152767  000200  037020'      17$:      BISB    #200,FLG.REG            ;                               5197
025056  062706  000012                        ADD     #12,SP                  ;                               5198
025062  000404                                BR      20$
025064  005204                      18$:      INC     R4                      ; COL.NUM                       5178
025066  020466  000012                19$:      CMP     R4,12(SP)               ; COL.NUM,FINISH
025072  003641                                BLE     12$
025074  005205                      20$:      INC     R5                      ; SEL.BNK                       5140
025076  020527  000003                        CMP     R5,#3                   ; SEL.BNK,*
025102  003002                                BGT     21$
025104  000167  177064                        JMP     3$
025110  016701  036772'              21$:      MOV     ML.ADDR,R1              ;                               5207
025114  016166  000010  000016              MOV     10(R1),16(SP)           ; *,ML.REG
025122  016600  000016                        MOV     16(SP),R0               ; ML.REG,MLREG
025126  152700  000040                        BISB    #40,R0                  ; *,MLREG
025132  016701  036772'                        MOV     ML.ADDR,R1
025136  010061  000010                        MOV     R0,10(R1)               ; MLREG,*
025142  016701  036772'                        MOV     ML.ADDR,R1
025146  016166  000010  000014              MOV     10(R1),14(SP)           ; *,ML.REG
025154  016600  000014                        MOV     14(SP),R0               ; ML.REG,MLREG
025160  016705  036776'                        MOV     ML.DUT,R5
025164  042705  177770                        BIC     #177770,R5
025170  142700  000007                        BICB    #7,R0                   ; *,MLREG
025174  050500                                BIS     R5,R0                   ; *,MLREG
025176  016701  036772'                        MOV     ML.ADDR,R1
025202  010061  000010                        MOV     R0,10(R1)               ; MLREG,*
025206  012702  023420                        MOV     #23420,R2               ; *,$$TMP2                      5210
025212  001411                      22$:      BEQ     25$
025214  016701  000000G                       MOV     L$DLY,R1                ; *,$$TMP1
025220  001404                                BEQ     24$
025222  005066  000042              23$:      CLR     42(SP)                  ; $$TMP
025226  005301                                DEC     R1                      ; $$TMP1
025230  001374                                BNE     23$
025232  005302                      24$:      DEC     R2                      ; $$TMP2
025234  000766                                BR      22$
025236  062706  000044              25$:      ADD     #44,SP                  ;                               5109
025242  000207                                RTS     PC

; Routine Size:  323 words
; Maximum stack depth per invocation:  29 words
```

```
;         5212  routine VER_BLAST (ARR$BNK_SEL) : novalue =
;         5213      begin
;         5214
;         5215  !++
;         5216  !  FUNCTIONAL DESCRIPTION: VERIFY THE PROM BLAST
;         5217  !
;         5218  !                          AFTER THE PROMS HAVE BEEN BLASTED THE
;         5219  !                          PROMS MUST BE READ TO INSURE THAT THEY
;         5220  !                          WERE WRITTEN CORRECTLY.
;         5221  !
;         5222  !                          THIS ROUTINE READS ALL THE BANKS PROM
;         5223  !                          DATA AND COMPARES IT TO THE IDENTICAL
;         5224  !                          PROM DATA STORED IN THE ERROR MAP.
;         5225  !
;         5226  !  FORMAL PARAMETERS:
;         5227  !                   ARR$BNK_SEL:
;         5228  !                          STORES THE SELECTED ARRAYS ARRAY AND BANK
;         5229  !                          SELECT NUMBERS.
;         5230  !--
;         5231
;         5232      loca
;         5233          PROM_ADRS,                              !STORES THE GENERATED PROM ADDRESS
;         5234          FINISH,                                 .VARIABLE ENDING SECTOR NUMBER
;         5235          ARR_SEL;                                .STORES WHICH ARRAY IS SELECTED FOR PM'ING
;         5236
;         5237      map
;         5238          ERROR_MAP : blockvector [4, 512] volatile;      !MAKE THE ERROR MAP LOOK LIKE THE BLAST TABLE
;         5239
;         5240      if .BNK_NUM_SEC eql 127 then FINISH = 127 else FINISH = 255;        !IS THIS A 16K CHIP
;         5241
;         5242      RD_PROM_MODE;                               .ENABLE PROM READ MODE
;         5243      ARR_SEL = .ARR$BNK_SEL<.ARR_SEL_POS, ARR$SEL_SIZE>; .LOAD THE ARRAY SELECT NUMBER
;         5244      PROM_ADRS = ZEROES;                         !CLEAR THE PROM ADDRES
;         5245      PROM_ADRS<11, 4>   .ARR_SEL;                !LOAD  THE ARRAY SELECT NUMBER
;         5246
;         5247      incr SEL_BNK from 0 to 3 do                 .LOOP ON ALL BANKS
;         5248          begin
;         5249          PROM_ADRS<10, 1> = ZERO;                !SELECT THE ROW ADDRESSES
;         5250          PROM_ADRS<8, 2> = .SEL_BNK;             .SELECT THE BANK
;         5251
;         5252          incr ROW_NUM from 0 to .FINISH do       !LOOK AT ALL ROW ADDRESSES IN THE BLAST TABLE
;         5253              begin
;         5254              PROM_ADRS<0, 8> = .ROW_NUM;         !LOAD THE ROW NUMBER
;         5255              WRT_RH (MLPA, PA_REG, .PROM_ADRS);  !SELECT THE PROM ADDRESS
;         5256              DELAY (ONE_US);                     !WAIT FOR THE PROM DATA TO COME OUT
;         5257
;         5258              if .ERROR_MAP [.SEL_BNK, .ROW_NUM, FULL_WRD] neq .ML_ADDR [MLPD, PD_REG]
;         5259              then
;         5260                  begin
;         5261                  ERRSF (ERR_8, CON_D_MSG, 0);    !REPORT THE ERROR
;         5262                  PRINTB (A_B_PRINT, .ARR_SEL, .SEL_BNK);
;         5263                  DEGRADE_MOD_REG [.SEL_BNK] = SET_FLG;
```

```
;         5264                    end;
;         5265
;         5266                end;
;         5267
;         5268            PROM_ADRS<10, 1> = ONE;                 !SELECT THE COLUMN ADDRESSES
;         5269
;         5270            incr COL_NUM from 0 to .FINISH do       !LOOK AT ALL THE COLUMNS
;         5271                begin
;         5272                PROM_ADRS<0, 8> = .COL_NUM;         !SELECT THE COLUMN ADDRESSES
;         5273                WRT_RH (MLPA, PA_REG, .PROM_ADRS);  !SELECT THE PROM ADDRESS
;         5274                DELAY (ONE_US);                     !WAIT FOR THE PROM DATA TO COME OUT
;         5275
;         5276                if .ERROR_MAP [.SEL_BNK, .COL_NUM + .COL_BASE, FULL_WRD] neq .ML_ADDR [MLPD, PD_REG]
;         5277                then
;         5278                    begin
;         5279                    ERRSF (ERR_8, CON_D_MSG, 0);    !REPORT THE ERROR
;         5280                    PRINTB (A_B_PRINT, .ARR_SEL, .SEL_BNK);
;         5281                    DEGRADE_MOD_REG [.SEL_BNK] = SET_FLG;
;         5282                    end;
;         5283
;         5284                end;
;         5285
;         5286            end;
;         5287
;         5288        CLR_MBUS;                                   .CLEAR OUT THE PROM READ MODE
;         5289        end;


                                        .SBTTL  VER.BLAST ROUTINE DECLARATIONS
025244  004167  000000G          VER.BLAST:
                                        JSR     R1,$SAVE5                       ;                       5212
025250  162706  000024                  SUB     #24,SP
025254  026727  037014' 000177          CMP     BNK.NUM.SEC,#177                ;                       5240
025262  001003                          BNE     1$
025264  012716  000177                  MOV     #177,(SP)                       ; *,FINISH
025270  000402                          BR      2$
025272  012716  000377          1$:     MOV     #377,(SP)                       ; *,FINISH
025276  016701  036772'          2$:     MOV     ML.ADDR,R1
025302  016166  000024  000020          MOV     24(R1),20(SP)                   ; *,ML.REG
025310  012700  000040                  MOV     #40,R0                          ; *,MLREG
025314  016701  036772'                  MOV     ML.ADDR,R1
025320  010061  000024                  MOV     R0,24(R1)                       ; MLREG,*
025324  012746  000044                  MOV     #44,-(SP)                       ;                       5243
025330  060616                          ADD     SP,(SP)                         ; ARR$BNK.SEL,*
025332  016746  037010'                  MOV     ARR.SEL.POS,-(SP)
025336  012746  000004                  MOV     #4,-(SP)
025342  005046                          CLR     -(SP)
025344  004767  000000G                  JSR     PC,BL$GT2
025350  010005                          MOV     R0,R5                           ; *,ARR.SEL
025352  005004                          CLR     R4                              ; PROM.ADRS             5244
025354  010503                          MOV     R5,R3                           ; ARR.SEL,*             5245
025356  000303                          SWAB    R3
```

```
025360  006303                          ASL     R3
025362  006303                          ASL     R3
025364  006303                          ASL     R3
025366  042703  103777                  BIC     #103777,R3
025372  042704  074000                  BIC     #74000,R4       ; *,PROM.ADRS
025376  050304                          BIS     R3,R4           ; *,PROM.ADRS
025400  005002                          CLR     R2              ; SEL.BNK                       5247
025402  010203                  3$:     MOV     R2,R3           ; SEL.BNK,*                     5250
025404  000303                          SWAB    R3
025406  042703  176377                  BIC     #176377,R3
025412  042704  003400                  BIC     #3400,R4        ; *,PROM.ADRS
02541o  050304                          BIS     R3,R4           ; *,PROM.ADRS
025420  010200                          MOV     R2,R0           ; SEL.BNK,*                     5258
025422  000300                          SWAB    R0
025424  105000                          CLRB    R0
025426  006300                          ASL     R0
025430  010066  000012                  MOV     R0,12(SP)
025434  005003                          CLR     R3              ; ROW.NUM                       5252
025436  000503                          BR      10$
025440  105004                  4$:     CLRB    R4              ; PROM.ADRS                     5254
025442  150304                          BISB    R3,R4           ; ROW.NUM,PROM.ADRS
025444  016701  036772'                 MOV     ML.ADDR,R1      ;                               5255
025450  016166  000020  000026          MOV     20(R1),26(SP)   ; *,ML.REG
025456  010400                          MOV     R4,R0           ; PROM.ADRS,MLREG
025460  016701  036772'                 MOV     ML.ADDR,R1
025464  010061  000020                  MOV     R0,20(R1)       ; MLREG,*
025470  012700  000001                  MOV     #1,R0           ; *,$$TMP2                      5256
025474  001411                  5$:     BEQ     8$
025476  016701  000000G                 MOV     L$DLY,R1        ; *,$$TMP1
025502  001404                          BEQ     7$
025504  005066  000032          6$:     CLR     32(SP)          ; $$TMP
025510  005301                          DEC     R1              ; $$TMP1
025512  001374                          BNE     6$
025514  005300                  7$:     DEC     R0              ; $$TMP2
025516  000766                          BR      5$
025520  010300                  8$:     MOV     R3,R0           ; ROW.NUM,*                     5258
025522  066600  000012                  ADD     12(SP),R0
025526  006300                          ASL     R0
025530  016701  036772'                 MOV     ML.ADDR,R1
025534  016166  000046  000024          MOV     46(R1),24(SP)   ; *,ML.REG
025542  026066  006626'  000024        CMP     ERROR.MAP(R0),24(SP)    ; *,ML.REG
025550  001435                          BEQ     9$
025552  104454                          TRAP    54              ;                               5261
025554  000010                          .WORD   10
025556  000510'                         .WORD   CON.D.MSG
025560  000000                          .WORD   0
025562  010246                          MOV     R2,-(SP)        ; SEL.BNK,*                     5262
025564  010546                          MOV     R5,-(SP)        ; ARR.SEL,*
025566  012746  002332'                 MOV     #A.B.PRINT,-(SP)
025572  012746  000003                  MOV     #3,-(SP)
025576  010600                          MOV     SP,R0           ; SP,*
025600  104414                          TRAP    14
```

```
025602  010200                          MOV     R2,R0                           ; SEL.BNK,*                     5263
025604  006200                          ASR     R0
025606  006200                          ASR     R0
025610  006200                          ASR     R0
025612  062700  037022'                 ADD     #DEGRADE.MOD.REG,R0
025616  010016                          MOV     R0,(SP)
025620  010246                          MOV     R2,-(SP)                        ; SEL.BNK,*
025622  042716  177770                  BIC     #177770,(SP)
025626  012746  000001                  MOV     #1,-(SP)
025632  011646                          MOV     (SP),-(SP)
025634  004767  000000G                 JSR     PC,BL$PU2
025640  062706  000016                  ADD     #16,SP                          ;                               5260
025644  005203                  9$:     INC     R3                              ; ROW.NUM                       5252
025646  020366  000010          10$:    CMP     R3,10(SP)                       ; ROW.NUM,FINISH
025652  003672                          BLE     4$
025654  052704  002000                  BIS     #2000,R4                        ; *,PROM.ADRS                   5268
025660  005003                          CLR     R3                              ; COL.NUM                       5270
025662  000505                          BR      17$
025664  105004                  11$:    CLRB    R4                              ; PROM.ADRS                     5272
025666  150304                          BISB    R3,R4                           ; COL.NUM,PROM.ADRS
025670  016701  036772'                 MOV     ML.ADDR,R1                      ;                               5273
025674  016166  000020  000022         MOV     20(R1),22(SP)                   ; *,ML.REG
025702  010400                          MOV     R4,R0                           ; PROM.ADRS,MLREG
025704  016701  036772'                 MOV     ML.ADDR,R1
025710  010061  000020                  MOV     R0,20(R1)                       ; MLREG,*
025714  012700  000001                  MOV     #1,R0                           ; *,$$TMP2                      5274
025720  001411                  12$:    BEQ     15$
025722  016701  000000G                 MOV     L$DLY,R1                        ; *,$$TMP1
025726  001404                          BEQ     14$
025730  005066  000032          13$:    CLR     32(SP)                          ; $$TMP
025734  005301                          DEC     R1                              ; $$TMP1
025736  001374                          BNE     13$
025740  005300                  14$:    DEC     R0                              ; $$TMP2
025742  000766                          BR      12$
025744  010300                  15$:    MOV     R3,R0                           ; COL.NUM,*                     5276
025746  066700  036770'                 ADD     COL.BASE,R0
025752  066600  000012                  ADD     12(SP),R0
025756  006300                          ASL     R0
025760  016701  036772'                 MOV     ML.ADDR,R1
025764  016166  000046  000020         MOV     46(R1),20(SP)                   ; *,ML.REG
025772  026066  006626' 000020         CMP     ERROR.MAP(R0),20(SP)            ; *,ML.REG
026000  001435                          BEQ     16$
026002  104454                          TRAP    54                              ;                               5279
026004  000010                          .WORD   10
026006  000510'                         .WORD   CON.D.MSG
026010  000000                          .WORD   0
026012  010246                          MOV     R2,-(SP)                        ; SEL.BNK,*                     5280
026014  010546                          MOV     R5,-(SP)                        ; ARR.SEL,*
026016  012746  002332'                 MOV     #A.B.PRINT,-(SP)
026022  012746  000003                  MOV     #3,-(SP)
026026  010600                          MOV     SP,R0                           ; SP,*
026030  104414                          TRAP    14
```

```
026032  010200                          MOV     R2,R0                           ; SEL.BNK,*                     5281
026034  006200                          ASR     R0
026036  006200                          ASR     R0
026040  006200                          ASR     R0
026042  062700  037022'                 ADD     #DEGRADE.MOD.REG,R0
026046  010016                          MOV     R0,(SP)
026050  010246                          MOV     R2,-(SP)                        ; SEL.BNK,*
026052  042716  177770                  BIC     #177770,(SP)
026056  012746  000001                  MOV     #1,-(SP)
026062  011646                          MOV     (SP),-(SP)
026064  004767  000000G                 JSR     PC,BL$PU2
026070  062706  000016                  ADD     #16,SP                          ;                               5278
026074  005203                  16$:    INC     R3                              ; COL.NUM                       5270
026076  020366  000010          17$:    CMP     R3,10(SP)                       ; COL.NUM,FINISH
026102  003670                          BLE     11$
026104  005202                          INC     R2                              ; SEL.BNK                       5247
026106  020227  000003                  CMP     R2,#3                           ; SEL.BNK,*
026112  003002                          BGT     18$
026114  000167  177262                  JMP     3$
026120  016701  036772'          18$:    MOV     ML.ADDR,R1                      ;                               5286
026124  016166  000010  000016          MOV     10(R1),16(SP)                   ; *,ML.REG
026132  016600  000016                  MOV     16(SP),R0                       ; ML.REG,MLREG
026136  152700  000040                  BISB    #40,R0                          ; *,MLREG
026142  016701  036772'                  MOV     ML.ADDR,R1
026146  010061  000010                  MOV     R0,10(R1)                       ; MLREG,*
026152  016701  036772'                  MOV     ML.ADDR,R1
026156  016166  000010  000014          MOV     10(R1),14(SP)                   ; *,ML.REG
026164  016600  000014                  MOV     14(SP),R0                       ; ML.REG,MLREG
026170  016705  036776'                  MOV     ML.DUT,R5
026174  042705  177770                  BIC     #177770,R5
026200  142700  000007                  BICB    #7,R0                           ; *,MLREG
026204  050500                          BIS     R5,R0                           ; *,MLREG
026206  016701  036772'                  MOV     ML.ADDR,R1
026212  010061  000010                  MOV     R0,10(R1)                       ; MLREG,*
026216  062706  000034                  ADD     #34,SP                          ;                               5213
026222  000207                          RTS     PC

; Routine Size:  248 words
; Maximum stack depth per invocation:  37 words
```

```
5290  routine VER_ERROR_MASK (ARR$BNK_SEL, TSTED_BNK) : novalue =
5291      begin
5292
5293  !++
5294  !  FUNCTIONAL DESCRIPTION: VERIFY THE ERROR MASK
5295  !                          AFTER THE PROMS HAVE BEEN BLASTED AND THE PROMS ARE READ FOR A GOOD BLAST THE
5296  !                          SELECTED ARRAY IS WRITE CHECKED AGAIN AND MADE SURE THAT ALL BLASTED ROWS AND
5297  !                          COLUMNS HAVE BEEN SUCCESSFULLY BLASTED.
5298  !                          UNC ERRORS ARE NOT TOLERATED IN ANY BANK AND WARRENTS THE ARRAY TO BE REPLACED.
5299  !                          SINGLE CHIP FAILURES IN DEGRADE MODE BANKS IS TOLERATED AND DOES NOT CONSTITURE AN ERROR.
5300  !                          SINGLE CHIP FAILURES IN NON-DEGRADE NODE BANKS IS NOT TOLERATED AND WILL CAUSE AN
5301  !                          ERROR AND THAT THE ARRAY BE REPLACED.  HOWEVER THE PROGRAM COULD BE RUN ON THE
5302  !                          SAME ARRAY IN HOPES OF FINDING AND MASKING THE ERROR.
5303  !                          ON THE SECOND PASS BEFORE REPLACING THE ARRAY.
5304  !
5305  !  FORMAL PARAMETERS:
5306  !                  ARR$BNK_SEL:
5307  !                          STORES THE SELECTED ARRAYS ARRAY AND BANK SELECT NUMBERS
5308  !--
5309
5310      local
5311          BNK_NUM,                                    !STORES THE SELECTED BANK NUMBER
5312          ARR_NUM;                                    !STORES THE SELECTED ARRAY NUMBER
5313
5314      ARR_NUM   .ARR$BNK_SEL<.ARR_SEL_POS, ARR$SEL_SIZE>; !LOAD THE ARRAY NUMBER
5315
5316      incr SEL_BNK from 0 to .TSTED_BNK - 1 do     .VERIFY ALL BANKS
5317          begin
5318          BNK_NUM = .ARR$BNK_SEL<.BNK_SEL_POS, BNK$SEL_SIZE>;     !LOAD THE BANK SEL NUMBER
5319
5320          if .BAD_BNK_REG [.BNK_NUM]                .IS THIS A BAD BANK
5321          then
5322              begin
5323              FB_CHIPS (.ARR$BNK_SEL);              !FIND ANY BAD BITS
5324
5325              if S_CHIP_TBL (ZERO) geq ZERO         !WERE ANY BAD BITS FOUND
5326              then
5327                  begin
5328
5329                  if .FLG_REG [F_UNC_ERR_FLG]       !WAS IT AN UNC ERROR
5330                  then
5331                      begin
5332                      ERRSF (ERR_11, UNC_ERR_MSG, 0);     !REPORT THE ERROR
5333                      PRINTB (REP_PRINT, .ARR_NUM);
5334                      end
5335                  else
5336                      begin
5337
5338                      if not .DEGRADE_MOD_REG [.BNK_NUM]  !IS THIS NOT A DEGRADE MODE BANK
5339                      then
5340                          begin
5341                          ERRSF (ERR_12, MEM_ERR_MSG, 0); .REPORT THE ERROR
```

```
:          5342                         PRINTB (REP_PRINT, .ARR_NUM);
:          5343                         end;
:          5344
:          5345                     end;
:          5346
:          5347                 end;
:          5348
:          5349             end;
:          5350
:          5351         ARR$BNK_SEL = .ARR$BNK_SEL + .INC_BNK;  !SELECT THE NEXT BANK
:          5352         end;
:          5353
:          5354     end;


