

M8743, M7981 MS11-L/M/P MEM DIAG
M8022 CZMSPB0

AH-T157B-MC
FICHE 1 OF 3

SEP 1982
COPYRIGHT © 1982
MADE IN USA



A large grid of approximately 15 columns and 15 rows of small, illegible text blocks. Each block appears to be a miniature version of a technical diagram or data table, consistent with the 'MEM DIAG' (Memory Diagram) title. The text is too small to be read, but the layout suggests a comprehensive set of diagnostic information for the MS11-L/M/P system.



The main body of the document consists of a dense grid of approximately 15 columns and 20 rows of small, illegible text or data fragments. Each cell in the grid appears to contain a small, structured piece of information, possibly a data point or a small table, but the text is too small to be read. The overall appearance is that of a microfiche or a high-density data storage format.

M8743, M7981 MS11-L/M/P MEM DIAG
M8022 CZMSPB0

AH-T157B-MC
FICHE 3 OF 3

SEP 1982
COPYRIGHT © 1982
MADE IN USA



[Faded vertical text on the left edge of the page, possibly bleed-through from the reverse side or a microfilm strip edge.]

2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39

.TITLE CZMSPB0 MS11-L/M/P MEMORY DIAG.
.NLIST TOC
.REM

IDENTIFICATION

PRODUCT CODE: AC-T156B-MC
PRODUCT NAME: CZMSPB0 MS11-L/M/P MEMORY DIAG
PRODUCT DATE: JULY 1982
MAINTAINER: ELECTRONIC STORAGE DIAGNOSTICS

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS MANUAL.

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED TO THE PURCHASER UNDER A LICENSE FOR USE ON A SINGLE COMPUTER SYSTEM AND CAN BE COPIED (WITH INCLUSION OF DIGITAL'S COPYRIGHT NOTICE) ONLY FOR USE IN SUCH SYSTEM, EXCEPT AS MAY OTHERWISE BE PROVIDED IN WRITING BY DIGITAL.

DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.

COPYRIGHT (C) 1981,1982 DIGITAL EQUIPMENT CORPORATION

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35
 TABLE OF CONTENTS

110-	4569	DEFINE	TRAPS
111-	4685	DEFINE	BASIC PDP11 STUFF
111-	4768	DEFINE	CACHE REGISTERS
111-	4775	DEFINE	CPU REGISTERS
111-	4778	DEFINE	MEMORY MANAGEMENT REGISTERS
113-	4908	DEFINE	UNIBUS MAP REGISTERS
113-	4976	DEFINE	SOFTWARE SWITCH & DISPLAY REGISTERS
113-	4980	DEFINE	CONTROL STATUS REGISTERS
113-	4983	DEFINE	PARAMETERS
115-	4990	MACRO	FATAL
115-	5010	MACRO	TYPE
117-	5028	MACRO	NEWTST
119-	5078	MACRO	\$\$NEWTST
119-	5099	MACRO	SUBTST
119-	5118	MACRO	\$\$SUBTST
121-	5133	MACRO	TYPOCT
123-	5173	MACRO	TYPOCS
125-	5229	MACRO	TYPDEC
126-	5271	MACRO	BMOV
128-	5337	MACRO	MAP
130-	5376	MACRO	SUPERVISOR
130-	5397	MACRO	USER
131-	5419	MACRO	TESTAREA
133-	5441	MACRO	SET4 & RES4
135-	5486	MACRO	DLEFT
137-	5510		TRAP CATCHER
137-	5518		ACT11 HOOKS
137-	5537		APT11 HOOKS
139-	5555	VARIABLES	INITIALIZED TO ZERO
141-	5728	VARIABLES	INITIALIZED TO NON ZERO
143-	5771		CONFIGURATION TABLE
144-	5798		***** MAIN *****
144-	5799		INITIALIZE VARIABLES TO ZERO
144-	5811		CLEAR NON-PROGRAM SPACE
145-	5824		TYPE OF SYSTEM SIZER
147-	5874		INITIALIZE VARIABLES TO NON ZERO
147-	5884		INITIALIZE VECTORS
149-	5904		INITIALIZE PATTERNS
149-	5933	SUBR	PLUG IN NULL PATTERNS
151-	5944		CLEAR THE CONFIGURATION TABLE
151-	5956		SIZE FOR A HARDWARE SWITCH REGISTER
152-	5974		SETUP ACT, APT, & XXDP
154-	5999		PROTECT PROGRAM & LOADERS
154-	6011		CHECK SYSTEM FOR CACHE
155-	6043		SETUP USER & SUPERVISOR STACK
155-	6061		GET SOFTWARE SWITCH REGISTER IF NECESSARY
155-	6068		GET MEMORY MANAGEMENT READY
157-	6074	T1	BIT TEST OF ALL CSR'S
158-	6131		DETERMINE TYPE OF ECC MEMORY
159-	6151		PRINT CSR REGISTER MAP
160-	6192		READ AND WRITE ALL CSR BITS
164-	6430		CLEAR ALL MEMORY SPACE FROM BANK 2 ON
166-	6460		MATCH ALL CSR'S WITH MEMORY
167-	6705	T2	TEST BANK 0 ACCESSES
167-	6734		ENABLE ECC FOR CORRECT TRAPS
169-	6742	T3	TEST BANKS 1-200 (OCTAL) FOR ZEROS & ONES
170-	6867		FIND SHADOW INHIBIT MODE POINTERS

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35
 TABLE OF CONTENTS

172- 6890	T4	ECC INHIBIT MODE POINTER TEST	
182- 7047		LEGAL CONFIGURATION CHECK	
184- 7161		PRINT CONFIGURATION DETAILS	
186- 7218		CHECK APT SIZING	
187- 7253	T5	DIAGNOSTIC MODE DISPATCH ROUTINE	
187- 7270	T6	UNIQUE BANK TEST	
187- 7284		FLUSH OUT DBE'S	
189- 7288		END OF PASS ROUTINE	
191- 7350		WRITE BACKGROUND PATTERNS	
193- 7364		MTEST MODES	
193- 7366		BANKS FORWARD,PATTERNS FORWARD	**RECURSIVE**
195- 7396		BANKS FORWARD,PATTERNS REVERSE	**RECURSIVE**
197- 7426		BANKS WORST FIRST,PATTERNS FORWARD	**RECURSIVE**
199- 7463		BANKS WORST FIRST,PATTERNS REVERSE	**RECURSIVE**
201- 7500		PATTERNS FORWARD,BANKS FORWARD	**RECURSIVE**
203- 7538		PATTERNS FORWARD,BANKS WORST FIRST	**RECURSIVE**
205- 7583		PATTERNS REVERSE,BANKS FORWARD	**RECURSIVE**
207- 7621		PATTERNS REVERSE,BANKS WORST FIRST	**RECURSIVE**
209- 7666	SUBR	SETUP MEMORY TEST	
211- 7686	SUBR	TEST ECC CSR LOGIC DISPATCH	
213- 7777		CHECK FOR SBE FREE LOCATIONS	
215- 7872		CSR PATTERN CASE STATEMENT	
217- 7918	SUBR	ECC TEST DISPATCH	
219- 7973	SUBR	PARITY TEST DISPATCH	
220- 8020		PATTERNS	
220- 8022		MEMORY TEST SETUP ROUTINES	
220- 8023	MT0000	SETUP DATA PATTERN TEST	
220- 8036	MT0001	SETUP ADDRESS TEST	
220- 8058	MT0002	SETUP COMPLEMENT ADDRESS TEST	
222- 8086	MT0003	SETUP 3 XOR 9 WORST CASE NOISE TEST	
222- 8122	MT0004	SETUP ROTATING ZEROS TEST	
222- 8140	MT0005	SETUP ROTATING ONES TEST	
224- 8162	MT0006	SETUP INITIAL DATA TEST	
224- 8169	MT0007	SETUP ADDRESS BIT TEST	
224- 8179	MT0010	SETUP BYTE ADDRESSING TEST	
226- 8188	MT0011	SETUP CREATE SINGLE BIT ERROR TEST	
226- 8197	MT0012	SETUP WRITE BYTE CLEARS SBE TEST	
226- 8212	MT0013	SETUP CREATE DOUBLE BIT ERROR TEST	
227- 8223	MT0014	SETUP BASIC DOUBLE BIT ERROR TEST	
229- 8237	MT0015	SETUP WRITE INHIBIT OF BYTE WITH DBE	
229- 8246	MT0016	SETUP WRITE INHIBIT OF WORD WITH DBE	
229- 8255	MT0017	SETUP HOLDING 1'S & 0'S	
231- 8262	MT0020	SETUP SYNDROMES TO CSR ON SINGLE BIT ERROR	
232- 8275	MT0021	SETUP MARCHING 0'S & 1'S TEST	
233- 8322	MT0022	SETUP REFRESH & SHIFTING DIAGONAL TEST	
233- 8330	MT0023	SHIFTING DIAGONAL TEST	
234- 8340	MT0024	SETUP FAST GALLOPING PATTERN TEST	
234- 8382	MT0025	SETUP INTERRUPT ENABLE TEST	
236- 8393	MT0026	SETUP RANDOM DATA TEST	
238- 8440	MT0027	UNIQUE BANK TEST	
240- 8511	MT0030	SETUP FLUSH OUT DBE'S TEST	
242- 8556	MT0031	SETUP SOB-A-LONG TEST	
244- 8585	MT0032	SETUP WRITE RECOVERY TEST	
246- 8649	MT0033	SETUP BRANCH GOBBLE TEST	
246- 8679	MT0034	SOFT ERROR - BACKGROUND PATTERN TEST	
246- 8709	MT0035	SETUP WORST CASE NOISE PARITY TEST	
247- 8730	MT0036	SETUP CORRECTION CODE TEST	

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35

TABLE OF CONTENTS

248- 8742	MT0037	SETUP ECC DISABLE TEST
248- 8754	MT0040	
249- 8757	MT0041	SETUP ADDRESS TO CSR ON DOUBLE BIT ERROR TEST
250- 8766	MT0042	SETUP EXTENDED UNIBUS ADDRESS TO CSR TEST
251- 8776	MT0043	SETUP WRITE BYTE CLEARS SBE TEST
251- 8784	MT0044	SETUP SHIFTING 1/0'S THROUGH THE CHECK BITS TEST
251- 8792	MT0045	SETUP SYNDROMES TO CSR ON DOUBLE BIT ERROR
251- 8800	MT0046	SETUP CHECK SINGLE BIT ERRORS WITH ECC DISABLED TET
251- 8808	MT0047	SETUP NO CSR UPDATE ON SBE WITH EXSISTING DBE TEST
253- 8819	MT0999	SETUP NULL TEST
253- 8824		CHECK FOR KAMIKAZE MODE
255- 8832	SUBR	EXECUTE PATTERN IN SUPERVISOR
259- 8903	MEMORY	TEST PATTERN ROUTINES
259- 8913	MTP000	BASIC DATA TEST
259- 8924	MTP001	ADDRESS TEST
259- 8936	MTP002	COMPLEMENT ADDRESS TEST (WRITE DOWN, READ UP)
261- 8950	MTPA03	3 XOR 9 WORST CASE NOISE TEST (WRITE)
261- 8973	MTPB03	3 XOR 9 WORST CASE NOISE TEST (READ)
263- 8991	MTPC03	TEST DATA SUBPROGRAM
263- 8999	MTPD03	TEST DATA SUBSUBPROGRAM
265- 9009	MTPA04	ROTATING ZEROS TEST
265- 9022	MTPB04	SUBR ROTATING BIT
265- 9031	MTP005	ROTATION ONES TEST
267- 9045	MTP006	INITIAL DATA TEST
269- 9088	MTP007	ADDRESS BIT TEST
271- 9128	MTP010	BYTE ADDRESSING TEST
273- 9164	MTP011	SINGLE BIT ERROR TEST
275- 9305	MTP012	WRITE BYTE CLEARS SBE TEST
277- 9387	MTP013	CREATE DOUBLE BIT ERROR TEST
282- 9478	MTP014	BASIC DOUBLE BIT ERROR TEST
283- 9526	MTP015	WRITE INHIBIT OF BYTE WITH DBE
285- 9625	MTP016	WRITE INHIBIT OF WORD WITH DBE
289- 9727	MTP017	HOLDING 1'S & 0'S TEST
291- 9761	MTP020	SYNDROMES TO CSR ON SINGLE BIT ERROR TEST
292- 9829	MTPA21	MARCHING 1'S & 0'S PATTERN TEST
296- 9899	MTP022	REFRESH & SHIFTING DIAGONAL TEST
297- 9971	SUBR	REFRESH DELAY
299- 9993	MTPA24	FAST GALLOPING PATTERN TEST
301-10037	MTPB24	FAST GALLOP PART B
301-10045	MTPC24	FAST GALLOP PART C
303-10055	MTP025	INTERRUPT ENABLE TEST
307-10149	MTPA26	RANDOM DATA (WRITE)
307-10156	MTPB26	RANDOM DATA (READ)
307-10174	RANDOM	NUMBER SUBPROGRAM
307-10187	RANDOM	NUMBER SUBSUBPROGRAM
309-10195	MT0030	FLUSH OUT DBE'S
309-10201	MTP031	SOB-A-LONG TEST
311-10252	MTP032	WRITE RECOVERY TEST
313-10272	MTP033	BRANCH GOBBLE TEST
314-10318	MTP034	SOFT ERROR - BACKGROUND PATTERN TEST
315-10330	MTP035	WORST CASE NOISE PARITY TEST
316-10362	MTPC36	CORRECTION CODE TEST
317-10415	MTP037	CHECK ECC DISABLE TEST
319-10436	MTP041	ADDRESS TO CSR ON DOUBLE BIT ERROR TEST
321-10477	MTP042	EXTENDED ADDRESS TO CSR ON ERROR TEST
322-10530	MTP043	WRITE BYTE CLEARS SINGLE BIT ERROR TEST
323-10571	MTP044	SHIFTING CHECK BITS THROUGH THE CSR TEST

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35
 TABLE OF CONTENTS

324-10654	MTP045	SYNDROMES TO CSR ON DOUBLE BIT ERROR TEST
325-10694	MTP046	CHECK SINGLE BIT ERRORS WITH ECC DISABLED
326-10770	MTP047	NO CSR UPDATE ON SBE WITH EXSISTING DBE
327-10814	MISC	SUBROUTINES
327-10816	SUBR	COPY R0 TO R4, R1 TO R3, & R2 TO R5
327-10822	FLIP	WARNING CONSTANTS IN WORST CASE NOISE TESTS
328-10849	SUBR	WRITE BACKGROUND
329-10868	SUBR	GET CSR INFORMATION FROM CONFIGURATION TABLE
331-10882	SUBR	PRINT CONFIGURATION MAP
333-10934	SUBR	TYPE CONFIGURATION
337-11064	TRAP	PARITY ERROR HANDLER
339-11096	TRAP	NON-EXISTANT MEMORY (HOLES) HANDLER
339-11116	TRAP	TIMEOUT (TRAP TO 4) HANDLER
339-11120	TRAP	MEMORY MANAGEMENT (TRAP TO 250) HANDLER
339-11123	TRAP	RESERVED INSTRUCTION HANDLER
339-11133	FIND	BAD SP, PC, & PSW FROM STACK
341-11141	TRAP	KERNEL TRAP HANDLER
341-11149	TRAP	ENERGIZE TRAP HANDLER
341-11153	TRAP	DEENERGIZE TRAP HANDLER
341-11157	TRAP	CACHON TRAP HANDLER
341-11164	TRAP	CACHOFF TRAP HANDLER
343-11172	TRAP	LOAD CSR TRAP HANDLER
343-11191	TRAP	READ CSR TRAP HANDLER
344-11199	TRAP	TEST (R1) & READ CSR CAREFULLY
346-11236	TRAP	ECC DISABLE ALL CSR'S TRAP HANDLER
346-11240	TRAP	ECC DISABLE OF 1 SELECTED CSR TRAP HANDLER
346-11244	TRAP	INITIALIZE ALL CSR'S TRAP HANDLER
346-11248	TRAP	INITIALIZE 1 SELECTED CSR TRAP HANDLER
346-11252	TRAP	ENABLE SBE PARITY TRAPS ON ALL CSR'S
346-11256	TRAP	ENABLE SBE PARITY TRAPS ON 1 SELECTED CSR
346-11260	TRAP	WRITE CHECKBITS THRU ALL CSR'S TRAP HANDLER
346-11265	TRAP	WRITE CHECKBITS THRU 1 SELECTED CSR TRAP HANDLER
348-11272	TRAP	WAS THERE A SBE ON ANY CSR TRAP HANDLER
348-11297	TRAP	WAS THERE A SBE IN 1 SELECTED CSR TRAP HANDLER
350-11307	TRAP	WAS THERE A DBE ON ANY CSR TRAP HANDLER
350-11332	TRAP	WAS THERE A DBE ON 1 SELECTED CSR TRAP HANDLER
352-11343	TRAP	CLEAR ALL ECC CSR'S TRAP HANDLER
352-11347	TRAP	CLEAR 1 SELECTED CSR TRAP HANDLER
352-11351	TRAP	ECC DISABLE, CHECK MODE, & WRITE CHECKBITS IN ALL CSR'S TRAP HANDLER
352-11356	TRAP	ECC DISABLE, CHECK MODE, & WRITE CHECKBITS IN 1 SELECTED CSR
354-11363	SUBR	WRITE IN ALL CSR'S
354-11378	TRAP	INVALIDATE BACKGROUND PATTERN
355-11387	TRAP	GENERATE AND TEST ERROR ADDRESS
355-11441	TRAP	ENABLE CHECK/SYNDROME BIT REGISTER
357-11448	SUBR	GENERATE CHECK BITS
361-11517	SUBR	MAPPER
361-11602	TRAP	MAP KERNEL (ALMOST 1 TO 1) TRAP HANDLER
363-11625	RELOCATE	PROGRAM
365-11730	UNRELOCATE	PROGRAM
365-11775	SETUP	LOWER 16K OF UNIBUS MAP
367-11788	MOVE	BANKS
369-11836	SUBR	MAP USER TO NEW BANK
369-11856	SUBR	SETUP KERNEL PAR'S FOR NEW BANK
369-11869	SUBR	MAP KERNAL PARS 4 AND 5 TO A BANK
369-11879	SUBR	SETUP KERNEL PAR'S FOR NEW LOADER BANK
369-11890	SUBR	UNMAP KERNAL PAR'S 4 AND 5
371-11897	SUBR	EXAMINE BANK

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35
 TABLE OF CONTENTS

373-11977	SUBR	BANK OK?	
373-11988	SUBR	INCREMENT PATTERN TESTING	
373-11996	SUBR	SET HIGHEST PATTERN TESTING TYPE	
373-12000	SUBR	INCREMENT BANK & TEST	
375-12007		BOOTSTRAP ROUTINE	
377-12036		HALT PROGRAM	
377-12045		SHUTDOWN DIAGNOSTIC	
377-12072		APT SHUTDOWN SEQUENCE	
379-12082		BLOCK MOVE SUBROUTINE	
380-12109		FIELD SERVICE MODE	
380-12111	SUBR	FIELD SERVICE	COMMAND MODE
382-12157	COMMAND 0	EXIT	
382-12179	FS	COMMAND 1	READ CSR
384-12194	FS	COMMAND 2	LOAD CSR
386-12218	FS	COMMAND 3	EXAMINE MEMORY
388-12260	FS	COMMAND 4	MODIFY MEMORY
390-12312	FS	COMMAND 5	SELECT BANK & PATTERN
391-12443	FS	COMMAND 6	TYPE CONFIGURATION MAP
393-12449	FS	COMMAND 7	SOB-A-LONG TEST
395-12490	FS	COMMAND 8	ERROR SUMMARY
397-12523	FS	COMMAND 9	REFRESH TEST
399-12564	FS	COMMAND 10	SET FILL COUNT
399-12574	FS	COMMAND 11	ENTER KAMIKAZE MODE
399-12579	FS	COMMAND 12	EXIT KAMIKAZE MODE
399-12585	FS	COMMAND 13	TURN CACHE OFF
399-12592	FS	COMMAND 14	TURN CACHE ON
400-12611	FS	COMMAND 15	TEST ONLY SELECTED BANKS
400-12631	FS	COMMAND 16	RESUME TESTING ALL BANKS
402-12645	FS	COMMAND 17	ENABLE TRACE
404-12651	FS	COMMAND 18	DISABLE TRACE
406-12657	SUBR	DETERMINE CORRECT CSR	
421-13225		ERROR DATA (SUPERVISOR) SETUP STUFF	
421-13239		DATA WAS 3 WORDS	
423-13280		GET DATA FROM ABORTED AREA IF POSSIBLE	
425-13296		POWER FAIL AUTO RESTART	
425-13297		ROUTINE POWER DOWN AND UP	
430-13485		POWER FAIL WHILE RELOCATED	
432-13512		POWER UP FROM BANK 0 TO RELOCATION	
434-13552		IO SUBROUTINES	
434-13554		ROUTINE TYPE	
449-14347		ERROR DATA SETUP	
454-14596		DATA WAS A WORD	
454-14608		DATA WAS A BYTE	
456-14621		DATA WAS A 7 BIT BYTE	
456-14636		DETERMINE XOR OF GOOD & BAD	
458-14645		LOG ERROR ON BAD BANK	
462-14740		ROUTINE SCOPE HANDLER	
463-14810	SUBR	DISPLAY	
465-14827		ROUTINE ERROR HANDLER	
468-14942		ROUTINE ERROR MESSAGE TYPEOUT	
476-15169	SUBR	DETAILED ERROR REPORT	
481-15311		ROUTINE BINARY TO OCTAL (ASCII) AND TYPE	
482-15389		ROUTINE CONVERT BINARY TO DECIMAL AND TYPE	
483-15446		ROUTINE TTY INPUT	
485-15541		CONTROL T	
485-15566		CONTROL S & CONTROL Q	
487-15686		ROUTINE READ AN OCTAL NUMBER FROM THE TTY	

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35

TABLE OF CONTENTS

487-15735	ROUTINE READ A DECIMAL NUMBER FROM THE TTY
488-15794	ROUTINE SAVE AND RESTORE R0-R5
489-15830	ROUTINE RANDOM NUMBER GENERATOR
491-15860	ROUTINE DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT
492-15902	TABLES
492-15904	APT MAILBOX-ETABLE
494-15986	ROUTINE TRAP DECODER
496-16013	TRAP TABLE
500-16116	TABLE ERROR POINTER
509-16404	ERROR DATA TAGS (DT)
511-16431	ERROR DATA FORMATS (DF)
513-16448	ERROR MESSAGES (EM)
515-16494	ERROR DATA HEADERS (DH)
517-16521	MESSAGES

41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64

REVISION HISTORY

=====

REVISION	DATE	AUTHOR	CHANGES
=====	=====	=====	=====
CZMSPA	1-JUN-82	IRA CHAVIS	NONE - NEW PROGRAM
CZMSPB	1-JULY-82	IRA CHAVIS	1-FIX FIELD SERVICE COMMAND 8 2-INCREASE TESTING LIMITS OF MT0044 FROM 16K TO EVERY 100 OCTAL ADDRESS IN 16K