                                        .SBTTL  VER.ERROR.MASK ROUTINE DECLARATIONS
026224  004167  000000G          VER.ERROR.MASK:
                                        JSR     R1,$SAVE4                       ;                               5290
026230  012746  000020                  MOV     #20,-(SP)                       ;                               5314
026234  060616                          ADD     SP,(SP)                         ; ARR$BNK.SEL,*
026236  016746  037010'                 MOV     ARR.SEL.POS,-(SP)
026242  012746  000004                  MOV     #4,-(SP)
026246  005046                          CLR     -(SP)
026250  004767  000000G                 JSR     PC,BL$GT2
026254  010004                          MOV     R0,R4                           ; *,ARR.NUM
026256  005001                          CLR     R1                              ; SEL.BNK                       5316
026260  000533                          BR      6$
026262  012746  000030          1$:     MOV     #30,-(SP)                       ;                               5318
026266  060616                          ADD     SP,(SP)                         ; ARR$BNK.SEL,*
026270  016746  037012'                 MOV     BNK.SEL.POS,-(SP)
026274  012746  000002                  MOV     #2,-(SP)
026300  005046                          CLR     -(SP)
026302  004767  000000G                 JSR     PC,BL$GT2
026306  010002                          MOV     R0,R2                           ; *,BNK.NUM
026310  010203                          MOV     R2,R3                           ; BNK.NUM,*                     5320
026312  006203                          ASR     R3
026314  006203                          ASR     R3
026316  006203                          ASR     R3
026320  010346                          MOV     R3,-(SP)
026322  062716  037016'                 ADD     #BAD.BNK.REG,(SP)
026326  010246                          MOV     R2,-(SP)                        ; BNK.NUM,*
026330  042716  177770                  BIC     #177770,(SP)
026334  012746  000001                  MOV     #1,-(SP)
026340  005046                          CLR     -(SP)
026342  004767  000000G                 JSR     PC,BL$GT2
026346  062706  000010                  ADD     #10,SP
026352  006000                          ROR     R0
026354  103067                          BCC     5$
026356  016646  000036                  MOV     36(SP),-(SP)                    ; ARR$BNK.SEL,*                 5323
026362  004767  174206                  JSR     PC,FB.CHIPS
026366  005046                          CLR     -(SP)                           ;                               5325
026370  004767  165224                  JSR     PC,S.CHIP.TBL
```

```
026374  005726                          TST     (SP)+
026376  005700                          TST     R0
026400  002454                          BLT     4$
026402  032767  000002  037020'         BIT     #2,FLG.REG                      ;                               5329
026410  001414                          BEQ     2$
026412  104454                          TRAP    54                              ;                               5332
026414  000013                          .WORD   13
026416  001112'                         .WORD   UNC.ERR.MSG
026420  000000                          .WORD   0
026422  010446                          MOV     R4,-(SP)                        ; ARR.NUM,*                     5333
026424  012746  002246'                 MOV     #REP.PRINT,-(SP)
026430  012746  000002                  MOV     #2,-(SP)
026434  010600                          MOV     SP,R0                           ; SP,*
026436  104414                          TRAP    14
026440  000432                          BR      3$                              ;                               5331
026442  010346                  2$:     MOV     R3,-(SP)                        ;                               5338
026444  062716  037022'                 ADD     #DEGRADE.MOD.REG,(SP)
026450  010246                          MOV     R2,-(SP)                        ; BNK.NUM,*
026452  042716  177770                  BIC     #177770,(SP)
026456  012746  000001                  MOV     #1,-(SP)
026462  005046                          CLR     -(SP)
026464  004767  000000G                 JSR     PC,BL$GT2
026470  062706  000010                  ADD     #10,SP
026474  006000                          ROR     R0
026476  103415                          BLO     4$
026500  104454                          TRAP    54                              ;                               5341
026502  000014                          .WORD
026504  001310'                         .WORD   MEM.ERR.MSG
026506  000000                          .WORD   0
026510  010446                          MOV     R4,-(SP)                        ; ARR.NUM,*                     5342
026512  012746  002246'                 MOV     #REP.PRINT,-(SP)
026516  012746  000002                  MOV     #2,-(SP)
026522  010600                          MOV     SP,R0                           ; SP,*
026524  104414                          TRAP    14
026526  062706  000006          3$:     ADD     #6,SP                           ;                               5340
026532  005726                  4$:     TST     (SP)+                           ;                               5322
026534  066766  037004' 000036  5$:     ADD     INC.BNK,36(SP)                  ; *,ARR$BNK.SEL                 5351
026542  062706  000010                  ADD     #10,SP                          ;                               5317
026546  005201                          INC     R1                              ; SEL.BNK                       5316
026550  020166  000024          6$:     CMP     R1,24(SP)                       ; SEL.BNK,TSTED.BNK
026554  002642                          BLT     1$
026556  062706  000010                  ADD     #10,SP                          ;                               5291
026562  000207                          RTS     PC                              ;                               5290

; Routine Size:  112 words
; Maximum stack depth per invocation:  18 words
```

```
;         5355  %sbttl 'INITIALIZE CODE SECTION'
;         5356  !
;         5357  ! INITIALIZE CODE SECTION
;         5358  !
;         5359  BGNINIT;
;         5360
;         5361  local
;         5362      !
;         5363      !HARDWARE PTABLE ENTRY TEMPORARY STORAGE LOCATIONS
;         5364      !
;         5365      SYS_HWTBL,                          !PROM MAINT THE ENTIRE SYSTEM
;         5366      ARR_HWTBL,                          !PROM MAINT A SINGLE ARRAY MODULE
;         5367      BNK_HWTBL,                          !PROM MAINT A SINGLE BANK
;         5368      BNK_NO_HWTBL,                       !BANK NUMBER TO BE PROM MAINT
;         5369      ARR_NO_HWTBL,                       !ARRAY MODULE NUMBER TO BE PROM MAINT
;         5370      RH_BASE_HWTBL,                      !RH BASE REGISTER ADDRESS
;         5371      DUT_HWTBL,                          !DIVICE NUMBER TO BE PROM MAINT
;         5372      OPTION_HWTBL,                       !DRIVE OPTION CODE 1 = ML-11A, 0 = ML-11B
;         5373      CORRECT_HWTBL,                      !ARE THE PROGRAM PARAMETERS CORRECT FLAG
;         5374      PTBL_PTR,                           !HARDWARE PTABLE POINTER
;         5375      LUN;                                !SUPERVISOR LOGICAL UNIT NUMBER VARIABLE
;         5376
;         5377  if .L$UNIT gtr ONE                      !WHERE MORE ONE PTABLE BUILT
;         5378  then
;         5379      begin
;         5380      ERRSF (ERR_9, HQ_ERR_MSG, 0);       !REPORT THE ERROR
;         5381      PRINTB (ONE_MSG, GTR_MSG);          !REPORT WHAT THE ERROR IS
;         5382      PRINTB (ONE_MSG, UNIT_SEL_MSG);     !REPORT THAT THE FIRST PTABLE BUILT WILL BE USED
;         5383      end;
;         5384
;         5385  L$UNIT = 0;                             !ONLY ONE DRIVE IS TO BE TESTED
;         5386  !
;         5387  ! BECAUSE OF THE INPACT OF THIS PROGRAM ON THE
;         5388  ! ML-11 SYSTEM ONLY THE START COMMAND WILL BE
;         5389  ! RECOGNIZED TO START PROGRAM EXECUTION
;         5390  !
;         5391
;         5392  if not (READEF (EF_START))              !WAS THE START COMMAND USED TO START THE PROGRAM
;         5393  then
;         5394      begin
;         5395      ERRSF (ERR_1, ILL_CMD_MSG, 0);      !REPORT THE ERROR
;         5396      PRINTB (ONE_MSG, START_MSG);        !REPORT WHAT THE ERROR IS
;         5397      DODU (.L$UNIT);                     !DROP THIS UNIT
;         5398      DOCLN;                              !GO TO THE CLEAN UP CODE SECTION
;         5399      end;
;         5400
;         5401  !
;         5402  ! THE CONTENTS OF THE FIRST P-TABLE BUILT (LUN 0 )
;         5403  ! WILL BE THE SELECTED PROGRAN PARAMETERS.  ALL OTHER REMAINING
;         5404  ! P-TABLES BUILT WILL BE IGNORED.
;         5405  !
;         5406  LUN - ZERO;
```

```
;         5407
;         5408  if (GPHARD (.LUN, PTBL_PTR)) eql ZERO          !IS THE FIRST PTABLE PRESENT
;         5409  then
;         5410      begin
;         5411      ERRSF (ERR_3, LUN_MISS_MSG, 0);           !REPORT THE ERROR
;         5412      PRINTB (ONE_MSG, SUPRES_MSG);             !REPORT SUPPERSS MESSAGE
;         5413      PRINTB (ONE_MSG, RET_DRS_MSG);            !REPORT RETURNING TO DRS>
;         5414      DODU (.L$UNIT);                           !DROP THIS UNIT
;         5415      DOCLN;                                    !GO TO THE DROP UNIT CODE SECTION
;         5416      end
;         5417  else
;         5418      begin
;         5419      !
;         5420      ! INPUT TO THE PROGRAM THE HARDWARE PTABLE
;         5421      ! ENTRIES AND STORE THEM INTO THE TEMPORARY
;         5422      ! STORAGE LOCATIONS
;         5423      !
;         5424      SYS_HWTBL = .((.PTBL_PTR) + %o'0');
;         5425      ARR_HWTBL = .((.PTBL_PTR) + %o'2');
;         5426      BNK_HWTBL = .((.PTBL_PTR) + %o'4');
;         5427      BNK_NO_HWTBL = .((.PTBL_PTR) + %o'6');
;         5428      ARR_NO_HWTBL = .((.PTBL_PTR) + %o'10');
;         5429      RH_BASE_HWTBL = .((.PTBL_PTR) + %o'12');
;         5430      DUT_HWTBL = .((.PTBL_PTR) + %o'14');
;         5431      OPTION_HWTBL = .((.PTBL_PTR) + %o'16');
;         5432      CORRECT_HWTBL = .((.PTBL_PTR) + %o'20');
;         5433      end;
;         5434
;         5435      !
;         5436  ! BEFORE WE START, LETS SEE IF THE OPERATOR
;         5437  ! ENTERED HIS PARAMETERS CORRECTLY.
;         5438  !
;         5439
;         5440  if not .CORRECT_HWTBL                          !ARE THE INPUTED PARAMETERS CORRECT
;         5441  then
;         5442      begin
;         5443      PRINTB (LFS);                             !PRINT SOME LINE FEEDS TO BE NEAT
;         5444      PRINTB (ONE_MSG, SUPRES_MSG);             !PRINT THE SUPPRESS MESSAGE
;         5445      PRINTB (ONE_MSG, RET_DRS_MSG);            !PRINT RETURNING TO DRS> MESSAGE
;         5446      DODU (.L$UNIT);                           !DROP THIS UNIT
;         5447      DOCLN;                                    !GO TO THE CLEAN UP CODE SECTION
;         5448      end
;         5449  else
;         5450      begin
;         5451      !
;         5452      ! BUILD THE RUN TIME PARAMETERS
;         5453      !
;         5454      ARR$BNK_SEL = ZEROES;                     !LOAD THE ARRAY & BANK SELECT VARIABLE WITH ZEROES
;         5455      ML_ADDR   .RH_BASE_HWTBL;                 !LOAD THE ML-11 BASE ADDRESS INTO ML_ADDR
;         5456      ML_DUT = .DUT_HWTBL;                      !LOAD ML_DUT WITH THE DIVICES DRIVE SELECT NUMBER
;         5457      !
;         5458      ! SET UP 16K OR 64K MOS RAM RUN TIME PARAMETERS
```

```
5459        .
5460
5461        if .OPTION_HWTBL eql ML11A                  !IS THIS A ML-11A OPTION 16K PARTS
5462        then
5463            begin
5464            COL_BASE = 128;                         !LOAD THE COLUMN BASE OFFSET
5465            INC_BNK = %o'200';                      .LOAD THE BANK INCREMENT VARIABLE
5466            INC_ARR = %o'1000';                     .LOAD THE ARRAY INCREMENT VARIABLE
5467            BNK_NUM_SEC = 127;                      !LOAD THE BANKS NUMBER OF SECTORS VARIABLE
5468            ARR_SEL_POS = 9;                        !LOAD THE ARRAY SELECT POSITION CONSTANT
5469            BNK_SEL_POS = 7;                        .LOAT THE BANK SELECT POSITION CONSTANT
5470            MAX_CHIP_COL = 128;                     .LOAD THE MAXIMUM COLUMNS PER CHIP VARIABLE
5471            end
5472        else
5473            begin
5474            COL_BASE = 256;                         .LOAD THE COLUMN BASE OFFSET VARIABLE
5475            INC_BNK = %o'1000';                     !LOAD THE BANK INCREMENT VARIABLE
5476            INC_ARR = %o'4000';                     .LOAD THE ARRAY INCREMENT VARIABLE
5477            BNK_NUM_SEC = 511;                      .LOAD THE BANKS NUMBER OF SECTORS VARIABLE
5478            ARR_SEL_POS = 11;                       .LOAD THE ARRAY SELECTION POSITION CONSTANT
5479            BNK_SEL_POS = 9;                        !LOAD THE BANK SELECTION POSITION CONSTANT
5480            MAX_CHIP_COL = 256;                     .LOAD THE MAXIMUM COLUMNS PER CHIP VARIABLE
5481            end;
5482
5483        .
5484        ! BUILD THE RUN TIME PARAMETER FOR MASKING
5485        ! EITHER THE ENTIRE SYSTEM, AN ENTIRE ARRAY OR
5486        ! A SINGLE BANK.
5487        !
5488
5489        if .SYS_HWTBL                               !IS THE ENTIRE ML-11 SYSTEM TO BE PROM MAINT
5490        then
5491            begin
5492            WRT_RH (MLCS2, DRV_SEL, .ML_DUT);       !SELECT THE DRIVE UNDER TEST
5493            LST_TSTED_ARR = .ML_ADDR [MLMR, SIZING] - 1;    .READ AND SAVE THE SIZING
5494            RAND_PASS   4;                          !DO TWO RANDOM DATA PASSES
5495            TSTED_BNK   4;                          .TEST FOUR BANKS PER ARRAY
5496            end
5497        else
5498            begin
5499
5500            if .ARR_HWTBL                           !IS A SINGLE ARRAY TO BE PROM MAINT
5501            then
5502                begin
5503                LST_TSTED_ARR = 0;                  .LOAD LOOP VARIABLE TO TEST ONLY ONE ARRAY
5504                RAND_PASS   8;                      !DO THREE PASSES OF RANDOM DATA
5505                TSTED_BNK = 4;                      !TEST ALL FOUR BANKS ON THE ARRAY
5506                ARR$BNK_SEL<.ARR_SEL_POS, ARR$SEL_SIZE> = .ARR_NO_HWTBL;    !SELECT WHICH ARRAY TO TEST
5507                end
5508            else
5509                begin
5510
```

```
;          5511                if .BNK_HWTBL                           .IS A SINGLE BANK TO BE PROM MAINT
;          5512                then
;          5513                    begin
;          5514                    LST_TSTED_ARR = 0;                  .LOAD LOOP VARIABLE TO TEST ONLY ONE ARRAY
;          5515                    RAND_PASS = 13;                     .DO FIVE PASSES OF RANDOM DATA
;          5516                    TSTED_BNK = 1;                      !TEST ONLY ONE BANK
;          5517                    ARR$BNK_SEL<.ARR_SEL_POS, ARR$SEL_SIZE> = .ARR_NO_HWTBL;
;          5518                                                        .SELECT WHICH ARRAY TO PROM MAINT
;          5519                    ARR$BNK_SEL<.BNK_SEL_POS, BNK$SEL_SIZE> = .BNK_NO_HWTBL;
;          5520                                                        !SELECT WHICH BANK TO PROM MAINT
;          5521                    end
;          5522                else
;          5523                    begin
;          5524                    .
;          5525                    . THE HARDWARE QUESTIONS WERE NOT ANSWERED
;          5526                    . CORRECTLY.  MASKING EITHER A SYSTEM, ARRAY
;          5527                    . OR BANK WAS NOT SELECTED.  REPORT THE ERROR
;          5528                    ! AND RERTURN TO DRS>.
;          5529                    .
;          5530                    ERRSF (ERR_2, HQ_ERR_MSG, 0);       !REPORT THE ERROR
;          5531                    PRINTB (ONE_MSG, HQ_MSG);           .REPORT WHAT THE ERROR IS
;          5532                    PRINTB (ONE_MSG, SUPRES_MSG);       !PRINT SUPPRESS MESSAGE
;          5533                    PRINTB (ONE_MSG, RET_DRS_MSG);      .PRINT RETURNING TO DRS> MESSAGE
;          5534                    DODU (.L$UNIT);                     .DROP THIS UNIT
;          5535                    DOCLN;                              !GO TO THE CLEAN UP CODE SECTION
;          5536                    end
;          5537
;          5538                end
;          5539
;          5540            end
;          5541
;          5542        end
;          5543
;          5544    ENDINIT;
```

```
                                        .SBTTL  LINIT INITIALIZE CODE SECTION
026564  004167  000000G          LINIT: JSR     R1,$SAVE5                ;                                  5354
026570  162706  000014                 SUB     #14,SP
026574  026727  000000G 000001         CMP     L$UNIT,#1                ;                                  5377
026602  003426                         BLE     1$
026604  104454                         TRAP    54                       ;                                  5380
026606  000011                         .WORD   11
026610  000142'                        .WORD   HQ.ERR.MSG
026612  000000                         .WORD   0
026614  012746  000560'               MOV     #GTR.MSG,-(SP)           ;                                  5381
026620  012746  001630'               MOV     #ONE.MSG,-(SP)
026624  012746  000002                 MOV     #2,-(SP)
026630  010600                         MOV     SP,R0                    ; SP,*
026632  104414                         TRAP    14
026634  012716  000634'               MOV     #UNIT.SEL.MSG,(SP)       ;                                  5382
026640  012746  001630'               MOV     #ONE.MSG,-(SP)
```

```
026644  012746  000002                    MOV     #2,-(SP)
026650  010600                            MOV     SP,R0                   ; SP,*
026652  104414                            TRAP    14
026654  062706  000012                    ADD     #12,SP                  ;                               5379
026660  005067  000000G             1$:    CLR     L$UNIT                  ;                               5385
026664  012700  000040                    MOV     #40,R0                  ;                               5392
026670  104447                            TRAP    47
026672  103422                            BCS     2$
026674  104454                            TRAP    54                      ;                               5395
026676  000001                            .WORD   1
026700  000000'                           .WORD   ILL.CMD.MSG
026702  000000                            .WORD   0
026704  012746  000104'                    MOV     #START.MSG,-(SP)        ;                               5396
026710  012746  001630'                    MOV     #ONE.MSG,-(SP)
026714  012746  000002                    MOV     #2,-(SP)
026720  010600                            MOV     SP,R0                   ; SP,*
026722  104414                            TRAP    14
026724  016700  000000G                   MOV     L$UNIT,R0               ;                               5397
026730  104451                            TRAP    51
026732  104444                            TRAP    44
026734  062706  000006                    ADD     #6,SP                   ;                               5394
026740  005000                      2$:   CLR     R0                      ; LUN                           5406
026742  104442                            TRAP    42                      ;                               5408
026744  010005                            MOV     R0,R5                   ; *,PTBL.PTR
026746  001033                            BNE     3$
026750  104454                            TRAP    54                      ;                               5411
026752  000003                            .WORD   3
026754  000412'                           .WORD   LUN.MISS.MSG
026756  000000                            .WORD   0
026760  012746  000330'                    MOV     #SUPRES.MSG,-(SP)       ;                               5412
026764  012746  001630'                    MOV     #ONE.MSG,-(SP)
026770  012746  000002                    MOV     #2,-(SP)
026774  010600                            MOV     SP,R0                   ; SP,*
026776  104414                            TRAP    14
027000  012716  000370'                    MOV     #RET.DRS.MSG,(SP)       ;                               5413
027004  012746  001630'                    MOV     #ONE.MSG,-(SP)
027010  012746  000002                    MOV     #2,-(SP)
027014  010600                            MOV     SP,R0                   ; SP,*
027016  104414                            TRAP    14
027020  016700  000000G                   MOV     L$UNIT,R0               ;                               5414
027024  104451                            TRAP    51
027026  104444                            TRAP    44
027030  062706  000012                    ADD     #12,SP                  ;                               5410
027034  000424                            BR      4$                      ;                               5408
027036  011566  000006            3$:     MOV     (R5),6(SP)              ; PTBL.PTR,SYS.HWTBL            5424
027042  016566  000002  000004          MOV     2(R5),4(SP)             ; *(PTBL.PTR),ARR.HWTBL         5425
027050  016566  000004  000002          MOV     4(R5),2(SP)             ; *(PTBL.PTR),BNK.HWTBL         5426
027056  016516  000006                    MOV     6(R5),(SP)              ; *(PTBL.PTR),BNK.NO.HWTBL      5427
027062  016501  000010                    MOV     10(R5),R1               ; *(PTBL.PTR),ARR.NO.HWTBL      5428
027066  016502  000012                    MOV     12(R5),R2               ; *(PTBL.PTR),RH.BASE.HWTBL     5429
027072  016503  000014                    MOV     14(R5),R3               ; *(PTBL.PTR),DUT.HWTBL         5430
027076  016504  000016                    MOV     16(R5),R4               ; *(PTBL.PTR),OPTION.HWTBL      5431
```

```
027102  016505  000020                        MOV     20(R5),R5                       ; *(PTBL.PTR),CORRECT.HWTBL             5432
027106  006005                          4$:   ROR     R5                              ; CORRECT.HWTBL                         5440
027110  103435                                BLO     5$                                                                      5443
027112  012746  001700'                       MOV     #LFS,-(SP)                      ;
027116  012746  000001                        MOV     #1,-(SP)
027122  010600                                MOV     SP,R0                           ; SP,*
027124  104414                                TRAP    14
027126  012716  000330'                       MOV     #SUPRES.MSG,(SP)                ;                                       5444
027132  012746  001630'                       MOV     #ONE.MSG,-(SP)
027136  012746  000002                        MOV     #2,-(SP)
027142  010600                                MOV     SP,R0                           ; SP,*
027144  104414                                TRAP    14
027146  012716  000370'                       MOV     #RET.DRS.MSG,(SP)               ;                                       5445
027152  012746  001630'                       MOV     #ONE.MSG,-(SP)
027156  012746  000002                        MOV     #2,-(SP)
027162  010600                                MOV     SP,R0                           ; SP,*
027164  104414                                TRAP    14
027166  016700  000000G                       MOV     L$UNIT,R0                       ;                                       5446
027172  104451                                TRAP    51
027174  104444                                TRAP    44
027176  062706  000014                        ADD     #14,SP                          ;                                       5442
027202  000537                                BR      8$                              ;                                       5440
027204  005067  037000'                 5$:   CLR     ARR$BNK.SEL                     ;                                       5454
027210  010267  036772'                       MOV     R2,ML.ADDR                      ; RH.BASE.HWTBL,*                       5455
027214  010367  036776'                       MOV     R3,ML.DUT                       ; DUT.HWTBL,*                           5456
027220  005304                                DEC     R4                              ; OPTION.HWTBL                          5461
027222  001026                                BNE     6$
027224  012767  000200  036770'               MOV     #200,COL.BASE                   ;                                       5464
027232  012767  000200  037004'               MOV     #200,INC.BNK                    ;                                       5465
027240  012767  001000  037006'               MOV     #1000,INC.ARR                   ;                                       5466
027246  012767  000177  037014'               MOV     #177,BNK.NUM.SEC                ;                                       5467
027254  012767  000011  037010'               MOV     #11,ARR.SEL.POS                 ;                                       5468
027262  012767  000007  037012'               MOV     #7,BNK.SEL.POS                  ;                                       5469
027270  012767  000200  036774'               MOV     #200,MAX.CHIP.COL               ;                                       5470
027276  000425                                BR      7$                              ;                                       5461
027300  012767  000400  036770'         6$:   MOV     #400,COL.BASE                   ;                                       5474
027306  012767  001000  037004'               MOV     #1000,INC.BNK                   ;                                       5475
027314  012767  004000  037006'               MOV     #4000,INC.ARR                   ;                                       5476
027322  012767  000777  037014'               MOV     #777,BNK.NUM.SEC                ;                                       5477
027330  012767  000013  037010'               MOV     #13,ARR.SEL.POS                 ;                                       5478
027336  012767  000011  037012'               MOV     #11,BNK.SEL.POS                 ;                                       5479
027344  012767  000400  036774'               MOV     #400,MAX.CHIP.COL               ;                                       5480
027352  032766  000001  000006         7$:   BIT     #1,6(SP)                        ; *,SYS.HWTBL                           5489
027360  001451                                BEQ     9$
027362  016705  036772'                       MOV     ML.ADDR,R5                      ;                                       5492
027366  016566  000010  0000'2                MOV     10(R5),12(SP)                   ; *,ML.REG
027374  016604  000012                        MOV     12(SP),R4                       ; ML.REG,MLREG
027400  016705  036776'                       MOV     ML.DUT,R5
027404  042705  177770                        BIC     #177770,R5
027410  142704  000007                        BICB    #7,R4                           ; *,MLREG
027414  050504                                BIS     R5,R4                           ; *,MLREG
027416  016705  036772'                       MOV     ML.ADDR,R5
```