66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87

OPERATIONAL SWITCH SETTINGS
SWITCH REGISTER DEFINITIONS

SWITCH	USE
15	HALT ON ERROR
14	LOOP ON TEST
13	INHIBIT ERROR TYPEOUTS
12	INHIBIT RELOCATION
11	QUICK VERIFY
10	BELL ON ERROR
9	LOOP ON ERROR
8	HALT PROGRAM (UNRELOCATED RESTORE LOADERS)
7	DETAILED ERROR REPORTS
6	INHIBIT CONFIGURATION MAP
5	LIMIT MAX ERRORS PER BANK
4	FAT TERMINAL (132 COLUMNS OR BETTER)
3	TEST MODE - SEE DOCUMENT
2	TEST MODE - SEE DOCUMENT
1	TEST MODE - SEE DOCUMENT
0	DETECT SINGLE BIT ERRORS

TABLE OF CONTENTS

89	
90	
91	
92	
93	1.0 GENERAL PROGRAM INFORMATION
94	
95	1.1 PROGRAM PURPOSE (ABSTRACT)
96	1.2 SYSTEM REQUIREMENTS
97	1.3 RELATED DOCUMENTS AND STANDARDS
98	1.4 DIAGNOSTIC HIERARCHY PREREQUISITES
99	1.5 ASSUMPTIONS
100	
101	2.0 OPERATING INSTRUCTIONS
102	
103	2.1 LOADING AND STARTING PROCEDURES
104	2.2 DEFAULT TEST SEQUENCE
105	2.3 SPECIAL ENVIRONMENTS
106	2.4 PROGRAM OPTIONS
107	2.5 EXECUTION TIMES
108	
109	3.0 ERROR INFORMATION
110	
111	3.1 ERROR REPORTING
112	3.2 ERROR ABBREVIATIONS
113	3.3 ERROR HALTS
114	
115	4.0 PROGRESS REPORTS
116	
117	5.0 CSR INFORMATION TABLES
118	
119	5.1 MS11-P CSR
120	5.2 MS11-L CSR
121	5.3 MS11-M CSR
122	
123	6.0 SUB-TEST SUMMARIES
124	
125	6.1 TESTS
126	6.2 PATTERNS
127	
128	7.0 PROGRAM FEATURES
129	
130	7.1 FAST DATA ACCESS RATES
131	7.2 BANK ZERO TESTING
132	7.3 MEMORY CONFIGURATION MAP
133	7.4 EVERYTHING YOU'VE ALWAYS WANTED TO KNOW ABOUT SUPERMAC ...
134	7.5 MEMORY MANAGEMENT MAPPING

136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177

1.0 GENERAL PROGRAM INFORMATION

1.1 PROGRAM PURPOSE (ABSTRACT)

- A. INTENDED FOR USE ON ALL PDP-11/24/44'S WHICH MEET THE CONDITIONS IN 1.2.1.
- B. THIS PROGRAM WILL BE USED BY SYSTEM MANAGERS AND OPERATORS TO DETERMINE THE CORRECT OPERATION OF MAIN MEMORY AND ALSO IT WILL BE PRIMARILY USED BY FIELD SERVICE AND MANUFACTURING TO ISOLATE FAILURES TO THE MEMORY AND TO ISOLATE FAILURES WITHIN THE MEMORY TO THE CORRECT CARD.
- C. THE OBJECT OF THIS SOFTWARE IS TO FUNCTIONALLY TEST AND VERIFY ALL MAIN MEMORY FUNCTIONS AS FAST AS POSSIBLE.
- D. THERE IS THE CAPABILITY OF TESTING MIXED CONFIGURATIONS (MS11-L, MS11-M AND MS11-P) ON THE SYSTEM.
- E. IT HAS SPECIAL A MAINTENANCE MODE (FIELD SERVICE MODE) TO PROVIDE SPECIFIC FUNCTIONAL CAPABILITIES.

1.2 SYSTEM REQUIREMENTS

1.2.1 HARDWARE REQUIREMENTS -

PDP-11-24/44 CPU WITH 22 BIT ADDRESSING AND AT LEAST 64K (16 BIT WORDS) OF MEMORY AND MEMORY MANAGEMENT.

NOTE

1. LIKE MEMORY TYPES MUST BE ON 16K WORD BOUNDARIES STARTING AT PHYSICAL ADDRESS 0.
2. PDP-11 SERIES 16/18 BIT PROCESSORS ARE NOT SUPPORTED.

179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222

1.2.2 SOFTWARE REQUIREMENTS -

THIS PROGRAM IS DESIGNED TO RUN STAND ALONE OR UNDER ANY OF THE FOLLOWING MONITORS:

XXDP
ACT
APT

1.3 RELATED DOCUMENTS AND STANDARDS

1. PDP-11/04/24/34/44/70 PROCESSOR HANDBOOK (EB-19402)
2. PDP-11/44 USER'S GUIDE (EK-11044-UG)
3. MS11-M USER'S GUIDE (EK-MS11M-UG-001)
4. MS11-L USERS GUIDE (EK-MS11L-UG-001)
5. MS11-P TECHNICAL MANUAL (EK-MS11P-TM-001)

1.4 DIAGNOSTIC HIERARCHY PREREQUISITES

IF THE PROGRAM IN ANY WAY MISBEHAVES, THEN:

1. TRY IT AGAIN WITH CACHE OFF (REFERENCE SECTION 2.4.3.1)
2. INHIBIT RELOCATION (REFERENCE SECTION 2.4.1)
3. TRY CPU DIAGNOSTICS
4. TRY MEMORY MANAGEMENT DIAGNOSTICS
5. TRY CACHE DIAGNOSTICS (WHERE APPLICABLE)
6. TRY UNIBUS MAP DIAGNOSTICS (WHERE APPLICABLE)

224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276

1.5 ASSUMPTIONS

THIS PROGRAM ASSUMES THE CORRECT OPERATION OF THE CPU, MEMORY MANAGEMENT, CACHE, AND THE UNIBUS MAP. THIS PROGRAM OCCUPIES (INITIALLY) BANK 0 (0-16K). THE XXDP LOADERS ARE IN BANK 1.

2.0 OPERATING INSTRUCTIONS

2.1 LOADING STARTING PROCEDURES

2.1.1 QUICK STARTING -

1. LOAD ADDRESS 200
2. SET SWITCH REGISTER FOR OPTIONS (NORMALLY 0)
3. START

NOTE

IF ON AN 11/24 USING MS11-L MEMORY BE SURE THAT THE PERIPHERAL PAGE JUMPER IS IN PLACE; FAILURE TO DO SO SENDS THE DIAGNOSTIC TO NEVER-NEVER LAND.

2.1.2 STOPPING -

1. SET SW8, AND/OR
2. TYPE CONTROL "C" (REFERENCE SECTION 2.4.4.1).

2.1.3 RESTARTING (PRESERVE CONFIGURATION TABLE) -

1. LOAD ADDRESS 202
2. SET SWITCH REGISTER FOR OPTIONS (NORMALLY 0)
3. START

278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304

2.1.4 SWITCH REGISTER OPTIONS -

SWITCH	USE
-----	-----
15	HALT ON ERROR
14	LOOP ON TEST
13	INHIBIT ERROR TYPEOUTS
12	INHIBIT RELOCATION
11	QUICK VERIFY
10	BELL ON ERROR
9	LOOP ON ERROR
8	HALT PROGRAM (UNRELOCATE RESTORE LOADERS)
7	DETAILED ERROR REPORTS
6	INHIBIT CONFIGURATION MAP
5	LIMIT MAX ERRORS PER BANK
4	FAT TERMINAL (132 COLUMNS OR BETTER)
3	TEST MODE - SEE DOCUMENT
2	TEST MODE - SEE DOCUMENT
1	TEST MODE - SEE DOCUMENT
0	DETECT SINGLE BIT ERRORS

306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349

2.2 DEFAULT TEST SEQUENCE

THE FOLLOWING TWO LISTS GIVE THE TEST PROTOCOL FOR PARITY AND ECC MEMORY. TESTS MARKED WITH A "*" ARE NOT NORMALLY RUN EXCEPT UNDER ACT OR APT, OR THROUGH A FIELD SERVICE COMMAND (REFERENCE SECTION 2.4.4.8).

2.2.1 TEST PROTOCOL FOR MS11-L PARITY MEMORY -

TEST	TEST NAME	TIME (SEC/16K)
34	SOFT ERROR TEST	<1
6	INITIAL DATA TEST	<1
17	HOLDING 1'S AND 0'S TEST	<1
7	ADDRESS BIT TEST	<1
1	ADDRESS TEST	<1
2	COMPLEMENT ADDRESS TEST	<1
3	3 XOR 9 TEST	1
4	ROTATING 0'S TEST	1
5	ROTATING 1'S TEST	1
21	MARCHING 1'S AND 0'S TEST	1
35	WORST CASE NOISE PARITY TEST	N/A
* 22	REFRESH TEST	10
* 23	SHIFTING DIAGONAL TEST	10
26	RANDOM DATA TEST	<1
* 24	FAST GALLOPING PATTERN TEST	20
* 31	SOB-A-LONG TEST	3
* 32	WRITE RECOVERY TEST	<1
* 33	BRANCH GOBBLE TEST	35
34	SOFT ERROR TEST	<1

351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401

2.2.2 TEST PROTOCOL FOR MS11-M ECC MEMORY -

TEST	TEST NAME	TIME (SEC/16K)
5	ROTATING 1'S TEST	1
@ 25	INTERRUPT ENABLE TEST	<1
+@ 11	SINGLE BIT ERROR TEST	<2
+@ 12	WRITE BYTE CLEARS SBE TEST	<1
+@ 13	CREATE DOUBLE BIT ERROR TEST	1
+@ 15	WRITE INHIBIT OF BYTE W/DBE TEST	1
+@ 16	WRITE INHIBIT OF WORD W/DBE TEST	<1
34	SOFT ERROR TEST	<1
6	INITIAL DATA TEST	<1
10	BYTE ADDRESS TEST	<1
17	HOLDING 1'S AND 0'S TEST	<1
7	ADDRESS BIT TEST	<1
1	ADDRESS TEST	<1
2	COMPLEMENT ADDRESS TEST	<1
4	ROTATING 0'S TEST	1
5	ROTATING 1'S TEST	1
21	MARCHING 0'S AND 1'S TEST	1
* 22	REFRESH TEST	10
26	RANDOM DATA TEST	<1
* 24	FAST GALLOPING PATTERN TEST	20
* 31	SOB-A-LONG TEST	3
* 32	WRITE RECOVERY TEST	<1
* 33	BRANCH GOBBLE TEST	35
34	SOFT ERROR TEST	<1

@ - RUN ONLY ON THE FIRST PASS WHEN UNDER ACT OR APT

+ - RUN TWICE FOR EACH 16K BANK IF INTERLEAVED

AT THE END OF EACH PASS THE PROGRAM WILL RUN CLEANUP PATTERNS #30, AND #27 FOR ALL BANKS.

403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447

2.2.3 TEST PROTOCOL FOR MS11-P ECC MEMEORY

PATTERN	PATTERN NAME	TIME (SEC/16K)
5	ROTATING 1'S TEST	1
34	SOFT ERROR TEST	<1
6	INITIAL DATA TEST	<1
44	SHIFTING CHECK BITS THRU THE CSR	5
14	BASIC DOUBLE BIT ERROR TEST	<1
45	SYNDROMES IN CSR ON DBE TEST	<1
36	CORRECTION CODE TEST	1
20	SYNDROMES IN CSR ON SBE TEST	1
37	CHECK ECC DISABLE TEST	<1
41	ADDRESS TO CSR ON DBE TEST	1
42	EXTENDED ADDRESS TO CSR TEST	<1
43	BYTE WRITE TEST	<1
46	CHECK SBE WITH ECC DISABLE TEST	<1
47	NO CSR UPDATE ON SBE WITH DBE TEST	<1
10	BYTE ADDRESS TEST	<1
17	HOLDING 1'S AND 0'S TEST	<1
7	ADDRESS BIT TEST	<1
1	ADDRESS TEST	<1
2	COMPLEMENT ADDRESS TEST	<1
4	ROTATING 0'S TEST	1
5	ROTATING 1'S TEST	1
21	MARCHING 0'S AND 1'S TEST	1
* 22	REFRESH TEST	10
26	RANDOM DATA TEST	<1
* 24	FAST GALLOPING PATTERN TEST	20
* 31	SOB-A-LONG TEST	3
* 32	WRITE RECOVERY TEST	<1
* 33	BRANCH GOBBLE TEST	35
34	SOFT ERROR TEST	<1

@ - RUN ONLY ON THE FIRST PASS WHEN UNDER ACT OR APT

AT THE END OF EACH PASS THE PROGRAM WILL RUN CLEANUP PATTERNS #30, AND #27 FOR ALL BANKS.

449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503

2.3 SPECIAL ENVIRONMENTS

2.3.1 XXDP -

THE FIRST PASS WILL BE A QUICK VERIFY PASS IF AND ONLY IF IT IS IN CHAIN MODE.

2.3.2 ACT APT AUTOMATIC MODE -

THE PROGRAM WILL NOT CREATE DOUBLE BIT ERRORS (DBE'S) AFTER THE 1ST PASS.

2.3.2.1 APT EXECUTION TIMES -

HERE ARE SOME MEASURED EXECUTION TIMES FOR AN 11/44 WITH CACHE UNDER APT

	1ST QV PASS	2ND PASS ONWARD
128K MS11-M (NON-INTERLEAVED)	10 MIN 5 SEC	7 MIN 40 SEC
128K MS11-L	9 MIN 50 SEC	7 MIN 30 SEC
256K MS11-M (INTERLEAVED)	19 MIN 50 SEC	14 MIN 45 SEC
512K MS11-P	NOT ESTABLISHED AT RELEASE TIME	

THE FIRST PASS WILL BE A QUICK VERIFY PASS

NOTE

EVEN THOUGH THE FIRST PASS IS A QV PASS IT TAKES LONGER THAN THE SUBSEQUENT NON-QV PASSES DUE TO THE FACT THAT IT IS RUNNING MORE PATTERNS, SOME OF WHICH (PATTERNS #24 AND #33 FOR EXAMPLE) CAN BE EXTREMELY TIME CONSUMING.

505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561

2.3.2.2 APT ENVIRONMENT TABLE -

THE FOLLOWING TABLE GIVES SOME OF THE STANDARD SETTINGS FOR THE APT E-TABLE. THEY MAY BE MODIFIED AS NOTED AS THE USER SEES FIT.

FIRST PASS RUN TIME:

THIS PARAMETER SHOULD BE SET ACCORDING TO THE AMOUNT AND TYPE OF MEMORY TO BE TESTED. THE ABOVE TABLE (APT EXECUTION TIMES) GIVES SOME MEASURED TIMES. FOR ANY PATTERNS DELETED (THROUGH USE OF THE DEVICE DESCRIPTOR WORDS) REFERENCE SECTION 2.2 FOR INDIVIDUAL PATTERN TIMES.

NOTE

THE TIMES GIVEN IN SECTION 2.2 ARE FOR 16K CHUNKS OF MEMORY, NOT 128K BOARDS!

LONGEST TEST TIME:

THIS PARAMETER SHOULD BE SET TO THE EXECUTION TIME OF THE LONGEST PATTERN BEING RUN. FOR THE DEFAULT CASE THIS IS 35 SECONDS FOR PATTERN #33.

ADDITIONAL RUN TIME:

NOT USED BY PROGRAM.

SOFTWARE ENVIRONMENT:

FOR APT AUTO MODE THIS PARAMETER SHOULD BE SET TO A "1". FOR DUMP MODE SET THIS TO A "0".

ENVIRONMENT MODE:

WHEN THIS PARAMETER IS SET TO A "0" THE PROGRAM DOES IT'S OWN SIZING. IF THE USERS SETS BIT #7 HOWEVER, HE MUST SPECIFY THE TYPES AND AMOUNTS OF MEMORY TO BE TESTED.

SWITCH 1:

THE DEFAULT SETTING OF THIS SWITCH IS "101". APT USES THIS AS THE SWITCH REGISTER FOR THE PROGRAM. REFERENCE SECTION 2.4.1 FOR MORE INFORMATION ON SWITCH SETTINGS.

SWITCH 2:

THIS SWITCH, IF SET TO ANY NON-ZERO NUMBER, IS USED TO LIMIT THE AMOUNT OF PASSES APT WILL MAKE. THE PROGRAM WILL HANG AFTER THIS COUNT HAS BEEN REACHED.

CPU OPTIONS:

NOT USED BY PROGRAM.

MEMORY TYPE N (N=1 TO 4)

IF BIT #7 OF ENVIRONMENT MODE IS SET THESE FOUR WORDS ARE USED TO LOG THE DIFFERENT TYPES OF MEMORY TO BE TESTED. IF BIT #7 IS NOT SET THESE LOCATION ARE NOT USED.

MAXIMUM ADDRESS N (N=1 TO 4)

THESE FOUR WORDS ARE USED IN CONJUNCTION WITH THE CORRESPONDING

562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610

MEMORY TYPE WORDS TO INDICATE THE HIGHEST ADDRESS THAT MEMORY TYPE OCCUPIES.

NOTE

THE ABOVE TWO PARAMETERS DO NOT ACTUALLY HAVE TO REPRESENT AN ACCURATE CONFIGURATION OF MEMORY. ALL THE PROGRAM LOOKS FOR IS AN ACCURATE TALLY OF MEMORY AMOUNT!

INTERRUPT VECTOR N (N=1 TO 2)
NOT USED BY PROGRAM.

BUS PRIORITY N (N=1 TO 2)
NOT USED BY PROGRAM.

BASE ADDRESS:
NOT USED BY PROGRAM.

DEVICE MAP:
NOT USED BY PROGRAM.

CONTROLLER DESCRIPTOR CODE N (N=1 TO 2)
NOT USED BY PROGRAM.

DEVICE DESCRIPTOR CODES:

THE DEVICE DESCRIPTOR CODES ARE USED BY THE PROGRAM TO DETERMINE WHICH PATTERNS IT WILL RUN. THE DEFAULT VALUES OF THESE WORDS ARE ALL "1"'S, INDICATING THAT ALL OF THE PATTERNS SHOWN IN SECTION 2.2 ARE EXECUTED (SAVE FOR EXCEPTIONS AS NOTED THERE). EACH SET OF WORDS CONTROLS A TABLE IN THE PROGRAM AS FOLLOWS:

DD WORDS	PROGRAM TABLE (SYMBOLIC LOCATION)
WORDS 0-1	MKCSRT
WORDS 2-3	MKPAT
WORDS 4-5	MJPAT

BIT #0 SET IN THE FIRST WORD INDICATES THAT THE FIRST PATTERN IN THE TABLE WILL BE EXECUTED, BIT #1 THE SECOND, BIT #2 THE THIRD,.... BIT #0 OF THE SECOND WORD INDICATES THAT THE 17TH ENTRY IN THE TABLE WILL BE EXECUTED, AND SO ON.

612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631

2.3.3 NO SBE FREE BANKS -

IF THE PROGRAM CANNOT FIND ANY SBE (SINGLE BIT ERROR) FREE LOCATIONS (IN NON-PROTECTED ECC MEMORY) IT WILL PRINT OUT AN ERROR MESSAGE AND CONTINUE TESTING BY-PASSING THE ECC LOGIC TESTS.

2.3.4 MIXED PARITY ECC CONFIGURATIONS -

THE PROGRAM WILL FUNCTION NORMALLY IN MIXED ENVIRONMENTS. THE SEQUENCE OF TESTING MAY SEEM STRANGE DUE TO THE RECURSIVE TEST MODE ALGORITHM (REFERENCE SECTIONS 2.4.1.1, 2.4.1.2, 2.4.1.3).

633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685

2.4 PROGRAM OPTIONS

2.4.1 SWITCH REGISTER DETAILS -

IF A HARDWARE SWITCH REGISTER IS NOT AVAILABLE THEN THE SOFTWARE SWITCH REGISTER IS IN LOCATION 176. IF UNDER APT IF BIT7 IS SET IN THE E-TABLE SYMBOLIC LOCATION '\$ENVM' THE APT SOFTWARE SWITCH REGISTER WILL BE USED (LOCATION \$\$SWREG).

TO CHANGE THE SOFTWARE SWITCH REGISTER CONTENTS: TYPE "CONTROL G". THIS WILL CAUSE DISPLAY THE CURRENT VALUE OF THE SWR AND PROMPT FOR THE OCTAL INPUT OF THE NEW SWR VALUE FROM THE TERMINAL. THIS ROUTINE WILL IGNORE YOU (NOT RESPOND TO CONTROL 'G') IF YOU HAVE A HARDWARE SWITCH REGISTER.

SW15 = HALT ON ERROR
(100000)

CONTINUING FROM THIS HALT WILL FIRST CHECK FOR A CHANGE IN THE SOFTWARE SWITCH REGISTER ("CONTROL G" IN THE TTY INPUT BUFFER) THEN IT WILL CONTINUE TESTING.

SW14 = LOOP ON TEST
(40000)

THIS WILL CAUSE LOOPING ON THE PRESENT TEST OR PATTERN (BACK TO LAST SCOPE TRAP). IF IN A PATTERN THEN THE LOOPING WILL BE FOR AN ENTIRE BANK OF 16K ADDRESSES.

SW13 = INHIBIT ERROR TYPEOUTS
(20000)

THIS WILL CAUSE RETURNS FROM THE ERROR ROUTINE WITHOUT THE TYPED MESSAGES. OTHER ON ERROR FUNCTIONS ARE NOT AFFECTED.

SW12 = INHIBIT RELOCATION
(10000)

THIS PREVENTS THE PROGRAM FROM MOVING AND CONSEQUENTLY PREVENTS THE PROGRAM FROM TESTING AT LEAST 32K OF MEMORY.

SW11 = QUICK VERIFY
(4000)

IF THIS SWITCH IS SELECTED APPROXIMATELY ONE 64TH OF THE POSSIBLE COMBINATIONS OF SBE'S DBE'S ARE TESTED.

EACH PASS COMPLETE TYPEOUT WILL INDICATE THIS MODE BY PRECEDING THE PASS NUMBER WITH "QV".

687 SW10 = BELL ON ERROR
688 (2000)
689 THIS CAUSES A BELL (OR BEEP OR CLICK) ON EACH ERROR
690 TRAP
691
692
693 SW9 = LOOP ON ERROR
694 (1000)
695 THIS WILL CAUSE LOOPING FROM FAILURE POINT BACK TO THE
696 LAST CORRECTLY INITIALIZED AREA OF THE CURRENT TEST.
697
698
699 SW8 = HALT PROGRAM
700 (400)
701 THIS INITIATES THE FOLLOWING SEQUENCE:
702
703 1. IF PROGRAM IS RELOCATED IT MOVES BACK TO BANK ZERO.
704 2. FLUSH OUT ALL POSSIBLE DBE'S.
705 3. TURNS OFF MEMORY MANAGEMENT.
706 4. RESTORE LOADERS.
707 5. UNMAP THE UNIBUS MAP (IF THERE IS ONE).
708 6. HALT IF UNDER APT OR ACT BRANCH SEL.
709
710
711
712
713
714
715
716
717
718 SW7 = DETAILED ERROR REPORTS
719 (200)
720 AFTER ANY NORMAL ERROR REPORT IS TYPED THIS OPTION
721 CAUSES THE CONTENTS OF THE FOLLOWING REGISTERS TO BE
722 TYPED:
723 R0, R1, R2, R3, R4, R5, SP, "CONTROL", "CPUERR"
724
725
726 SW6 = INHIBIT CONFIGURATION MAP
727 (100)
728 THIS INHIBITS THE PRINTING OF A MAP SHOWING THE MEMORY
729 CONFIGURATION - REFERENCE SECTION 7.3
730
731
732 SW5 = LIMIT MAX ERRORS PER BANK
733 (40)
734 THIS WILL LIMIT THE NUMBER OF ERROR TYPEOUTS PER BANK.
735 THE DEFAULT IS 10. DECIMAL, HOWEVER THIS CAN BE
736 CHANGED BY CHANGING LOCATION "ERRMAX" MANUALLY.
737
738

740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792

SW4 = FAT TERMINAL
(20)

THIS INFORMS THE PROGRAM THAT THE CONSOLE TERMINAL HAS
A WIDTH OF AT LEAST 132 COLUMNS (LA36 WITH WIDE PAPER).

SW3-1 = TEST MODE

TEST MODES DETERMINE THE RECURSION ALGORITHM TO BE USED
DURING PATTERN TESTS.

MODE NAME DESCRIPTION

(0)	0	BAFPAF	BANKS FORWARD, PATTERNS FORWARD
(2)	1	BAFPAR	BANKS FORWARD, PATTERNS REVERSE
(4)	2	BAWPAF	BANKS WORST FIRST, PATTERNS FORWARD.
(6)	3	BAWPAR	BANKS WORST FIRST, PATTERNS REVERSE.
(10)	4	PAFBAF	PATTERNS FORWARD, BANKS FORWARD
(12)	5	PAFBAW	PATTERNS FORWARD, BANKS WORST FIRST
(14)	6	PARBAF	PATTERNS REVERSE, BANKS FORWARD
(16)	7	PARBAW	PATTERNS REVERSE, BANKS WORST FIRST.

FOR MORE DETAILS REFERENCE SECTION 2.4.1.1, 2.4.1.2 AND
2.4.1.3.

SW0 = DETECT SINGLE BIT ERRORS (SBI'S)
(1)

FOR MANUFACTURING PURPOSES THIS SWITCH SHOULD ALWAYS BE
ON. FOR FIELD SERVICE PURPOSES THIS SWITCH SHOULD
ALWAYS BE OFF.

THIS SWITCH WILL ALLOW ALL ECC SINGLE BIT ERRORS TO BE
REPORTED BY DISABLING ERROR CORRECTION.

ERROR PRINTOUTS OF SBE'S ARE NOT DISTINGUISHABLE FROM
DBE'S.

NOTE

IF DOUBLE BIT ERRORS ARE FOUND IN THE MEMORY,
THIS SWITCH SHOULD BE SET TO MAKE SURE THAT NEW
DATA CAN BE WRITTEN TO THE DBE LOCATIONS.

2.4.1.1 TEST MODE EXAMPLE -

EXAMPLE ANALYSIS OF MODE 5 'PAFBAW'. ASSUME BANKS 0 1 ARE MS11-L
AND BANKS 2,3,4, 5 ARE MS11-M.

794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845

ASSUME ALSO THAT BANK 3 IS KNOWN BAD BY THE PROGRAM VIA THE SIZING
ROUTINE OR PREVIOUS RUNS THE TESTING SEQUENCE WOULD BE AS FOLLOWS:

;TEST MS11-M MEMORY TYPES FIRST
;TEST KNOWN BAD MEMORY (BANK 3)

TEST 17.	BANK 3
TEST 7.	BANK 3
TEST 1.	BANK 3
TEST 2.	BANK 3
TEST 4.	BANK 3
TEST 5.	BANK 3
TEST 21.	BANK 3
TEST 20.	BANK 3
TEST 22.	BANK 3
TEST 26.	BANK 3

;TEST PRESUMED GOOD MEMORY (BANKS 2,4,5)

TEST 17.	BANK 2
TEST 7.	BANK 2
TEST 1.	BANK 2
TEST 2.	BANK 2
TEST 4.	BANK 2
TEST 5.	BANK 2
TEST 21.	BANK 2
TEST 20.	BANK 2
TEST 22.	BANK 2
TEST 26.	BANK 2
TEST 17.	BANK 4
TEST 7.	BANK 4
TEST 1.	BANK 4
TEST 2.	BANK 4
TEST 4.	BANK 4
TEST 5.	BANK 4
TEST 21.	BANK 4
TEST 20.	BANK 4
TEST 22.	BANK 4
TEST 26.	BANK 4
TEST 17.	BANK 5
TEST 7.	BANK 5
TEST 1.	BANK 5
TEST 2.	BANK 5
TEST 4.	BANK 5
TEST 5.	BANK 5
TEST 21.	BANK 5
TEST 20.	BANK 5
TEST 22.	BANK 5
TEST 26.	BANK 5

847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898

;RELOCATE TEST PROGRAM SPACE (BANK 0 & 1)

TEST 1,	BANK 0
TEST 2,	BANK 0
TEST 3,	BANK 0
TEST 4,	BANK 0
TEST 5,	BANK 0
TEST 26,	BANK 0
TEST 1,	BANK 1
TEST 2,	BANK 1
TEST 3,	BANK 1
TEST 4,	BANK 1
TEST 5,	BANK 1
TEST 26,	BANK 1

NOTE

THIS IS AN EXAMPLE NOT AN ACTUAL SEQUENCE.

THE TEST SEQUENCE WAS FORWARD (THE SIMPLE PATTERNS FIRST, COMPLEX TESTS LAST) SEQUENCE OF PATTERNS (MS11-M = 17, 7, 1, 2, 4, 5, 21, 20, 22, 26)(MS11-L = 1, 2, 3, 4, 5, 26).

IF THE BANK SELECTION IS FORWARD THE BANKS WILL BE TESTED IN THE FOLLOWING ORDER:

1. ECC BANKS THAT ARE NOT PROTECTED OR PROGRAM SPACE (FROM 0 TO 167).
2. PARITY BANKS THAT ARE NOT PROGRAM SPACE (FROM 0 TO 167).
3. THE PROGRAM NOW RELOCATES TESTS:
4. ECC BANKS THAT WERE PROTECTED OR PROGRAM SPACE (FROM 0 TO 167).
5. PARITY BANKS THAT WERE PROGRAM SPACE (FROM 0 TO 167).

IF BANK SELECTION IS WORST FIRST THE CONFIGURATION TABLE WILL BE CONSULTED AND BANKS WILL BE TESTED IN THE FOLLOWING ORDER.

1. ECC BANKS THAT ARE KNOWN BAD AND ARE NOT PROTECTED OR PROGRAM SPACE (FROM 0 TO 167).
2. PARITY BANKS THAT ARE KNOWN BAD AND ARE NOT PROGRAM SPACE (FROM 0 TO 167).
3. ECC BANKS THAT ARE PRESUMED GOOD AND ARE NOT PROTECTED OR

900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953

PROGRAM SPACE (FROM 0 TO 167).

4. PARITY BANKS THAT ARE PRESUMED GOOD AND ARE NOT PROGRAM SPACE (FROM 0 TO 167).
5. THE PROGRAM NOW RELOCATES TESTS:
6. ECC BANKS THAT ARE KNOWN BAD AND WERE PROTECTED OR PROGRAM SPACE (FROM 0 TO 167).
7. PARITY BANKS THAT ARE KNOWN BAD AND WERE PROGRAM SPACE (FROM 0 TO 167).
8. ECC BANKS THAT ARE PRESUMED GOOD AND WERE PROTECTED OR PROGRAM SPACE (FROM 0 TO 167).
9. PARITY BANKS THAT ARE PRESUMED GOOD AND WERE PROGRAM SPACE (FROM 0 TO 167).

2.4.1.2 TEST MODE DETAILS -

MODE 0 = 'BAFPAF' BANKS FORWARD, PATTERNS FORWARD

THIS IS THE DEFAULT AND SIMPLEST MODE.

THIS MODE TESTS EACH BANK COMPLETELY FROM 0 TO 167 EXCEPT THOSE REQUIRING RELOCATION*.

WHILE TESTING EACH BANK THE PATTERNS ARE RUN WITH THE SIMPLE ONES FIRST BUILDING TO THE MORE COMPLEX.

MODE 1 = 'BAFPAR' = BANKS FORWARD, PATTERNS REVERSE

THIS MODE TESTS EACH BANK COMPLETELY FROM 0 TO 167 EXCEPT THOSE REQUIRING RELOCATION*.

WHILE TESTING EACH BANK THE PATTERNS ARE RUN WITH THE MOST COMPLEX ONES FIRST, WORKING TO THE SIMPLE ONES.

MODE 2 = 'BAWPAF' = BANKS WORST FIRST, PATTERNS FORWARD

THIS MODE FIRST TESTS EACH KNOWN BAD BANK COMPLETELY FROM 0 TO 167 EXCEPT THOSE REQUIRING RELOCATION*, THEN PRESUMED GOOD BANKS ARE TESTED FROM 0 TO 167 EXCEPT THOSE REQUIRING RELOCATION*.

WHILE TESTING EACH BANK THE PATTERNS ARE RUN WITH THE SIMPLE ONES FIRST, BUILDING TO THE MORE COMPLEX.

MODE 3 = 'BAWPAR' = BANKS WORST FIRST, PATTERNS REVERSE

THIS MODE FIRST TESTS EACH KNOWN BAD BANK COMPLETELY

955 FROM 0 TO 167 EXCEPT THOSE REQUIRING RELOCATION*, THEN
 956 PRESUMED GOOD BANKS ARE TESTED FROM 0 TO 167 EXCEPT
 957 THOSE REQUIRING RELOCATION*.
 958
 959 WHILE TESTING EACH BANK THE PATTERNS ARE RUN WITH THE
 960 MOST COMPLEX ONES FIRST, WORKING TO THE SIMPLE ONES.
 961
 962 MODE 4 = 'PAFBAF' = PATTERNS FORWARD, BANKS FORWARD
 963
 964 THIS MODE TESTS EACH PATTERN COMPLETELY WITH THE SIMPLE
 965 ONES FIRST, BUILDING TO THE MORE COMPLEX.
 966
 967 WHILE TESTING EACH PATTERN THE BANKS ARE RUN FROM 0 TO
 968 167 EXCEPT THOSE REQUIRING RELOCATION*.
 969
 970
 971 MODE 5 = 'PAFBAW' = PATTERNS FORWARD, BANKS WORST FIRST
 972
 973 THIS MODE TESTS EACH PATTERN COMPLETELY WITH THE SIMPLE
 974 ONES FIRST, BUILDING TO THE MORE COMPLEX.
 975
 976 WHILE TESTING EACH PATTERN FIRST EACH KNOWN BAD BANK
 977 FROM 0 TO 167 EXCEPT THOSE REQUIRING RELOCATION* IS
 978 RUN, THEN PRESUMED GOOD BANKS ARE RUN FROM 0 TO 167
 979 EXCEPT THOSE REQUIRING RELOCATION*.
 980
 981 MODE 6 = 'PARBAF' = PATTERNS REVERSE, BANKS FORWARD
 982
 983 THIS MODE TESTS EACH PATTERN COMPLETELY WITH THE MOST
 984 COMPLEX ONES FIRST, WORKING TO THE SIMPLE ONES.
 985
 986 WHILE TESTING EACH PATTERN THE BANKS ARE RUN FROM 0 TO
 987 167 EXCEPT THOSE REQUIRING RELOCATION*.
 988
 989 MODE 7 = 'PARBAW' = PATTERNS REVERSE, BANKS WORST FIRST
 990
 991 THIS MODE TESTS EACH PATTERN COMPLETELY WITH THE MOST
 992 COMPLEX ONES FIRST, WORKING TO THE SIMPLE ONES.
 993
 994 WHILE TESTING EACH PATTERN FIRST EACH KNOWN BAD BANK
 995 FROM 0 TO 167 EXCEPT THOSE THAT REQUIRE RELOCATION* IS
 996 RUN, THEN PRESUMED GOOD BANKS ARE RUN FROM 0 TO 167
 997 EXCEPT THOSE REQUIRING RELOCATION*.
 998
 999
 1000
 1001
 1002
 1003
 1004
 1005
 1006
 1007

NOTE

* RELOCATION IS REQUIRED TO TEST THE BANK(S) IN PROGRAM SPACE AND ALSO TO TEST ANY ECC BANKS PROTECTED BY DIAGNOSTIC CHECKMODE WITH THE INHIBIT MODE POINTER OFF (ZERO)!

1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048

2.4.1.3 TEST MODE APPLICATIONS -

1. TO VERIFY CORRECT OPERATION OF THE MEMORY SYSTEM USE MODE 0 'BAFPAF'.

ADVANTAGES: EASY TO UNDERSTAND.

DISADVANTAGES: IN CASE OF A FAILING BANK, IT MAY TAKE A LONG TIME TO FIND THE FAILURE.

2. TO GET DETAILED ERROR INFORMATION ON KNOWN BAD BANKS (FOUND BY SIZING ROUTINE) USE MODE 2 'BAWPAF'.

ADVANTAGES: SEEKS BAD BANKS. EASY TO UNDERSTAND.

DISADVANTAGES: FAILURES OTHER THAN ZEROS ONES MAY TAKE A LONG TIME TO FIND.

3. TO GET GOOD ERROR INFO ON ANY MEMORY PROBLEM FAST USE MODE 4 'PAFBAF'.

ADVANTAGES: COVERS ALL BANKS FAST. EASY TO UNDERSTAND.

DISADVANTAGES: FAILURES FROM ONLY COMPLEX PATTERNS MAY TAKE A LONG TIME TO FIND.

4. TO FIND ANY PROBLEM FAST USE MODE 7 'PARBAW'.

ADVANTAGES: COVERS ALL BANKS FAST.

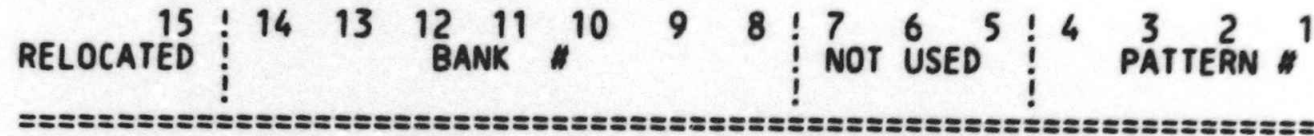
DISADVANTAGES: DIFFICULT TO UNDERSTAND FAILURES REPORTED ARE NOT NECESSARILY THE MOST BASIC FAILURE MODES.

1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100

2.4.2 DISPLAY REGISTER -

A SOFTWARE DISPLAY REGISTER EXISTS IN LOCATION 174 IN ADDITION TO ANY HARDWARE DISPLAY EXISTENCE.

DISPLAY FIELDS ARE AS FOLLOWS:



PATTERN # = THE NUMBER OF THE PATTERN PRESENTLY BEING RUN. ALL PATTERNS ARE DESCRIBED IN SECTION 6.2. ANY PATTERN CAN BE FOUND IN THE DIAGNOSTIC BY LOOKING UP THE SYMBOLIC TAGS 'MTOONN' AND 'MTPONN' - WHERE 'NN' IS THE TEST NUMBER. MTOONN REFERS TO THE ROUTINE THAT SETS UP FOR THE TEST PATTERN WHEREAS MTPONN IS THE ACTUAL PATTERN ITSELF.

NOTE

THE PATTERN # IS NOT NECESSARILY AN INDICATION OF DEGREE OF DIFFICULTY.

BANK = THE NUMBER OF THE BANK (16K) OF MEMORY UNDER TEST (0-167). THESE BITS DIRECTLY MAP TO PHYSICAL ADDRESS BITS (21:15).

RELOCATED = THIS BIT INDICATES THAT THE PROGRAM IS RELOCATED AND NO LONGER IN BANK 0. IT WILL BE RELOCATED TO THE FIRST KNOWN GOOD NON-PROTECTED MEMORY BANK INDICATED ON THE CONFIGURATION MAP (REFERENCE SECTION 7.3).

NOTE

ANOTHER WAY TO OBTAIN THIS INFORMATION IS TO TYPE A CONTROL/T AT THE CONSOLE (REFERENCE SECTION 2.4.4.5).

1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158

2.4.3 SPECIAL MEMORY LOCATIONS -

2.4.3.1 CACHE CONSTANT -

THE CACHE CONSTANT IS LOCATED AT SYMBOLIC LOCATION "CACHK" AND IS USED TO ENABLE CACHE.

NOTE

BIT 0 IN THE CACHE CONSTANT HAS NO EFFECT SINCE IT IS UNCONDITIONALLY SET BY THE PROGRAM WHENEVER IT TRIES TO ENABLE CACHE.

2.4.3.2 CONFIGURATION TABLE

THE CONFIGURATION TABLE IS LOCATED AT SYMBOLIC LOCATION "CGNFIG" AND HAS THE FOLLOWING FORMAT:

CONFIG: FIRST 16K CONFIGURATION WORDS (2 EACH)
2ND 16K CONFIGURATION WORDS (2 EACH)
200TH 16K CONFIGURATION WORDS (2 EACH)

CONFIGURATION WORDS:

LOW:	BIT 0	ERRORS PRESENT
	BIT 1	MEMORY EXISTS
	BIT 2-4	RESERVED
	BIT 5	SKIP ECC LOGIC TESTS FLAG (1=SKIP)
	BIT 6	PROTECTED REGION OF AN ECC MEMORY
	BIT 7	PROTECTED (PROGRAM SPACE)
	BIT 8-11	CSR CODE
	BIT 12-15	INTERLEAVED CSR CODE
MED:	BIT 0-7	NUMBER OF ERRORS
	BIT 8-10	MEMORY TYPE
	BIT 11	CSR TESTED OK
	BIT 12	INTERLEAVE ENABLED
	BIT 13	"BACKGROUND PATTERN VALID" FLAG
	BIT 14	BANK SELECTED FOR TEST BY FIELD SERVICE MODE
	BIT 15	LOADERS HOME BANK

THIS TABLE IS USED AS THE SOURCE FOR THE CONFIGURATION MAP (REFERENCE. SECTION 7.3).

1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216

2.4.4 TERMINAL COMMANDS -

2.4.4.1 CONTROL 'C'

THIS COMMAND WILL:

1. IF SWITCH 8 (HALT PROGRAM) IN THE SWITCH REGISTER IS SET HALT THE PROGRAM.
2. IF SWITCH 8 IS NOT SET, UNRELOCATE IF PROGRAM WAS RELOCATED.
3. FLUSH OUT ANY DBE'S.
4. TURN OFF MEMORY MANAGEMENT.
5. ATTEMPT TO BOOT RK05 DRIVE 0.
6. FAILING 4, ATTEMPT TO BOOT RK04 DRIVE 1.
7. FAILING 5, GO TO 4.

THIS COMMAND WILL ONLY BE RECOGNIZED AT THE COMPLETION OF THE CURRENT TEST OR PATTERN, OR AT THE END OF A LINE OF AN ERROR MESSAGE.

2.4.4.2 CONTROL 'K' (KILL ERROR PRINTOUT AND SKIP PATTERN)

THIS COMMAND WILL ALLOW YOU TO STOP AN ERROR PRINTOUT AND SKIP TO THE NEXT PATTERN. THIS IS HANDY, FOR EXAMPLE, WHEN YOU HAVE A WHOLE BANK FULL OF ERRORS, HAVE GOTTEN ENOUGH INFORMATION, AND WISH TO SKIP TO THE NEXT PATTERN.

2.4.4.3 CONTROL 'T' (TELL ME WHAT'S HAPPENING)

THIS COMMAND WILL PRINT OUT THE INFORMATION ENCODED IN THE DISPLAY REGISTER. THIS IS MAINLY INTENDED FOR CPU'S WITHOUT A HARDWARE DISPLAY REGISTER.

EXAMPLE:

BANK = 17 TEST = 46
RELOCATED BANK= 0 PAT= 26

BY USE OF FIELD SERVICE COMMAND 17 "TRACE" CAN BE SET SO THAT IT WILL AUTOMATICALLY TYPE OUT THE BANK AND PATTERN NUMBERS AS EACH PATTERN IS RUN. (REFERENCE SECTION 2.4.4.8.18).

1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262

2.4.4.4 CONTROL 'S' (STOP)

THIS COMMAND WILL STOP TYPEOUT (SOON) AND WILL WAIT FOR A CONTROL 'Q'.

2.4.4.5 CONTROL 'Q' (QUINTINUE)

THIS COMMAND WILL CONTINUE TYPING THAT HAS BEEN STOPPED BY CONTROL 'S'. IF THERE HAS BEEN NO CONTROL 'S' TYPED THEN THIS COMMAND IS IGNORED.

2.4.4.6 CONTROL 'F' (FIELD SERVICE MODE)

THIS COMMAND WILL CAUSE YOU TO ENTER A MODE WHICH LOOKS FOR SUB COMMANDS.

WHEN THE PROGRAM IS LOOKING FOR A SUB COMMAND ANY NUMBER THAT IS NOT A LEGAL COMMAND WILL CAUSE A MINI HELP MESSAGE TO BE TYPED. THEREFORE WHEN IN DOUBT TYPE 99 (CR) AND YOU WILL GET HELP.

NOTE

TYPING JUST CARRIAGE RETURN IS A DEFAULT COMMAND 0.

2.4.4.7.1 FIELD SERVICE COMMAND 0 (EXIT)

THIS COMMAND WILL EXIT FIELD SERVICES MODE AND RETURN TO WHATEVER TASK IT WAS IN PRIOR TO TYPING CONTROL 'F'. NOTE TYPING JUST CARRIAGE RETURN IS A DEFAULT COMMAND 0.

1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313

2.4.4.7.2 FIELD SERVICE COMMAND 1 (READ CSR)

THIS COMMAND WILL TYPEOUT THE CONTENTS OF THE CSR.

IF THERE IS MORE THAN ONE CSR ON THE CPU (OR IF THE PROGRAM HAS NOT DETERMINED THE CSR STATUS YET), IT WILL ASK YOU "WHICH CSR(0-F)" TO WHICH YOU MUST RESPOND WITH AN HEXIDECIMAL NUMBER FROM 0 TO F. NOTE TYPING JUST CARRIAGE RETURN IS A DEFAULT 0.

IF THE CSR YOU SELECT CAUSES A TRAP TO 4 THE PROGRAM WILL TYPE "THIS CSR DOES NOT EXIST".

NOTE

CSR REFERENCES ARE DONE IN ACCORDANCE WITH SECTION 5.0.

2.4.4.7.3 FIELD SERVICE COMMAND 2 (LOAD CSR)

THIS COMMAND WILL ENABLE YOU TO LOAD THE CSR.

IF THERE IS MORE THAN ONE CSR ON THE CPU (OR IF THE PROGRAM HAS NOT YET DETERMINED THE CSR STATUS YET) IT WILL ASK YOU "WHICH CSR(0-F)" TO WHICH YOU MUST RESPOND WITH AN HEXIDECIMAL NUMBER FROM 0 TO F. NOTE TYPING JUST CARRIAGE RETURN IS A DEFAULT 0.