```
027422  010465  000010                          MOV     R4,10(R5)                       ; MLREG,*                       5493
027426  016705  036772'                         MOV     ML.ADDR,R5                      ;
027432  016566  000024  000010                  MOV     24(R5),10(SP)                   ; *,ML.REG
027440  016605  000010                          MOV     10(SP),R5                       ; ML.REG,*
027444  006205                                  ASR     R5
027446  006205                                  ASR     R5
027450  006205                                  ASR     R5
027452  000305                                  SWAB    R5
027454  042705  177740                         BIC     #177740,R5
027460  005305                                  DEC     R5
027462  010567  037032'                         MOV     R5,LST.TSTED.ARR                                                5494
027466  012767  000004  037034'                 MOV     #4,RAND.PASS                    ;                               5495
027474  012767  000004  037002'                 MOV     #4,TSTED.BNK                    ;                               5489
027502  000532                          8$:     BR      14$                             ;                               5500
027504  032766  000001  000004          9$:     BIT     #1,4(SP)                        ; *,ARR.HWTBL
027512  001422                                  BEQ     10$                                                             5503
027514  005067  037032'                         CLR     LST.TSTED.ARR                   ;                               5504
027520  012767  000010  037034'                 MOV     #10,RAND.PASS                   ;                               5505
027526  012767  000004  037002'                 MOV     #4,TSTED.BNK                    ;                               5506
027534  012746  037000'                         MOV     #ARR$BNK.SEL,-(SP)              ;
027540  016746  037010'                         MOV     ARR.SEL.POS,-(SP)
027544  012746  000004                          MOV     #4,-(SP)
027550  010146                                  MOV     R1,-(SP)                        ; ARR.NO.HWTBL,*
027552  004767  000000G                         JSR     PC,BL$PU2
027556  000502                                  BR      13$                             ;                               5498
027560  032766  000001  000002          10$:    BIT     #1,2(SP)                        ; *,BNK.HWTBL                   5511
027566  001434                                  BEQ     11$                                                             5514
027570  005067  037032'                         CLR     LST.TSTED.ARR                   ;                               5515
027574  012767  000015  037034'                 MOV     #15,RAND.PASS                   ;                               5516
027602  012767  000001  037002'                 MOV     #1,TSTED.BNK                    ;                               5517
027610  012746  037000'                         MOV     #ARR$BNK.SEL,-(SP)              ;
027614  016746  037010'                         MOV     ARR.SEL.POS,-(SP)
027620  012746  000004                          MOV     #4,-(SP)
027624  010146                                  MOV     R1,-(SP)                        ; ARR.NO.HWTBL,*
027626  004767  000000G                         JSR     PC,BL$PU2
027632  012716  037000'                         MOV     #ARR$BNK.SEL,(SP)               ;                               5519
027636  016746  037012'                         MOV     BNK.SEL.POS,-(SP)
027642  012746  000002                          MOV     #2,-(SP)
027646  016646  000014                          MOV     14(SP),-(SP)                    ; BNK.NO.HWTBL,*
027652  004767  000000G                         JSR     PC,BL$PU2
027656  000440                                  BR      12$                             ;                               5509
027660  104454                          11$:    TRAP    54                              ;                               5530
027662  000002                                  .WORD   2
027664  000142'                                 .WORD   HQ.ERR.MSG
027666  000000                                  .WORD   0
027670  012746  000216'                         MOV     #HQ.MSG,-(SP)                   ;                               5531
027674  012746  001630'                         MOV     #ONE.MSG,-(SP)
027700  012746  000002                          MOV     #2,-(SP)
027704  010600                                  MOV     SP,R0                           ; SP,*
027706  104414                                  TRAP    14
027710  012716  000330'                         MOV     #SUPRES.MSG,(SP)                ;                               5532
027714  012746  001630'                         MOV     #ONE.MSG,-(SP)
```

```
027720  012746  000002                          MOV     #2,-(SP)
027724  010600                                  MOV     SP,R0                   ; SP,*
027726  104414                                  TRAP    14
027730  012716  000370'                         MOV     #RET.DRS.MSG,(SP)       ;                       5533
027734  012746  001630'                         MOV     #ONE.MSG.-(SP)
027740  012746  000002                          MOV     #2,-(SP)
027744  010600                                  MOV     SP,R0                   ; SP,*
027746  104414                                  TRAP    14
027750  016700  000000G                         MOV     L$UNIT,R0               ;                       5534
027754  104451                                  TRAP    51
027756  104444                                  TRAP    44
027760  062706  000006          12$:    ADD     #6,SP                   ;                       5509
027764  062706  000010          13$:    ADD     #10,SP                  ;                       5498
027770  062706  000014          14$:    ADD     #14,SP                  ;                       5354
027774  000207                                  RTS     PC

; Routine Size:  325 words
; Maximum stack depth per invocation:  19 words



                                        .SBTTL  L$INIT INITIALIZE CODE SECTION
027776  004767  176562          L$INIT::JSR     PC,LINIT                ;                       5542
030002  104411                                  TRAP    11
030004  000207                                  RTS     PC

; Routine Size:  4 words
; Maximum stack depth per invocation:  0 words
```

```
5545  %sbttl 'MAIN PROM MAINT CONTROL CODE'
5546
5547  ! MAIN PROM MAINT CONTROL CODE
5548  !
5549  BGNTST;
5550
5551  local
5552      TEMP_TSTED_BNK,                          !TEMPORARY STORAGE FOR THE NUMBER OF TESTED BANKS PER ARRAY
5553      TEMP_ARR$BNK_SEL;                        !TEMPORARY STORAGE FOR THE ARRAY & BANK SELECT VARIABLE
5554
5555  !
5556  ! PROMPT THE OPERATOR THAT THE
5557  ! PROM MAINTANENCE PROGRAM HAS BEGUN
5558  ! AND WHICH DRIVE IS SELECTED.
5559  !
5560  PRINTB (ONE_MSG, BGN_MSG);
5561  PRINTB (DRV_SEL_PRINT, .ML_DUT);
5562  !
5563  ! THIS OUTER LOOP WILL PM ALL SELECTED
5564  ! ARRAYS. IT WILL PM FROM EITHER ONE ARRAY TO THE SELECTED
5565  ! DRIVES MAX ARRAYS PRESENT.
5566  !
5567
5568  incr ARR_SEL from 0 to .LST_TSTED_ARR do      !TEST ALL SELECTED ARRAYS
5569      begin
5570      BREAK;
5571      DEGRADE_MOD_REG = ZEROES;                 !CLEAR THE DEGRADE MODE REGISTER
5572      BAD_BNK_REG = ZEROES;                     !CLEAR THE BAD BANK REGISTERS FLAGS
5573      FLG_REG = ZEROES;                         !CLEAR THE FLAG REGISTER
5574      I_BLAST_TBL ();                           !INTI THE BLAST TABLE
5575      TEMP_TSTED_BNK = .TSTED_BNK;              !LOAD NUMBER OF TESTED BANKS INTO TEMP STORAGE
5576      TEMP_ARR$BNK_SEL = .ARR$BNK_SEL;          !LOAD THE ARRAY & BANK SEL INTO TEMP STORAGE
5577      !
5578      ! THIS INNER LOOP WILL FIND NEW FAILING ROWS &
5579      ! COLUMNS WITHIN BAD CHIPS IN THE BANKS OF THE SELECTED
5580      ! ARRAYS.  EITHER 4 BANKS, IF PM'ING SYS/ARRAY
5581      ! OR 1 BANK IF PM'ING A SINGLE BANK WILL BE TESTED.
5582      !
5583      !
5584      ! PROMPT THE OPERATOR WHICH ARRAY IS PRESENTLY SELECTED
5585      !
5586      PRINTB (ARR_SEL_PRINT, .ARR$BNK_SEL<.ARR_SEL_POS, ARR$SEL_SIZE>);
5587
5588      do                                        !PROM MAINT ALL THE FAILING CHIPS IN THIS BANK
5589          begin
5590          PRINTB (BNK_SEL_PRINT, .TEMP_ARR$BNK_SEL<.BNK_SEL_POS, BNK$SEL_SIZE>);  !PRINT
5591          FB_CHIPS (.TEMP_ARR$BNK_SEL);         !FIND THE BAD CHIPS IN THIS BANK
5592          PM_THIS_BANK (.TEMP_ARR$BNK_SEL);     !PROM MAINT THE BAD CHIPS IN THIS BANK
5593          TEMP_ARR$BNK_SEL = .TEMP_ARR$BNK_SEL + .INC_BNK;    !INCREMENT TO THE NEXT BANK
5594          TEMP_TSTED_BNK = .TEMP_TSTED_BNK - 1;  !DECREMENT THE TESTED BANK COUNT
5595          end
5596      until (.TEMP_TSTED_BNK eql ZERO) or (.FLG_REG [F_ABORT_ARRAY]);     .REPEAT UNTIL  ALL BANKS ARE TESTED
```

```
;          5597
;          5598        !
;          5599        ! THE BLAST TABLE HAS NOW BEEN LOADED WITH ALL
;          5600        ! NEWLY FAILING ROWS AND COLUMNS WITHIN THIS TESTED ARRAY.
;          5601        !
;          5602
;          5603        if .FLG_REG [F_ABORT_ARRAY]                 !WAS THE ABORT ARRAY FLAG SET DURING PROM MAINT
;          5604        then
;          5605            PRINTB (ONE_MSG, ABORT_MSG)             !REPORT THAT PROM MAINT FOR THIS ARRAY IS ABORTED
;          5606        else
;          5607            begin
;          5608            IN_BLAST_TBL (.ARR$BNK_SEL, .TSTED_BNK);        .INTERIGATE THE BLAST TABLE
;          5609
;          5610            if .BAD_BNK_REG eql ZERO                !ARE THERE ANY ADDITIONAL ERRORS TO BLAST
;          5611            then
;          5612                PRINTB (ONE_MSG, NO_AD_MSG)         .PRINT NO ADDITIONAL ERRORS MESSAGE
;          5613            else
;          5614                begin
;          5615                CAL_CHK_SUM ();                     !CALCULATE THE NEW PROM DATA CHECK SUMS
;          5616                BL_PROMS (.ARR$BNK_SEL),            !BLAST THE PROMS WITH THE NEWLY FAILING ROWS/COLS
;          5617
;          5618                if .FLG_REG [F_D_CLK_TIME_OUT] then exitloop;
;          5619
;          5620                VER_BLAST (.ARR$BNK_SEL);           !VERIFY THAT THE BLAST WAS SUCCESSFUL
;          5621                VER_ERROR_MASK (.ARR$BNK_SEL, .TSTED_BNK);
;          5622                                                    !VERIFY THAT THE BLAST MASKED OUT THE NEW FAILING ROWS/COLS
;          5623                end;
;          5624
;          5625            end;
;          5626
;          5627        ARR$BNK_SEL = .ARR$BNK_SEL + .INC_ARR;      !INCREMENT TO THE NEXT ARRAY
;          5628        end;
;          5629
;          5630    !
;          5631    ! PROMPT THE OPERATOR THAT THE
;          5632    ! PROGRAM EXECUTION HAS ENDED
;          5633    !
;          5634    PRINTB (ONE_MSG, END_MSG);                      !REPORT THAT THE PROGRAM IS COMPLETED
;          5635    PRINTB (ONE_MSG, RET_DRS_MSG);                  !PRINT RETURNING TO DRS> MESSAGE
;          5636    DODU (.L$UNIT);                                 !DROP THIS UNIT
;          5637    DOCLN;                                          !GO TO THE CLEAR UP CODE SECTION
;          5638    ENDTST;


                                                .SBTTL  $T1 MAIN PROM MAINT CONTROL CODE
030006  004167  000000G                 $T1:    JSR     R1,$SAVE4                       ;                       5544
030012  012746  000052'                         MOV     #BGN.MSG,-(SP)                  ;                       5560
030016  012746  001630'                         MOV     #ONE.MSG,-(SP)
030022  012746  000002                          MOV     #2,-(SP)
030026  010600                                  MOV     SP,R0                           ; SP,*
030030  104414                                  TRAP    14
030032  016716  C36776'                         MOV     ML.DUT,(SP)                     ;                       5561
```

```
030036  012746  001736'                        MOV     #DRV.SEL.PRINT,-(SP)
030042  012746  000002                        MOV     #2,-(SP)
030046  010600                                MOV     SP,R0                   ; SP,*
030050  104414                                TRAP    14
030052  016704  037032'                        MOV     LST.TSTED.ARR,R4        ;                               5568
030056  005001                                CLR     R1                      ; ARR.SEL
030060  000572                                BR      9$
030062  104422                        1$:     TRAP    22                      ;                               5569
030064  005067  037022'                        CLR     DEGRADE.MOD.REG         ;                               5571
030070  005067  037016'                        CLR     BAD.BNK.REG             ;                               5572
030074  005067  037020'                        CLR     FLG.REG                 ;                               5573
030100  004767  154466                        JSR     PC,I.BLAST.TBL          ;                               5574
030104  016703  037002'                        MOV     TSTED.BNK,R3            ; *,TEMP.TSTED.BNK              5575
030110  016702  037000'                        MOV     ARR$BNK.SEL,R2          ; *,TEMP.ARR$BNK.SE             5576
030114  012746  037000'                        MOV     #ARR$BNK.SEL,-(SP)      ;                               5586
030120  016746  037010'                        MOV     ARR.SEL.POS,-(SP)
030124  012746  000004                        MOV     #4,-(SP)
030130  005046                                CLR     -(SP)
030132  004767  000000G                       JSR     PC,BL$GT2
030136  010016                                MOV     R0,(SP)
030140  012746  001770'                        MOV     #ARR.SEL.PRINT,-(SP)
030144  012746  000002                        MOV     #2,-(SP)
030150  010600                                MOV     SP,R0                   ; SP,*
030152  104414                                TRAP    14
030154  010246                        2$:     MOV     R2,-(SP)                ; TEMP.ARR$BNK.SE,*             5590
030156  016746  037012'                        MOV     BNK.SEL.POS,-(SP)
030162  012746  000002                        MOV     #2,-(SP)
030166  005046                                CLR     -(SP)
030170  004767  000000G                       JSR     PC,BL$GT1
030174  010016                                MOV     R0,(SP)
030176  012746  002036'                        MOV     #BNK.SEL.PRINT,-(SP)
030202  012746  000002                        MOV     #2,-(SP)
030206  010600                                MOV     SP,R0                   ; SP,*
030210  104414                                TRAP    14
030212  010216                                MOV     R2,(SP)                 ; TEMP.ARR$BNK.SE,*             5591
030214  004767  172354                        JSR     PC,FB.CHIPS
030220  010216                                MOV     R2,(SP)                 ; TEMP.ARR$BNK.SE,*             5592
030222  004767  172416                        JSR     PC,PM.THIS.BANK
030226  066702  037004'                        ADD     INC.BNK,R2              ; *,TEMP.ARR$BNK.SE             5593
030232  005303                                DEC     R3                      ; TEMP.TSTED.BNK                5594
030234  062706  000014                        ADD     #14,SP                  ;                               5589
030240  005703                                TST     R3                      ; TEMP.TSTED.BNK                5596
030242  001404                                BEQ     3$
030244  032767  000100  037020'                BIT     #100,FLG.REG
030252  001740                                BEQ     2$
030254  032767  000100  037020'        3$:     BIT     #100,FLG.REG            ;                               5603
030262  001411                                BEQ     4$
030264  012746  000766'                        MOV     #ABORT.MSG,-(SP)        ;                               5605
030270  012746  001630'                        MOV     #ONE.MSG,-(SP)
030274  012746  000002                        MOV     #2,-(SP)
030300  010600                                MOV     SP,R0                   ; SP,*
030302  104414                                TRAP    14
```

```
030304  000452                          BR      8$                      ;                               5603
030306  016746  037000'           4$:   MOV     ARR$BNK.SEL,-(SP)       ;                               5608
030312  016746  037002'                 MOV     TSTED.BNK,-(SP)
030316  004767  164254                  JSR     PC,IN.BLAST.TBL
030322  005767  037016'                 TST     BAD.BNK.REG             ;                               5610
030326  001012                          BNE     5$
030330  012746  001036'                 MOV     #NO.AD.MSG,-(SP)        ;                               5612
030334  012746  001630'                 MOV     #ONE.MSG,-(SP)
030340  012746  000002                  MOV     #2,-(SP)
030344  010600                          MOV     SP,R0                   ; SP,*
030346  104414                          TRAP    14
030350  005726                          TST     (SP)+
030352  000426                          BR      7$                      ;                               5610
030354  004767  172616           5$:   JSR     PC,CAL.CHK.SUM          ;                               5615
030360  016746  037000'                 MOV     ARR$BNK.SEL,-(SP)       ;                               5616
030364  004767  173446                  JSR     PC,BL.PROMS
030370  105767  037020'                 TSTB    FLG.REG                 ;                               5618
030374  100003                          BPL     6$
030376  062706  000022                  ADD     #22,SP
030402  000423                          BR      10$
030404  016716  037000'           6$:   MOV     ARR$BNK.SEL,(SP)        ;                               5620
030410  004767  174630                  JSR     PC,VER.BLAST
030414  016716  037000'                 MOV     ARR$BNK.SEL,(SP)        ;                               5621
030420  016746  037002'                 MOV     TSTED.BNK,-(SP)
030424  004767  175574                  JSR     PC,VER.ERROR.MASK
030430  005726                    7$:   TST     (SP)+                   ;                               5607
030432  066767  037006' 037000'  8$:   ADD     INC.ARR,ARR$BNK.SEL     ;                               5627
030440  062706  000022                  ADD     #22,SP                  ;                               5569
030444  005201                          INC     R1                      ; ARR.SEL                       5568
030446  020104                    9$:   CMP     R1,R4                   ; ARR.SEL,*
030450  003604                          BLE     1$
030452  012716  000274'           10$:  MOV     #END.MSG,(SP)           ;                               5634
030456  012746  001630'                 MOV     #ONE.MSG,-(SP)
030462  012746  000002                  MOV     #2,-(SP)
030466  010600                          MOV     SP,R0                   ; SP,*
030470  104414                          TRAP    14
030472  012716  000370'                 MOV     #RET.DRS.MSG,(SP)       ;                               5635
030476  012746  001630'                 MOV     #ONE.MSG,-(SP)
030502  012746  000002                  MOV     #2,-(SP)
030506  010600                          MOV     SP,R0                   ; SP,*
030510  104414                          TRAP    14
030512  016700  000000G                 MOV     L$UNIT,R0               ;                               5636
030516  104451                          TRAP    51
030520  104444                          TRAP    44
030522  062706  000022                  ADD     #22,SP                  ;                               5544
030526  000207                          RTS     PC

; Routine Size:  169 words
; Maximum stack depth per invocation:  22 words
```

```
                                                .SBTTL  T1 MAIN PROM MAINT CONTROL CODE

030530  004767  177252                  T1::
030530                                    1$:   JSR     PC,$T1                          ;                                5637
030534  104466                                  TRAP    66
030536  006000                                  ROR     R0
030540  103773                                  BLO     1$
030542  000207                                  RTS     PC

; Routine Size:  6 words
; Maximum stack depth per invocation:  0 words
```

```
;          5639  %sbttl 'CLEAN UP CODE SECTION'
;          5640  .
;          5641  . CLEAN UP CODE SECTION
;          5642  .
;          5643  BGNCLN;
;          5644  CLR_MBUS;
;          5645  ENDCLN;


                                                .SBTTL  LCLEAN CLEAN UP CODE SECTION
030544  010146                          LCLEAN: MOV     R1,-(SP)                ;                       5638
030546  016700  036772'                        MOV     ML.ADDR,R0              ;                       5643
030552  016046  000010                        MOV     10(R0),-(SP)            ; *,ML.REG
030556  011601                                MOV     (SP),R1                 ; ML.REG,MLREG
030560  152701  000040                        BISB    #40,R1                  ; *,MLREG
030564  016700  036772'                        MOV     ML.ADDR,R0
030570  010160  000010                        MOV     R1,10(R0)               ; MLREG,*
030574  016700  036772'                        MOV     ML.ADDR,R0
030600  016046  000010                        MOV     10(R0),-(SP)            ; *,ML.REG
030604  012601                                MOV     (SP)+,R1                ; ML.REG,MLREG
030606  016700  036776'                        MOV     ML.DUT,R0
030612  042700  177770                        BIC     #177770,R0
030616  142701  000007                        BICB    #7,R1                   ; *,MLREG
030622  050001                                BIS     R0,R1                   ; *,MLREG
030624  016700  036772'                        MOV     ML.ADDR,R0
030630  010160  000010                        MOV     R1,10(R0)               ; MLREG,*
030634  005726                                TST     (SP)+                   ;                       5638
030636  012601                                MOV     (SP)+,R1
030640  000207                                RTS     PC

; Routine Size:  31 words
; Maximum stack depth per invocation:  4 words



                                                .SBTTL  L$CLEAN CLEAN UP CODE SECTION
030642  004767  177676                  L$CLEAN::
                                                JSR     PC,LCLEAN               ;                       5644
030646  104412                                TRAP    12
030650  000207                                RTS     PC