IF THE CSR YOU SELECT CAUSES A TRAP TO 4 THE PROGRAM WILL TYPE "THIS CSR DOES NOT EXIST".

THE CSR WILL BE READ AND DISPLAYED AS IN COMMAND 1.

THE PROGRAM WILL THEN ASK YOU FOR THE "CSR?" TO WHICH YOU MUST RESPOND WITH AN OCTAL NUMBER. NOTE TYPING JUST CARRIAGE RETURN IS A DEFAULT 0.

THE PROGRAM WILL THEN LOAD THE CSR AND READ IT AGAIN DISPLAYING ITS NEW CONTENTS.

1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365

2.4.4.7.4 FIELD SERVICE COMMAND 3 (EXAMINE MEMORY)

THIS COMMAND WILL ALLOW YOU TO EXAMINE ANY PHYSICAL ADDRESS AND DOES THE NECESSARY MEMORY MANAGEMENT MAPPING FOR YOU.

THE PROGRAM WILL ASK YOU FOR THE 'PHYSICAL ADDRESS (0-17757776)' TO WHICH YOU MUST RESPOND WITH AN OCTAL NUMBER.

IF THE ADDRESS ACCESS CAUSES A TRAP TO 4 THE PROGRAM WILL TYPE 'TIMEOUT TRAP'. IF THE ADDRESS ACCESS CAUSES A TRAP TO 114 THE PROGRAM WILL TYPE 'PARITY ABORT'.

THE CONTENTS OF YOUR PHYSICAL ADDRESS WILL BE TYPED.

2.4.4.7.5 FIELD SERVICE COMMAND 4 (MODIFY MEMORY)

THIS COMMAND ALLOWS YOU TO MODIFY ANY PHYSICAL ADDRESS AND DOES THE NECESSARY MEMORY MANAGEMENT MAPPING FOR YOU.

THE PROGRAM WILL ASK YOU FOR THE 'PHYSICAL ADDRESS (0-17757776)' TO WHICH YOU MUST RESPOND WITH AN OCTAL NUMBER.

IF THE ADDRESS ACCESS CAUSES A TRAP TO 4 THE PROGRAM WILL TYPE 'TIMEOUT TRAP'. IF THE ADDRESS ACCESS CAUSES A TRAP TO 114 THE PROGRAM WILL TYPE 'PARITY ABORT'.

THE PROGRAM WILL TYPE 'OLD DATA WAS' AND THE CONTENTS OF YOUR PHYSICAL ADDRESS.

THE PROGRAM WILL THEN TYPE 'INPUT NEW DATA' TO WHICH YOU MUST RESPOND WITH AN OCTAL NUMBER. NOTE TYPING JUST CARRIAGE RETURN IS A DEFAULT 0.

THE PROGRAM WILL ATTEMPT TO WRITE THIS NEW DATA INTO YOUR PHYSICAL ADDRESS AFTER WHICH IT WILL READ IT AGAIN AND TYPE 'DATA IS NOW' AND THE NEW CONTENTS OF YOUR PHYSICAL ADDRESS.

NOTE

IF YOU CAN'T CHANGE THE DATA, THAT WOULD INDICATE THAT YOU HAVE A DOUBLE BIT ERROR IN THAT DOUBLE WORD PAIR.

1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416

2.4.4.7.6 FIELD SERVICE COMMAND 5 (SELECT BANK TEST)

THIS COMMAND ALLOWS YOU TO RUN ANY BANK WITH ANY PATTERN FOREVER.

THE PROGRAM WILL ASK YOU 'BANK(0-167)' TO WHICH YOU MUST RESPOND WITH AN OCTAL NUMBER. IF THE BANK IS NOT ACCESSIBLE. THE PROGRAM WILL TYPE 'BANK NOT ACCESSIBLE' AND ASK QUESTION OVER.

THE PROGRAM WILL THEN ASK 'TEST (0-47)' TO WHICH YOU MUST RESPOND WITH AN OCTAL NUMBER.

NOTE

ANY PATTERN CAN BE RUN INCLUDING THOSE THAT ARE NOT PART OF THE APT E-TABLE DEFAULTS (REFERENCE SECTION 6.2.1). IF YOU SELECT PATTERN 0, THE PROGRAM WILL ASK 'TEST 0 DATA IS?' TO WHICH YOU MUST RESPOND WITH AN OCTAL NUMBER.

IF THE BANK YOU SELECTED REQUIRES RELOCATION THE PROGRAM WILL TYPE 'BANK REQUIRES RELOCATION' AND EXIT THIS COMMAND. NOTE NORMALLY THIS IS TRUE FOR BANK 0.

THE PROGRAM WILL THEN ARM THE CONSOLE KEYBOARD FOR INTERRUPTS AND TYPE 'TO ESCAPE TYPE ANY KEY!'.

THE TEST PATTERN WILL BE ENTERED AND RUN UNTIL A CONSOLE KEY IS DEPRESSED TO ESCAPE THIS LOOP.

2.4.4.7.7 FIELD SERVICE COMMAND 6 (TYPE CONFIGURATION MAP)

THIS COMMAND TYPES THE CONFIGURATION MAP.

THIS IS USEFUL AFTER A LONG RUN (OVERNIGHT) TO SEE ALL THE BANKS THAT ARE MARKED AS BAD. (ESPECIALLY IF YOUR CONSOLE IS A VIDEO TERMINAL).

FOR A DETAILED EXPLANATION OF THE MAP REFERENCE SECTION 7.3.

1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470

2.4.4.7.8 FIELD SERVICE COMMAND 7 (SOB-A-LONG TEST)

THIS COMMAND ALLOWS EXECUTION OF THE SOB-A-LONG TEST ON ALL NON-PROTECTED BANKS REFERENCE SECTION 6.2.2.26. OPERATION IS IDENTICAL TO COMMAND 5 EXCEPT THAT NO PATTERN OR BANK IS ENTERED AND EACH PASS CAUSES A BELL.

2.4.4.7.9 FIELD SERVICE COMMAND 8 (ERROR SUMMARY)

THIS COMMAND TYPES OUT THE NUMBER OF PASSES AND THE TOTAL NUMBER OF ERRORS. IF THERE WERE ANY ERRORS IT WILL TYPE OUT THE BANKS AND THE NUMBER OF ERRORS PER BANK UP TO 255 DECIMAL.

THIS BECOMES USEFUL AFTER LONG RUNS (ALL NIGHT) ON SYSTEMS WITH A VIDEO CONSOLE TERMINAL.

2.4.4.7.10 FIELD SERVICE COMMAND 9 (REFRESH TEST)

THIS COMMAND ALLOWS EXECUTION OF THE REFRESH TEST ON ALL NON-PROTECTED BANKS REFERENCE SECTION 6.2.2.19. OPERATION IS IDENTICAL TO COMMAND 5 EXCEPT THAT NO PATTERN OR BANK IS ENTERED AND EACH PASS CAUSES A BELL.

2.4.4.7.11 FIELD SERVICE COMMAND 10 (SET FILL COUNT)

THIS COMMAND ALLOWS SETTING OF THE TERMINAL FILL COUNT (NECESSARY FOR LA30'S, ASR33'S, AND VT05'S). IT IS NORMALLY SET TO ZERO FOR LA36'S, VT52'S, VT100'S, ETC.

2.4.4.7.12 FIELD SERVICE COMMAND 11 (ENTER KAMIKAZE MODE)

THIS COMMAND ALLOWS YOU TO RUN PATTERNS THAT ARE NORMALLY NOT EXECUTED UNLESS UNDER APT OR ACT. THEY ARE USUALLY VERY TIME CONSUMING AND CAN RESULT IN FAILURES THAT ARE FATAL TO THE PROGRAM. IN EFFECT YOU ARE TRYING TO FIND A HARDWARE FAILURE REGARDLESS OF THE CONSEQUENCES. NOTE THAT MOST CRASHES DO NOT WIPE OUT THE DISPLAY INFORMATION WHICH IS TELLING YOU WHAT THE PROGRAM WAS DOING JUST PRIOR TO FAILURE. THERE ARE TWO WAYS TO DIE HERE - IMPATIENCE AND CRASHES.

1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527

2.4.4.7.13 FIELD SERVICE COMMAND 12 (EXIT KAMIKAZE MODE)
RETURN TO THE DEFAULT MODE OF TESTING (UNDO COMMAND 12).

2.4.4.7.14 FIELD SERVICE COMMAND 13 (TURN CACHE OFF)
THIS CHANGES THE CACHE CONSTANT TO BYPASS CACHE (REFERENCE SECTION 2.4.3.1).

2.4.4.8.15 FIELD SERVICE COMMAND 14 (TURN CACHE ON)
THIS CHANGES THE CACHE CONSTANT TO USE CACHE (REFERENCE SECTION 2.4.3.1).

2.4.4.7.16 FIELD SERVICE COMMAND 15 (TEST ONLY SELECTED BANKS)
THIS COMMAND ALLOWS YOU TO CENTER THE TEST EFFORT ON ONLY THOSE BANKS THAT YOU ARE TROUBLESHOOTING. YOU MAY ALSO TEST BANKS THAT REQUIRE RELOCATION AND WERE INACCESSABLE VIA COMMAND 5.

2.4.4.7.17 FIELD SERVICE COMMAND 16 (RESUME TESTING ALL BANKS)
RETURN TO THE DEFAULT MODE OF TESTING (UNDO COMMAND 15).

2.4.4.7.18 FIELD SERVICE COMMAND 17 (RESUME TESTING ALL BANKS)
ENABLE "TRACE". AFTER EXITING FIELD SERVICE MODE, THE PROGRAM WILL TYPE OUT THE BANK AND PATTERN NUMBERS AS EACH PATTERN IS RUN.

2.4.4.7.19 FIELD SERVICE COMMAND 18 (RESUME TESTING ALL BANKS)
DISABLE "TRACE". (UNDO COMMAND 17).

1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585

2.5 EXECUTION TIMES

2.5.1 TYPICAL (SYSTEM) -

EXECUTION TIME DEPENDS ON MANY VARIABLES; HOWEVER HERE ARE SOME MEASURED TIMES ON AN 11/44 WITH CACHE:

128K WORDS OF MS11-L MEMORY
 NORMAL PASS 0 MIN 50 SEC
 QUICK VERIFY 0 MIN 50 SEC
 KAMIKAZE MODE 10 MIN 5 SEC
 KAMIKAZE QV 10 MIN 5 SEC

128K WORDS OF MS11-M MEMORY (NON-INTERLEAVED)
 NORMAL PASS 2 MIN 25 SEC
 QUICK VERIFY 1 MIN 0 SEC
 KAMIKAZE MODE 11 MIN 0 SEC
 KAMIKAZE QV 10 MIN 30 SEC

128K WORDS OF MS11-M MEMORY (INTERLEAVED)
 NORMAL PASS 3 MIN 55 SEC
 QUICK VERIFY 1 MIN 50 SEC
 KAMIKAZE MODE 22 MIN 0 SEC
 KAMIKAZE QV 20 MIN 5 SEC

512K WORDS OF MS11-P MEMORY
 NORMAL PASS 4 MIN 50 SEC
 QUICK VERIFY 4 MIN 25 SEC
 KAMIKAZE MODE 44 MIN 0 SEC
 KAMIKAZE QV 39 MIN 0 SEC

2.5.2 CALCULATIONS (SYSTEM)

NORMAL PASS
 ADD 18 SEC PER 16K BANK OF NON-INTEREAVED MS11-M
 ADD 15 SEC PER 16K BANK OF INTERLEAVED MS11-M
 ADD 6 SEC PER 16K BANK OF MS11-L
 ADD 37 SEC PER 64K BANK OF MS11-P

QUICK VERIFY PASS
 ADD 8 SEC PER 16K BANK OF NON-INTERLEAVED MS11-M
 ADD 7 SEC PER 16K BANK OF INTERLEAVED MS11-M
 ADD 6 SEC PER 16K BANK OF MS11-L
 ADD 33 SEC PER 64K BANK OF MS11-P

KAMIKAZE MODE
 ADD 10 MIN. PER 128K WORDS FOR APPROXIMATE PASS TIMES.

1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637

2.5.3 TYPICAL (TESTS)

TEST TIME	DESCRIPTION
-----	-----
MT0000 :<1 SEC	DATA PATTERN TEST
MT0001 :<1 SEC	ADDRESS TEST
MT0002 :<1 SEC	COMPLEMENT ADDRESS TEST
MT0003 : 1 SEC	3 XOR 9 WORST CASE NOISE TEST
MT0004 : 1 SEC	ROTATING ZEROS TEST
MT0005 : 1 SEC	ROTATING ONES TEST
MT0006 :<1 SEC	INITIAL DATA TEST
MT0007 :<1 SEC	ADDRESS BIT TEST
MT0010 :<1 SEC	BYTE ADDRESSING TEST
MT0011 :<2 SEC	CREATE SINGLE BIT ERROR TEST
MT0012 :<1 SEC	WRITE BYTE CLEARS SBE TEST
MT0013 : 1 SEC	CREATE DOUBLE BIT ERROR TEST
MT0014 : 1 SEC	BASIC DOUBLE BIT ERROR TEST
MT0015 : 1 SEC	WRITE INHIBIT OF BYTE WITH DBE
MT0016 :<1 SEC	WRITE INHIBIT OF WORD WITH DBE
MT0017 :<1 SEC	HOLDING 1'S 0'S TEST
MT0020 : 1 SEC	SYNDROMES TO CSR ON SINGLE BIT ERROR TEST
MT0021 : 1 SEC	MARCHING 0'S 1'S TEST
MT0022 :10 SEC	REFRESH TEST
MT0023 :10 SEC	SHIFTING DIAGONAL TEST
MT0024 :20 SEC	FAST GALLOPING PATTERN TEST
MT0025 :<1 SEC	INTERRUPT ENABLE TEST
MT0026 :<1 SEC	RANDOM DATA TEST
MT0027 : 1 SEC	UNIQUE BANK TEST
MT0030 : 1 SEC	FLUSH OUT DBE'S TEST
MT0031 : 3 SEC	SOB-A-LONG TEST
MT0032 :<1 SEC	WRITE RECOVERY TEST
MT0033 :35 SEC	BRANCH GOBBLE TEST
MT0034 :<1 SEC	SOFT ERROR TEST
MT0035 :<1 SEC	WORST CASE PARITY TEST
MT0036 : 1 SEC	CORRECTION CODE TEST
MT0037 :<1 SEC	CHECK ECC DISABLE TEST
MT0041 : 1 SEC	ADDRESS TO CSR ON DBC TEST
MT0042 :<1 SEC	EXTENDED ADDRESS TO CSR ON ERROR TEST
MT0043 :<1 SEC	WRITE BYTE TEST
MT0044 : 5 SEC	SHIFTING CHECKBITS THROUGH CSR TEST
MT0045 :<1 SEC	SYNDROME BITS TO THE CSR ON A DBE TEST
MT0046 : 1 SEC	CHECK SINGLE BIT ERRORS WITH ECC DISABLED TEST
MT0047 :<1 SEC	NO CSR UPDATE WITH EXISTING DBE TEST

1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691

3.0 ERROR INFORMATION

3.1 ERROR REPORTING

MOST ERRORS ARE REPORTED USING THE EMT TRAP AND HANDLER PROVIDED BY SYSMAC.SML. MOST ERRORS WILL BE OF THE 'MEMORY DATA ERROR' TYPE WHICH WILL BE DESCRIBED HERE. MEMORY DATA ERRORS WILL ALSO CAUSE THE BANK TO BE MARKED AS BAD IN THE CONFIGURATION TABLE.

OTHER ERRORS ARE BEST EXPLAINED BY REFERENCING THE SPECIFIC TYPEOUT AND IF NECESSARY THE PROGRAM LISTING.

EXAMPLE 1:

MEMORY DATA ERROR

PC	BANK	VADD	PADD	GOOD	BAD	XOR	CSR	MTYP	INT	PAT
022132	37	060006	03700006	000000	000100	000100	0		-	06
022132	37	060006	03700006	000000	000100	000100	0		-	06
022132	37	060006	03700006	000000	000100	000100	0		-	06
022132	37	060006	03700006	000000	000100	000100	0		-	06

WHILE TESTING BANK 37 AT VIRTUAL ADDRESS 60006 (VIRTUAL ADDRESSES ARE ALWAYS BETWEEN 60000 AND 157776 FOR MAPPING PURPOSES), PHYSICAL ADDRESS 3700006 (THAT'S BANK 37 PHYSICAL 6 WITHIN THE BANK) WITH PATTERN 6 (INITIAL DATA TEST), THE GOOD DATA EXPECTED WAS 0 BUT THE DATA ACTUALLY READ (BAD) WAS 100, THE EXCLUSIVE OR AT GOOD BAD YIELDS 100 WHICH INDICATES ONLY FAILING BIT(S) (BIT 6). IT IS AN MS11-P (ECC) MEMORY AND IT'S NOT INTERLEAVED. THE CSR IS LOCATED AT 172000.

EXAMPLE 2:

MEMORY DATA ERROR

PC	BANK	VADD	PADD	GOOD	BAD	XOR	CSR	MTYP?	INT	PAT
022132	35	060000	03500000	000000	000001	000001	0	M	1	06
022132	35	060002	03500002	000000	000100	000100	0	M	1	06
022132	35	060006	03500006	000000	000100	000100	0	M	1	06

WHILE TESTING BANK 35, VIRTUAL ADDRESS 60000, PHYSICAL ADDRESS 3700000 WITH PATTERN 6 (INITIAL DATA TEST), THE GOOD DATA EXPECTED WAS 0 BUT THE DATA ACTUALLY READ (BAD) WAS 1, THE EXCLUSIVE OR AT GOOD BAD YIELDS 1 WHICH INDICATES ONLY FAILING BIT(S) (BIT 0). IT IS AN MS11-M (ECC) MEMORY AND IT'S INTERLEAVED; SO SINCE ADDRESS BIT 1 WAS NOT ASSERTED, THE CSR IS LOCATED AT 172000.

WHILE ALSO IN BANK 35, VIRTUAL ADDRESSES 60002 AND 60006 WERE EXPECTED TO HAVE 0, BUT THE DATA READ WAS 100, THE EXCLUSIVE OR OF GOOD BAD YIELDS 100 WHICH INDICATES ONE FAILING BIT (BIT 6). SINCE IT IS INTERLEAVED MS11-M MEMORY, AND ADDRESS BIT 1 IS ASSERTED, THE CSR IS LOCATED AT 172102 (CSR NUMBER 1 UNDER THE INT COLUMN)

NOTE

SUBSEQUENT ERRORS OF THE SAME TEST DO NOT TYPE A NEW HEADING.

1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743

3.2 ERROR ABBREVIATIONS

THE FOLLOWING IS A LIST OF ALL ABBREVIATIONS USED IN ERROR REPORTS.

# OF ERRORS	NUMBER OF ERRORS THAT WERE DETECTED.
1ST ADD	FIRST ADDRESS THAT FAILED.
ARRAY	THE ARRAY NUMBER THAT WAS LOCKED UP IN THE MS11-M CSR.
APT#	THE # OF CPU'S APT EXPECTS ON THE SYSTEM.
APTCORE	APT CORE SIZE.
APTMOS	APT MOS SIZE.
BAD	BAD DATA.
BAD-WD1	BAD WORD #1 OF A DOUBLE WORD DATA VALUE.
BAD-WD2	BAD WORD #2 OF A DOUBLE WORD DATA VALUE.
BAD-CHK	BAD CHECK CODE BITS.
BANK	THE BANK NUMBER. BANKS ARE 16K WORDS LONG.
BD-CC	BAD CHECK CODE BITS.
CHKBITS	THE 7 BIT VALUE OF THE CHECK CODE BITS.
CONTRL	THE CACHE CONTROL REGISTER.
CPUERR	CPU ERROR REGISTER.
CSR	CONTROL AND STATUS REGISTER.
CSRNO	CSR NUMBER (0-F HEXIDECIMAL).
DATARG	THE CACHE DATA REGISTER.
DBE	DOUBLE BIT ERROR (UNCORRECTABLE ERROR).
DEV ADD	DEVICE ADDRESS.
ECC	ERROR CORRECTABLE CODE.
GD-CC	GOOD CHECK CODE BITS.
GD-CHK	GOOD CHECK CODE BITS.
GD-WD1	GOOD WORD #1 OF A DOUBLE WORD DATA VALUE.
GD-WD2	GOOD WORD #2 OF A DOUBLE WORD DATA VALUE.
GOOD	GOOD DATA.
INT	INTERLEAVED (ADDRESS BIT 1 ASSERTED) CSR NUMBER.
LSIZE	MS11-L SIZE.
MEMERR	MEMORY ERROR REGISTER.
MMR0	MEMORY MANAGEMENT REGISTER #0.
MMR1	MEMORY MANAGEMENT REGISTER #1.
MMR2	MEMORY MANAGEMENT REGISTER #2.
MMR3	MEMORY MANAGEMENT REGISTER #3.
MSIZE	MS11-M SIZE.
MTYP	MEMORY TYPE (MS11-L, MS11-M, OR MS11-P).
PADD	PHYSICAL ADDRESS (ASSERTED BY THE PROGRAM AFTER MAPPING).
PAT	PATTERN NUMBER.
PC	PROGRAM COUNTER AT THE TIME THE ERROR OCCURRED.
SBE	SINGLE BIT ERROR (CORRECTABLE ERROR).
VADD	VIRTUAL ADDRESS (ASSERTED BY THE PROGRAM BEFORE MAPPING).
WROTE1	THE DATA THAT WAS WRITTEN INTO THE 1ST HALF OF A DOUBLE WORD.
WROTE2	THE DATA THAT WAS WRITTEN INTO THE 2ND HALF OF A DOUBLE WORD.
XOR	EXCLUSIVE OR OF THE GOOD AND BAD DATA. SHOWS THE BAD BITS.
AUT	ADDRESS UNDER TEST

1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768

3.3 ERROR HALTS

THERE ARE SEVERAL HALTS IN THE PROGRAM.

ALL UNUSED TRAP VECTORS CONTAIN A TRAP CATCHER (.WORD .+2,HALT).

AN UNDEFINED TRAP INSTRUCTION HALTS AT SYMBOLIC LOCATION "\$HALT2".

THE APT DOWN LOAD SEQUENCE WILL HALT AT SYMBOLIC LOCATION "APTHLT".

HALT ON ERROR OPTION (SW15 SET) AT SYMBOLIC LOCATION "\$HALT".

HALT PROGRAM (SW8 SET) AT SYMBOLIC LOCATION "\$EXHALT".

POWER FAIL WILL NORMALLY HALT AT THE END OF THE SHUT DOWN SEQUENCE (SYMBOLIC LOCATION "\$DOWN").

POWER FAIL HAS A FATAL HALT AT SYMBOLIC LOCATION "\$ILLUP" WHICH CAN BE CAUSED BY POWER UP OCCURRING BEFORE POWER DOWN SEQUENCE COMPLETED OR BY POWER DOWN BEFORE A POWER UP SEQUENCE IS COMPLETED.