; Routine Size:  4 words
; Maximum stack depth per invocation:  0 words


;          5646  end
;          5647
;          5648  eludom

                                        ; OTS external references
```

```
                .GLOBL  BL$GT2, $SAVE5, $SAVE4, $SAVE3
                .GLOBL  $SAVE2, BL$PU2, BL$GT1, BL$DIV
                .GLOBL  BL$MOD



; Size:         6357 code + 8970 data words
; Run Time:     01:28.3
; Elapsed Time: 02:51.0
; Memory Used:  131 pages
; Compilation Complete
```

SEQ 0240

$$LAS1	Pmskl2	432+#									
		432+#									
	Pmskl3	432+#									
		432+#									
	Pmskl4	445+#									
		446+#									
$$TMP	Pmskl2	259+	260+#								
		259+	260+#								
	Pmskl3	259+	260+#								
		259+	260+#								
	Pmskl4	272+	273+#								
		273+	274+#								
$$TMP1	Pmskl2	257+									
		257+									
	Pmskl3	257+									
		257+									
	Pmskl4	270+									
		271+									
$$TMP2	Pmskl2	252+	253+#	254+	263+#						
		252+	253+#	254+	263+#						
	Pmskl3	252+	253+#	254+	263+#						
		252+	253+#	254+	263+#						
	Pmskl4	265+	266+#	267+	276+#						
		266+	267+#	268+	277+#						
$BGNSEG	Pmskl2	334+#	459+	484+	496+	662+					
		334+#	459+	484+	496+	662+					
	Pmskl3	334+#	459+	484+	496+	662+					
		334+#	459+	484+	496+	662+					
	Pmskl4	347+#	472+	497+	509+	675+					
		348+#	473+	498+	510+	676+					
$BGNSUB	Pmskl2	333+#	454+	501+	506+	640+	648+				
		333+#	454+	501+	506+	640+	648+				
	Pmskl3	333+#	454+	501+	506+	640+	648+				
		333+#	454+	501+	506+	640+	648+				
	Pmskl4	346+#	467+	514+	519+	653+	661+				
		347+#	468+	515+	520+	654+	662+				
$BGNTST	Pmskl2	332+#	464+#	473+	619+#	631+					
		332+#	464+#	473+	619+#	631+					
	Pmskl3	332+#	464+#	473+	619+#	631+					
		332+#	464+#	473+	619+#	631+					
	Pmskl4	345+#	477+#	486+	632+#	644+					
		346+#	478+#	487+	633+#	645+					
$CODE$	Pmskl2	952+	962+	967+	1003+	1060+#	1061+	1064+	1074+	1081+#	1082+	1084+
		952+	962+	967+	1003+	1060+#	1061+	1064+	1074+	1081+#	1082+	1084+
		1094+	1101+	1134+	1140+	1143+	1148+	1150+	1152+	1157+	1159+	1164+
		1094+	1101+	1134+	1140+	1143+	1148+	1150+	1152+	1157+	1159+	1164+
		1166+	1168+	1176+	1178+	1180+	1182+	1184+	1186+	1188+	1190+	1192+
		1166+	1168+	1176+	1178+	1180+	1182+	1184+	1186+	1188+	1190+	1192+
		1194+	1196+	1201+	1203+	1205+	1210+	1215+	1217+	1219+	1221+	1226+
		1194+	1196+	1201+	1203+	1205+	1210+	1215+	1217+	1219+	1221+	1226+
		1228+	1230+	1232+	1234+	1236+	1238+	1253+	1259+	1283+#	1294+	1302+
		1228+	1230+	1232+	1234+	1236+	1238+	1253+	1259+	1283+#	1294+	1302+
		1310+	1317+	1322+	1348+	1385+	1420+	1489+				
		1310+	1317+	1322+	1348+	1385+	1420+	1489+				
	Pmskl3	952+	962+	967+	1003+	1060+#	1061+	1064+	1074+	1081+#	1082+	1084+
		952+	962+	967+	1003+	1060+#	1061+	1064+	1074+	1081+#	1082+	1084+
		1094+	1101+	1134+	1140+	1143+	1148+	1150+	1152+	1157+	1159+	1164+
		1094+	1101+	1134+	1140+	1143+	1148+	1150+	1152+	1157+	1159+	1164+
		1166+	1168+	1176+	1178+	1180+	1182+	1184+	1186+	1188+	1190+	1192+
		1166+	1168+	1176+	1178+	1180+	1182+	1184+	1186+	1188+	1190+	1192+

		1194+	1196+	1201+	1203+	1205+	1210+	1215+	1217+	1219+	1221+	1226+
		1194+	1196+	1201+	1203+	1205+	1210+	1215+	1217+	1219+	1221+	1 SEQ 0241
		1228+	1230+	1232+	1234+	1236+	1238+	1253+	1259+	1283+#	1294+	1302+
		1228+	1230+	1232+	1234+	1236+	1238+	1253+	1259+	1283+#	1294+	1302+
		1310+	1317+	1322+	1348+	1385+	1420+	1489+				
		1310+	1317+	1322+	1348+	1385+	1420+	1489+				
	Pmskl4	965+	975+	980+	1016+	1073+#	1074+	1077+	1087+	1094+#	1095+	1097+
		966+	976+	981+	1017+	1074+#	1075+	1078+	1088+	1095+#	1096+	1098+
		1107+	1114+	1147+	1153+	1156+	1161+	1163+	1165+	1170+	1172+	1177+
		1108+	1115+	1148+	1154+	1157+	1162+	1164+	1166+	1171+	1173+	1178+
		1179+	1181+	1189+	1191+	1193+	1195+	1197+	1199+	1201+	1203+	1205+
		1180+	1182+	1190+	1192+	1194+	1196+	1198+	1200+	1202+	1204+	1206+
		1207+	1209+	1214+	1216+	1218+	1223+	1228+	1230+	1232+	1234+	1239+
		1208+	1210+	1215+	1217+	1219+	1224+	1229+	1231+	1233+	1235+	1240+
		1241+	1243+	1245+	1247+	1249+	1251+	1266+	1272+	1296+#	1307+	1315+
		1242+	1244+	1246+	1248+	1250+	1252+	1267+	1273+	1297+#	1308+	1316+
		1323+	1330+	1335+	1361+	1398+	1433+	1502+				
		1324+	1331+	1336+	1362+	1399+	1434+	1503+				
$END_LINK	Pmskl2	385+*										
		385+*										
	Pmskl3	385+*										
		385+*										
	Pmskl4	398+*										
		399+*										
$GLOB$	Pmskl2	1075+#	1096+	1106+								
		1075+#	1096+	1106+								
	Pmskl3	1075+#	1096+	1106+								
		1075+#	1096+	1106+								
	Pmskl4	1088+#	1109+	1119+								
		1089+#	1110+	1120+								
$L	Pmskl2	1291+#										
		1291+#										
	Pmskl3	1291+#	1561	1566	1582							
		1291+#	1561	1566	1582							
	Pmskl4	1304+#										
		1305+#										
OWN	Pmskl2	1076+	1095+#	1287+								
		1076+	1095+#	1287+								
	Pmskl3	1076+	1095+#	1287+								
		1076+	1095+#	1287+								
	Pmskl4	1089+	1108+#	1300+								
		1090+	1109+#	1301+								
$POINT	Pmskl2	1002+#	1006+#	1055+								
		1002+#	1006+#	1055+								
	Pmskl3	1002+#	1006+#	1055+								
		1002+#	1006+#	1055+								
	Pmskl4	1015+#	1019+#	1068+								
		1016+#	1020+#	1069+								
$POPLEV	Pmskl2	339+#	470+	632+	653+	654+						
		339+#	470+	632+	653+	654+						
	Pmskl3	339+#	470+	632+	653+	654+						
		339+#	470+	632+	653+	654+						
	Pmskl4	352+#	483+	645+	666+	667+						
		353+#	484+	646+	667+	668+						
$PUSHLEV	Pmskl2	338+#	469+	482+	485+	632+						
		338+#	469+	482+	485+	632+						
	Pmskl3	338+#	469+	482+	485+	632+						
		338+#	469+	482+	485+	632+						
	Pmskl4	351+#	482+	495+	498+	645+						
		352+#	483+	496+	499+	646+						

						3						
$SEGNUM	Pmskl2	337+#	449+	472+	483+							
		337+#	449+	472+	483+							
	Pmskl3	337+#	449+	472+	483+							
		337+#	449+	472+	483+							
	Pmskl4	350+#	462+	485+	496+							
		351+#	463+	486+	497+							
$SUBNUM	Pmskl2	336+#	449+	471+	507+							
		336+#	449+	471+	507+							
	Pmskl3	336+#	449+	471+	507+							
		336+#	449+	471+	507+							
	Pmskl4	349+#	462+	484+	520+							
		350+#	463+	485+	521+							
$TSTNUM	Pmskl2	335+#	449+	474+	626+	628+	630+					
		335+#	449+	474+	626+	628+	630+					
	Pmskl3	335+#	449+	474+	626+	628+	630+					
		335+#	449+	474+	626+	628+	630+					
	Pmskl4	348+#	462+	487+	639+	641+	643+					
		349+#	463+	488+	640+	642+	644+					
XYZ	Pmskl2	383+	391+	392+	407+	416+						
		383+	391+	392+	407+	416+						
	Pmskl3	383+	391+	392+	407+	416+						
		383+	391+	392+	407+	416+						
	Pmskl4	396+	404+	405+	420+	429+						
		397+	405+	406+	421+	430+						
%REMAINING	Pmskl2	438+	695+	702+	709+	716+	728+	1055+				
		438+	695+	702+	709+	716+	728+	1055+				
	Pmskl3	438+	695+	702+	709+	716+	728+	1055+				
		438+	695+	702+	709+	716+	728+	1055+				
	Pmskl4	451+	708+	715+	722+	729+	741+	1068+				
		452+	709+	716+	723+	730+	742+	1069+				
A	Pmskl2	723+	726+	728+	889+	919+	951+	953+	1467+			
		723+	726+	728+	889+	919+	951+	953+	1467+			
	Pmskl3	723+	726+	728+	889+	919+	951+	953+	1467+			
		723+	726+	728+	889+	919+	951+	953+	1467+			
	Pmskl4	736+	739+	741+	902+	932+	964+	966+	1480+	3808	3848	3958
		737+	740+	742+	903+	933+	965+	967+	1481+	3809	3849	3959
		4959	4976	5008	5031	5054	5073	5090	5127	5140	5168	5197
		4960	4977	5009	5032	5055	5074	5091	5128	5141	5169	5198
ABORT_MSG	Pmskl4	2045#	5604									
		2046#	5605									
ADJACENT_CNT	Pmskl4	3172	3204#	3218#	3240							
		3173	3205#	3219#	3241							
ADR	Pmskl2	224+#										
		224+#										
	Pmskl3	224+#										
		224+#										
	Pmskl4	237+#										
		238+#										
ALL	Pmskl2	1007+	1508									
		1007+	1508									
	Pmskl3	1007+										
		1007+										
	Pmskl4	1020+	1678#	2526								
		1021+	1679#	2527								
ALL_BAD	Pmskl4	3630	3681#	3683	3685	3696	3697	3698	3704	3705	3706	3715
		3631	3682#	3684	3686	3697	3698	3699	3705	3706	3707	3716
		3717	3754#	3756	3758	3761	3762	3763	3771	3821	3823#	3900#
		3718	3755#	3757	3759	3762	3763	3764	3772	3822	3824#	3901#
		3902	3904	3907	3913	3916	3919	4017	4515	4520#	4522	4525#
		3903	3905	3908	3914	3917	3920	4018	4516	4521#	4523	4526#

	Pmskl4	1313+#	
		1314+#	
BGNHW	Pmskl2	1080+#	1544
		1080+#	1544
	Pmskl3	1080+#	
		1080+#	
	Pmskl4	1093+#	
		1094+#	
BGNINIT	Pmskl2	345+#	
		345+#	
	Pmskl3	345+#	
		345+#	
	Pmskl4	358+#	5358
		359+#	5359
BGNMSG	Pmskl2	514+#	
		514+#	
	Pmskl3	514+#	
		514+#	
	Pmskl4	527+#	2170
		528+#	2171
BGNPROT	Pmskl2	951+#	1594
		951+#	1594
	Pmskl3	951+#	
		951+#	
	Pmskl4	964+#	
		965+#	
BGNPTAB	Pmskl2	422+#	
		422+#	
	Pmskl3	422+#	
		422+#	
	Pmskl4	435+#	
		436+#	
BGNRPT	Pmskl2	375+#	
		375+#	
	Pmskl3	375+#	1627
		375+#	1627
	Pmskl4	388+#	
		389+#	
BGNSEG	Pmskl2	481+#	
		481+#	
	Pmskl3	481+#	
		481+#	
	Pmskl4	494+#	
		495+#	
BGNSETUP	Pmskl2	399+#	
		399+#	
	Pmskl3	399+#	
		399+#	
	Pmskl4	412+#	
		413+#	
BGNSFT	Pmskl2	1308+#	
		1308+#	
	Pmskl3	1308+#	1599
		1308+#	1599
	Pmskl4	1321+#	
		1322+#	
BGNSRV	Pmskl2	529+#	
		529+#	
	Pmskl3	529+#	
		529+#	

	Pmskl4	542+#	
		543+#	
BGNSUB	Pmskl2	495+#	
		495+#	
	Pmskl3	495+#	
		495+#	
	Pmskl4	508+#	
		509+#	
BGNSW	Pmskl2	1059+#	1576
		1059+#	1576
	Pmskl3	1059+#	
		1059+#	
	Pmskl4	1072+#	
		1073+#	
BGNTST	Pmskl2	453+#	
		453+#	
	Pmskl3	453+#	
		453+#	
	Pmskl4	466+#	5548
		467+#	5549
BGN_MSG	Pmskl4	2029#	5559
		2030#	5560
BITO	Pmskl2	198+#	
		198+#	
	Pmskl3	198+#	
		198+#	
	Pmskl4	211+#	
		212+#	
BITOO	Pmskl2	187+#	198+
		187+#	198+
	Pmskl3	187+#	198+
		187+#	198+
	Pmskl4	200+#	211+
		201+#	212+
BIT01	Pmskl2	186+#	197+
		186+#	197+
	Pmskl3	186+#	197+
		186+#	197+
	Pmskl4	199+#	210+
		200+#	211+
BIT02	Pmskl2	185+#	196+
		185+#	196+
	Pmskl3	185+#	196+
		185+#	196+
	Pmskl4	198+#	209+
		199+#	210+
BIT03	Pmskl2	184+#	195+
		184+#	195+
	Pmskl3	184+#	195+
		184+#	195+
	Pmskl4	197+#	208+
		198+#	209+
BIT04	Pmskl2	183+#	194+
		183+#	194+
	Pmskl3	183+#	194+
		183+#	194+
	Pmskl4	196+#	207+
		197+#	208+
BIT05	Pmskl2	182+#	193+
		182+#	193+

	Pmskl3	182+#	193+
		182+#	193+
	Pmskl4	195+#	206+
		196+#	207+
BIT06	Pmskl2	181+#	192+
		181+#	192+
	Pmskl3	181+#	192+
		181+#	192+
	Pmskl4	194+#	205+
		195+#	206+
BIT07	Pmskl2	180+#	191+
		180+#	191+
	Pmskl3	180+#	191+
		180+#	191+
	Pmskl4	193+#	204+
		194+#	205+
BIT08	Pmskl2	179+#	190+
		179+#	190+
	Pmskl3	179+#	190+
		179+#	190+
	Pmskl4	192+#	203+
		193+#	204+
BIT09	Pmskl2	178+#	189+
		178+#	189+
	Pmskl3	178+#	189+
		178+#	189+
	Pmskl4	191+#	202+
		192+#	203+
BIT1	Pmskl2	197+#	
		197+#	
	Pmskl3	197+#	
		197+#	
	Pmskl4	210+#	
		211+#	
BIT10	Pmskl2	177+#	
		177+#	
	Pmskl3	177+#	
		177+#	
	Pmskl4	190+#	
		191+#	
BIT11	Pmskl2	176+#	
		176+#	
	Pmskl3	176+#	
		176+#	
	Pmskl4	189+#	
		190+#	
BIT12	Pmskl2	175+#	
		175+#	
	Pmskl3	175+#	
		175+#	
	Pmskl4	188+#	
		189+#	
BIT13	Pmskl2	174+#	
		174+#	
	Pmskl3	174+#	
		174+#	
	Pmskl4	187+#	
		188+#	
BI 14	Pmskl2	173+#	
		173+#	

	Pmskl3	173+# 173+#			
	Pmskl4	186+# 187+#			
BIT15	Pmskl2	172+# 172+#			
	Pmskl3	172+# 172+#			
	Pmskl4	185+# 186+#			
BIT2	Pmskl2	196+# 196+#			
	Pmskl3	196+# 196+#			
	Pmskl4	209+# 210+#			
BIT3	Pmskl2	195+# 195+#			
	Pmskl3	195+# 195+#			
	Pmskl4	208+# 209+#			
BIT4	Pmskl2	194+# 194+#			
	Pmskl3	194+# 194+#			
	Pmskl4	207+# 208+#			
BIT5	Pmskl2	193+# 193+#			
	Pmskl3	193+# 193+#			
	Pmskl4	206+# 207+#			
BIT6	Pmskl2	192+# 192+#			
	Pmskl3	192+# 192+#			
	Pmskl4	205+# 206+#			
BIT7	Pmskl2	191+# 191+#			
	Pmskl3	191+# 191+#			
	Pmskl4	204+# 205+#			
BIT8	Pmskl2	190+# 190+#			
	Pmskl3	190+# 190+#			
	Pmskl4	203+# 204+#			
BIT9	Pmskl2	189+# 189+#			
	Pmskl3	189+# 189+#			
	Pmskl4	202+# 203+#			
BITS_TSTED	Pmskl4	4583 4584	4642# 4643#	4658 4659	4665# 4666#

B_11	Pmskl4	1845#	
		1846#	
B_12	Pmskl4	1846#	
		1847#	
B_13	Pmskl4	1847#	
		1848#	
B_14	Pmskl4	1848#	
		1849#	
B_15	Pmskl4	1849#	
		1850#	
B_16	Pmskl4	1854#	
		1855#	
B_17	Pmskl4	1855#	
		1856#	
B_18	Pmskl4	1856#	
		1857#	
B_19	Pmskl4	1857#	
		1858#	
B_2	Pmskl4	1836#	
		1837#	
B_20	Pmskl4	1858#	
		1859#	
B_21	Pmskl4	1859#	
		1860#	
B_22	Pmskl4	1860#	
		1861#	
B_23	Pmskl4	1861#	
		1862#	
B_24	Pmskl4	1862#	
		1863#	
B_25	Pmskl4	1863#	
		1864#	
B_26	Pmskl4	1864#	
		1865#	
B_27	Pmskl4	1865#	
		1866#	
B_28	Pmskl4	1866#	
		1867#	
B_29	Pmskl4	1867#	
		1868#	
B_3	Pmskl4	1837#	
		1838#	
B_30	Pmskl4	1868#	
		1869#	
B_31	Pmskl4	1869#	
		1870#	
B_32	Pmskl4	1822#	
		1823#	
B_33	Pmskl4	1823#	
		1824#	
B_34	Pmskl4	1824#	
		1825#	
B_35	Pmskl4	1825#	
		1826#	
B_36	Pmskl4	1826#	4756
		1827#	4757
B_37	Pmskl4	1827#	4788
		1828#	4789
B_38	Pmskl4	1828#	4819
		1829#	4820

	Pmskl4	23+#	524+									
		24+#	525+									
C$CEFG	Pmskl2	45+#										
		45+#										
	Pmskl3	45+#										
		45+#										
	Pmskl4	58+#										
		59+#										
C$CLCK	Pmskl2	58+#	842+									
		58+#	842+									
	Pmskl3	58+#	842+									
		58+#	842+									
	Pmskl4	71+#	855+									
		72+#	856+									
C$CLEAN	Pmskl2	18+#	593+									
		18+#	593+									
	Pmskl3	18+#	593+									
		18+#	593+									
	Pmskl4	31+#	606+									
		32+#	607+									
C$CLOS	Pmskl2	37+#	760+									
		37+#	760+									
	Pmskl3	37+#	760+									
		37+#	760+									
	Pmskl4	50+#	773+									
		51+#	774+									
C$CLP1	Pmskl2	14+#	305+									
		14+#	305+									
	Pmskl3	14+#	305+									
		14+#	305+									
	Pmskl4	27+#	318+									
		28+#	319+									
C$COUNT	Pmskl2	342+#	1132+	1139+	1331+	1345+	1348+	1368+	1382+	1385+	1405+	1417+
		342+#	1132+	1139+	1331+	1345+	1348+	1368+	1382+	1385+	1405+	1417+
		1420+	1435+									
		1420+	1435+									
	Pmskl3	342+#	1132+	1139+	1331+	1345+	1348+	1368+	1382+	1385+	1405+	1417+
		342+#	1132+	1139+	1331+	1345+	1348+	1368+	1382+	1385+	1405+	1417+
		1420+	1435+									
		1420+	1435+									
	Pmskl4	355+#	1145+	1152+	1344+	1358+	1361+	1381+	1395+	1398+	1418+	1430+
		356+#	1146+	1153+	1345+	1359+	1362+	1382+	1396+	1399+	1419+	1431+
		1433+	1448+									
		1434+	1449+									
C$CVEC	Pmskl2	38+#	685+									
		38+#	685+									
	Pmskl3	38+#	685+									
		38+#	685+									
	Pmskl4	51+#	698+									
		52+#	699+									
C$DCLN	Pmskl2	44+#	812+									
		44+#	812+									
	Pmskl3	44+#	812+									
		44+#	812+									
	Pmskl4	57+#	825+									
		58+#	826+									
C$DODU	Pmskl2	49+#	804+									
		49+#	804+									
	Pmskl3	49+#	804+									
		49+#	804+									

	Pmskl4	62+#	817+
		63+#	818+
C$DRP*	Pmskl2	28+#	796+
		28+#	796+
	Pmskl3	28+#	796+
		28+#	796+
	Pmskl4	41+#	809+
		42+#	810+
C$DU	Pmskl2	51+#	603+
		51+#	603+
	Pmskl3	51+#	603+
		51+#	603+
	Pmskl4	64+#	616+
		65+#	617+
C$EDIT	Pmskl2	1120+#	1189+
		1120+#	1189+
	Pmskl3	1120+#	1189+
		1120+#	1189+
	Pmskl4	1133+#	1202+
		1134+#	1203+
C$ERDF	Pmskl2	53+#	117+
		53+#	117+
	Pmskl3	53+#	117+
		53+#	117+
	Pmskl4	66+#	130+
		67+#	131+
C$ERMRD	Pmskl2	54+#	127+
		54+#	127+
	Pmskl3	54+#	127+
		54+#	127+
	Pmskl4	67+#	140+
		68+#	141+
C$ERROR	Pmskl2	56+#	103+
		56+#	103+
	Pmskl3	56+#	103+
		56+#	103+
	Pmskl4	69+#	116+
		70+#	117+
C$ERSF	Pmskl2	52+#	137+
		52+#	137+
	Pmskl3	52+#	137+
		52+#	137+
	Pmskl4	65+#	150+
		66+#	151+
C$ERSCFT	Pmskl2	55+#	147+
		55+#	147+
	Pmskl3	55+#	147+
		55+#	147+
	Pmskl4	68+#	160+
		69+#	161+
C$ESCAPE	Pmskl2	16+#	311+
		16+#	311+
	Pmskl3	16+#	311+
		16+#	311+
	Pmskl4	29+#	324+
		30+#	325+
C$ESEG	Pmskl2	13+#	661+
		13+#	661+
	Pmskl3	13+#	661+
		13+#	661+

SEQ 0255

	Pmskl4	26+#	674+		
		27+#	675+		
C$FSJB	Pmskl2	11+#	647+		
		11+#	647+		
	Pmskl3	11+#	647+		
		11+#	647+		
	Pmskl4	24+#	660+		
		25+#	661+		
C$ESST	Pmskl2	9+#	629+		
		9+#	629+		
	Pmskl3	9+#	629+		
		9+#	629+		
	Pmskl4	22+#	642+		
		23+#	643+		
C$EXIT	Pmskl2	34+#	318+	326+	
		34+#	318+	326+	
	Pmskl3	34+#	318+	326+	
		34+#	318+	326+	
	Pmskl4	47+#	331+	339+	
		48+#	332+	340+	
C$GETB	Pmskl2	30+#	775+		
		30+#	775+		
	Pmskl3	30+#	775+		
		30+#	775+		
	Pmskl4	43+#	788+		
		44+#	789+		
C$GETW	Pmskl2	31+#	770+		
		31+#	770+		
	Pmskl3	31+#	770+		
		31+#	770+		
	Pmskl4	44+#	783+		
		45+#	784+		
C$GMAN	Pmskl2	43+#	868+	88?+	897+
		43+#	868+	881+	897+
	Pmskl3	43+#	868+	881+	897+
		43+#	868+	88?+	897+
	Pmskl4	56+#	881+	894+	910+
		57+#	882+	895+	911+
C$GPHRD	Pmskl2	42+#	752+		
		42+#	752+		
	Pmskl3	42+#	752+		
		42+#	752+		
	Pmskl4	55+#	765+		
		56+#	766+		
C$GPLO	Pmskl2	32+#			
		32+#			
	Pmskl3	32+#			
		32+#			
	Pmskl4	45+#			
		46+#			
C$GPRI	Pmskl2	40+#	791+		
		40+#	791+		
	Pmskl3	40+#	791+		
		40+#	791+		
	Pmskl4	53+#	804+		
		54+#	805+		
C$INIT	Pmskl2	17+#	563+		
		17+#	563+		
	Pmskl3	17+#	563+		
		17+#	563+		

	Pmskl4	30+#	576+							
		31+#	577+							
C$INLP	Pmskl2	24+#	291+							
		24+#	291+							
	Pmskl3	24+#	291+							
		24+#	291+							
	Pmskl4	37+#	304+							
		38+#	305+							
C$ITCN	Pmskl2	948+#	975+	978+	980+	983+	989+	991+	993+	995+
		948+#	975+	978+	980+	983+	989+	991+	993+	995+
	Pmskl3	948+#	975+	978+	980+	983+	989+	991+	993+	995+
		948+#	975+	978+	980+	983+	989+	991+	993+	995+
	Pmskl4	961+#	988+	991+	993+	996+	1002+	1004+	1006+	1008+
		962+#	989+	992+	994+	997+	1003+	1005+	1007+	1009+
C$MANI	Pmskl2	48+#	298+							
		48+#	298+							
	Pmskl3	48+#	298+							
		48+#	298+							
	Pmskl4	61+#	311+							
		62+#	312+							
C$MEM	Pmskl2	33+#	284+							
		33+#	284+							
	Pmskl3	33+#	284+							
		33+#	284+							
	Pmskl4	46+#	297+							
		47+#	298+							
C$MSG	Pmskl2	27+#	519+							
		27+#	519+							
	Pmskl3	27+#	519+							
		27+#	519+							
	Pmskl4	40+#	532+							
		41+#	533+							
C$OPEN	Pmskl2	36+#	765+							
		36+#	765+							
	Pmskl3	36+#	765+							
		36+#	765+							
	Pmskl4	49+#	778+							
		50+#	779+							
C$PNTB	Pmskl2	20+#	695+							
		20+#	695+							
	Pmskl3	20+#	695+							
		20+#	695+							
	Pmskl4	33+#	708+							
		34+#	709+							
C$PNTF	Pmskl2	23+#	716+							
		23+#	716+							
	Pmskl3	23+#	716+							
		23+#	716+							
	Pmskl4	36+#	729+							
		37+#	730+							
C$PNTS	Pmskl2	22+#	709+							
		22+#	709+							
	Pmskl3	22+#	709+							
		22+#	709+							
	Pmskl4	35+#	722+							
		36+#	723+							
C$PNTX	Pmskl2	21+#	702+							
		21+#	702+							
	Pmskl3	21+#	702+							
		21+#	702+							

	Pmskl4	34+#	715+
		35+#	716+
C$QIO	Pmskl2	59+#	
		59+#	
	Pmskl3	59+#	
		59+#	
	Pmskl4	72+#	
		73+#	
C$RDBU	Pmskl2	15+#	277+
		15+#	277+
	Pmskl3	15+#	277+
		15+#	277+
	Pmskl4	28+#	290+
		29+#	291+
C$REFG	Pmskl2	47+#	820+
		47+#	820+
	Pmskl3	47+#	820+
		47+#	820+
	Pmskl4	60+#	833+
		61+#	834+
C$RESERV	Pmskl2	8+#	
		8+#	
	Pmskl3	8+#	
		8+#	
	Pmskl4	21+#	
		22+#	
C$RESET	Pmskl2	35+#	246+
		35+#	246+
	Pmskl3	35+#	246+
		35+#	246+
	Pmskl4	48+#	259+
		49+#	260+
C$REVISION	Pmskl2	1120+#	1189+
		1120+#	1189+
	Pmskl3	1120+#	1189+
		1120+#	1189+
	Pmskl4	1133+#	1202+
		1134+#	1203+
C$RFLA	Pmskl2	25+#	270+
		25+#	270+
	Pmskl3	25+#	270+
		25+#	270+
	Pmskl4	38+#	283+
		39+#	284+
C$RPT	Pmskl2	29+#	613+
		29+#	613+
	Pmskl3	29+#	613+
		29+#	613+
	Pmskl4	42+#	626+
		43+#	627+
C$SEFG	Pmskl2	46+#	
		46+#	
	Pmskl3	46+#	
		46+#	
	Pmskl4	59+#	
		60+#	
C$SPRI	Pmskl2	41+#	677+
		41+#	677+
	Pmskl3	41+#	677+
		41+#	677+

```
D                Pmskl2   911+
                          911+
                 Pmskl3   911+    1560     1565
                          911+    1560     1565
                 Pmskl4   924+
                          925+
D$ERR1           Pmskl2   732+#
                          732+#
                 Pmskl3   732+#
                          732+#
                 Pmskl4   745+#
                          746+#
D$ERR2           Pmskl2   734+#
                          734+#
                 Pmskl3   734+#
                          734+#
                 Pmskl4   747+#
                          748+#
D$ERR3           Pmskl2   736+#
                          736+#
                 Pmskl3   736+#
                          736+#
                 Pmskl4   749+#
                          750+#
D$ERR4           Pmskl2   738+#
                          738+#
                 Pmskl3   738+#
                          738+#
                 Pmskl4   751+#
                          752+#
D$ERR5           Pmskl2   740+#
                          740+#
                 Pmskl3   740+#
                          740+#
                 Pmskl4   753+#
                          754+#
D$ERR6           Pmskl2   742+#
                          742+#
                 Pmskl3   742+#
                          742+#
                 Pmskl4   755+#
                          756+#
D$ERR7           Pmskl2   744+#
                          744+#
                 Pmskl3   744+#
                          744+#
                 Pmskl4   757+#
                          758+#
D$ERR8           Pmskl2   746+#
                          746+#
                 Pmskl3   746+#
                          746+#
                 Pmskl4   759+#
                          760+#
D$ERR9           Pmskl2   748+#
                          748+#
                 Pmskl3   748+#
                          748+#
                 Pmskl4   761+#
                          762+#
```

DEGRADE_MOD_REG	Pmskl4	2005	4307#	4353#	4394#	4435#	4549#	5262#	5280#	5337	5570#	SEQ 0263
		2006	4308#	4354#	4395#	4436#	4550#	5263#	5281#	5338	5571#	
DELAY	Pmskl2	249+#										
		249+#										
	Pmskl3	249+#										
		249+#										
	Pmskl4	262+#	4081	4092	5159	5188	5209	5255	5273			
		263+#	4082	4093	5160	5189	5210	5256	5274			
DEPO	Pmskl2	1111+	1141+									
		1111+	1141+									
	Pmskl3	1111+	1141+									
		1111+	1141+									
	Pmskl4	1124+	1154+									
		1125+	1155+									
DESC	Pmskl2	1257+	1258+	1260+								
		1257+	1258+	1260+								
	Pmskl3	1257+	1258+	1260+								
		1257+	1258+	1260+								
	Pmskl4	1270+	1271+	1273+								
		1271+	1272+	1274+								
DESCRIPT	Pmskl2	1257+#										
		1257+#										
	Pmskl3	1257+#	1501									
		1257+#	1501									
	Pmskl4	1270+#										
		1271+#										
DEST	Pmskl2	769+	770+	774+	775+	779+	781+	782+#				
		769+	770+	774+	775+	779+	781+	782+#				
	Pmskl3	769+	770+	774+	775+	779+	781+	782+#				
		769+	770+	774+	775+	779+	781+	782+#				
	Pmskl4	782+	783+	787+	788+	792+	794+	795+#				
		783+	784+	788+	789+	793+	795+	796+#				
DEVTYP	Pmskl2	1251+#										
		1251+#										
	Pmskl3	1251+#	1498									
		1251+#	1498									
	Pmskl4	1264+#										
		1265+#										
DFLT	Pmskl2	862+	866+	873+	879+	886+	895+	927+	928+	930+	1326+	1339+
		862+	866+	873+	879+	886+	895+	927+	928+	930+	1326+	1339+
		1363+	1376+	1400+	1412+							
		1363+	1376+	1400+	1412+							
	Pmskl3	862+	866+	873+	879+	886+	895+	927+	928+	930+	1326+	1339+
		862+	866+	873+	879+	886+	895+	927+	928+	930+	1326+	1339+
		1363+	1376+	1400+	1412+							
		1363+	1376+	1400+	1412+							
	Pmskl4	875+	879+	886+	892+	899+	908+	940+	941+	943+	1339+	1352+
		876+	880+	887+	893+	900+	909+	941+	942+	944+	1340+	1353+
		1376+	1389+	1413+	1425+							
		1377+	1390+	1414+	1426+							
DFPTBL	Pmskl2	1544										
		1544										
DIAG_REG_SEL	Pmskl4	3036	3058#	3062#	3077							
		3037	3059#	3063#	3078							
DIF	Pmskl4	3504	3509#	3527#	3557	3575	3580#	3598#	3628			
		3505	3510#	3528#	3558	3576	3581#	3599#	3629			
DIF_COLS	Pmskl4	3667	3688#	3693								
		3668	3689#	3694								
DIF_ROWS	Pmskl4	3668	3690#	3693								
		3669	3691#	3694								

EE_PRINT	Pmskl4	2089#	2184			
		2090#	2185			
EFN	Pmskl2	819+	820+			
		819+	820+			
	Pmskl3	819+	820+			
		819+	820+			
	Pmskl4	832+	833+			
		833+	834+			
EF_CONTINUE	Pmskl2	205+#				
		205+#				
	Pmskl3	205+#				
		205+#				
	Pmskl4	218+#				
		219+#				
EF_NEW	Pmskl2	206+#				
		206+#				
	Pmskl3	206+#				
		206+#				
	Pmskl4	219+#				
		220+#				
EF_PWR	Pmskl2	207+#				
		207+#				
	Pmskl3	207+#				
		207+#				
	Pmskl4	220+#				
		221+#				
EF_RESTART	Pmskl2	204+#				
		204+#				
	Pmskl3	204+#				
		204+#				
	Pmskl4	217+#				
		218+#				
EF_START	Pmskl2	203+#				
		203+#				
	Pmskl3	203+#				
		203+#				
	Pmskl4	216+#	5391			
		217+#	5392			
EL_PRINT	Pmskl4	2090#	2185			
		2091#	2186			
EL_REG	Pmskl4	1882#				
		1883#				
EMT_ESLOAD	Pmskl2	1131+#	1220+			
		1131+#	1220+			
	Pmskl3	1131+#	1220+			
		1131+#	1220+			
	Pmskl4	1144+#	1233+			
		1145+#	1234+			
ENABE	Pmskl4	1638#	2238	2265	2375	2485
		1639#	2239	2266	2376	2486
ENDAU	Pmskl2	568+#				
		568+#				
	Pmskl3	568+#	1680			
		568+#	1680			
	Pmskl4	581+#				
		582+#				
ENDAUTO	Pmskl2	578+#				
		578+#				
	Pmskl3	578+#	1647			
		578+#	1647			