1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821

4.0 PROGRESS REPORTS

PASS COMPLETE TYPEOUTS AS FOLLOWS:

END PASS	#	0
END PASS	#	1
END PASS	#QV	2

NOTE

PASS 2 WAS FLAGGED AS A QUICK VERIFY PASS. (BECAUSE OF A CHANGE IN SW5)

TO OBTAIN PROGRESS REPORTS WHILE EXECUTING, TYPING A CONTROL "T" WILL PRINT OUT THE INFORMATION ENCODED IN THE DISPLAY REGISTER.

EXAMPLE:

BANK= 2 TEST= 34

REFERENCE SECTION 2.4.4.7.18 FOR MORE INFORMATION ON TRACING.

5.0 CSR INFORMATION TABLES

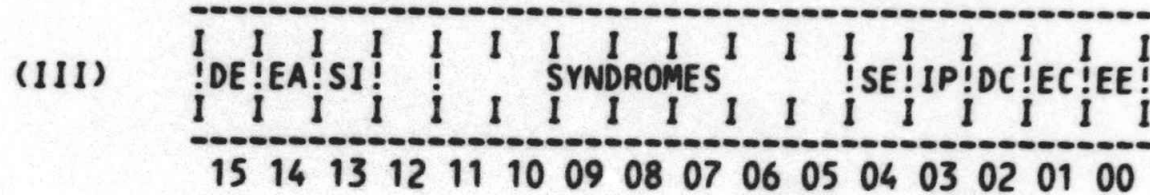
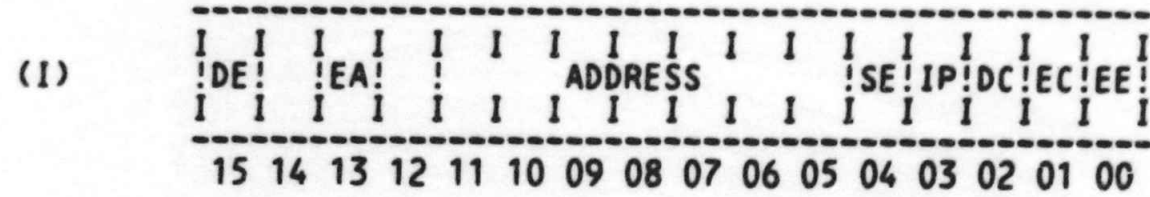
THE FOLLOWING IS A PICTURE VIEW OF THE CURRENT CONTROL STATUS REGISTERS WHICH CAN BE TESTED BY THIS PROGRAM. IT SHOWS BIT ASSIGNMENTS AND DEFINITIONS TO PROVIDE A HANDY REFERENCE, AND SHOWS THE SIMILARITIES AND DIFFERENCES BETWEEN EACH ONE:

NOTE

ALL UNUSED BITS IN EACH CSR ARE EQUAL TO ZERO.

1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876

5.1 MS11-P CSR



BIT ASSIGNMENTS ARE DEFINED AS FOLLOWS:

BIT15 UNCORRECTABLE ERROR
ON A READ TO MEMORY (ECC DISABLE BIT = 0), THIS BIT IS SET IF A DOUBLE ERROR OCCURS. THE ERROR ADDRESS IS STORED IN THE CSR. SETTING THIS BIT ALSO TURNS ON A RED LED AT THE REAR OF THE CARD FOR A VISUAL INDICATION. THIS BIT IS ALSO SET IN ECC DISABLE MODE IF A SERR OR DERR OCCURS.

BIT14 EUB ERROR ADDRESS
WITH BIT 14 = 1, A READ TO THE CSR WILL FETCH ADDRESS A21 THROUGH A18. WHEN BIT 14 = 1, DIAGNOSTIC DATA MAY NOT BE LOADED INTO THE SYNDROME REGISTER.

1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931

BIT13 SET INHIBIT MODE
WHEN THIS BIT IS SET TO
A '1', IT ENABLES THE INHIBIT
MODE POINTER TO INHIBIT
EITHER THE FIRST OR SECOND
16K FROM EVER GOING INTO THE
DIAG CHECK OR ECC DISABLE
MODE.

BITS05-10 CHECK BIT STORAGE (CSR II)
CHECK BIT STORAGE (DIAG CK
BIT 2 = 1)
WHEN IN THE DIAGNOSTIC CHECK
MODE THESE BITS ARE USED TO STORE
THE CHECK BITS TO BE WRITTEN
INTO MEMORY OR THE CHECK BITS
READ FROM MEMORY. IF A DOUBLE
ERROR OR SINGLE ERROR OCCURS
WHEN IN THE DIAGNOSTIC CHECK
MODE AND ECC DISABLE BIT 1 = 0,
THEN THE CHECK BITS ARE STORED
IN THE CSR TOGETHER WITH
THE DOUBLE OR SINGLE ERROR
BIT. THESE BITS ARE WRITEABLE
IN DIAGNOSTIC MODE. A '1'
IS STORED IN BIT 11 IF CSR
02, CSR 13, AND CSR 14 ARE
SET TO INDICATE THAT THE
MEMORY UNDER TEST IS A MS11-P.

BITS05-11 UNISUS ADDRESS STORAGE (CSR I)
(DIAG CK BITS 2 = 0, ECC
DISABLE BIT 1 = 0)

IF A DOUBLE OR SINGLE ERROR
OCCURS ON A READ CYCLE, THEN
ADDRESS BITS ALL THROUGH
A17 ARE STORED IN THESE BITS
THESE BITS ARE READ ONLY ON
THE CONDITION THAT SERR (CSR 4)
OR DERR (CSR 15) IS SET BUT
CSR 14 IS NOT SET.

EUB ADDRESS STORAGE (DIAG CK
BIT 2 = 0), ECC DISABLE BIT
1 = 0 OR 1).

IF A DOUBLE OR SINGLE ERROR
OCCURS ON A READ CYCLE,
ADDRESS BITS A17 THROUGH
A11 ARE STORED IN CSR BITS
11 THROUGH 5 AND ADDRESS
BITS A21 THROUGH A18 ARE

1933
 1934
 1935
 1936
 1937
 1938
 1939
 1940
 1941
 1942
 1943
 1944
 1945
 1946
 1947
 1948
 1949
 1950
 1951
 1952
 1953
 1954
 1955
 1956
 1957
 1958
 1959
 1960
 1961
 1962
 1963
 1964
 1965
 1966
 1967
 1968
 1969
 1970
 1971
 1972
 1973
 1974
 1975
 1976
 1977
 1978
 1979
 1980
 1981
 1982
 1983
 1984
 1985
 1986

STORED IN A BACKUP REGISTER.
 THE EUB ERROR ADDRESS
 RETRIEVAL BIT (CSR 14) IS
 USED TO OBTAIN THE TOTAL
 ERROR ADDRESS AS FOLLOWS:

WITH CSR BIT 14 = 0 A READ TO
 THE CSR WILL OBTAIN A17
 THROUGH A11 FROM CSR BITS 11
 THROUGH 5.

CSR BIT 14 CAN THEN BE SET
 TO A '1' AND A READ TO THE
 CSR WILL THEN READ A21
 THROUGH A18 FROM CSR BITS 8
 THROUGH 5 AND 0'S FROM CSR
 BITS 11 THROUGH 9.

ADDRESS BITS A21 THROUGH
 A11 ARE OBTAINED
 TO LOCATE THE DOUBLE
 ERROR TO A 1K SEGMENT OF
 MEMORY.

THE EUB ADDRESS A21
 THROUGH A18 IS READ ONLY
 WHENEVER CSR 14 = 1.

BIT05-10 SYNDROME STORAGE (CSR III)
 IF A DOUBLE OR SINGLE ERROR
 OCCURS ON A READ OR WRITE
 BYTE CYCLE, AND IF CSR BIT
 2 IS SET TO A '0' SYNDROME
 BITS X, 0, 1, 2, 4 AND 8
 AND STORED IN CSR BITS 5
 THROUGH 10. TO READ THE
 SYNDROME BITS FROM CSR, BIT
 YOU MUST READ THE ERROR
 ADDRESS, THEN SET 2 OF
 THE CSR MUST BE SET TO
 A '1' (DIAGNOSTIC MODE) AND
 THE CSR READ AGAIN. THIS OPERATION
 WILL ALLOW SYNDROME BITS
 FOR A SINGLE OR DOUBLE
 FAILURE TO BE READ INSTEAD
 OF THE ADDRESS BITS NORMALLY
 READ WHEN CSR 02 IS SET TO '0'.

BIT04 SINGLE ERROR
 IF ON A READ TO MEMORY A
 SBE OCCURS, THE ERROR
 ADDRESS A21-A11 AND
 THE ERROR SYNDROMES WILL

1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040

BE LOGGED IN CSR BITS 5-11 UNLESS THE UNCORRECTABLE ERROR CSR 15 IS SET. THE ERROR ADDRESS WILL BE LOGGED UNCONDITIONALLY IN THE ECC DISABLE MODE. THIS BIT IS NOT SET IF INHIBIT MODE (BIT 13 = 1) IS SET AND DIAGNOSTIC MODE (BIT 02 = 1) IS SET.

BIT03 INHIBIT MODE POINTER THIS BIT WORKS IN CONJUNCTION WITH THE SET INHIBIT MODE (BIT 13). WHEN BIT 13 IS SET TO A 1, A 16K PORTION OF MEMORY IS INHIBITED FROM OPERATING IN THE ECC DISABLE MODE OR DIAGNOSTIC CHECK MODE.

THE INHIBIT MODE POINTER INDICATES WHICH 16K IS BEING INHIBITED, I.E., BIT 3 = 0 THE FIRST 16K OF MEMORY IS INHIBITED, BIT 3 = 1, THE SECOND 16K OF MEMORY IS INHIBITED.

WITH BIT 13 SET TO A 0, BIT 3 BECOMES INOPERATIVE.

BIT03, IN CONJUNCTION WITH BIT 13, THEREFORE ALLOWS A 16K CHUNK OF MEMORY TO ALWAYS HAVE ECC COVERAGE. THE SYSTEMS DIAGNOSTIC CAN THEREFORE RESIDE IN THIS PROTECTED PORTION OF MEMORY AND CAN DISABLE ECC AND/OR RUN THE DIAGNOSTIC CHECK MODE IN THE REST OF MEMORY WITHOUT ITSELF BECOMING VULNERABLE TO SINGLE ERRORS. THIS BIT IS A READ/WRITE BIT RESET BY POWER UP AND BUS INIT.

BIT02 DIAGNOSTIC CHECK MODE THIS MODE ALLOWS A MEANS OF FORCING A SINGLE OR DOUBLE ERROR IN A DESIRED LOCATION. IT ALSO PROVIDES A MEANS OF EXAMINING THE CHECK BITS AND THE SYNDROME IN A GIVEN LOCATION.

2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094

THE CHECK BITS DESIRED FOR A GIVEN DATA PATTERN ARE WRITTEN INTO BITS 5 THROUGH 11 OF THE CSR. A WORD OR WRITE BYTE MEMORY WILL WRITE THE CHECK BITS FROM THE CSR TO THE MOS ARRAY (CSR 2 = 1) INSTEAD OF THE CHECK BITS GENERATED ON THE DATA TO BE WRITTEN. SINGLE ERRORS ON THE READ PORTION OF THE DATOB CYCLE ARE CORRECTED.

A READ TO THE MEMORY WILL READ THE CHECK BITS STORED IN MEMORY AND CLOCK THEM INTO THE CSR.

IF A DOUBLE ERROR OR SINGLE ERROR OCCURS THE DERR OR SERR BIT IN THE CSR IS SET AND THE ERROR SYNDROME BITS READ FROM ECC ARE STORED IN CSR BITS 10-5 AS WELL AS THE ADDRESS BITS. IN DIAGNOSTIC CHECK MODE THE ERROR SYNDROME BITS WILL BE READ WHEN CSR BITS 10-5 ARE READ.

THIS BIT IS A READ/WRITE BIT AND IS RESET ON POWER UP AND BUS INIT.

BIT01 DISABLE CORRECTION MODE IF THIS BIT IS SET, NO SINGLE ERRORS WILL BE CORRECTED. A SINGLE ERROR WILL SET CSR 4 AND CSR 15 OR A DOUBLE ERROR WILL SET CSR 15 AND ASSERT BUS PBL IF CSR 00 IS ASSERTED. THE 1K BLOCK OF ADDRESS WHERE THE ERROR OCCURS WILL ALSO BE STORED IN THE CSR. THE PRIORITY OF A SERR AND DERR WILL BE THE SAME, I.E., THE LAST ERROR INFORMATION WILL ALWAYS BE STORED UNLESS A DERR PRECEDES A SERR. IF A DOUBLE ERROR OCCURS DURING A WRITE BYTE CYCLE, THE WRITE PORTION OF THE CYCLE WILL NOT BE ABORTED. THE CHECK BITS WRITTEN WILL

2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119

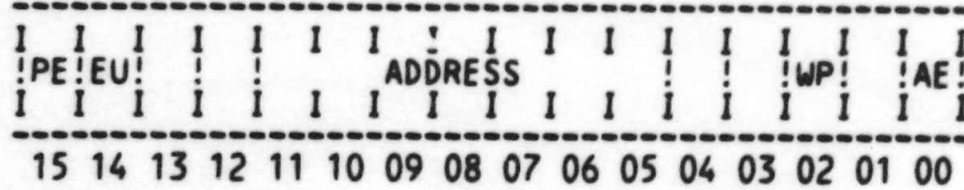
HAVE BEEN GENERATED ON THE DATA WRITTEN. THIS MEANS THAT IF A SINGLE OR DOUBLE ERROR EXISTED IN THE LOCATION ACCESSED, IT WOULD BE CLEARED (UNLESS THE ERRORS WERE HARD).

THIS BIT IS A DIAGNOSTIC AID TO ALLOW WRITING AND READING DATA FROM MEMORY WITHOUT INTERFERENCE FROM THE ERROR CORRECTION LOGIC.

BIT00 UNCORRECTABLE ERROR INDICATION ENABLE
IF A DOUBLE ERROR OCCURS WITH ECC ENABLED OR A SINGLE ERROR OR DOUBLE ERROR WITH ECC DISABLED, ON A READ CYCLE TO THE MEMORY AND THIS BIT IS SET, THEN BUS PBL WILL BE ASSERTED.

2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175

5.2 MS11-L CSR



BIT ASSIGNMENTS ARE DEFINED AS FOLLOWS:

BIT15 PARITY ERROR

BIT14 EUB ERROR RETRIEVAL IF THE MEMORY IS ON AN EXTENDED UNIBUS, WHEN BIT14 IS ZERO, THE LOW ORDER FAILING ADDRESSES ARE AVAILABLE (BITS 11-17); WHEN BIT14 IS ONE, THE HIGH ORDER FAILING ADDRESSES ARE AVAILABLE (BITS 18-21 OF ADDRESS). IF THE MEMORY IS ON A UNIBUS, A JUMPER DISABLES THIS BIT SO THAT IT IS READ ONLY, AND EQUAL TO ZERO.

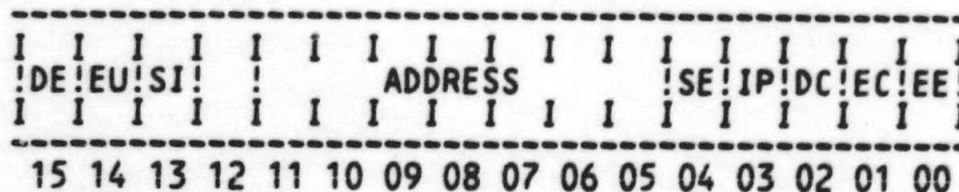
BITS 11-5 ERROR ADDRESS WITH BIT14 SET, THEY CONTAIN THE HIGH ORDER PARITY ERROR ADDRESS (BITS 21-18 OF ADDRESS); WITH BIT14 CLEARED, THEY CONTAIN THE LOW ORDER PARITY ERROR ADDRESS (BITS 17-11 OF ADDRESS).

BIT02 WRITE WRONG PARITY NORMAL PARITY (ODD) WHEN CLEAR; OTHER PARITY (EVEN) WHEN SET.

BIT00 ACTION ENABLE NO ACTION WHEN CLEAR; TRAP TO VECTOR 114 WHEN SET.

2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230

5.3 MS11-M CSR



BIT ASSIGNMENTS ARE DEFINED AS FOLLOWS:

BIT15 UNCORRECTABLE ERROR THIS BIT IS SET IF A DBE OCCURS, AND THE ERROR ADDRESS IS STORED IN THE CSR. THIS BIT IS ALSO SET IN THE ECC DISABLE MODE IF AN SBE OR DBE OCCURS.

BIT14 EUB ERROR RETRIEVAL IF THE MEMORY IS ON AN EXTENDED UNIBUS, WHEN BIT14 IS ZERO AND EITHER BIT4 OR BIT 15 IS A ONE, THE LOW ORDER FAILING ADDRESSES ARE AVAILABLE (BITS 11-17); WHEN BIT14 IS ONE, THE HIGH ORDER FAILING ADDRESSES ARE AVAILABLE (BITS 18-21 OF ADDRESS). IF THE MEMORY IS ON A UNIBUS, A JUMPER DISABLES THIS BIT SO THAT IT IS READ ONLY, AND EQUAL TO ZERO.

BIT13 SET INHIBIT MODE WHEN THIS BIT IS SET TO A 1, IT ENABLES THE INH MODE POINTER TO INHIBIT EITHER THE FIRST OR SECOND 16K FROM EVER GOING INTO THE DIAG. CHECK OR ECC DISABLE MODE. WHEN THIS BIT IS SET TO A 0, IT ALLOWS THE DIAG. CHECK MODE

2232
 2233
 2234
 2235
 2236
 2237
 2238
 2239
 2240
 2241
 2242
 2243
 2244
 2245
 2246
 2247
 2248
 2249
 2250
 2251
 2252
 2253
 2254
 2255
 2256
 2257
 2258
 2259
 2260
 2261
 2262
 2263
 2264
 2265
 2266
 2267
 2268
 2269
 2270
 2271
 2272
 2273
 2274
 2275
 2276
 2277
 2278
 2279
 2280
 2281
 2282
 2283
 2284
 2285

AND/OR ECC DISABLE
 MODE TO OPERATE OVER
 THE ENTIRE MEMORY ON
 THE BOARD.

BITS 11-5 ERROR
 ADDRESS WITH BIT02
 CLEARED AND BIT14 SET,
 THEY CONTAIN THE HIGH
 ORDER ERROR ADDRESS
 (BITS 21-18); WHEN
 BIT02 AND BIT14 ARE
 CLEARED, THEY CONTAIN
 THE LOW ORDER ERROR
 ADDRESS (BITS 17-11);
 WHEN BIT02 IS SET THEY
 CONTAINS CHECK BITS
 FOR ECC.

BIT04 SINGLE ERROR SET
 WHENEVER SINGLE ERROR
 OCCURS.

BIT03 INHIBIT MODE
 POINTER THE INHIBIT
 MODE POINTER WORKS IN
 CONJUNCTION WITH THE
 SET INHIBIT MODE BIT.
 WHEN BIT13 IS SET TO A
 1, A 16K PORTION OF
 MEMORY IS INHIBITTED
 FROM OPERATING IN THE
 ECC DISABLE MODE OR
 DIAGNOSTIC CHECK MODE.
 THE INHIBIT MODE
 POINTER INDICATES
 WHICH 16K IS BEING IN-
 HIBITED; E.G.-IF BIT3
 =1, THE SECOND 16K OF
 MEMORY IS INHIBITTED.
 WHEN BIT13 IS SET TO A
 0, BIT3 BECOMES
 INOPERATIVE.

BIT02 DIAGNOSTIC CHECK
 MODE WHEN SET ENABLES
 READ-WRITE OF CHECK
 BITS(SEE BITS 11-5).
 IF A DBE OCCURS IN
 THIS MODE (WITH BIT1=0
), BIT15 IS SET, BUT
 THE CHECK BITS READ

2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317

ARE STORED IN BITS
11-5, NOT THE DBE
ADDRESS BITS.

BIT01 DISABLE ERROR
CORRECTION WHEN SET NO
SINGLE ERROR CORRECTI-
ON TAKES PLACE. A
SINGLE BIT ERROR WILL
SET BIT04 AND BIT15
AND ASSERT BUS PBL L
IF BIT00 IS ASSERTED;
A DOUBLE ERROR WILL
SET SET BIT15 AND AS-
SERT BUS PBL L IF
BIT00 IS ASSERTED.
THE ERROR ADDRESS IS
STORED IN THE CSR, AND
CORRECT CHECK BITS ARE
GENERATED AND STORED
ON A WRITE.

BIT00 UNCORRECTABLE
ERROR ENABLE WHEN SET
ENABLES TRAP TO VECTOR
114 ON UNCORRECTABLE
ERROR.

2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363

6.0 SUB-TEST SUMMARIES

6.1 TESTS

TEST 1

BIT TEST OF ALL CSR'S/MATCH ALL CSR'S WITH MEMORY
(CSR ACCESS MAY CAUSE WRONG TYPE OF TRAPS)

TEST 2

TEST BANK 0 ACCESSES
FAILURES ARE FATAL.

TEST 3

TEST BANKS 1-167 (OCTAL) FOR ZEROS AND ONES
ERRORS ARE NOT TYPED HERE - ONLY LOGGED IN
THE CONFIGURATION TABLE

TEST 4

ECC INHIBIT MODE POINTER TEST

TEST 5

DIAGNOSTIC MODE DISPATCH ROUTINE
THIS TEST RUNS ALL THE PATTERNS IN THE
MODE SELECTED.

TEST 6

UNIQUE BANK TEST
PATTERN 27 IS RUN

2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410

6.2 TESTS

6.2.1 GENERAL TEST INFORMATION

ACTUAL TESTS ARE IDENTIFIED BY SYMBOLIC LOCATIONS 'MTPXY' WHERE X MAY BE ANY SUB PROGRAM INDICATOR (A,B,C,ETC) OR 0 AND Y WILL BE THE NUMBER OF THE TEST.

SETUP PROCEDURES FOR EACH TEST ARE IDENTIFIED BY SYMBOLIC LOCATIONS 'MTOOY' WHERE Y WILL BE THE NUMBER OF THE TEST.

TESTS RESIDE IN 4 SCRIPTS THAT ARE SCANNED FOR EXECUTION. SYMBOLIC LOCATION 'MKCSRT' IS A TABLE OF TESTS THAT CAN RUN ONCE FOR EACH ECC BANK (TWICE FOR INTERLEAVED MS11-M'S). SYMBOLIC LOCATION 'MKPAT' IS A TABLE OF TESTS THAT CAN RUN ON EACH BANK OF ECC MEMORY. SYMBOLIC LOCATION 'MJPAT' IS A TABLE OF TESTS THAT CAN RUN ON EACH BANK OF PARITY MEMORY. SYMBOLIC LOCATION 'FSPAT' IS A TABLE OF TESTS THAT CAN BE RUN IN FIELD SERVICE MODE (COMMAND 5).

THE 1ST 3 SCRIPTS ARE COMPLETELY CONTROLLED BY THE APT E-TABLE (EVEN IF NOT RUNNING UNDER APT). MODIFICATIONS TO THIS TABLE CAN BE MADE (1) WITH APT, OR (2) MANUALLY.