```
                    Pmskl4   591+#
                             592+#
ENDCLN              Pmskl2   588+#
                             588+#
                    Pmskl3   588+#
                             588+#
                    Pmskl4   601+#  5644
                             602+#  5645
ENDDU               Pmskl2   598+#
                             598+#
                    Pmskl3   598+#  1663
                             598+#  1663
                    Pmskl4   611+#
                             612+#
ENDHRD              Pmskl2  1316+#
                            1316+#
                    Pmskl3  1316+#  1584
                            1316+#  1584
                    Pmskl4  1329+#
                            1330+#
ENDHW               Pmskl2  1093+#  1556
                            1093+#  1556
                    Pmskl3  1093+#
                            1093+#
                    Pmskl4  1106+#
                            1107+#
ENDINIT             Pmskl2   558+#
                             558+#
                    Pmskl3   558+#
                             558+#
                    Pmskl4   571+#  5543
                             572+#  5544
ENDMSG              Pmskl2   553+#
                             553+#
                    Pmskl3   553+#
                             553+#
                    Pmskl4   566+#  2186
                             567+#  2187
ENDPROT             Pmskl2   957+#  1601
                             957+#  1601
                    Pmskl3   957+#
                             957+#
                    Pmskl4   970+#
                             971+#
ENDPTAB             Pmskl2   442+#
                             442+#
                    Pmskl3   442+#
                             442+#
                    Pmskl4   455+#
                             456+#
ENDRPT              Pmskl2   608+#
                             608+#
                    Pmskl3   608+#  1629
                             608+#  1629
                    Pmskl4   621+#
                             622+#
ENDSEG              Pmskl2   652+#
                             652+#
                    Pmskl3   652+#
                             652+#
```

	Pmskl4	665+#								
		666+#								
ENDSETUP	Pmskl2	411+#								
		411+#								
	Pmskl3	411+#								
		411+#								
	Pmskl4	424+#								
		425+#								
ENDSFT	Pmskl2	1321+#								
		1321+#								
	Pmskl3	1321+#	1612							
		1321+#	1612							
	Pmskl4	1334+#								
		1335+#								
ENDSRV	Pmskl2	537+#	549+							
		537+#	549+							
	Pmskl3	537+#	549+							
		537+#	549+							
	Pmskl4	550+#	562+							
		551+#	563+							
ENDSRV_MODPRI	Pmskl2	541+#								
		541+#								
	Pmskl3	541+#								
		541+#								
	Pmskl4	554+#								
		555+#								
ENDSUB	Pmskl2	639+#								
		639+#								
	Pmskl3	639+#								
		639+#								
	Pmskl4	652+#								
		653+#								
ENDSW	Pmskl2	1073+#	1581							
		1073+#	1581							
	Pmskl3	1073+#								
		1073+#								
	Pmskl4	1086+#								
		1087+#								
ENDTST	Pmskl2	618+#								
		618+#								
	Pmskl3	618+#								
		618+#								
	Pmskl4	631+#	5637							
		632+#	5638							
END_MSG	Pmskl4	2033#	5633							
		2034#	5634							
END_SEC	Pmskl4	3174	3188#	3195#	3206	3820	3864#	3870#	3879#	3885#	3889
		3175	3189#	3196#	3207	3821	3865#	3871#	3880#	3886#	3890
ENTER_BOARD	Pmskl3	1547	1561							
		1547	1561							
EQUALS	Pmskl2	167+#	1506							
		167+#	1506							
	Pmskl3	167+#	1492							
		167+#	1492							
	Pmskl4	180+#	2018							
		181+#	2019							
ERRBL*	Pmskl2	1105+#								
		1105+#								
	Pmskl3	1105+#								
		1105+#								

	Pmskl4	1118+#									
		1119+#									
ERRDF	Pmskl2	111+#									
		111+#									
	Pmskl3	111+#									
		111+#									
	Pmskl4	124+#									
		125+#									
ERRHRD	Pmskl2	121+#									
		121+#									
	Pmskl3	121+#									
		121+#									
	Pmskl4	134+#									
		135+#									
ERRMSG	Pmskl2	1104+									
		1104+									
	Pmskl3	1104+									
		1104+									
	Pmskl4	1117+									
		1118+									
ERRNBR	Pmskl2	1103+									
		1103+									
	Pmskl3	1103+									
		1103+									
	Pmskl4	1116+									
		1117+									
ERRNUM	Pmskl2	111+	117+	121+	127+	131+	137+	141+	147+		
		111+	117+	121+	127+	131+	137+	141+	147+		
	Pmskl3	111+	117+	121+	127+	131+	137+	141+	147+		
		111+	117+	121+	127+	131+	137+	141+	147+		
	Pmskl4	124+	130+	134+	140+	144+	150+	154+	160+		
		125+	131+	135+	141+	145+	151+	155+	161+		
ERROR	Pmskl2	102+#									
		102+#									
	Pmskl3	102+#									
		102+#									
	Pmskl4	115+#									
		116+#									
ERROR_MAP	Pmskl4	1973	2599	3083#	3210	3215	3892	3928#	3937#	3941#	3949#	3953#
		1974	2600	3084#	3211	3216	3893	3929#	3938#	3942#	3950#	3954#
		4064	4082#	4093#	4121	4128	4161	4165	4217	4247	4259	4276
		4065	4083#	4094#	4122	4129	4162	4166	4218	4248	4260	4277
		4289	4962	5006#	5029#	5052#	5071#	5088#	5097#	5237	5257	5275
		4290	4963	5007#	5030#	5053#	5072#	5089#	5098#	5238	5258	5276
ERRSF	Pmskl2	131+#									
		131+#									
	Pmskl3	131+#									
		131+#									
	Pmskl4	144+#	2389	2408	2446	2459	2495	2508	4305	4351	4392	4433
		145+#	2390	2409	2447	2460	2496	2509	4306	4352	4393	4434
		4546	4553	4927	5164	5193	5260	5278	5331	5340	5379	5394
		4547	4554	4928	5165	5194	5261	5279	5332	5341	5380	5395
		5410	5529									
		5411	5530									
ERRSOFT	Pmskl2	141+#									
		141+#									
	Pmskl3	141+#									
		141+#									
	Pmskl4	154+#									
		155+#									

ERRTBL	Pmskl2	1100+#	1525				
		1100+#	1525				
	Pmskl3	1100+#					
		1100+#					
	Pmskl4	1113+#					
		1114+#					
ERRTYP	Pmskl2	1102+	1122+	1128+			
		1102+	1122+	1128+			
	Pmskl3	1102+	1122+	1128+			
		1102+	1122+	1128+			
	Pmskl4	1115+	1135+	1141+			
		1116+	1136+	1142+			
ERR_1	Pmskl4	1585#	5394				
		1586#	5395				
ERR_10	Pmskl4	1594#	5164	5193			
		1595#	5165	5194			
ERR_11	Pmskl4	1595#	5331				
		1596#	5332				
ERR_12	Pmskl4	1596#	5340				
		1597#	5341				
ERR_13	Pmskl4	1597#	2389	2408	2446	2459	2495 2508
		1598#	2390	2409	2447	2460	2496 2509
ERR_2	Pmskl4	1586#	5529				
		1587#	5530				
ERR_3	Pmskl4	1587#	5410				
		1588#	5411				
ERR_4	Pmskl4	1588#	4927				
		1589#	4928				
ERR_5	Pmskl4	1589#	4546				
		1590#	4547				
ERR_6	Pmskl4	1590#	4553				
		1591#	4554				
ERR_7	Pmskl4	1591#	4305	4351	4392	4433	
		1592#	4306	4352	4393	4434	
ERR_8	Pmskl4	1592#	5260	5278			
		1593#	5261	5279			
ERR_9	Pmskl4	1593#	5379				
		1594#	5380				
ER_PRINT	Pmskl4	2084#	2179				
		2085#	2180				
ESCAPE	Pmskl2	310+#					
		310+#					
	Pmskl3	310+#					
		310+#					
	Pmskl4	323+#					
		324+#					
EVL	Pmskl2	222+#					
		222+#					
	Pmskl3	222+#					
		222+#					
	Pmskl4	235+#					
		236+#					
EXIT	Pmskl2	316+#					
		316+#					
	Pmskl3	316+#					
		316+#					
	Pmskl4	329+#					
		330+#					
EXIT_TST	Pmskl2	324+#					
		324+#					

		2938#	2953#	2968#	2983#	2994#	3064	3084#	4543	4548#	4555#	4633#
		2939#	2954#	2969#	2984#	2995#	3065	3085#	4544	4549#	4556#	
		4862#	4866	4900#	4907#	4920	4935	5167#	5196#	5328	5572#	5595
		4863#	4867	4901#	4908#	4921	4936	5168#	5197#	5329	5573#	5596
		5602	5617									
		5603	5618									
FORDEC	Pmskl2	965+	988+#	995+								
		965+	988+#	995+								
	Pmskl3	965+	988+#	995+								
		965+	988+#	995+								
	Pmskl4	978+	1001+#	1008+								
		979+	1002+#	1009+								
FOUR_MSG	Pmskl4	2062#										
		2063#										
FULL_WRD	Pmskl4	2157#	2540	3210	3464	4082	4093	4126	4128	4129	4165	4166
		2158#	2541	3211	3465	4083	4094	4127	4129	4130	4166	4167
		4974	4980	5005	5006	5007	5028	5029	5030	5051	5052	5053
		4975	4981	5006	5007	5008	5029	5030	5031	5052	5053	5054
		5070	5071	5072	5087	5088	5089	5096	5097	5152	5157	5180
		5071	5072	5073	5088	5089	5090	5097	5098	5153	5158	5181
		5186	5257	5275								
		5187	5258	5276								
FUNC	Pmskl4	1727#	2128	2243	2270	2380	2441	2490				
		1728#	2129	2244	2271	2381	2442	2491				
FUNC_1	Pmskl4	1542#										
		1543#										
FUNC_2	Pmskl4	1543#	2128									
		1544#	2129									
FUNC_3	Pmskl4	1544#										
		1545#										
FUNC_4	Pmskl4	1545#	2243	2380								
		1546#	2244	2381								
FUNC_5	Pmskl4	1546#	2270	2441								
		1547#	2271	2442								
FUNC_6	Pmskl4	1547#	2490									
		1548#	2491									
F_ABORT_ARRAY	Pmskl4	1612#	4555	4935	5595	5602						
		1613#	4556	4936	5596	5603						
F_ALL_BAD_CHIP	Pmskl4	1610#	4543	4548	4900							
		1611#	4544	4549	4901							
F_BLST_TBL_ENTERED	Pmskl4	1609#										
		1610#										
F_D_CLK_TIME_OUT	Pmskl4	1613#	5167	5196	5617							
		1614#	5168	5197	5618							
F_ERR_MAP_ENTERED	Pmskl4	1608#	3084	4907	4920							
		1609#	3085	4908	4921							
F_FLG_ERR	Pmskl4	1606#										
		1607#										
F_HWTBL	Pmskl2	1552										
		1552										
F_RAND_DATA	Pmskl4	1611#	2792	2803	2818	2833	2848	2863	2878	2893	2908	2923
		1612#	2793	2804	2819	2834	2849	2864	2879	2894	2909	2924
		2938	2953	2968	2983	2994	3064					
		2939	2954	2969	2984	2995	3065					
F_UNC_ERR_FLG	Pmskl4	1607#	4633	4862	4866	5328						
		1608#	4634	4863	4867	5329						
G$CNTOP	Pmskl2	89+#	1344+	1381+	1416+							
		89+#	1344+	1381+	1416+							
	Pmskl3	89+#	1344+	1381+	1416+							
		89+#	1344+	1381+	1416+							

	Pmskl4	102+#	1357+	1394+	1429+
		103+#	1358+	1395+	1430+
G$DISP	Pmskl2	69+#	1490+		
		69+#	1490+		
	Pmskl3	69+#	1490+		
		69+#	1490+		
	Pmskl4	82+#	1503+		
		83+#	1504+		
G$EXCP	Pmskl2	92+#	1434+		
		92+#	1434+		
	Pmskl3	92+#	1434+		
		92+#	1434+		
	Pmskl4	105+#	1447+		
		106+#	1448+		
G$HILIM	Pmskl2	94+#	1446+		
		94+#	1446+		
	Pmskl3	94+#	1446+		
		94+#	1446+		
	Pmskl4	107+#	1459+		
		108+#	1460+		
G$LOLIM	Pmskl2	93+#	1439+		
		93+#	1439+		
	Pmskl3	93+#	1439+		
		93+#	1439+		
	Pmskl4	106+#	1452+		
		107+#	1453+		
G$NO	Pmskl2	73+#	931+		
		73+#	931+		
	Pmskl3	73+#	931+		
		73+#	931+		
	Pmskl4	86+#	944+		
		87+#	945+		
G$OFFSET	Pmskl2	85+#	1336+	1373+	1409+
		85+#	1336+	1373+	1409+
	Pmskl3	85+#	1336+	1373+	1409+
		85+#	1336+	1373+	1409+
	Pmskl4	98+#	1349+	1386+	1422+
		99+#	1350+	1387+	1423+
G$OFSIZE	Pmskl2	84+#	1482+		
		84+#	1482+		
	Pmskl3	84+#	1482+		
		84+#	1482+		
	Pmskl4	97+#	1495+		
		98+#	1496+		
G$PRMA	Pmskl2	67+#	892+	1336+	
		67+#	892+	1336+	
	Pmskl3	67+#	892+	1336+	
		67+#	892+	1336+	
	Pmskl4	80+#	905+	1349+	
		81+#	906+	1350+	
G$PRMD	Pmskl2	68+#	876+	1373+	
		68+#	876+	1373+	
	Pmskl3	68+#	876+	1373+	
		68+#	876+	1373+	
	Pmskl4	81+#	889+	1386+	
		82+#	890+	1387+	
G$PRML	Pmskl2	66+#	865+	1409+	
		66+#	865+	1409+	
	Pmskl3	66+#	865+	1409+	
		66+#	865+	1409+	

	Pmskl4	79+#	878+	1422+							
		80+#	879+	1423+							
G$RADA	Pmskl2	81+#	920+								
		81+#	920+								
	Pmskl3	81+#	920+								
		81+#	920+								
	Pmskl4	94+#	933+								
		95+#	934+								
G$RADB	Pmskl2	77+#	904+								
		77+#	904+								
	Pmskl3	77+#	904+								
		77+#	904+								
	Pmskl4	90+#	917+								
		91+#	918+								
G$RADD	Pmskl2	79+#	912+								
		79+#	912+								
	Pmskl3	79+#	912+								
		79+#	912+								
	Pmskl4	92+#	925+								
		93+#	926+								
G$RADL	Pmskl2	80+#	865+	916+							
		80+#	865+	916+							
	Pmskl3	80+#	865+	916+							
		80+#	865+	916+							
	Pmskl4	93+#	878+	929+							
		94+#	879+	930+							
G$RADO	Pmskl2	78+#	908+								
		78+#	908+								
	Pmskl3	78+#	908+								
		78+#	908+								
	Pmskl4	91+#	921+								
		92+#	922+								
G$XFER	Pmskl2	70+#	1286+								
		70+#	1286+								
	Pmskl3	70+#	1286+								
		70+#	1286+								
	Pmskl4	83+#	1299+								
		84+#	1300+								
G$YES	Pmskl2	74+#	929+								
		74+#	929+								
	Pmskl3	74+#	929+								
		74+#	929+								
	Pmskl4	87+#	942+								
		88+#	943+								
GEN_RANDOM_DATA	Pmskl4	2188*	2294	2637	2812	2827	2842	2857	2872	2887	2902	2917
		2189*	2295	2638	2813	2828	2843	2858	2873	2888	2903	2918
		2932	2947	2962	2977							
		2933	2948	2963	2978							
GETBYTE	Pmskl2	774+#										
		774+#										
	Pmskl3	774+#										
		774+#										
	Pmskl4	787+#										
		788+#										
GETPRI	Pmskl2	790+#										
		790+#										
	Pmskl3	790+#										
		790+#										
	Pmskl4	803+#										
		804+#										

GETWORD	Pmskl2	769+#					
		769+#					
	Pmskl3	769+#					
		769+#					
	Pmskl4	782+#					
		783+#					
GET_DATA	Pmskl2	770+	775+	779+#			
		770+	775+	779+#			
	Pmskl3	770+	775+	779+#			
		770+	775+	779+#			
	Pmskl4	783+	788+	792+#			
		784+	789+	793+#			
GMANIA	Pmskl2	886+#					
		886+#					
	Pmskl3	886+#					
		886+#					
	Pmskl4	899+#					
		900+#					
GMANID	Pmskl2	873+#					
		873+#					
	Pmskl3	873+#					
		873+#					
	Pmskl4	886+#					
		887+#					
GMANIL	Pmskl2	862+#					
		862+#					
	Pmskl3	862+#					
		862+#					
	Pmskl4	875+#					
		876+#					
GO	Pmskl4	1726#					
		1727#					
GP$ATHI	Pmskl2	1461+#					
		1461+#					
	Pmskl3	1461+#					
		1461+#					
	Pmskl4	1474+#					
		1475+#					
GP$ATLO	Pmskl2	1455+#					
		1455+#					
	Pmskl3	1455+#					
		1455+#					
	Pmskl4	1468+#					
		1469+#					
GP$DISP	Pmskl2	1489+					
		1489+					
	Pmskl3	1489+					
		1489+					
	Pmskl4	1502+					
		1503+					
GP$_COUNT	Pmskl2	1247+#	1328+	1348+	1365+	1385+	1402+	1420+
		1247+#	1328+	1348+	1365+	1385+	1402+	1420+
	Pmskl3	1247+#	1328+	1348+	1365+	1385+	1402+	1420+
		1247+#	1328+	1348+	1365+	1385+	1402+	1420+
	Pmskl4	1260+#	1341+	1361+	1378+	1398+	1415+	1433+
		1261+#	1342+	1362+	1379+	1399+	1416+	1434+
GPHARD	Pmskl2	751+#						
		751+#						
	Pmskl3	751+#						
		751+#						

	Pmskl4	170+# 171+#		
IOLN	Pmskl2	158+# 158+#		
	Pmskl3	158+# 158+#		
	Pmskl4	171+# 172+#		
IOPL	Pmskl2	162+# 162+#		
	Pmskl3	162+# 162+#		
	Pmskl4	175+# 176+#		
IOSB	Pmskl2	160+# 160+#		
	Pmskl3	160+# 160+#		
	Pmskl4	173+# 174+#		
IRDY	Pmskl4	1752# 1753#		
IS	Pmskl2	163+ 163+		
	Pmskl3	163+ 163+		
	Pmskl4	176+ 177+		
ISR	Pmskl2	226+#	529+	530+*
		226+#	529+	530+*
	Pmskl3	226+#	529+	530+*
		226+#	529+	530+*
	Pmskl4	239+#	542+	543+*
		240+#	543+	544+*
IXE	Pmskl2	231+# 231+#		
	Pmskl3	231+# 231+#		
	Pmskl4	244+# 245+#		
I_BLAST_TBL	Pmskl4	2602*	5573	
		2603*	5574	
I_CHIP_TBL	Pmskl4	2515*	4863	
		2516*	4864	
I_COL_CNT_TBL	Pmskl4	2543*	3824	
		2544*	3825	
I_COL_PTR	Pmskl4	2574*	3862	3877
		2575*	3863	3878
I_ERROR_MAP	Pmskl4	2588*	4219	4909
		2589*	4220	4910
I_HWTBL	Pmskl2	1555 1555		
I_REM_TBL	Pmskl4	2557*	3674	
		2558*	3675	
I_TMP_BLST_TBL	Pmskl4	2529*	4519	
		2530*	4520	
KIND	Pmskl2	840+	842+	
		840+	842+	
	Pmskl3	840+	842+	
		840+	842+	

	Pmskl4	853+	855+	
		854+	856+	
L	Pmskl2	828+	915+	1410+
		828+	915+	1410+
	Pmskl3	828+	915+	1410+
		828+	915+	1410+
	Pmskl4	841+	928+	1423+
		842+	929+	1424+
LSACP	Pmskl2	1230+		
		1230+		
	Pmskl3	1230+		
		1230+		
	Pmskl4	1243+		
		1244+		
LSAPT	Pmskl2	1178+		
		1178+		
	Pmskl3	1178+		
		1178+		
	Pmskl4	1191+		
		1192+		
LSAU	Pmskl2	570+*	1118+	1208+
		570+*	1118+	1208+
	Pmskl3	570+*	1118+	1208+
		570+*	1118+	1208+
	Pmskl4	583+*	1131+	1221+
		584+*	1132+	1222+
LSAUT	Pmskl2	1205+		
		1205+		
	Pmskl3	1205+		
		1205+		
	Pmskl4	1218+		
		1219+		
LSAUTO	Pmskl2	580+*	1118+	1231+
		580+*	1118+	1231+
	Pmskl3	580+*	1118+	1231+
		580+*	1118+	1231+
	Pmskl4	593+*	1131+	1244+
		594+*	1132+	1245+
LSCCP	Pmskl2	1228+		
		1228+		
	Pmskl3	1228+		
		1228+		
	Pmskl4	1241+		
		1242+		
LSCLCK	Pmskl2	842+	852+#	
		842+	852+#	
	Pmskl3	842+	852+#	
		842+	852+#	
	Pmskl4	855+	865+#	
		856+	866+#	
LSCLEAN	Pmskl2	590+*	1117+	1229+
		590+*	1117+	1229+
	Pmskl3	590+*	1117+	1229+
		590+*	1117+	1229+
	Pmskl4	603+*	1130+	1242+
		604+*	1131+	1243+
LSCLRVEC	Pmskl2	685+	689+#	
		685+	689+#	
	Pmskl3	685+	689+#	
		685+	689+#	

	Pmskl4	698+	702+#	
		699+	703+#	
L$CO	Pmskl2	1168+		
		1168+		
	Pmskl3	1168+		
		1168+		
	Pmskl4	1181+		
		1182+		
L$DEPO	Pmskl2	1142+#		
		1142+#		
	Pmskl3	1142+#		
		1142+#		
	Pmskl4	1155+#		
		1156+#		
L$DESC	Pmskl2	1118+	1218+	1258+
		1118+	1218+	1258+
	Pmskl3	1118+	1218+	1258+
		1118+	1218+	1258+
	Pmskl4	1131+	1231+	1271+
		1132+	1232+	1272+
L$DESP	Pmskl2	1217+		
		1217+		
	Pmskl3	1217+		
		1217+		
	Pmskl4	1230+		
		1231+		
L$DEVP	Pmskl2	1194+		
		1194+		
	Pmskl3	1194+		
		1194+		
	Pmskl4	1207+		
		1208+		
L$DISPATCH	Pmskl2	967+	1126+	1181+
		967+	1126+	1181+
	Pmskl3	967+	1126+	1181+
		967+	1126+	1181+
	Pmskl4	980+	1139+	1194+
		981+	1140+	1195+
L$DLV	Pmskl2	251+	257+	1236+
		251+	257+	1236+
	Pmskl3	251+	257+	1236+
		251+	257+	1236+
	Pmskl4	264+	270+	1249+
		265+	271+	1250+
L$DORLN	Pmskl2	812+	816+#	
		812+	816+#	
	Pmskl3	812+	816+#	
		812+	816+#	
	Pmskl4	825+	829+#	
		826+	830+#	
L$DODU	Pmskl2	804+	808+#	
		804+	808+#	
	Pmskl3	804+	808+#	
		804+	808+#	
	Pmskl4	817+	821+#	
		818+	822+#	
L$DORP	Pmskl2	796+	800+#	
		796+	800+#	
	Pmskl3	796+	800+#	
		796+	800+#	

```
                    Pmskl4   809+     813+#
                             810+     814+#
L$DTP               Pmskl2  1180+
                            1180+
                    Pmskl3  1180+
                            1180+
                    Pmskl4  1193+
                            1194+
L$DTVP              Pmskl2  1176+
                            1176+
                    Pmskl3  1176+
                            1176+
                    Pmskl4  1189+
                            1190+
L$DU                Pmskl2   600+*  1118+   1213+
                             600+*  1118+   1213+
                    Pmskl3   600+*  1118+   1213+
                             600+*  11 8+   1213+
                    Pmskl4   613+*  1131+   1226+
                             614+*  1132+   1227+
L$DUT               Pmskl2  1210+
                            1210+
                    Pmskl3  1210+
                            1210+
                    Pmskl4  1223+
                            1224+
L$DVTVP             Pmskl2  1118+   1195+   1252+
                            1118+   1195+   1252+
                    Pmskl3  1118+   1195+   1252+
                            1118+   1195+   1252+
                    Pmskl4  1131+   1208+   1265+
                            1132+   1209+   1266+
L$EF                Pmskl2  1190+
                            1190+
                    Pmskl3  1190+
                            1190+
                    Pmskl4  1203+
                            1204+
L$ENVI              Pmskl2  1184+
                            1184+
                    Pmskl3  1184+
                            1184+
                    Pmskl4  1197+
                            1198+
L$ERROR             Pmskl2   103+     108+#
                             103+     108+#
                    Pmskl3   103+     108+#
                             103+     108+#
                    Pmskl4   116+     121+#
                             117+     122+#
L$ERRTBL            Pmskl2  1128+#   1224+
                            1128+#   1224+
                    Pmskl3  1128+#   1224+
                            1128+#   1224+
                    Pmskl4  1141+#   1237+
                            1142+#   1238+
L$ETP               Pmskl2  1221+
                            1221+
                    Pmskl3  1221+
                            1221+
```

	Pmskl4	1234+		
		1235+		
L$EXP1	Pmskl2	1186+		
		1186+		
	Pmskl3	1186+		
		1186+		
	Pmskl4	1199+		
		1200+		
L$EXP4	Pmskl2	1201+		
		1201+		
	Pmskl3	1201+		
		1201+		
	Pmskl4	1214+		
		1215+		
L$EXP5	Pmskl2	1203+		
		1203+		
	Pmskl3	1203+		
		1203+		
	Pmskl4	1216+		
		1217+		
L$GPHARD	Pmskl2	752+	756+#	
		752+	756+#	
	Pmskl3	752+	756+#	
		752+	756+#	
	Pmskl4	765+	769+#	
		766+	770+#	
L$HARD	Pmskl2	1118+	1151+	1304+#
		1118+	1151+	1304+#
	Pmskl3	1118+	1151+	1304+#
		1118+	1151+	1304+#
	Pmskl4	1131+	1164+	1317+#
		1132+	1165+	1318+#
L$HIME	Pmskl2	1238+		
		1238+		
	Pmskl3	1238+		
		1238+		
	Pmskl4	1251+		
		1252+		
L$HPCP	Pmskl2	1150+		
		1150+		
	Pmskl3	1150+		
		1150+		
	Pmskl4	1163+		
		1164+		
L$HPTP	Pmskl2	1157+		
		1157+		
	Pmskl3	1157+		
		1157+		
	Pmskl4	1170+		
		1171+		
L$HRDLN	Pmskl2	1302+	1303+	1304+
		1302+	1303+	1304+
	Pmskl3	1302+	1303+	1304+
		1302+	1303+	1304+
	Pmskl4	1315+	1316+	1317+
		1316+	1317+	1318+
L$HW	Pmskl2	1088+	1130+#	1158+
		1088+	1130+#	1158+
	Pmskl3	1088+	1130+#	1158+
		1088+	1130+#	1158+

	Pmskl4	1101+	1143+#	1171+	
		1102+	1144+#	1172+	
L$HWLEN	Pmskl2	1084+	1085+	1124+	1130+
		1084+	1085+	1124+	1130+
	Pmskl3	1084+	1085+	1124+	1130+
		1084+	1085+	1124+	1130+
	Pmskl4	1097+	1098+	1137+	1143+
		1098+	1099+	1138+	1144+
L$ICP	Pmskl2	1226+			
		1226+			
	Pmskl3	1226+			
		1226+			
	Pmskl4	1239+			
		1240+			
L$INI	Pmskl2	560+*	1117+	1227+	
		560+*	1117+	1227+	
	Pmskl3	560+*	1117+	1227+	
		560+*	1117+	1227+	
	Pmskl4	573+*	1130+	1240+	
		574+*	1131+	1241+	
L$INLOOP	Pmskl2	291+	294+#		
		291+	294+#		
	Pmskl3	291+	294+#		
		291+	294+#		
	Pmskl4	304+	307+#		
		305+	308+#		
L$ISR	Pmskl2	530+#	534+#		
		530+#	534+#		
	Pmskl3	530+#	534+#		
		530+#	534+#		
	Pmskl4	543+#	547+#		
		544+#	548+#		
L$LADP	Pmskl2	1164+			
		1164+			
	Pmskl3	1164+			
		1164+			
	Pmskl4	1177+			
		1178+			
L$LAST	Pmskl2	395+#	435+	1118+	1165+
		395+#	435+	1118+	1165+
	Pmskl3	395+#	435+	1118+	1165+
		395+#	435+	1118+	1165+
	Pmskl4	408+#	448+	1131+	1178+
		409+#	449+	1132+	1179+
L$LOAD	Pmskl2	1219+			
		1219+			
	Pmskl3	1219+			
		1219+			
	Pmskl4	1232+			
		1233+			
L$LUN	Pmskl2	1215+			
		1215+			
	Pmskl3	1215+			
		1215+			
	Pmskl4	1228+			
		1229+			
L$MANUAL	Pmskl2	298+	301+#		
		298+	301+#		
	Pmskl3	298+	301+#		
		298+	301+#		

	Pmskl4	311+	314+#	
		312+	315+#	
L$MEMORY	Pmskl2	284+	287+#	
		284+	287+#	
	Pmskl3	284+	287+#	
		284+	287+#	
	Pmskl4	297+	300+#	
		298+	301+#	
L$MREV	Pmskl2	1188+		
		1188+		
	Pmskl3	1188+		
		1188+		
	Pmskl4	1201+		
		1202+		
L$MSG	Pmskl2	519+#	526+#	
		519+#	526+#	
	Pmskl3	519+#	526+#	
		519+#	526+#	
	Pmskl4	532+#	539+#	
		533+#	540+#	
L$NAME	Pmskl2	1134+		
		1134+		
	Pmskl3	1134+		
		1134+		
	Pmskl4	1147+		
		1148+		
L$NDHRD	Pmskl2	1301+	1303+	1317+
		1301+	1303+	1317+
	Pmskl3	1301+	1303+	1317+
		1301+	1303+	1317+
	Pmskl4	1314+	1316+	1330+
		1315+	1317+	1331+
L$NDHW	Pmskl2	1083+	1085+	1094+#
		1083+	1085+	1094+#
	Pmskl3	1083+	1085+	1094+#
		1083+	1085+	1094+#
	Pmskl4	1096+	1098+	1107+#
		1097+	1099+	1108+#
L$NDSFT	Pmskl2	1309+	1311+	1322+
		1309+	1311+	1322+
	Pmskl3	1309+	1311+	1322+
		1309+	1311+	1322+
	Pmskl4	1322+	1324+	1335+
		1323+	1325+	1336+
L$NDSW	Pmskl2	1062+	1065+	1074+#
		1062+	1065+	1074+#
	Pmskl3	1062+	1065+	1074+#
		1062+	1065+	1074+#
	Pmskl4	1075+	1078+	1087+#
		1076+	1079+	1088+#
L$PRIO	Pmskl2	1182+		
		1182+		
	Pmskl3	1182+		
		1182+		
	Pmskl4	1195+		
		1196+		
L$PROT	Pmskl2	952+	1125+	1233+
		952+	1125+	1233+
	Pmskl3	952+	1125+	1233+
		952+	1125+	1233+

	Pmskl4	965+	1138+	1246+
		966+	1139+	1247+
L$PRT	Pmskl2	1232+		
		1232+		
	Pmskl3	1232+		
		1232+		
	Pmskl4	1245+		
		1246+		
L$READBUS	Pmskl2	277+	280+#	
		277+	280+#	
	Pmskl3	277+	280+#	
		277+	280+#	
	Pmskl4	290+	293+#	
		291+	294+#	
L$READEF	Pmskl2	820+	824+#	
		820+	824+#	
	Pmskl3	820+	824+#	
		820+	824+#	
	Pmskl4	833+	837+#	
		834+	838+#	
L$REPP	Pmskl2	1196+		
		1196+		
	Pmskl3	1196+		
		1196+		
	Pmskl4	1209+		
		1210+		
L$REV	Pmskl2	1140+	1142+	
		1140+	1142+	
	Pmskl3	1140+	1142+	
		1140+	1142+	
	Pmskl4	1153+	1155+	
		1154+	1156+	
L$RFLAGS	Pmskl2	270+	273+#	
		270+	273+#	
	Pmskl3	270+	273+#	
		270+	273+#	
	Pmskl4	283+	286+#	
		284+	287+#	
L$RPT	Pmskl2	610+*	1117+	1199+
		610+*	1117+	1199+
	Pmskl3	610+*	1117+	1199+
		610+*	1117+	1199+
	Pmskl4	623+*	1130+	1212+
		624+*	1131+	1213+
L$SETPRI	Pmskl2	677+	681+#	
		677+	681+#	
	Pmskl3	677+	681+#	
		677+	681+#	
	Pmskl4	690+	694+#	
		691+	695+#	
L$SETVEC	Pmskl2	669+	673+#	
		669+	673+#	
	Pmskl3	669+	673+#	
		669+	673+#	
	Pmskl4	682+	686+#	
		683+	687+#	
L$SFTLN	Pmskl2	1310+	1311+	1312+
		1310+	1311+	1312+
	Pmskl3	1310+	1311+	1312+
		1310+	1311+	1312+

	Pmskl4	1323+	1324+	1325+	
		1324+	1325+	1326+	
L$SOFT	Pmskl2	1117+	1155+	1312+#	
		1117+	1155+	1312+#	
	Pmskl3	1117+	1155+	1312+#	
		1117+	1155+	1312+#	
	Pmskl4	1130+	1168+	1325+#	
		1131+	1169+	1326+#	
L$SPC	Pmskl2	1192+			
		1192+			
	Pmskl3	1192+			
		1192+			
	Pmskl4	1205+			
		1206+			
L$SPCP	Pmskl2	1152+			
		1152+			
	Pmskl3	1152+			
		1152+			
	Pmskl4	1165+			
		1166+			
L$SPTP	Pmskl2	1159+			
		1159+			
	Pmskl3	1159+			
		1159+			
	Pmskl4	1172+			
		1173+			
L$STA	Pmskl2	1166+			
		1166+			
	Pmskl3	1166+			
		1166+			
	Pmskl4	1179+			
		1180+			
L$SW	Pmskl2	1068+	1129+#	1162+	
		1068+	1129+#	1162+	
	Pmskl3	1068+	1129+#	1162+	
		1068+	1129+#	1162+	
	Pmskl4	1081+	1142+#	1175+	
		1082+	1143+#	1176+	
L$SWLEN	Pmskl2	1064+	1065+	1123+	1129+
		1064+	1065+	1123+	1129+
	Pmskl3	1064+	1065+	1123+	1129+
		1064+	1065+	1123+	1129+
	Pmskl4	1077+	1078+	1136+	1142+
		1078+	1079+	1137+	1143+
L$TEST	Pmskl2	1234+			
		1234+			
	Pmskl3	1234+			
		1234+			
	Pmskl4	1247+			
		1248+			
L$TIML	Pmskl2	1148+			
		1148+			
	Pmskl3	1148+			
		1148+			
	Pmskl4	1161+			
		1162+			
L$UNIT	Pmskl2	1143+			
		1143+			
	Pmskl3	1143+			
		1143+			

	Pmskl4	1156+	1950	2391	2410	2448	2461	2497	2510	5376	5384#	5396
		1157+	1951	2392	2411	2449	2462	2498	2511	5377	5385#	5 SEQ 0287
		5413	5445	5533	5635							
		5414	5446	5534	5636							
LAB	Pmskl2	1264+	1265+	1269+	1270+	1274+	1275+	1279+	1281+	1284+	1285+	1291+
		1264+	1265+	1269+	1270+	1274+	1275+	1279+	1281+	1284+	1285+	1291+
		1296+										
		1296+										
	Pmskl3	1264+	1265+	1269+	1270+	1274+	1275+	1279+	1281+	1284+	1285+	1291+
		1264+	1265+	1269+	1270+	1274+	1275+	1279+	1281+	1284+	1285+	1291+
		1296+										
		1296+										
	Pmskl4	1277+	1278+	1232+	1283+	1287+	1288+	1292+	1294+	1297+	1298+	1304+
		1278+	1279+	1283+	1284+	1288+	1289+	1293+	1295+	1298+	1299+	1305+
		1309+										
		1310+										
LASTAD	Pmskl2	381+#										
		381+#										
	Pmskl3	381+#										
		381+#										
	Pmskl4	394+#										
		395+#										
LAU	Pmskl2	352+*	572+									
		352+*	572+									
	Pmskl3	352+*	572+									
		352+*	572+									
	Pmskl4	365+*	585+									
		366+*	586+									
LAUTO	Pmskl2	358+*	582+									
		358+*	582+									
	Pmskl3	358+*	582+									
		358+*	582+									
	Pmskl4	371+*	595+									
		372+*	596+									
LBT	Pmskl4	1763#										
		1764#										
LCLEAN	Pmskl2	364+*	592+									
		364+*	592+									
	Pmskl3	364+*	592+									
		364+*	592+									
	Pmskl4	377+*	605+									
		378+*	606+									
LDU	Pmskl2	370+*	602+									
		370+*	602+									
	Pmskl3	370+*	602+									
		370+*	602+									
	Pmskl4	383+*	615+									
		384+*	616+									
LF	Pmskl4	2063#										
		2064#										
LFS	Pmskl4	2064#	5442									
		2065#	5443									
LINIT	Pmskl2	346+*	562+									
		346+*	562+									
	Pmskl3	346+*	562+									
		346+*	562+									
	Pmskl4	359+*	575+									
		360+*	576+									
LOADTS	Pmskl2	969+	974+#	983+								
		969+	974+#	983+								

	Pmskl3	969+	974+#	983+								
		969+	974+#	983+								
	Pmskl4	982+	987+#	996+								
		983+	988+#	997+								
LOC	Pmskl2	269+	270+#									
		269+	270+#									
	Pmskl3	269+	270+#									
		269+	270+#									
	Pmskl4	282+	283+#									
		283+	284+#									
LOE	Pmskl2	234+#										
		234+#										
	Pmskl3	234+#										
		234+#										
	Pmskl4	247+#										
		248+#										
LOT	Pmskl2	223+#										
		223+#										
	Pmskl3	223+#										
		223+#										
	Pmskl4	236+#										
		237+#										
LOW	Pmskl2	873+	881+	886+	897+	1326+	1341+	1363+	1378+	1432+	1440+	1442+
		873+	881+	886+	897+	1326+	1341+	1363+	1378+	1432+	1440+	1442+
	Pmskl3	873+	881+	886+	897+	1326+	1341+	1363+	1378+	1432+	1440+	1442+
		873+	881+	886+	897+	1326+	1341+	1363+	1378+	1432+	1440+	1442+
	Pmskl4	886+	894+	899+	910+	1339+	1354+	1376+	1391+	1445+	1453+	1455+
		887+	895+	900+	911+	1340+	1355+	1377+	1392+	1446+	1454+	1456+
LRPT	Pmskl2	376+*	612+									
		376+*	612+									
	Pmskl3	376+*	612+									
		376+*	612+									
	Pmskl4	389+*	625+									
		390+*	626+									
LST_PAT	Pmskl4	2742	2779	4893	4910#	4914#						
		2743	2780	4894	4911#	4915#						
LST_REM_ENTRY	Pmskl4	3255	3289	3293	3300	3304	3307	3320	3331	3335	3342	3346
		3256	3290	3294	3301	3305	3308	3321	3332	3336	3343	3347
		3349	3362	3383	3387	3389	3402	3416	3420	3422	3470	3511
		3350	3363	3384	3388	3390	3403	3417	3421	3423	3471	3512
		3514	3516	3559	3582	3585	3587	3666	3675#	3676#	3679	3688
		3515	3517	3560	3583	3586	3588	3667	3676#	3677#	3680	3689
		3690	3699	3700#	3707	3708#						
		3691	3700	3701#	3708	3709#						
LST_TSTED_ARR	Pmskl4	2012	5492#	5502#	5513#	5567						
		2013	5493#	5503#	5514#	5568						
LUN	Pmskl4	5374	5405#	5407								
		5375	5406#	5408								
LUN_MISS_MSG	Pmskl4	2036#	5410									
		2037#	5411									
L_1_0_DATA	Pmskl4	2646*	4602	4605								
		2647*	4603	4606								
L_CHIP_TBL	Pmskl4	2664*	4625	4660	4759	4791	4822					
		2665*	4626	4661	4760	4792	4823					
L_ERROR_MAP	Pmskl4	3004*	4915									
		3005*	4916									
L_FAILING_CHIP	Pmskl4	2742*	4914									
		2743*	4915									
L_RANDOM_DATA	Pmskl4	2616*	4609									
		2617*	4610									

	Pmskl3	283+	284+#									
		283+	284+#									
	Pmskl4	296+	297+#									
		297+	298+#									
MEMORY	Pmskl2	283+#										
		283+#										
	Pmskl3	283+#										
		283+#										
	Pmskl4	296+#										
		297+#										
MEM_ERR_MSG	Pmskl4	2050#	5340									
		2051#	5341									
ML11A	Pmskl4	1635#	5460									
		1636#	5461									
MLAS	Pmskl4	1565#	2180									
		1566#	2181									
MLBA	Pmskl4	1560#	2175	2242	2269	2379	2440	2489				
		1561#	2176	2243	2270	2380	2441	2490				
MLBAE	Pmskl4	1579#										
		1580#										
MLCS1	Pmskl4	1558#	2128	2173	2243	2270	2380	2382	2401	2441	2443	2456
		1559#	2129	2174	2244	2271	2381	2383	2402	2442	2444	2457
		2490	2492	2505								
		2491	2493	2506								
MLCS2	Pmskl4	1562#	2122	2123	2177	2239	2266	2376	2386	2405	2437	2486
		1563#	2123	2124	2178	2240	2267	2377	2387	2406	2438	2487
		4638	5491									
		4639	5492									
MLCS3	Pmskl4	1580#										
		1581#										
MLD1	Pmskl4	1574#	2305	2342								
		1575#	2306	2343								
MLD2	Pmskl4	1575#	2306	2343								
		1576#	2307	2344								
MLDA	Pmskl4	1561#	2176	2241	2268	2378	2439	2488				
		1562#	2177	2242	2269	2379	2440	2489				
MLDB	Pmskl4	1568#										
		1569#										
MLDS	Pmskl4	1563#	2178	2399	2454	2503						
		1564#	2179	2400	2455	2504						
MLDT	Pmskl4	1570#	2182									
		1571#	2183									
MLE1	Pmskl4	1572#										
		1573#										
MLE2	Pmskl4	1573#	2307	2344	4756	4788	4819					
		1574#	2308	2345	4757	4789	4820					
MLEE	Pmskl4	1576#	2184	4621	4624	4630						
		1577#	2185	4622	4625	4631						
MLEL	Pmskl4	1577#	2185									
		1578#	2186									
MLER	Pmskl4	1564#	2179									
		1565#	2180									
MLLA	Pmskl4	1566#										
		1567#										
MLMR	Pmskl4	1569#	2133	2135	2137	2139	2141	2143	2145	2147	2149	2181
		1570#	2134	2136	2138	2140	2142	2144	2146	2148	2150	2182
		5161	5190	5492								
		5162	5191	5493								
MLPA	Pmskl4	1567#	4080	4091	5156	5185	5254	5272				
		1568#	4081	4092	5157	5186	5255	5273				

MLPD	Pmskl4	1578#	4082	4093	4749	4781	4812	5157	5186	5257	5275	
		1579#	4083	4094	4750	4782	4813	5158	5187	5258	5276	SEQ 0291
MLREG	Pmskl4	2112	2114#	2115#	2116							
		2113	2115#	2116#	2117							
MLSN	Pmskl4	1571#	2183									
		1572#	2184									
MLWC	Pmskl4	1559#	2174	2240	2267	2377	2438	2487				
		1560#	2175	2241	2268	2378	2439	2488				
ML_ADDR	Pmskl4	1987	2114	2116	2173	2174	2175	2176	2177	2178	2179	2180
		1988	2115	2117	2174	2175	2176	2177	2178	2179	2180	2181
		2181	2182	2183	2184	2185	2382	2386	2399	2401	2405	2443
		2182	2183	2184	2185	2186	2383	2387	2400	2402	2406	2444
		2454	2456	2492	2503	2505	3077	4082	4093	4621	4624	4630
		2455	2457	2493	2504	2506	3078	4083	4094	4622	4625	4631
		4638	4749	4756	4781	4788	4812	4819	5161	5190	5257	5275
		4639	4750	4757	4782	4789	4813	4820	5162	5191	5258	5276
		5454#	5492									
		5455#	5493									
ML_ALL	Pmskl4	1898#	2114	2173	2174	2175	2176	2177	2178	2179	2180	2181
		1899#	2115	2174	2175	2176	2177	2178	2179	2180	2181	2182
		2182	2183	2184	2185							
		2183	2184	2185	2186							
ML_DUT	Pmskl4	1992	2123	2239	2266	2376	2437	2486	5455#	5491	5560	
		1993	2124	2240	2267	2377	2438	2487	5456#	5492	5561	
ML_REG	Pmskl4	1921	1923#	1924								
		1922	1924#	1925								
MCL	Pmskl4	1762#										
		1763#										
MOST_OFTEN	Pmskl4	3470	3555#	3559	3626#							
		3471	3556#	3560	3627#							
MR_PRINT	Pmskl4	2086#	2181									
		2087#	2182									
MR_REG	Pmskl4	1803#	2133	2135								
		1804#	2134	2136								
MSGADR	Pmskl2	862+	868+	873+	881+	886+	897+	1326+	1349+	1363+	1386+	1400+
		862+	868+	873+	881+	886+	897+	1326+	1349+	1363+	1386+	1400+
		1421+										
		1421+										
	Pmskl3	862+	868+	873+	881+	886+	897+	1326+	1349+	1363+	1386+	1400+
		862+	868+	873+	881+	886+	897+	1326+	1349+	1363+	1386+	1400+
		1421+										
		1421+										
	Pmskl4	875+	881+	886+	894+	899+	910+	1339+	1362+	1376+	1399+	1413+
		876+	882+	887+	[illegible]	900+	911+	1340+	1363+	1377+	1400+	1414+
		1434+										
		1435+										
MSGPTR	Pmskl2	111+	117+	121+	127+	131+	137+	141+	147+			
		111+	117+	121+	127+	131+	137+	141+	147+			
	Pmskl3	111+	117+	121+	127+	131+	137+	141+	147+			
		111+	117+	121+	127+	131+	137+	141+	147+			
	Pmskl4	124+	130+	134+	140+	144+	150+	154+	160+			
		125+	131+	135+	141+	145+	151+	155+	161+			
MULT	Pmskl2	249+	253+									
		249+	253+									
	Pmskl3	249+	253+									
		249+	253+									
	Pmskl4	262+	266+									
		263+	267+									
MXF	Pmskl4	1749#										
		1750#										

	Pmskl3	1333+	1370+	1466+#			
		1333+	1370+	1466+#			
	Pmskl4	1346+	1383+	1479+#			
		1347+	1384+	1480+#			
NO_ODD_ADR	Pmskl2	1334+	1371+	1407+	1473+#		
		1334+	1371+	1407+	1473+#		
	Pmskl3	1334+	1371+	1407+	1473+#		
		1334+	1371+	1407+	1473+#		
	Pmskl4	1347+	1384+	1420+	1486+#		
		1348+	1385+	1421+	1487+#		
NUM	Pmskl2	399+	401+	406+			
		399+	401+	406+			
	Pmskl3	399+	401+	406+			
		399+	401+	406+			
	Pmskl4	412+	414+	419+			
		413+	415+	420+			
O	Pmskl2	907+					
		907+					
	Pmskl3	907+	1572	1573			
		907+	1572	1573			
	Pmskl4	920+	1917	1923	2108	2114	2116
		921+	1918	1924	2109	2115	2117
O$APTS	Pmskl2	942+#	1011+	1036+			
		942+#	1011+	1036+			
	Pmskl3	942+#	1011+	1036+			
		942+#	1011+	1036+			
	Pmskl4	955+#	1024+	1049+			
		956+#	1025+	1050+			
O$AU	Pmskl2	943+#	1012+	1040+	1206+		
		943+#	1012+	1040+	1206+		
	Pmskl3	943+#	1012+	1040+	1206+		
		943+#	1012+	1040+	1206+		
	Pmskl4	956+#	1025+	1053+	1219+		
		957+#	1026+	1054+	1220+		
O$BGNRPT	Pmskl2	940+#	1009+	1028+	1197+		
		940+#	1009+	1028+	1197+		
	Pmskl3	940+#	1009+	1028+	1197+		
		940+#	1009+	1028+	1197+		
	Pmskl4	953+#	1022+	1041+	1210+		
		954+#	1023+	1042+	1211+		
O$BGNSFT	Pmskl2	939+#	1008+	1024+	1153+		
		939+#	1008+	1024+	1153+		
	Pmskl3	939+#	1008+	1024+	1153+		
		939+#	1008+	1024+	1153+		
	Pmskl4	952+#	1021+	1037+	1166+		
		953+#	1022+	1038+	1167+		
O$DU	Pmskl2	944+#	1013+	1044+	1211+		
		944+#	1013+	1044+	1211+		
	Pmskl3	944+#	1013+	1044+	1211+		
		944+#	1013+	1044+	1211+		
	Pmskl4	957+#	1026+	1057+	1224+		
		958+#	1027+	1058+	1225+		