EXAMPLE E-TABLE SEGMENT:

:THE FOLLOWING LOCATIONS SPECIFY WHICH TESTS
:ARE TO BE RUN FOR PARTICULAR MEMORIES
:
:REFERENCE THE TABLE LISTED BELOW TO RELATE BITS TO TESTS.
:BIT0 SET WILL RUN THE FIRST ENTRY IN THE TABLE, BIT0 SET
:IN THE SECOND WORD WILL RUN THE 17TH ENTRY IN THE TABLE...
:
:NOTE**NULL TESTS DO NOT TAKE ANY TIME

		RECOMMENDED VALUE	
\$DDW0:	.WORD	177777	:ECC CSR TESTS 177777 TABLE = MKCSRT:
\$DDW1:	.WORD	177777	:ECC CSR TESTS 177777 TABLE = MKCSRT:
\$DDW2:	.WORD	177777	:ECC TESTS 103777 TABLE = MKPAT:
\$DDW3:	.WORD	177777	:ECC TESTS 177777 TABLE = MKPAT:
\$DDW4:	.WORD	177777	:PARITY TESTS 003777 TABLE = MJPAT:
\$DDW5:	.WORD	177777	:PARITY TESTS 177774 TABLE = MJPAT:

2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441

6.2.2 SPECIFIC TESTS

6.2.2.1 TEST 0 BASIC DATA TEST

WRITES READS R2 INTO A 16K BANK.

THIS IS USED FOR ZEROS AND ONES TESTING AND IN FIELD SERVICE MODE FOR ANY CONSOLE SELECTED TEST.

IT CAN EXECUTE OUT OF THE USER INSTRUCTION PAR'S.

NOTE

IT IS FREQUENTLY MODIFIED DYNAMICALLY SUCH THAT (1) IT RETURNS AFTER WRITING ONLY (THE 1ST NOP IS REPLACED WITH A RETURN) OR (2) IT ONLY COUNTS ERRORS (THE CODE PERRO2 AND NOP ARE REPLACED WITH INC @#PATERR).

2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460

6.2.2.2 TEST 1 ADDRESS TEST

WRITES READS AN INCREMENTING PATTERN EQUIVALENT TO PHYSICAL
ADDRESSED INTO A 16K BANK.

IT CAN EXECUTE OUT OF THE USER INSTRUCTION PAR'S.

2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475

6.2.2.3 TEST 2 COMPLEMENT ADDRESS TEST

WRITES THE COMPLEMENT OF THE PHYSICAL ADDRESS FROM HIGH ADDRESSES TO LOW (WRITE DOWN) AND READS FROM LOW ADDRESSES TO HIGH (READ UP).

THIS PROVIDES THE COMPLEMENT OF THE COVERAGE OF TEST 1 IN BOTH DATA PATTERN AND ADDRESSING SEQUENCE.

IT CAN EXECUTE OUT OF THE USER INSTRUCTION PAR'S.

2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494

6.2.2.4 TEST 3 3 XOR 9

WRITES READS A TEST THAT COMPLEMENTS AS ADDRESS BITS 3 AND 9 CHANGE.

THIS TEST IS RUN 4 TIMES (1) WITH ZEROS ONES, (2) WITH ONES ZEROS, (3) WITH 401 ONES, AND (4) WITH ONES 401. THE TEST OF THE 401 IS TO FORCE A THE PARITY BITS TO BECOME INVOLVED.

IT CAN EXECUTE OUT OF THE USER DATA PDR'S, THE USER INSTRUCTION PAR'S, THE KERNEL DATA PAR'S AND THE SUPERVISOR DATA PAR'S.

2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519

6.2.2.5 TEST 4 ROTATING ZEROS TEST

WRITES A BACKGROUND PATTERN OF ONES. ROTATES A ZERO CARRY BIT LEFT THRU EACH PAR OF BYTES (18 TIMES) AND THEN CHECKS THAT THE CARRY IS ZERO AND THE WORD (2 BYTES) IS STILL ALL ONES.

IT CAN EXECUTE OUT OF THE USER DATA PAR'S AND THE KERNEL DATA PAR'S.

NOTE

IT IS NOT UNCOMMON TO OBSERVE THE GOOD DATA EQUAL TO THE BAD DATA. THIS INDICATES THAT THE CARRY WAS NOT CLEAR AFTER 18 ROLB'S.

2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545

6.2.2.6 TEST 5 ROTATING ONES TEST

WRITES A BACKGROUND PATTERN OF ZEROS. ROTATES A ONE CARRY BIT LEFT THRU EACH PAIR OF BYTES (18 TIMES) AND THEN CHECKS THAT THE CARRY IS A ONE AND THE WORD (2 BYTES) IS STILL ALL ZEROS.

THIS PROVIDES THE COMPLEMENT OF THE COVERAGE OF TEST 4 IN DATA.

IT CAN EXECUTE OUT OF THE USER DATA PAR'S AND THE KERNEL DATA PAR'S.

NOTE

IT IS NOT UNCOMMON TO OBSERVE THE GOOD DATA EQUAL THE BAD DATA. THIS INDICATES THAT THE CARRY WAS NOT SET AFTER 18 ROLB'S.

2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560

6.2.2.7 TEST 6 INITIAL DATA TEST

WRITES READS A DOUBLE WORD FIRST WITH ALL BITS 0 EXCEPT 1 (FOR EVERY BIT POSITION), SECOND WITH ALL BITS 1 EXCEPT 1 (FOR EVERY BIT POSITION).

THIS IS A VERY QUICK CHECK OF THE DATA PATHS.

2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586

6.2.2.8 TEST 7 ADDRESS BIT TEST

WRITES A BACKGROUND OF ALL ZEROS.

READ ADDRESS 1 FOR A 0 BYTE.

COMPLEMENT ADDRESS 1.

READ ADDRESS 1 FOR A NON 0 BYTE.

FOR EACH ADDRESS BIT POSITION FROM BIT 1:

VIRTUAL (2, 4, 10, 20, 40, 100, 200, 400, 1000, 2000, 4000, 10000,
60000, 20000)

PHYSICAL (60002, 60004, 60010, 60020, 60040, 60100, 60200, 60400,
61000, 62000, 64000, 70000, 140000, 100000)

READ ADDRESS FOR A 0 WORD.

COMPLEMENT ADDRESS CONTENTS.

READ ADDRESS FOR A NON-ZERO WORD.

THIS IS A VERY QUICK CHECK OF THE ADDRESS BIT UNIQUENESS.

2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611

6.2.2.9 TEST 10 BYTE ADDRESSING TEST

WITH ECC DISABLED.
WRITES ALL ONES TO A DOUBLE WORD.
FOR EACH OF THE 4 BYTES IN THE DOUBLE WORD.
CLEARS ONE BYTE.
READS ALL 4 BYTES FROM DOUBLE WORD.
CHECKS FOR ONLY PROPER BYTE CLEAR.
ALL OTHER BYTES SET TO ALL ONES.

THIS IS ONLY DONE ON ONE DOUBLE WORD ADDRESS.

NOTE

THIS IS RUN FOR ECC MEMORY ONLY

2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667

6.2.2.10 TEST 11 SINGLE BIT ERROR TEST

1. CREATE A SINGLE BIT ERROR.
2. READ DATA UNCORRECTED (WITH ECC DISABLE).
3. CHECK THAT SBE AND DBE FLAGS ARE SET, AND THE ERROR ADDRESS IS LATCHED.
4. READ FIRST WORD OF DATA CORRECTED (WITH ECC ENABLED)
5. CHECK THAT THE CSR SINGLE BIT ERROR FLAG WAS SET, AND THE ERROR ADDRESS WAS LATCHED.
6. CLEAR SBE FLAG.
7. READ SECOND WORD OF DATA CORRECTED (WITH ECC ENABLED).
8. CHECK THAT THE CSR SINGLE BIT ERROR FLAG WAS SET.
9. DO (1-7) FOR A SINGLE BIT ERROR IN EACH OF 32 POSITIONS OF A DOUBLE WORD.
I.E. (32 TIMES)
10. IF NOT IN QUICK VERIFY MODE THEN DO (1-8) FOR DATA CONSISTING OF 1 BIT SET IN EACH OF 32 POSITIONS OF A DOUBLE WORD.
I.E. (32 X 32 = 1024 TIMES)
11. DO (1-9) FOR COMPLEMENTED DATA (1 BIT CLEAR IN EACH OF 32 POSITIONS OF A DOUBLE WORD).
I.E. (1024 X 2 = 2048 TIMES)
OR (32 X 2 = 64 TIMES (QUICK VERIFY))
12. DO (1-7) FOR A DOUBLE WORD EQUAL TO (000000,000000), AND ALL POSSIBLE SINGLE BIT ERROR COMBINATIONS FORCED INTO THE CHECK BITS (CSR BITS 5-11).
13. CLEAR ANY ERRORS OUT OF TEST LOCATIONS.

THIS INSURES THAT ALL SINGLE BIT ERRORS CAN BE CORRECTED AND DETECTED.

NOTE

THIS TEST IS RUN FOR MS11-M MEMORY ONLY

2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708

6.2.2.11 TEST 12 WRITE BYTE CLEARS SBE TEST

1. CREATE A SINGLE BIT ERROR.
2. WRITE A BYTE OF DOUBLE WORD TO ONES.
3. READ A BYTE OF DOUBLE WORD.
4. IF THIS IS MS11-M, THE SBE FLAG SHOULD BE SET.
5. THE BYTE SHOULD HAVE BEEN EQUAL TO ONES.
6. DO (1-5) FOR EACH OF THE 4 BYTES OF THE DOUBLE WORD
7. DO (1-6) FOR A SINGLE BIT ERROR IN EACH OF 32 POSITIONS OF A DOUBLE WORD
I.E. (32 TIMES)
8. IF NOT IN QUICK VERIFY MODE THEN DO (1-7) FOR DATA CONSISTING OF 1 BIT SET IN EACH OF 32 POSITIONS OF A DOUBLE WORD.
I.E. (32 X 32 = 1024 TIMES)
9. CLEAR ANY ERRORS OUT OF TEST LOCATIONS.

THIS INSURES THAT SINGLE BIT ERRORS IN THE DATA PORTION (NOT IN CHECKBITS) CAN BE CLEARED BY WRITING THE CORRESPONDING BYTE AND THAT WRITING ANY OTHER BYTE DOES NOT CHANGE THE EXISTING SINGLE BIT ERROR.

NOTE

THIS TEST IS RUN FOR MS11-M MEMORY ONLY.

2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762

6.2.2.12 TEST 13 CREATE DOUBLE BIT ERROR TEST

1. CREATE A DOUBLE BIT ERROR.
2. ACCESS THE DATA (TST INSTRUCTION).
3. CHECK THAT THE CSR DBE FLAG IS SET, AND THE ERROR ADDRESS IS LATCHED.
4. INITIALIZE CSR TO ALLOW PARITY TRAPS ON DBE'S.
5. ACCESS THE DATA (TST INSTRUCTION).
6. CHECK THAT A PARITY TRAP OCCURRED.
7. DO (1-6) FOR THE 2ND BIT OF EACH DOUBLE BIT ERROR IN EACH OF 32 POSITIONS OF A DOUBLE WORD LESS THE ONE POSITION OF THE 1ST BAD BIT.
I.E. (31 TIMES)
8. IF NOT IN QUICK VERIFY MODE THEN DO (1-7) FOR THE 1ST BIT OF EACH OF DOUBLE BIT ERROR IN EACH OF 32 POSITIONS OF A DOUBLE WORD.
I.E. (31 X 32 = 992 TIMES)
9. DO (1-8) FOR COMPLEMENTED DATA (ONES VERSUS ZEROS IN DOUBLE WORD)
I.E. (992 X 2 = 1984 TIMES)
OR (31 X 2 = 62 TIMES (QUICK VERIFY))
10. DO (1-6) FOR A DOUBLE WORD EQUAL TO (000000,000000), AND ALL POSSIBLE DOUBLE BIT ERROR COMBINATIONS FORCED INTO EACH OF THE CHECK BITS (CSR BITS 5-11).
11. CLEAR ANY ERRORS OUT OF TEST LOCATIONS.

THIS INSURES THAT ALL DOUBLE BIT ERRORS CAN BE CREATED AND DETECTED AND CAUSE TRAPS.

NOTE

- 1) THIS TEST IS RUN ON THE MS11-M ONLY.
- 2) THIS TEST IS ONLY RUN DURING THE FIRST (QV) PASS WHEN UNDER ACT OR APT, AND IS RUN FOR ECC MEMORY ONLY.

2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801

6.2.2.13

TEST 14 BASIC DOUBLE BIT ERROR TEST

1. WRITE THE CSR TO ENABLE DIAG MODE WITH A DOUBLE BIT ERROR CHECK BITS OF 110011 AND UNCORRECTABLE ERROR INDICATION ENABLED.
2. WRITE FIRST AUT IN A 16K BANK WITH DATA OF ALL ZERO'S. THIS WILL WRITE THE CHECK BITS IN (1)
3. READ ADDRESS. THIS SHOULD CAUSE A DOUBLE BIT ERROR. BUS PBL IS ASSERTED AND WE CHECK FOR A PARITY TRAP TO OCCUR.
4. READ THE CSR FOR CHECK BITS IN (1) AND UNCORRECTABLE ERROR INDICATOR.
5. WRITE ONES TO THE HIGH BYTE OF THE ADDRESS UNDER TEST. SINCE A DBE EXISTS AT THIS ADDRESS THE WRITE SHOULD BE ABORTED.
6. READ ADDRESS AND CHECK FOR A PARITY TRAP TO OCCUR AS A RESULT OF (5)
7. REPEAT 5 AND 6 FOR DATA OF ONES IN THE LOW BYTE AND CHECK FOR WRITE ABORT AND PARITY TRAP.

THIS TEST CHECKS TO SEE IF A DOUBLE BIT ERROR WILL BE ABORTED AND A BYTE WRITE OF A DOUBLE BIT ERROR WILL BE ABORTED.

NOTE

THIS TEST IS ONLY RUN FOR THE MS11-P

2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854

6.2.2.14 TEST 15 WRITE INHIBIT OF BYTE WITH DBE

1. CREATE A DOUBLE BIT ERROR.
2. DO A MOVB IMMEDIATE TO TEST BYTE.
3. CHECK THAT DOUBLE WORD IS STILL BAD (UNCHANGED-WITH DBE).
4. DO (2-3) ON ALL 4 BYTES OF DOUBLE WORD.
5. DO (1-4) FOR THE 2ND BIT OF EACH DOUBLE BIT ERROR IN EACH OF 32 POSITIONS OF A DOUBLE WORD LESS THE ONE POSITION OF THE 1ST BAD BIT.
I.E. (31 TIMES)
6. IF NOT IN QUICK VERIFY MODE THEN DO (1-5) FOR THE 1ST BIT OF EACH DOUBLE BIT ERROR IN EACH OF 32 POSITIONS OF A DOUBLE WORD.
I.E. (31 X 32 = 922 TIMES)
7. DO (1-6) FOR COMPLEMENTED DATA (ONES VERSUS ZEROS IN DOUBLE WORD).
I.E. (992 X 2 = 1984 TIMES)
OR (31 X 2 = 62 TIMES (QUICK VERIFY))
8. DO (1-4) FOR A DOUBLE WORD EQUAL TO (000000,000000), AND ALL POSSIBLE DOUBLE BIT ERROR COMBINATIONS FORCED INTO THE CHECK BITS (CSR BITS 5-11).
9. CLEAR ANY ERRORS OUT OF TEST LOCATIONS.

THIS INSURES THAT NO DOUBLE BIT ERROR CAN BE CLEARED BY A MOVB TO ANY AFFECTED BYTE.

NOTE

- 1) THIS TEST IS RUN ON THE MS11-M ONLY.
- 2) THIS TEST IS ONLY RUN DURING THE FIRST (QV) PASS WHEN UNDER ACT OR APT, AND IS RUN FOR ECC MEMRY ONLY.

2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907

6.2.2.15 TEST 16 WRITE INHIBIT OF WORD WITH DBE TEST

1. CREATE A DOUBLE BIT ERROR.
2. DO MOV IMMEDIATE ON TEST LOCATION.
3. CHECK THAT DOUBLE WORD IS STILL BAD (UNCHANGED-WITH DBE).
4. DO (2-3) ON BOTH DOUBLE WORDS.
5. DO (1-4) FOR THE 2ND BIT OF EACH DOUBLE BIT ERROR IN EACH OF 32 POSITIONS OF A DOUBLE WORD LESS THE ONE POSITION OF THE 1ST BAD BIT.
I.E. (31 TIMES)
6. IF NOT IN QUICK VERIFY MODE THEN DO (1-5) FOR THE 1ST BIT OF EACH DOUBLE BIT ERROR IN EACH OF 32 POSITIONS OF A DOUBLE WORD.
I.E. (32 X 32 = 992 TIMES)
7. DO (1-6) FOR COMPLEMENTED DATA (ONES VERSUS ZEROS IN DOUBLE WORD).
I.E. (992 X 2 = 1984 TIMES)
OR (31 X 2 = 62 TIMES (QUICK VERIFY))
8. DO (1-4) FOR A DOUBLE WORD EQUAL TO (000000,000000), AND ALL POSSIBLE DOUBLE BIT ERROR COMBINATIONS FORCED INTO THE CHECK BITS (CSR BITS 5-11).
9. CLEAR ANY ERRORS OUT OF TEST LOCATIONS.

THIS INSURES THAT NO DOUBLE BIT ERROR CAN BE CLEARED BY A MOV TO ANY AFFECTED WORD.

NOTE

- 1) THIS TEST IS RUN ON THE MS11-M ONLY
- 2) THIS TEST IS ONLY RUN DURING THE FIRST (QV) PASS WHEN UNDER ACT OR APT, AND IS RUN FOR ECC MEMORY ONLY.

2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924

6.2.2.16 TEST 17 HOLDING 1'S 0'S TEST

1. WRITE A 16K BANK WITH ALTERNATING BYTES OF ZEROS ONES
WRITING A BYTE AT A TIME.
2. READ EACH WORD FOR CORRECT TEST.
3. DO (1-2) AGAIN FOR A COMPLEMENT TEST.

THIS CHECKS THE MEMORY FOR THE CAPABILITY OF HOLDING 0'S 1'S.

2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963

6.2.2.17 TEST 20 SYNDROME BITS TO THE CSR ON A SBE TEST

1. WRITE CSR WITH CHECK BITS TO CORRECT BIT 0 OF THE FIRST AUT 16K BANK FROM A 0 TO A 1 WITH DIAG MODE.
2. WRITE AUT WITH DATA OF 0'S CREATING A SBE.
3. CLEAR CSR.
4. READ THE AUT TO CLOCK THE ADDRESS AND SYNDROMES INTO THE CSR.
5. READ THE CSR FOR THE SBE INDICATOR, BIT 4.
6. WRITE THE CSR TO DIAG MODE TO CLOCK THE SYNDROME BITS INTO CSR BITS 5-11.
7. READ THE CSR FOR THE PROPER SYNDROME BITS.
8. REPEAT 1-7 FOR ALL 16 DATA BITS.
9. REPEAT 1-8 FOR DATA OF ONES SO THAT A CORRECTION WILL OCCUR FROM A 1 TO A ZERO.

THIS TEST CHECKS TO SEE THAT THE EDC CHIP CAN DETECT SINGLE BIT ERRORS FOR ALL 16 DATA BITS BY CHECKING FOR CSR BIT#4 AND THAT THE PROPER SYNDROME BITS ARE PLACED IN THE CSR.

NOTE

THIS TEST IS ONLY RUN FOR THE MS11-P

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

6.2.2.18 TEST 21 MARCHING 0'S 1'S TEST

1. WRITE A BACKGROUND OF ALTERNATING BYTES OF ZEROS ONES
2. FOR THE 16K BANK ADDRESSING DOWN
 - (A) READ CHECK A WORD
 - (B) BYTE SWAP A WORD
 - (C) READ CHECK A WORD
3. FOR THE 16K BANK ADDRESSING UP
 - (A) READ CHECK A WORD
 - (B) BYTE SWAP A WORD
 - (C) READ CHECK A WORD
4. FOR THE 16K BANK ADDRESSING UP
 - (A) READ CHECK A WORD
 - (B) BYTE SWAP A WORD
 - (C) READ CHECK A WORD
5. FOR THE 16K BANK ADDRESSING DOWN
 - (A) READ CHECK A WORD
 - (B) BYTE SWAP A WORD
 - (C) READ CHECK A WORD

THIS CHECKS THE INTEGRITY OF THE 32 BIT DOUBLE WORDS.
IT CAN EXECUTE OUT OF THE USER DATA PAR'S.

NOTE

IT IS NOT UNCOMMON TO SEE A MISLEADING ERROR TYPEOUT BECAUSE THE SECOND TEST IN EACH CASE IS BASED UPON A BYTESWAP OF THE FIRST TEST WHICH MAY OR MAY NOT HAVE FAILED. IF THE ERROR REPORT INDICATES ERRORS IN PAIRS WITH THE BAD BIT IN THE SECOND REPORT BEING THE SAME BIT POSITION RELATIVE TO A BYTE THEN YOU SHOULD IGNORE THE SECOND ERROR REPORT.

3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068

6.2.2.19 TEST 22 REFRESH TEST

- 1. WRITE A DIAGONAL TEST OF ONES ON EVERY KDIAG(TH) STRIPE
WRITE ZEROS ELSEWHERE.

THIS TEST IS ON ADDRESSES NOT BIT POSITIONS.

EXAMPLE:

ADDRESS

LSB'S

MSB'S

0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
1	0	0	0	1	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
1	0	0	0	1	0	0	0

NOTE

EXAMPLE USES KDIAG OF VALUE 4 MORE TYPICAL IS A VALUE OF 8. CONSULT THE SYMBOLIC DEFINITION OF 'KDIAG' IN THE PROGRAM LISTING TO BE SURE.

- 2. DISTURB EACH ROW FOR > 3.2MS
- 3. READ CHECK DIAGONAL PATTERN
- 4. DO (1-3) KDIAG TIMES MOVING THE PLACEMENT OF THE DIAGONAL STRIPE TO COVER ALL ADDRESS POSITIONS.
- 5. DO (1-4) FOR A COMPLEMENT PATTERN (ZEROS IN A BACKGROUND OF ONES)

NOTE

THIS TEST IS NOT NORMALLY EXECUTED EXCEPT UNDER APT OR ACT. IT MAY BE INVOKED VIA FIELD SERVICE COMMAND 13 (KAMIKAZE MODE).

3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3090

6.2.2.20 TEST 23 SHIFTING DIAGONAL TEST

SIMILAR IN OVERALL OPERATION TO TEST 22 EXCEPT IT DOES NOT DELAY
FOR REFRESH AND DISTURB ROWS.

NOTE

THIS TEST IS NOT NORMALLY EXECUTED
EXCEPT UNDER APT OR ACT. IT MAY BE
INVOKED VIA FIELD SERVICE COMMAND 13
(KAMIKAZE MODE).