O$ERRTBL	Pmskl2	945+#	1014+	1048+	1222+		
		945+#	1014+	1048+	1222+		
	Pmskl3	945+#	1014+	1048+	1222+		
		945+#	1014+	1048+	1222+		
	Pmskl4	958+#	1027+	1061+	1235+		
		959+#	1028+	1062+	1236+		
O$GNSW	Pmskl2	941+#	1010+	1032+	1160+		
		941+#	1010+	1032+	1160+		

	Pmskl3	941+#	1010+	1032+	1160+							
		941+#	1010+	1032+	1160+							
	Pmskl4	954+#	1023+	1045+	1173+							
		955+#	1024+	1046+	1174+							
OSPOINTER	Pmskl2	947+#	1016+	1020+	1025+	1029+	1033+	1037+	1041+	1045+	1049+	1053+
		947+#	1016+	1020+	1025+	1029+	1033+	1037+	1041+	1045+	1049+	1053+
		1112+										
		1112+										
	Pmskl3	947+#	1016+	1020+	1025+	1029+	1033+	1037+	1041+	1045+	1049+	1053+
		947+#	1016+	1020+	1025+	1029+	1033+	1037+	1041+	1045+	1049+	1053+
		1112+										
		1112+										
	Pmskl4	960+#	1029+	1033+	1038+	1042+	1046+	1050+	1054+	1058+	1062+	1066+
		961+#	1030+	1034+	1039+	1043+	1047+	1051+	1055+	1059+	1063+	1067+
		1125+										
		1126+										
OSSETUP	Pmskl2	946+#	1015+	1052+	1144+							
		946+#	1015+	1052+	1144+							
	Pmskl3	946+#	1015+	1052+	1144+							
		946+#	1015+	1052+	1144+							
	Pmskl4	959+#	1028+	1065+	1157+							
		960+#	1029+	1066+	1158+							
OFFSET	Pmskl2	1455+	1457+	1461+	1463+	1473+	1474+	1481+	1482+			
		1455+	1457+	1461+	1463+	1473+	1474+	1481+	1482+			
	Pmskl3	1455+	1457+	1461+	1463+	1473+	1474+	1481+	1482+			
		1455+	1457+	1461+	1463+	1473+	1474+	1481+	1482+			
	Pmskl4	1468+	1470+	1474+	1476+	1486+	1487+	1494+	1495+	2202	2207#	2213#
		1469+	1471+	1475+	1477+	1487+	1488+	1495+	1496+	2203	2208#	2214#
		2214#	3038	3046#	3055	3062	3064#	3089	4584	4592#	4610#	4670
		2215#	3039	3047#	3056	3063	3065#	3090	4585	4593#	4611#	4671
		4700	4703#	4731#	4767	4798	4829					
		4701	4704#	4732#	4768	4799	4830					
OFFSET_SIZE	Pmskl2	1335+	1372+	1408+	1481+#							
		1335+	1372+	1408+	1481+#							
	Pmskl3	1335+	1372+	1408+	1481+#							
		1335+	1372+	1408+	1481+#							
	Pmskl4	1348+	1385+	1421+	1494+#							
		1349+	1386+	1422+	1495+#							
OLDPC	Pmskl2	530+										
		530+										
	Pmskl3	530+										
		530+										
	Pmskl4	543+										
		544+										
OLDPSW	Pmskl2	530+	547+#									
		530+	547+#									
	Pmskl3	530+	547+#									
		530+	547+#									
	Pmskl4	543+	560+#									
		544+	561+#									
ONE	Pmskl4	1552#	2122	2143	2309	3077	3078	3083	3133	3136	3215	3289
		1553#	2123	2144	2310	3078	3079	3084	3134	3137	3216	3290
		3331	3383	3416	3511	3582	3892	3928	3937	3941	3949	3953
		3332	3384	3417	3512	3583	3893	3929	3938	3942	3950	3954
		4086	4594	4655	4715	5175	5267	5376				
		4087	4595	4656	4716	5176	5268	5377				
ONES	Pmskl4	1551#	2801	4605	4726							
		1552#	2802	4606	4727							
ONE_MSG	Pmskl4	2059#	2390	2409	2447	2460	2496	2509	5165	5166	5194	5195
		2060#	2391	2410	2448	2461	2497	2510	5166	5167	5195	5196

		3619	3622#	3627
		3620	3623#	3628
PRI	Pmskl2	230+#	1111+	1183+
		230+#	1111+	1183+
	Pmskl3	230+#	1111+	1183+
		230+#	1111+	1183+
	Pmskl4	243+#	1124+	1196+
		244+#	1125+	1197+
PRI00	Pmskl2	218+#	1510	
		218+#	1510	
	Pmskl3	218+#		
		218+#		
	Pmskl4	231+#		
		232+#		
PRI01	Pmskl2	217+#		
		217+#		
	Pmskl3	217+#		
		217+#		
	Pmskl4	230+#		
		231+#		
PRI02	Pmskl2	216+#		
		216+#		
	Pmskl3	216+#		
		216+#		
	Pmskl4	229+#		
		230+#		
PRI03	Pmskl2	215+#		
		215+#		
	Pmskl3	215+#		
		215+#		
	Pmskl4	228+#		
		229+#		
PRI04	Pmskl2	214+#		
		214+#		
	Pmskl3	214+#		
		214+#		
	Pmskl4	227+#		
		228+#		
PRI05	Pmskl2	213+#		
		213+#		
	Pmskl3	213+#		
		213+#		
	Pmskl4	226+#		
		227+#		
PRI06	Pmskl2	212+#		
		212+#		
	Pmskl3	212+#		
		212+#		
	Pmskl4	225+#		
		226+#		
PRI07	Pmskl2	211+#		
		211+#		
	Pmskl3	211+#		
		211+#		
	Pmskl4	224+#		
		225+#		
PRINTB	Pmskl2	692+#		
		692+#		
	Pmskl3	692+#		
		692+#		

	Pmskl4	705+#	2171	2172	2173	2174	2175	2176	2177	2178	2179	2180
		706+#	2172	2173	2174	2175	2176	2177	2178	2179	2180	2 SEQ 0298
		2181	2182	2183	2184	2185	2390	2409	2447	2460	2496	2509
		2182	2183	2184	2185	2186	2391	2410	2448	2461	2497	2510
		4306	4352	4393	4434	4547	4554	4928	5165	5166	5194	5195
		4307	4353	4394	4435	4548	4555	4929	5166	5167	5195	5196
		5261	5279	5332	5341	5380	5381	5395	5411	5412	5442	5443
		5262	5280	5333	5342	5381	5382	5396	5412	5413	5443	5444
		5444	5530	5531	5532	5559	5560	5585	5589	5604	5611	5633
		5445	5531	5532	5533	5560	5561	5586	5590	5605	5612	5634
		5634										
		5635										
PRINTF	Pmskl2	713+#										
		713+#										
	Pmskl3	713+#										
		713+#										
	Pmskl4	726+#										
		727+#										
PRINTS	Pmskl2	706+#										
		706+#										
	Pmskl3	706+#										
		706+#										
	Pmskl4	719+#										
		720+#										
PRINTX	Pmskl2	699+#										
		699+#										
	Pmskl3	699+#										
		699+#										
	Pmskl4	712+#										
		713+#										
PRIO	Pmskl2	790+	791+#									
		790+	791+#									
	Pmskl3	790+	791+#									
		790+	791+#									
	Pmskl4	803+	804+#									
		804+	805+#									
PRIOR	Pmskl2	668+	669+	676+	677+							
		668+	669+	676+	677+							
	Pmskl3	668+	669+	676+	677+							
		668+	669+	676+	677+							
	Pmskl4	681+	682+	689+	690+							
		682+	683+	690+	691+							
PRIORITY	Pmskl2	541+	542+	547+								
		541+	542+	547+								
	Pmskl3	541+	542+	547+								
		541+	542+	547+								
	Pmskl4	554+	555+	560+								
		555+	556+	561+								
PROM_ADRS	Pmskl4	4060	4069#	4070#	4074#	4075#	4079#	4080	4086#	4090#	4091	5130
		4061	4070#	4071#	4075#	4076#	4080#	4081	4087#	4091#	4092	5131
		5136#	5137#	5146#	5147#	5155#	5156	5175#	5184#	5185	5232	5243#
		5137#	5138#	5147#	5148#	5156#	5157	5176#	5185#	5186	5233	5244#
		5244#	5248#	5249#	5253#	5254	5267#	5271#	5272			
		5245#	5249#	5250#	5254#	5255	5268#	5272#	5273			
PROM_DIS	Pmskl4	2144#										
		2145#										
PROM_RW	Pmskl4	2140#										
		2141#										
P^$NUM	Pmskl2	405+#	406+	412+	423+	429+	435+	438+	444+			
		405+#	406+	412+	423+	429+	435+	438+	444+			

	Pmskl3	405+#	406+	412+	423+	429+	435+	438+	444+			
		405+#	406+	412+	423+	429+	435+	438+	444+			
	Pmskl4	418+#	419+	425+	436+	442+	448+	451+	457+			
		419+#	420+	426+	437+	443+	449+	452+	458+			
PTABS	Pmskl2	424+	427+#									
		424+	427+#									
	Pmskl3	424+	427+#									
		424+	427+#									
	Pmskl4	437+	440+#									
		438+	441+#									
PTBL_PTR	Pmskl4	5373	5407	5423	5424	5425	5426	5427	5428	5429	5430	5431
		5374	5408	5424	5425	5426	5427	5428	5429	5430	5431	5432
PTR	Pmskl4	3814	3861#	3876#	3921	4002#	4003					
		3815	3862#	3877#	3922	4003#	4004					
PURGE	Pmskl4	3304	3305	3346	3347							
		3305	3306	3347	3348							
PUR_LOC	Pmskl4	3293	3296	3300	3304	3335	3338	3342	3346			
		3294	3297	3301	3305	3336	3339	3343	3347			
P_DIS	Pmskl4	1797#	2145									
		1798#	2146									
P_RW	Pmskl4	1796#	2141									
		1797#	2142									
Q	Pmskl2	157+	158+	159+	160+	161+	162+					
		157+	158+	159+	160+	161+	162+					
	Pmskl3	157+	158+	159+	160+	161+	162+					
		157+	158+	159+	160+	161+	162+					
	Pmskl4	170+	171+	172+	173+	174+	175+					
		171+	172+	173+	174+	175+	176+					
QD_0_SUM	Pmskl4	4204	4300#	4302	4308							
		4205	4301#	4303	4309							
QD_1_SUM	Pmskl4	4205	4295#	4334#	4348	4354						
		4206	4296#	4335#	4349	4355						
QD_2_SUM	Pmskl4	4206	4296#	4317#	4389	4395						
		4207	4297#	4318#	4390	4396						
QD_3_SUM	Pmskl4	4207	4297#	4363#	4419#	4430	4436					
		4208	4298#	4364#	4420#	4431	4437					
QD_LOOP	Pmskl4	3840										
		3841										
QD_NUM_SEARCH	Pmskl4	3817	3830#	3835#	3851							
		3818	3831#	3836#	3852							
QD_OFFSET	Pmskl4	3818	3838#	3842#	3851							
		3819	3839#	3843#	3852							
QD_PAIR_LOOP	Pmskl4	3816	3829#	3834#	3840	3924						
		3817	3830#	3835#	3841	3925						
QD_SEARCH	Pmskl4	3851	3855	3932								
		3852	3856	3933								
QUANTITY	Pmskl4	3255	3300	3304	3305	3307	3320	3342	3346	3347	3349	
		3256	3301	3305	3306	3308	3321	3343	3347	3348	3350	
RADIX	Pmskl2	873+	877+	886+	889+	893+	1326+	1333+	1337+	1363+	1370+	1374+
		873+	877+	886+	889+	893+	1326+	1333+	1337+	1363+	1370+	1374+
		1466+	1467+									
		1466+	1467+									
	Pmskl3	873+	877+	886+	889+	893+	1326+	1333+	1337+	1363+	1370+	1374+
		873+	877+	886+	889+	893+	1326+	1333+	1337+	1363+	1370+	1374+
		1466+	1467+									
		1466+	1467+									
	Pmskl4	886+	890+	899+	902+	906+	1339+	1346+	1350+	1376+	1383+	1387+
		887+	891+	900+	903+	907+	1340+	1347+	1351+	1377+	1384+	1388+
		1479+	1480+									
		1480+	1481+									

ROW_NUM	Pmskl4	3073	3077	3081	3082	3171	3186#	3193#	3208	3213	3246#	3844
		3074	3078	3082	3083	3172	3187#	3194#	3209	3214	3247#	3 SEQ 0302
		3846	3847	3913	3916	3970	3978	3986	3994	4077	4079	4082
		3847	3848	3914	3917	3971	3979	3987	3995	4078	4080	4083
		4240	4243	4247	4269	4272	4276	4310	4312	4315	4356	4358
		4241	4244	4248	4270	4273	4277	4311	4313	4316	4357	4359
		4361	4397	4399	4402	4438	4440	4443	5149	5152	5155	5157
		4362	4398	4400	4403	4439	4441	4444	5150	5153	5156	5158
		5251	5253	5257								
		5252	5254	5258								
ROW_PURGE	Pmskl4	3320*	3699									
		3321*	3700									
ROW_SEL	Pmskl4	3320	3338									
		3321	3339									
ROW_SORT	Pmskl4	3402*	3585									
		3403*	3586									
ROW_SUM	Pmskl4	3671	3690	3699	3700							
		3672	3691	3700	3701							
RO_CL	Pmskl4	1654#	2569	2570	3305	3308	3347	3350	3394	3395	3396	3427
		1655#	2570	2571	3306	3309	3348	3351	3395	3396	3397	3428
		3428	3429									
		3429	3430									
RPT	Pmskl2	1027+										
		1027+										
	Pmskl3	1027+										
		1027+										
	Pmskl4	1040+										
		1041+										
R_C	Pmskl4	1710#	3131	3696	3704	3761	3907					
		1711#	3132	3697	3705	3762	3908					
R_C_NO	Pmskl4	1709#	3133	3136	3697	3705	3762	3913	3916			
		1710#	3134	3137	3698	3706	3763	3914	3917			
S	Pmskl4	1917	1925									
		1918	1926									
SC	Pmskl4	1720#	2382	2401	2443	2456	2492	2505				
		1721#	2383	2402	2444	2457	2493	2506				
SEARCH	Pmskl4	3461	3464									
		3462	3465									
SEC_CNT	Pmskl4	2299	2346	2639	2641	2660	2661	4614	4617	4643	4743	4775
		2300	2347	2640	2642	2661	2662	4615	4618	4644	4744	4776
		4806										
		4807										
SEC_NUM	Pmskl4	3069	3083	3206	3210	3215	3222	3889	3892	3909	3965	3970
		3070	3084	3207	3211	3216	3223	3890	3893	3910	3966	3971
		3978	3986	3994								
		3979	3987	3995								
SEED1	Pmskl4	1943	2204	4589#	4704#	4904#						
		1944	2205	4590#	4705#	4905#						
SEED2	Pmskl4	1944	2205	4590#	4705#	4905#						
		1945	2206	4591#	4706#	4906#						
SEED3	Pmskl4	1945	2206	4591#	4706#	4906#						
		1946	2207	4592#	4707#	4907#						
SFL_BNK	Pmskl4	4072	4075	4082	4093	4224	5139	5143	5147	5152	5157	5180
		4073	4076	4083	4094	4225	5140	5144	5148	5153	5158	5181
		5186	5246	5249	5257	5261	5262	5275	5279	5280	5315	
		5187	5247	5250	5258	5262	5263	5276	5280	5281	5316	
SERIAL_REG	Pmskl4	1812#										
		1813#										
SETPRI	Pmskl2	676+#										
		676+#										

		1394+	1399+	1417+	1422+	1424+	1426+	1429+	1434+	1447+		
		1395+	1400+	1418+	1423+	1425+	1427+	1430+	1435+	1448+		
T$EXCP	Pmskl2	1245+#	1329+	1351+	1366+	1388+	1403+	1439+	1446+			
		1245+#	1329+	1351+	1366+	1388+	1403+	1439+	1446+			
	Pmskl3	1245+#	1329+	1351+	1366+	1388+	1403+	1439+	1446+			
		1245+#	1329+	1351+	1366+	1388+	1403+	1439+	1446+			
	Pmskl4	1258+#	1342+	1364+	1379+	1401+	1416+	1452+	1459+			
		1259+#	1343+	1365+	1380+	1402+	1417+	1453+	1460+			
T$FREE	Pmskl2	391+	393+	416+								
		391+	393+	416+								
	Pmskl3	391+	393+	416+								
		391+	393+	416+								
	Pmskl4	404+	406+	429+								
		405+	407+	430+								
T$HILIM	Pmskl2	1243+#	1349+	1386+	1447+	1449+						
		1243+#	1349+	1386+	1447+	1449+						
	Pmskl3	1243+#	1349+	1386+	1447+	1449+						
		1243+#	1349+	1386+	1447+	1449+						
	Pmskl4	1256+#	1362+	1399+	1460+	1462+						
		1257+#	1363+	1400+	1461+	1463+						
T$LOLIM	Pmskl2	1244+#	1349+	1386+	1440+	1442+						
		1244+#	1349+	1386+	1440+	1442+						
	Pmskl3	1244+#	1349+	1386+	1440+	1442+						
		1244+#	1349+	1386+	1440+	1442+						
	Pmskl4	1257+#	1362+	1399+	1453+	1455+						
		1258+#	1363+	1400+	1454+	1456+						
T$PTHV	Pmskl2	401+#	1117+	1146+								
		401+#	1117+	1146+								
	Pmskl3	401+#	1117+	1146+								
		401+#	1117+	1146+								
	Pmskl4	414+#	1130+	1159+								
		415+#	1131+	1160+								
T$RAP	Pmskl2	241+	246+	563+	573+	583+	593+	603+	613+	629+	760+	858+#
		241+	246+	563+	573+	583+	593+	603+	613+	629+	760+	858+#
	Pmskl3	241+	246+	563+	573+	583+	593+	603+	613+	629+	760+	858+#
		241+	246+	563+	573+	583+	593+	603+	613+	629+	760+	858+#
	Pmskl4	254+	259+	576+	586+	596+	606+	616+	626+	642+	773+	871+#
		255+	260+	577+	587+	597+	607+	617+	627+	643+	774+	872+#
T$RAPRO	Pmskl2	305+	311+	318+	326+	491+	511+	647+	661+	665+#		
		305+	311+	318+	326+	491+	511+	647+	661+	665+#		
	Pmskl3	305+	311+	318+	326+	491+	511+	647+	661+	665+#		
		305+	311+	318+	326+	491+	511+	647+	661+	665+#		
	Pmskl4	318+	324+	331+	339+	504+	524+	660+	674+	678+#		
		319+	325+	332+	340+	505+	525+	661+	675+	679+#		
T$TEMP	Pmskl2	340+#	866+	867+	877+	878+	879+	880+	893+	894+	895+	896+
		340+#	866+	867+	877+	878+	879+	880+	893+	894+	895+	896+
		1337+	1338+	1339+	1340+	1374+	1375+	1376+	1377+	1410+	1411+	1412+
		1337+	1338+	1339+	1340+	1374+	1375+	1376+	1377+	1410+	1411+	1412+
		1413+	1474+	1475+								
		1413+	1474+	1475+								
	Pmskl3	340+#	866+	867+	877+	878+	879+	880+	893+	894+	895+	896+
		340+#	866+	867+	877+	878+	879+	880+	893+	894+	895+	896+
		1337+	1338+	1339+	1340+	1374+	1375+	1376+	1377+	1410+	1411+	1412+
		1337+	1338+	1339+	1340+	1374+	1375+	1376+	1377+	1410+	1411+	1412+
		1413+	1474+	1475+								
		1413+	1474+	1475+								
	Pmskl4	353+#	879+	880+	890+	891+	892+	893+	906+	907+	908+	909+
		354+#	880+	881+	891+	892+	893+	894+	907+	908+	909+	910+
		1350+	1351+	1352+	1353+	1387+	1388+	1389+	1390+	1423+	1424+	1425+
		1351+	1352+	1353+	1354+	1388+	1389+	1390+	1391+	1424+	1425+	1426+

	Pmskl4	110+#	1278+								
		111+#	1279+								
X$FALSE	Pmskl2	99+#	1270+								
		99+#	1270+								
	Pmskl3	99+#	1270+								
		99+#	1270+								
	Pmskl4	112+#	1283+								
		113+#	1284+								
X$OFFSET	Pmskl2	86+#	1285+								
		86+#	1285+								
	Pmskl3	86+#	1285+								
		86+#	1285+								
	Pmskl4	99+#	1298+								
		100+#	1299+								
X$TRUE	Pmskl2	98+#	1275+								
		98+#	1275+								
	Pmskl3	98+#	1275+								
		98+#	1275+								
	Pmskl4	111+#	1288+								
		112+#	1289+								
X$XFER_SKIP	Pmskl2	1242+#	1296+								
		1242+#	1296+								
	Pmskl3	1242+#	1296+								
		1242+#	1296+								
	Pmskl4	1255+#	1309+								
		1256+#	1310+								
XFER	Pmskl2	1264+#									
		1264+#									
	Pmskl3	1264+#									
		1264+#									
	Pmskl4	1277+#									
		1278+#									
XFERF	Pmskl2	1269+#									
		1269+#									
	Pmskl3	1269+#	1556								
		1269+#	1556								
	Pmskl4	1282+#									
		1283+#									
XFERT	Pmskl2	1274+#									
		1274+#									
	Pmskl3	1274+#	1542	1547							
		1274+#	1542	1547							
	Pmskl4	1287+#									
		1288+#									
X_MSG	Pmskl4	2055#									
		2056#									
X_PRINT	Pmskl4	2093#									
		2094#									
X_TMP_BLST_TBL	Pmskl4	3094*	4538								
		3095*	4539								
X_TO_REM_TBL	Pmskl4	3140*	3676								
		3141*	3677								
YES	Pmskl2	928+									
		928+									
	Pmskl3	928+	1572	1583	1610						
		928+	1572	1583	1610						
	Pmskl4	941+									
		942+									
ZERO	Pmskl4	1554#	2295	2636	3064	3067	3077	3078	3083	3133	3137	3179
		1555#	2296	2637	3065	3068	3078	3079	3084	3134	3138	3180

L 8

		3192	3204	3210	3215	3464	3508	3579	3675	3679	3860	3875
		3193	3205	3211	3216	3465	3509	3580	3676	3680	3861	3 SEQ 0310
		3892	3928	3937	3941	3949	3953	4074	4126	4235	4236	4237
		3893	3929	3938	3942	3950	3954	4075	4127	4236	4237	4238
		4238	4319	4323	4336	4365	4369	4381	4405	4409	4422	4446
		4239	4320	4324	4337	4366	4370	4382	4406	4410	4423	4447
		4450	4462	4703	4704	4717	4895	4897	4904	4910	4935	4974
		4451	4463	4704	4705	4718	4896	4898	4905	4911	4936	4975
		4978	4979	4996	5002	5019	5025	5042	5048	5061	5067	5078
		4979	4980	4997	5003	5020	5026	5043	5049	5062	5068	5079
		5084	5146	5152	5180	5248	5324	5405	5407	5595	5609	
		5085	5147	5153	5181	5249	5325	5406	5408	5596	5610	
ZEROES	Pmskl4	1553#	2526	2540	2554	2569	2570	2585	2599	2613	2790	3764
		1554#	2527	2541	2555	2570	2571	2586	2600	2614	2791	3765
		4069	4589	4592	4596	4602	4642	4723	4918	5136	5243	5453
		4070	4590	4593	4597	4603	4643	4724	4919	5137	5244	5454
		5570	5571	5572								
		5571	5572	5573								

PSECT Storage Map

Name	Loc	Length	/Start	Length	File	Module	Ident
	002000	000000					
$CODE$	002000	032224					
			002000	000200	PMSKL2	BSKEL2	REV A
			002200	000274	PMSKL3	BSKEL3	REV A
			002474	000046	B16ROT	B16ROT	2.6
			002542	000142	B16PG2	B16PG2	2.3
			002704	000130	B16PG4	B16PG4	2.4
			003034	000316	B16MUL	B16MUL	2.7
			003352	030652	PMSKL4	BSKEL4	REV A
OWN	034224	037036					
$PLIT$	073262	004430					
			073262	000442	PMSKL3	BSKEL3	REV A
			073724	003766	PMSKL4	BSKEL4	REV A
XXX	077712	000324					
			077712	000076	RANDOM	RANDOM	
			100010	000106	B16SAV	B16SAV	2.3
			100116	000120	B16PN1	B16PN1	
XYZ	100236	000010					
			100236	000010	LSTAD	LSTAD	NONE
. ABS.	100246	000000					

Map of Global Symbols

Name	Value	PSECT	Module	ABS Value	
$SAVE2	000000	XXX	B16SAV	100010	$CODE$
$SAVE3	000014	XXX	B16SAV	100024	$CODE$
$SAVE4	000032	XXX	B16SAV	100042	$CODE$
$SAVE5	000052	XXX	B16SAV	100062	$CODE$
BL$DIV	000224	$CODE$	B16MUL	003260	$CODE$
BL$GT1	000000	XXX	B16PN1	100116	$CODE$
BL$GT2	000000	$CODE$	B16PG2	002542	$CODE$
BL$MOD	000236	$CODE$	B16MUL	003272	$CODE$
BL$PU2	000000	$CODE$	B16PG4	002704	$CODE$
BL$SHF	000250	$CODE$	B16MUL	003304	XXX ,$CODE$
L$AU	000264	$CODE$	BSKEL3	002464	$CODE$
L$AUTO	000240	$CODE$	BSKEL3	002440	$CODE$
L$CLEA	030642	$CODE$	BSKEL4	034214	$CODE$
L$DESC	000036	$CODE$	BSKEL3	002236	$CODE$
L$DLY	000116	$CODE$	BSKEL2	002116	$CODE$
L$DU	000252	$CODE$	BSKEL3	002452	$CODE$
L$DVTY	000000	$CODE$	BSKEL3	002200	$CODE$
L$HARD	000070	$CODE$	BSKEL3	002270	$CODE$
L$INIT	027776	$CODE$	BSKEL4	033350	$CODE$
L$LAST	000004	XYZ	LSTAD	100242	$CODE$
L$RPT	000226	$CODE$	BSKEL3	002426	$CODE$
L$SOFT	000214	$CODE$	BSKEL3	002414	$CODE$
L$UNIT	000012	$CODE$	BSKEL2	002012	$CODE$
RANDAT	000074	XXX	RANDOM	100006	$CODE$
RANGEN	000000	XXX	RANDOM	077712	$CODE$
SEED1	000066	XXX	RANDOM	100000	$CODE$
SEED2	000070	XXX	RANDOM	100002	$CODE$
SEED3	000072	XXX	RANDOM	100004	$CODE$
T$PTHV	000000		LSTAD	000000	$CODE$
T1	030530	$CODE$	BSKEL4	034102	$CODE$