3092
3093
3094
3095
3096
3097
3098
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3110
3111

6.2.2.21 TEST 24 FAST GALLOPING PATTERN TEST

THIS DOES A CLASSICAL GALLOPING PATTERN EXCEPT THAT ADDRESSING IS INCREMENTED BY 400 OCTAL (EVERY 64TH DOUBLE WORD)

NOTE

THIS TEST IS NOT NORMALLY EXECUTED EXCEPT UNDER APT OR ACT. IT MAY BE INVOKED VIA FIELD SERVICE COMMAND 13 (KAMIKAZE MODE).

3113
3114
3115
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168

6.2.2.22 TEST 25 INTERRUPT ENABLE TEST

1. SET CSR TO ALLOW UNCORRECTABLE ERROR TRAPS.
2. ACCESS TEST DOUBLE WORDS.
3. CHECK THAT NO UNCORRECTABLE ERROR TRAP OCCURRED.
4. ENABLE CSR FOR SBE TRAPS.
5. ACCESS TEST DOUBLE WORDS.
6. CHECK THAT NO SBE TRAP OCCURRED.
7. WRITE A SBE IN 1 BYTE.
8. DISABLE CSR TRAPS.
9. ACCESS TEST DOUBLE WORDS.
10. CHECK THAT NO TRAPS OCCURRED.
11. ENABLE CSR FOR SBE TRAPS.
12. ACCESS TEST DOUBLE WORDS.
13. CHECK TO INSURE TRAP OCCURRED.
14. DO (7-13) FOR THE 3 OTHER BYTES IN THE DOUBLE WORD.
15. CREATE A DBE IN 1 BYTE.
16. DISABLE CSR TRAPS.
17. ACCESS THE TEST DOUBLE WORD.
18. CHECK THAT NO TRAPS OCCURRED.
19. ENABLE CSR FOR DBE TRAPS.
20. ACCESS THE TEST DOUBLE WORD.
21. CHECK TO INSURE TRAP OCCURRED.
22. ENABLE CSR FOR SBE TRAPS.
23. ACCESS THE TEST DOUBLE WORD.
24. CHECK TO INSURE TRAP OCCURRED.
25. DO (15-24) FOR THE 3 OTHER BYTES IN THE DOUBLE WORD.

THIS INSURES THAT SBE'S DBE'S GIVE THE CORRECT TYPE OF TRAPS.

NOTE
THIS TEST IS RUN FOR MS11-M MEMORY ONLY.

3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3200

6.2.2.23 TEST 26 RANDOM DATA TEST

WRITE RANDOM DATA IN A 16K BANK WHILE INCREMENTING THE ADDRESSES.

READ CHECK RANDOM DATA.

THIS ROUTINE REGENERATES THE SAME RANDOM NUMBERS BY USING THE SAME

SEED AS THE WRITE SEQUENCE. AFTER THE READ CHECK THE SEED IS UPDATED SO THAT THE NEXT USE OF THIS PATTERN WILL NOT INVOKE THE SAME SEQUENCE OF RANDOM NUMBERS.

IF YOU WISH TO CHANGE THE RANDOM SEQUENCE SO THAT IT IS DIFFERENT THAN ANY OTHER RUN IN THE SAME CONFIGURATION THEN THERE ARE 2 WAYS OF DOING SO.

1. MODIFY SYMBOLIC LOCATIONS "SEEDHI" AND "SEEDLO" TO ANY NUMBER YOU LIKE.
2. ENTER FIELD SERVICE MODE AND EXECUTE THIS TEST (COMMAND 5) ON SOME (ANY GOOD) BANK FOR A SHORT TIME (30 SEC OR SO).

THIS CAN EXECUTE OUT OF THE USER DATA PAR'S, THE KERNEL DATA PAR'S, AND THE SUPERVISOR DATA PAR'S.

3202
3203
3204
3205
3206
3207
3208
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218

6.2.2.24 TEST 27 UNIQUE BANK TEST

THIS TEST USES TEST 0 TO WRITE READ THE BANK NUMBER IN EACH BANK.

IT DOES NOT TEST BANKS THAT REQUIRE RELOCATION TO TEST.

IT DOES NOT RUN AS PART OF ANY SCRIPT BUT RATHER IS ALWAYS RUN AFTER NORMAL PATTERN TESTS ARE COMPLETE.

3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3230
3231
3232
3233
3234
3235
3236

6.2.2.25 TEST 30 FLUSH OUT DBE'S TEST

THIS READS EACH LOCATION THEN MOVES THE OLD VALUE BACK IN. THIS IS DONE WITH ECC DISABLED AND THEREFORE CORRECTS ANY DBE'S OR SBE'S (IF POSSIBLE).

IT DOES NOT RUN AS PART OF ANY SCRIPT BUT RATHER IS ALWAYS RUN JUST PRIOR TO THE END OF PASS CODE, AS PART OF A CONTROL 'C' (BOOT) COMMAND, AS PART OF END OF PASS SHUTDOWN FOR ACT OR XXDP CHAIN MODE, AS PART OF HANGING SEQUENCE AFTER AN ERROR IF UNDER ACT OR APT, AND AS PART OF A SHUTDOWN SEQUENCE DIRECTED BY SWITCH 8 (HALT PROGRAM).

3238
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3290

6.2.2.26 TEST 31 SOB-A-LONG TEST

RATIONALIZATION

IN ORDER TO CONCENTRATE THE MEMORY CYCLES OF A TEST INTO A PARTICULAR ADDRESS, WE MUST CUT THE OVERHEAD CYCLES TO A MINIMUM. FREQUENTLY, THE INSTRUCTION ITSELF MAY PROVIDE ADEQUATE DATA OR SET UP A BACKGROUND IN WHICH ANY COMPLEMENTED BIT MAY FIND IT HARD TO SURVIVE.

THE SOB INSTRUCTION IS THE ONLY PDP-11 INSTRUCTION THAT IS (1) A SINGLE OPERAND, (2) CAN BE REPEATEDLY EXECUTED AT THE SAME PC AND, (3) CAN ESCAPE THIS REPETITIOUS LOOP.

HENCE, IT CAN BE POSSIBLE TO SOB A MOS CELL TO DEATH (OR AT LEAST BRAIN WASH HIM), AND TO SOB A CORE INTO OVER-HEATING (OR AT LEAST WARM DISCOMFORT).

THE SOB ROUTINE WILL BE LOADED AND CALLED WITH RO SET EQUAL TO THE SOB CONSTANT "SOBK", R1 SET EQUAL TO THE COMPLEMENT OF A "SOB RO,." INSTRUCTION "100776".

SIMPLIFIED SOB EXAMPLE:

```

1$:      SOB          RO,1$          ;SOB TILL RO UNDERFLOWS
        MOV          R1,1$          ;WRITE COMPLEMENT OF SOB
        CMP          R1,1$          ;READ CHECK NOT SOB
        BEQ          2$              ;SKIP IF OK
        SOBFAIL      ;TRAP REPORT ERROR
2$:      SOBMOV1     ;CODE TO GET SELF MOVED
        SOBMOV2     ;FORWARD 1 WORD AND RUN AGAIN
        SOBMOV3
        SOBMOV4
        SOBMOV...

```

THE VALUE OF THE SOB CONSTANT CAN BE FOUND AT SYMBOLIC LOCATION "SOBK" (TYPICAL 25 DECIMAL).

THIS TEST IS NOT IN THE NORMAL SCRIPT OF EXECUTION BUT MAY BE ADDED VIA THE APT E-TABLE, REFERENCE SYMBOLIC LOCATIONS "MKPAT", "MJPAT", "\$DDW2-5". FIELD SERVICE MODE COMMAND 8 IS THE NORMAL METHOD OF RUNNING THIS PATTERN.

NOTE

THIS TEST IS NOT NORMALLY EXECUTED EXCEPT UNDER APT OR ACT. IT MAY BE INVOKED VIA FIELD SERVICE COMMAND 13 (KAMIKAZE MODE).

3292
3293
3294
3295
3296
3297
3298
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3340
3341

6.2.2.27 TEST 32 WRITE RECOVERY TEST

THIS TEST CAUSES A WRITE, READ, WRITE, READ, ... TO OCCUR IN MEMORY AND IF THE 1ST, 3RD, 5TH, ... READ IS BAD THE PROGRAM MAY BOMB OR IF THE 2ND, 4TH, 6TH, ... READ IS BAD THE PROGRAM WILL GRACEFULLY TYPE OUT THE ERROR.

WRITE RECOVERY TEST

THIS TEST DIFFERS FROM OTHER TESTS IN THAT IT CONSISTS OF A SMALL TEST PROGRAM ACTUALLY RUNNING IN THE BANK UNDER TEST. THE PROGRAM IS SELF MODIFYING AND MAY BE DIFFICULT TO DEBUG. TO AID IN THE DEBUG, REMEMBER THAT THE BANK AND MARGIN ARE BEING DISPLAYED. THIS WILL ALLOW THE USER TO AT LEAST SEE WHICH MEMORY BANK FAILED.

THE TEST CONSISTS OF 1/2 OF THE BANK STORED WITH 'MOV R2,-(PC)' AND THE OTHER 1/2 CONTAINING '177667'. '177667' IS THE COMPLEMENT OF 'JMP (R0)' INSTRUCTION. R2 CONTAINS 'COM -(R1)' INSTRUCTION ON ENTRY TO THE BANK AND R1 CONTAINS THE HIGHEST TEST ADDRESS IN THAT BANK.

IF YOU UNDERSTAND THIS SO FAR THE REST IS EASY.

THE TEST EXECUTION IS AS FOLLOWS:

1. THE 'MOV R2,-(PC)' INSTRUCTION EXECUTES STORING THE CONTENTS OF R2 IN THE ADDRESS IT VACATED (DUE TO -(PC).
2. SINCE R2 CONTAINS A 'COM -(R1)' INSTRUCTION IT COMPLEMENTS THE HIGHEST ADDRESS UNDER TEST. THIS ADDRESS CONTAINED '177667' SO AFTER THE COM -(R1) IT EQUALS 110 CLEVERLY THIS IS THE 'JMP (R0)' INSTRUCTION.
3. THIS SEQUENCE CONTINUES UNTIL THE 'MOV R2,-(PC)' INSTRUCTIONS REACH THE MIDDLE OF THE TEST BANK. THEN THE 'JMP (R0)' INSTRUCTION IS MET AND EXECUTED. R0 CONTAINED THE RETURN ADDRESS BACK TO TEST 13.
4. THESE STEPS ARE REPEATED FOR EACH BANK UNDER TEST.

NOTE

THIS TEST IS NOT NORMALLY EXECUTED EXCEPT UNDER APT OR ACT. IT MAY BE INVOKED VIA FIELD SERVICE COMMAND 13 (KAMIKAZE MODE).

3343
3344
3345
3346
3347
3348
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3390
3391

6.2.2.28 TEST 33 BRANCH GOBBLE TEST

THIS TEST LOADS A SMALL ROUTINE INTO THE MEMORY UNDER TEST. THE ROUTINE MOVES ITSELF ALONG IN MEMORY ONE WORD AFTER EACH PASS SO THAT WHEN IT REACHES THE END EVERY INSTRUCTION HAS EXECUTED FROM EVERY LOCATION WITH THE EXCEPTION OF THE BEGINNING AND END OF EACH TEST AREA.

THE BRANCH GOBBLE'S GENERAL FORMAT AFTER YOU ELIMINATE SETUP CODE AND CODE TO MOVE THE PROGRAM ALONG IS AS FOLLOWS.

```

BGTEST: 0                               ;TEST WORD
BRGOBB: SEC                             ;INC LOW BYTE
        ADCB                            ;END LOOP AFTER 128 TIMES
        BMI                               ;INC HIGH BYTE
        INCB                             ;LOOP 128 TIMES
        BR                                ;BRANCH IF V-BIT SET (SHOULD BE)
1$:     BVS                               ;ERROR TRAP
        ERROR                            ;CLEAR V-BIT
2$:     CLV                               ;INC HIGH BYTE ONE LAST TIME
        INCB                             ;BRANCH IF C-BIT SET (SHOULD NOT BE)
        BCS                               ;BRANCH IF V-BIT CLEAR (SHOULD NOT BE)
        BVC                               ;BRANCH IF N-BIT SET (SHOULD BE)
        BMI                               ;ERROR TRAP
3$:     ERROR
4$:     RETURN
        BGTEST                            1$
        BRGOBB                          BGTEST+1
        1$                               BRGOBB
        2$                               2$
        3$                               BGTEST
        4$                               3$
        3$                               3$
        4$                               4$

```

THIS CODE ORIGINALLY CAME FROM THE PDP-11 FAMILY INSTRUCTION EXERCISER DZQKA-A. THE FIRST MOS MEMORIES WERE SUSCEPTIBLE TO THIS SECTION OF THAT DIAGNOSTIC AND IT HAS BEEN AN IMPORTANT MEMORY EXERCISER EVER SINCE.

NOTE

THIS TEST IS NOT NORMALLY EXECUTED EXCEPT UNDER APT OR ACT. IT MAY BE INVOKED VIA FIELD SERVICE COMMAND 13 (KAMIKAZE MODE).

3393
3394
3395
3396
3397
3398
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3410
3411
3412
3413
3414
3415
3416
3417

6.2.2.29 TEST 34 SOFT ERROR TEST

RATIONALIZATION

MOS CHIPS HAVE A FAILURE MODE IN WHICH THEY CAN RANDOMLY PICK OR DROP BITS. THIS IS CAUSED BY ALPHA PARTICLES BOMBARDING THE CELL. IF THE CELL IS VERY SMALL (AND THEY ARE) THEN THE ELECTRONS DISPLACED BY THE ALPHA PARTICLE ARE SUFFICIENT TO CAUSE THE CELL TO CHANGE FROM A ONE TO A ZERO OR FROM A ZERO TO A ONE.

THIS TEST IS CONTROLLED BY THE MAIN PROGRAM SO THAT IT IS USED TO CREATE A TEST OF 125252 AND 52525 ON ALTERNATE PASSES OF THE PROGRAM. THE CONFIGURATION TABLE IS USED TO FLAG BANKS THAT HAVE THE TEST INVALIDATED BECAUSE ANOTHER TEST WAS WRITTEN OVER THIS BACKGROUND.

THIS TEST IS NOTHING MORE THAN A CLEVER USE OF TEST 0.

3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3440
3441
3442
3443
3444
3445
3446
3447

6.2.2.30 TEST 35 WORST CASE PARITY TEST

1. FORCE WRITE WRONG PARITY IN EACH 1K WORD BLOCK OF THE MEMORY UNDER TEST.
2. READ WITH PARITY TRAPPING ENABLED, MAKING SURE THAT A TRAP OCCURRS.
3. MAKE SURE ERROR ADDRESS BITS ARE SET CORRECTLY.
4. WRITE GOOD PARITY WITHOUT TRAPPING, AND MAKE SURE NO TRAP OCCURRS WHEN READ.

NOTE

THIS TEST IS RUN FOR PARITY MEMORY WHICH IS NOT CONTROLLED BY THE SAME CSR AS THE PROGRAM.

3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3470
3471
3472
3473
3474
3475

6.2.2.31

TEST 36 CORRECTION CODE TEST

1. WRITE CSR WITH CHECK BITS TO CORRECT BIT 0 OF THE FIRST ADDRESS IN A 16K BANK FROM A 0 TO A 1 WITH DIAG MODE.
2. WRITE AUT WITH DATA OF 0'S.
3. READ AUT FOR CORRECTION OF BIT 0 FROM A 0 TO A 1.
4. REPEAT 1-3 FOR ALL 16 DATA BITS.
5. REPEAT 1-4 FOR DATA OF ONES SO THAT A CORRECTION WILL OCCUR FROM A 1 TO A ZERO.

THIS TEST CHECKS TO SEE THAT THE EDC CHIP CAN CORRECT SINGLE BIT ERRORS FOR ALL 16K DATA BITS FROM A 1 TO A 0 AND VISA VERSA.

NOTE

THIS TEST IS ONLY RUN FOR THE MS11-P

3477
3478
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3499
3500
3501

6.2.2.32 TEST 37 CHECK ECC DISABLE TEST

1. WRITE CSR WITH ECC DISABLE, DIAG MODE, AND SBE CHECK BITS OF 000010.
2. WRITE AUT WITH DATA OF ZERO'S. THIS SHOULD WRITE CHECK BITS TO MEMORY.
3. READ AUT FOR DATA OF ZEROS INSURING NO CORRECTION WAS MADE.

NOTE

THIS TEST IS RUN ON THE MS11-P ONLY.

3503
3504
3505
3506
3507
3508
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3530

6.2.2.33 TEST 41 ADDRESS TO CSR ON DBE TEST

1. WRITE CSR WITH ECC DISABLE, DIAG MODE, AND DOUBLE BIT ERROR CHECK BITS OF 010011
2. WRITE AUT WITH DATA OF ZEROS CREATING A DBE.
3. READ AUT TO DETECT DBE AND TO CLOCK ADDRESS INTO CSR
4. READ CSR FOR CORRECT ADDRESS IN BITS 5-11.
5. INCREMENT ADDRESS BY 1K AND REPEAT 1-4 UNTIL 16K IS DONE.

THIS TEST INSURES THAT THE CORRECT ADDRESS APPEARS IN CSR BITS 5-11 ON A DBE

NOTE

THIS TEST IS RUN ON A MS11-P ONLY.

3532
3533
3534
3535
3536
3537
3538
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3560
3561
3562
3563
3564
3565
3566

6.2.2.34 TEST 42 EXTENDED ADDRESS TO CSR ON ERROR TEST

1. WRITE CSR WITH SBE CHECK BITS OF 000010 WITH DIAGNOSTIC MODE.
2. WRITE LOW ADDRESS IN A 16K BANK WITH DATA OF ZEROS CREATING A SBE.
3. CLEAR THE CSR.
4. READ ADDRESS TO DETECT SBE.
5. READ CSR FOR CORRECT ADDRESS AND THE SBE INDICATOR BIT #4.
6. ENABLE CSR BIT 14 TO CHECK THE EXTENDED ADDRESS BITS.
7. READ CSR FOR CORRECT ADDRESS BITS
8. REPEAT 1-7 WITH A TEST ADDRESS THAT IS THE HIGHEST IN A 16K BANK.

THIS TEST CHECKS TO SEE THAT THE CORRECT ADDRESS BITS APPEAR IN THE CSR.
THIS IS ALSO REPEATED FOR THE EXTENDED ADDRESS FUNCTION IN THE CSR.

NOTE

THIS TEST IS ONLY RUN FOR THE MS11-P

3568
3569
3570
3571
3572
3573
3574
3575
3576
3577
3578
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3599
3600
3601
3602
3603
3604
3605
3606
3607

6.2.2.35 TEST 43 WRITE BYTE TEST

1. WRITE CSR TO DIAG MODE WITH CHECK BITS OF 001100. THESE CORRESPOND TO DATA OF ZEROS.
2. WRITE FIRST AUT WITH DATA OF ONE IN BIT ZERO. THE WRITE EFFECTIVELY CREATES A SBE IN BYTE 0.
3. CLEAR THE CSR
4. WRITE BYTE 1 OF THE AUT WITH DATA OF ALL ONES.
5. READ CSR TO CHECK FOR SBE INDICATION.
6. WRITE THE CSR TO DIAG MODE.
7. READ THE AUT TO CHECK FOR THE CORRECT DATA -- ALL ONES IN HIGH BYTE AND ALL ZEROS IN LOW BYTE.
8. READ THE CSR TO CHECK FOR CORRECT CHECK BITS CORRESPONDING TO THE DATA READ IN (7). THESE CHECK BITS ARE 000110.
9. REPEAT (1)-(8) THIS TIME CREATING AN ERROR IN BYTE 1 (2) AND WRITING BYTE 0 IN (4).

THIS TEST CHECKS TO SEE THAT A SBE WILL BE CORRECTED DURING THE READ PORTION OF THE BYTE WRITE AND THAT CORRECT CHECKBITS WILL BE GENERATED ON THE WRITE.

NOTE

THIS TEST IS ONLY RUN FOR THE MS11-P

3609
3610
3611
3612
3613
3614
3615
3616
3617
3618
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638
3639
3640
3641
3642
3643
3644
3645
3646
3647
3648
3649
3650
3651
3652

6.2.2.36 TEST 44 SHIFTING CHECKBITS THROUGH THE CSR TEST

1. WRITE CSR TO DIAG MODE TO ENABLE CHECKBIT REGISTER.
2. WRITE CSR WITH CHECK BITS OF 000001, ECC DISABLE AND DIAG MODE.
3. WRITE MEMORY WITH DATA OF ZEROS. THIS SHOULD WRITE THE CHECK BITS INTO MEMORY.
4. COMPLEMENT CHECK BITS PATTERN AND WRITE CSR AS IN (2).
5. READ CSR FOR COMPLEMENT CHECK BIT PATTERN.
6. READ MEMORY TO READ CHECK BITS WRITTEN IN (2) INTO CSR.
7. READ CSR FOR CORRET CHECK BITS WRITTEN IN (2).
8. SHIFT CHECK BIT PATTERN AND REPEAT (1-7) TILL CSR BITS 5-10 ARE DONE.
9. COMPLEMENT CHECK BIT PATTERN IN (2) AND REPEAT (1-8) SHIFTING A ZERO THROUGH A FIELD OF ONES.
10. REPEAT 1-9 FOR EVERY 100 OCTAL LOCATIONS IN 16K

THIS TEST CHECKS THE ABILITY TO READ CHECK BITS FROM THE CSR TO MEMORY AND BACK. THE TEST IS DONE TWICE. ONCE SHIFTING A FIELD OF A ONE THROUGH A FIELD OF ZEROS AND A ZERO THROUGH A FIELD OF ONES. THIS TESTS THE CHECKBIT/SYNDROME BIT REGISTER AND CHECK BIT RAM'S. THIS TEST IS DONE FOR EVERY 100 OCTAL LOCATIONS IN 16K.

NOTE

THIS TEST IS ONLY RUN FOR THE MS11-P

3654
3655
3656
3657
3658
3659
3660
3661
3662
3663
3664
3665
3666
3667
3668
3669
3670
3671
3672
3673
3674
3675
3676
3677
3678
3679
3680
3681
3682
3683
3684
3685
3686
3687
3688
3689

6.2.2.37 TEST 45 SYNDROME BITS TO THE CSR ON A DBE TEST

1. WRITE CSR WITH DIAG MODE TO ENABLE CHECK/SYNDROME BIT REGISTER.
2. WRITE CSR WITH DBE CHECK BITS OF 110011 WITH DIAG MODE.
3. WRITE MEMORY WITH DATA OF ZEROS CREATING A DBE.
4. CLEAR CSR.
5. READ MEMORY TO DETECT DBE.
6. READ CSR FOR UNCORRECTABLE ERROR INDICATOR.
7. WRITE CSR TO DIAG MODE TO READ SYNDROME BITS INTO CSR.
8. READ CSR FOR CORRECT SYNDROME BITS OF 111111.
9. REPEAT (1-8) WITH MULTIPLE BIT ERROR CHECK BITS OF 111100 AND CORRESPONDING SYNDROME BITS OF 110000.

THIS TEST CHECKS THE ABILITY OF THE CSR TO DETECT A DBE AND READ FOR THE PROPER SYNDROME BITS GENERATED BY THE EDC CHIP. THIS TEST IS THEN REPEATED WITH CHECK BITS CORRESPONDING TO A MULTIPLE BIT ERROR.

NOTE

THIS TEST IS ONLY RUN FOR THE MS11-P

3691
3692
3693
3694
3695
3696
3697
3698
3699
3700
3701
3702
3703
3704
3705
3706
3707
3708
3709
3710
3711
3712
3713
3714
3715
3716
3717
3718
3719
3720
3721
3722
3723
3724
3725
3726
3727

6.2.2.38 TEST 46 CHECK SINGLE BIT ERRORS WITH ECC DISABLED TEST

1. WRITE CSR WITH CHECK BITS TO CORRECT BIT 0 OF THE FIRST ADDRESS IN A 16K BANK FROM A 0 TO A 1 WITH DIAG MODE AND ECC DISABLED.
2. WRITE AUT WITH DATA OF 0'S THUS CREATING A SBE.
3. WRITE THE CSR TO ECC DISABLE.
4. READ AUT TO DETECT SBE.
5. CHECK TO SEE THAT NO TRAP OCCURED.
6. READ CSR TO SEE THAT UNCORRECTABLE ERROR (CSR15) IS SET.
7. REPEAT 1-6 FOR ALL 16 DATA BITS.
8. REPEAT 1-7 FOR DATA OF ONES SO THAT A CORRECTION WILL OCCUR FROM A 1 TO A ZERO.
9. REPEAT 1-8 EXCEPT IN STEPS (3) THE CSR IS WRITTEN TO ECC DISABLE AND BUS PBL ENABLE AND (5) WE CHECK FOR TRAPS.

THIS TEST CHECKS TO SEE THAT SBE ARE TREATED A UNCORRECTABLE ERRORS WITH ECC DISABLE. THE TEST IS REPEATED 2 TIMES, ONCE WITH TRAPS DISABLED AND AGAIN WITH IT ENABLED. THIS IS DONE FOR ALL 16 POSSIBLE SBE CONDITIONS.

NOTE

THIS TEST IS ONLY RUN FOR THE MS11-P

3729
3730
3731
3732
3733
3734
3735
3736
3737
3738
3739
3740
3741
3742
3743
3744
3745
3746
3747
3748
3749
3750
3751
3752
3753
3754
3755
3756
3757
3758
3759
3760
3761
3762

6.2.2.39 TEST 47 NO CSR UPDATE ON SBE WITH EXSISTING DBE TEST

1. WRITE THE CSR TO DIAG MODE TO ENABLE CHECKBIT/SYNDROME BIT REGISTER.
2. WRITE THE CSR WITH DBE CHECK BITS OF 110011 AND DIAG MODE.
3. WRITE MEMORY WITH DATA OF ZEROS CREATING A DBE.
4. WRITE CSR WITH SBE CHECK BITS OF 000010 AND DIAG MODE.
5. WRITE MEMORY 4K ABOVE ADDRESS IN (3) CREATING A SBE.
6. CLEAR CSR.
7. READ MEMORY WITH ADDRESS IN (3) TO DETECT DBE.
8. READ CSR FOR CORRECT ADDRESS AND UNCORRECTABLE ERROR INDICATOR
9. READ MEMORY WITH ADDRESS IN (5) TO DETECT SBE.
10. READ CSR FOR SBE INDICATOR AND NO CHANGE IN DBE STATUS IN CSR IN (8)

THIS TEST CHECKS TO SEE THAT NO UPDATE WILL OCCUR IN THE CSR WITH A SBE IN MEMORY WHEN A DBE ALREADY EXISTS.

NOTE

THIS TEST IS ONLY RUN FOR THE MS11-P

3764
3765
3766
3767
3768
3769
3770
3771
3772
3773
3774
3775
3776

6.2.2.40 TEST 999 NULL TEST

THIS IS AN INSTANT RETURN ADDED TO PRESERVE THE SOFTWARE STRUCTURE.

THIS TEST REPLACES ANY REAL TESTS WHEN THE APT E-TABLE DOES NOT SPECIFY A TEST TO BE RUN.

3778
3779
3780
3781
3782
3783
3784
3785
3786
3787
3788
3789
3790
3791
3792
3793
3794
3795
3796
3797
3798
3799
3800
3801
3802
3803
3804
3805
3806
3807
3808
3809
3810
3811
3812
3813
3814
3815
3816
3817
3818
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829
3830
3831

7.0 PROGRAM FEATURES

7.1 FAST DATA ACCFSS RATES

ONE OF THE MAIN AREAS OF CONCERN IN TESTING MEMORY IN SYSTEMS ENVIRONMENTS IS SPEED. ONE OF THE PRIME REASONS THAT SYSTEM PROGRAMS LIKE RSTS, IAS AND MUMPS CAN CRASH DUE TO MEMORY FAILURES NOT DETECTABLE BY MEMORY DIAGNOSTICS (0-124K, 0-2 MEG, ETC.) IS BECAUSE OF MULTIPLE NPR DEVICES CONTENDING FOR THE BUS. AFTER SOME DELAY A NPR DEVICE BECOMES BUS MASTER AND DOES SEVERAL MEMORY TRANSFERS AT MEMORY DATA RATES.

ON THE OTHER HAND MOST DIAGNOSTICS WHEN WRITING READING AND/OR CHECKING PATTERNS SPEND MOST OF THEIR TIME FETCHING INSTRUCTIONS AND OPERANDS OUT OF THEIR PROGRAM SPACE AND PROPORTIONALLY LITTLE TIME ACCESSING THE MEMORY UNDER TEST.

THIS DIAGNOSTIC'S ERROR DETECTING ABILITIES HAVE BEEN OPTIMIZED AROUND THE PRIMARY DESIGN CRITERIA OF SPEED. TO THIS END THE FOLLOWING STEPS HAVE BEEN TAKEN.

7.1.1 FAST CITY

UTILIZATION OF MEMORY MANAGEMENT REGISTERS AS NON MEMORY BUS, NON UNIBUS, BIPOLAR MEMORY. SINCE USER MODE IS ONLY USED FOR RELOCATION AND DATA SPACE IS NEVER USED, THEN SUBROUTINES CAN BE EXECUTED FROM THE UIPAR'S, UDPAR'S, KDPAR'S, SDPAR'S AND WITH SOME BIT PATTERN RESTRICTIONS THE UIPDR'S, UDPDR'S, KDPDR'S, AND SDPDR'S.

THE PROGRAM RUNS IN KERNEL MODE AND PATTERNS ARE EXECUTED IN SUPERVISOR MODE FOR MAPPING PURPOSES. ALL CORE PATTERNS AND SOME MOS PATTERNS ARE SUBROUTINES THAT ARE MOVED TO THIS BIPOLAR REGION REFERRED TO IN THE PROGRAM AS FAST CITY.

NOTE

18-BIT PDP-11'S CANNOT EXECUTE FROM THE PAR'S BECAUSE THEIR PAR'S ARE ONLY 12 BITS WIDE; THEY ALSO HAVE NO SUPERVISOR MODE. THEREFORE, ALL PATTERNS ARE EXECUTED IN MEMORY, USING USER MODE (REFERENCE SECTION 7.5).

7.1.2 SOB'S

3833
3834
3835
3836
3837
3838
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878
3879
3880
3881
3882
3883
3884
3885
3886

UTILIZATION OF THE FULL PDP-11 INSTRUCTION SET TO SPEED PATTERN ALGORITHMS (PRINCIPALLY THE SOB).

7.1.3 CACHE

CACHE IS USED BETWEEN PATTERN TESTS TO DECREASE PROGRAM PASS TIMES. CACHE CAN BE DEFEATED BY THE OPERATOR (REFERENCE SECTION 2.4.3.1).

7.2 BANK ZERO TESTING

BANK ZERO HAS BEEN TRADITIONALLY NEGLECTED BY MEMORY DIAGNOSTICS FOR THE FOLLOWING REASON.

THE VECTOR SPACE EXISTS THERE AND ALL TRAPS MUST NOT ACCESS TEST PATTERN DATA. IF THE AREA IS TESTED THE DIAGNOSTIC MUST NOT USE ANY TRAPS, AND IT IS AGAINST THE RULES FOR POWER TO FAIL.

SYSTEMS WITH MEMORY MANAGEMENT CAN OVERCOME THIS BECAUSE ALL TRAPS ARE TO KERNEL VIRTUAL SPACE EVEN IF THE POWER SHOULD FAIL (CAUTION MUST BE OBSERVED BECAUSE POWER UP GOES TO PHYSICAL ADDRESS 24 (BECAUSE THE MEMORY MANAGEMENT UNIT COMES UP OFF)).

HOWEVER, CATCH 22 IS THAT THE DIAGNOSTIC IS NOT APT COMPATIBLE IN THIS MODE BECAUSE APT ACCESSES PHYSICAL MEMORY LOCATIONS.

THE PDP-11/44 CAN OVER COME THIS BECAUSE THE UNIBUS MAP CAN FOOL APT.

BECAUSE OF THE PREVIOUS ARGUMENTS THIS PROGRAM DOES NOT RELOCATE IN THE TRUE SENSE OF THE WORD (I.E. NO POSITION INDEPENDENT CODE WAS WRITTEN (AT LEAST NOT ON PURPOSE)), BUT RATHER THIS PROGRAM MOVES AND REMAPS (HEREAFTER REFERRED TO AS RELOCATES). THIS ENABLES THE COMPLETE TESTING OF BANK ZERO OR ANY OTHER PROGRAM SPACE OR PRIVILEGED SPACE EXACTLY AS ALL OTHER BANKS ARE TESTED. (THE CONDITIONAL TEST TO SEE IF A BANK IS PROTECTED IS COMPLEMENTED WHEN RELOCATED).

NOTE

THE PROGRAM WILL RELOCATE ONLY IN THE FIRST PASS UNDER APT; AFTER THIS, THE PROGRAM WILL REMAIN FIXED IN BANKS 0 AND 1.

7.3 MEMORY CONFIGURATION MAP

3888
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3910
3911
3912
3913
3914
3915
3916
3917
3918
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928
3929
3930
3931
3932
3933
3934
3935
3936
3937
3938
3939
3940

THIS MAP IS PRINTED OUT IMMEDIATELY AFTER SIZING THE MEMORY UNLESS SW6 IS SET (REFERENCE SECTION 2.4.1). IT CAN ALSO BE PRINTED AT ANY LATER TIME IN FIELD SERVICE MODE (REFERENCE SECTION 2.4.4.8.7)

EXAMPLE:

```

                                MEMORY CONFIGURATION MAP
                                16K BANKS
                                1     2     3     4     5     6     7
                                0123456701234567012345670123456701234567012345670123
ERRORS                          XX
INTRLV  -----3333333333333333-----
MEMTYPE LLLLLLLLMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
CSR     00000000111111112222222222222222224444444444444444444444444444444444444444
PROTECT PP      I      I      P      I
                                0     1     2     3     4     5     6
                                4567012345670123456701234567012345670123456701234567
ERRORS
INTRLV  ----
MEMTYPE PPPP
CSR     4444
PROTECT

```

DISPLAYED ARE BANKS 0-73 OCTAL (2 MEG WORDS). IF THE FAT TERMINAL SWITCH WAS SET (REFERENCE SECTION 2.4.1) THEN ALL BANKS (0-167) WOULD BE SHOWN. IF THIS WAS AN 18-BIT PDP-11 (EG - 11/34), ONLY BANKS 0-7 WOULD BE PRINTED.
THE FIELDS:

ERRORS:

THE SIZING ROUTINE COULD NOT WRITE ZEROS AND ONES IN BANKS 10 11, HENCE THEY ARE MARKED AS BAD WITH X'S.

INTRLV:

THERE IS INTERLEAVING ON BANKS 20-37, WITH CSR 2 (172104) CONTROLLING THE ADDRESS BIT 1 NON-ASSERTED ADDRESSES, AND CSR 3 (172106) CONTROLLING THE ADDRESS BIT 1 ASSERTED ADDRESSES.

ERRORS:

MEMTYPE:

BANKS 0-7 ARE MEMORY TYPE L (MS-11L), AND BANKS 10-37 ARE MEMORY TYPE M (MS11-M) AND BANKS 40-77 ARE MEMORY TYPE P(MS11-P). BANKS 100-167 DO NOT EXIST.

CSR:

3942
3943
3944
3945
3946
3947
3948
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958
3959
3960
3961
3962
3963
3964
3965
3966
3967
3968
3969
3970
3971
3972
3973
3974
3975
3976
3977
3978
3979
3980
3981
3982
3983
3984
3985
3986
3987
3988
3989
3990
3991
3992
3993

BANKS 0-7 ARE ASSIGNED TO CSR 172100, 10-17 TO CSR 172102, AND 20-37 TO INTERLEAVED CSR'S 172104 AND 172106 AND BANKS 40-77 ARE ASSIGNED CSR 17210.

PROTECT:

BANKS 0 AND 1 ARE PROTECTED BECAUSE THEY ARE PROGRAM SPACE. BANK 0 AND 1 CAN ALSO BE PROTECTED BECAUSE THEY ARE IN THE BOTTOM 16K OF AN MS11-M CSR. THE PROTECTION IS HIERARCHICAL AND PROGRAM SPACE OVERSHADOWS MS11-M PROTECTION. BANKS 0 AND 1 WILL NOT BE TESTED UNTIL THE PROGRAM RELOCATES. IF

ANY BANK IS PROTECTED BY MS11-M AND NOT BECAUSE IT IS IN PROGRAM SPACE IT WILL HAVE AN "I" TYPED IN THIS ROW. THIS IS TO POINT OUT WHERE THE PROTECTED BANKS START FOR EACH ECC CSR. NOTE THE "P" AT BANK 30; THIS POINTS OUT THE "SHADOW" PROTECTION WHICH OCCURS WHEN TWO MS11-M MEMORIES ARE INTERLEAVED. THEREFORE, BANK 30 WILL NOT BE TESTED UNTIL THE PROGRAM HAS RELOCATED.

7.4 EVERYTHING YOU'VE ALWAYS WANTED TO KNOW ABOUT SUPERMAC ...

SUPER-MAC IS A SET OF STRUCTURED PROGRAMMING MACROS THAT ALLOWS PROGRAMS TO BE WRITTEN IN A HIGH LEVEL, EASILY UNDERSTOOD LANGUAGE.

AS A GENERAL RULE, MOST SUPER-MAC STATEMENTS CAN BE SINGLE-LINE STATEMENTS OR MULTIPLE-LINE (NESTED) BLOCK STATEMENTS. A SINGLE-LINE STATEMENT MUST BE COMPLETED ON ONE SOURCE LINE; NO CONTINUATION LINES ARE ALLOWED. SINGLE-LINE STATEMENTS SHOULD BE AS SHORT AND SIMPLE AS POSSIBLE. COMMENTS MAY ALSO BE INCLUDED ON A SOURCE LINE. ALL THE GENERAL RULES, CONDITIONS, ETC., THAT GOVERN MACRO-11 ALSO GOVERN SUPER-MAC. SPACING ON A SOURCE LINE IS VERY IMPORTANT. THE ELEMENTS SHOULD BE SEPARATED BY A COMMA OR A SPACE. TABS SHOULD NEVER BE USED FOR SPACING. FOR EXAMPLE: THE EXPRESSION A+B IS INTERPRETED DIFFERENT THAN A + B.

ALL THE CONDITIONAL STATEMENTS CAN BE WRITTEN AS MULTIPLE-LINE NESTED BLOCKS. EACH LEVEL OF NESTING WITHIN A BLOCK MUST BE TERMINATED WITH AN ASSOCIATED END STATEMENT. EACH LEVEL OF NESTING SHOULD BE INDENTED TWO SPACES.

USER WRITTEN MACROS OR ASSEMBLY LANGUAGE INSTRUCTIONS MAY BE INCLUDED IN A PROGRAM IF DESIRED. AS A DEBUGGING AID, IF THE SYMBOL LST\$\$ IS DEFINED, IT WILL CAUSE GENERATED CODE AND LABELS TO BE LISTED. ALL PROGRAMS MUST BEGIN WITH THE MACRO CALL SMACIT. THIS CALL INITIALIZES SUPER-MAC. ALL LEGAL PDP-11 SOURCE AND DESTINATION OPERANDS ARE LEGAL IN SUPER-MAC.

3995
3996
3997
3998
3999
4000
4001
4002
4003
4004
4005
4006
4007
4008
4009
4010
4011
4012
4013
4014
4015
4016
4017
4018
4019
4020
4021
4022
4023
4024
4025
4026
4027
4028
4029
4030
4031
4032
4033
4034
4035
4036
4037
4038
4039
4040
4041
4042
4043
4044
4045
4046
4047
4048

7.4.1 SAMPLE SOURCE FILE -

```
.ENABL ABS
.ENABL AMA
.MCALL .SUPER
.SUPER
.LST$$=0
BITS=40
A: 0
B: 0
C: 0
D: 0
E: 0
F: 0
G: 0
H: 0
I: 0
J: 0
.PAGE
:LET EXAMPLES
LET R0 := A
LET B := C + D
LET E := F + 1
LET G := H + 2
LET J := J + 01
LET A :B= B
:IF EXAMPLES
IF A IS TRUE
MOV 23,D
END ;OF IF A
IF B IS FALSE
MOV 34,E
END ;OF IF B
IF A EQ B THEN LET C := D
IF A LT B
MOV C,D
ELSE
MOV E,D
END ;OF IF A
IF A EQ B AND C NE D
MOV F,G
END ;OF IF A
IF A EQ B OR C NE D
MOV F,G
END ;OF IF A
IFB A EQ B AND C EQ 1
MOV H,J
ELSE
MOV E,J
END ;OF IFB A
IFB A EQ B ANDB C EQ 1
MOV H,J
ELSE
MOV E,J
```

```

4050                                     END ;OF IFB A
4051                                     IF RESULT IS EQ
4052                                     MOV A,B
4053                                     END ;OF IF RESULT
4054                                     IF BITS SET.IN A
4055                                     MOV B,C
4056                                     END ;OF IF BITS
4057                                     IF BITS OFF.IN A
4058                                     MOV C,D
4059                                     END ;OF IF BITS
4060 :ON.ERROR IS LIKE AN IF STATEMENT ON THE C-BIT
4061 :ON.ERROR EXAMPLES
4062 ON.ERROR
4063     MOV A,B
4064 ELSE
4065     MOV C,B
4066 END ;OF ON.ERROR
4067 ON.NOERROR
4068
4069
4070     MOV C,B
4071 ELSE
4072     MOV A,B
4073 END ;OF ON.NOERROR
4074 ON.ERROR THEN LET A :B= B
4075 :FOR EXAMPLES
4076 FOR I := -5 TO 23
4077     INC A
4078 END ;OF FOR I
4079 FOR RO := 0 TO 140 BY 4
4080     DEC A(RO)
4081 END ;OF FOR RO
4082 FOR I := 133 DOWNT0 3 BY 2
4083     ADD A,B
4084 END ;OF FOR I
4085 :BEGIN EXAMPLES
4086 BEGIN ALPHA
4087     FOR RO := 0 TO 167
4088         MOV B A(RO),B
4089         IF B LT 0 THEN LEAVE ALPHA
4090     END ;OF FOR RO
4091     FOR RO := 400 TO 567
4092         IF B GE 0 THEN LEAVE ALPHA
4093     END ;OF FOR RO
4094 END ALPHA
4095 :$RETURN EXAMPLES
4096 $RETURN
4097 $RETURN ERROR
4098 $RETURN NOERROR
4099 :CASE EXAMPLES
4100 MOV A,RO
4101 CASE RO
4102     A

```

4104
4105
4106
4107
4108
4109
4110
4111
4112
4113
4114
4115
4116
4117
4118
4119
4120
4121
4122
4123
4124
4125
4126
4127
4128
4129
4130
4131
4132
4133
4134
4135
4136
4137
4138
4139
4140
4141
4142
4143
4144
4145
4146
4147
4148
4149
4150
4151
4152
4153
4154
4155
4156

B
C
D
E
F
END ;OF CASE R0

.END

7.4.2 SAMPLE LISTING FILE (WITH NO EXPANDED MACROS) - -
.MAIN. MACRO M1111 01-APR-79 16:41 PAGE 2

1	000000				.ENABL ABS
2					.ENABL AMA
3					.MCALL .SUPER
4	000000				.SUPER
5					:LST\$\$=0
6		000040			BIT5=40
7	000000	000000		A:	0
8	000002	000000		B:	0
9	000004	000000		C:	0
10	000006	000000		D:	0
11	000010	000000		E:	0
12	000012	000000		F:	0
13	000014	000000		G:	0
14	000016	000000		H:	0
15	000020	000000		I:	0
16	000022	000000		J:	0

.MAIN. MACRO M1111 01-APR-79 16:41 PAGE 3

18					:LET EXAMPLES
19	000024				LET RO := A
20	000030				LET B := C + D
21	000044				LET E := F + 1
22	000056				LET G := H + 2
23	000072				LET J := J + J1
24	000100				LET A :B= B
25					:IF EXAMPLES
26	000106				IF A IS TRUE
27	000114	012737	000023	000006	MOV 23,D
28	000122				END ;OF IF A
29	000122				IF B IS FALSE
30	000130	012737	000034	000010	MOV 34,E
31	000136				END ;OF IF B
32	000136				IF A EQ B THEN LET C := D
33	000154				IF A LT B
34	000164	013737	000004	000006	MOV C,D
35	000172				ELSE

4158	36	000174	013737	000010	000006
4159	37	000202			
4160	38	000202			
4161	39	000222	013737	000012	000014
4162	40	000230			
4163	41	000230			
4164	42	000250	013737	000012	000014
4165	43	000256			
4166	44	000256			
4167	45	000276	013737	000016	000022
4168	46	000304			
4169	47	000306	013737	000010	000022
4170	48	000314			
4171	49	000314			
4172	50	000334	013737	000016	000022
4173	51	000342			
4174	52	000344	013737	000010	000022
4175	53	000352			
4176	54	000352			
4177	55	000354	013737	000000	000002
4178	56	000362			
4179	57	000362			
4180	58	000372	013737	000002	000004
4181	59	000400			
4182	60	000400			
4183	61	000410	013737	000004	000006
4184	62	000416			
4185	63				
4186	64				
4187	65	000416			
4188	66	000420	013737	000000	000002
4189	67	000426			
4190	68	000430	013737	000004	000002
4191	69	000436			
4192	70	000436			
4193	71	000440	013737	000004	000002
4194	72	000446			
4195	73	000450	013737	000000	000002
4196	74	000456			

```

MOV E,D
END ;OF IF A
IF A EQ B AND C NE D
MOV F,G
END ;OF IF A
IF A EQ B OR C NE D
MOV F,G
END ;OF IF A
IFB A EQ B AND C EQ 1
MOV H,J
ELSE
MOV E,J
END ;OF IFB A
IFB A EQ B ANDB C EQ 1
MOV H,J
ELSE
MOV E,J
END ;OF IFB A
IF RESULT IS EQ
MOV A,B
END ;OF IF RESULT
IF BITS SET.IN A
MOV B,C
END ;OF IF BITS
IF BITS OFF.IN A
MOV C,D
END ;OF IF BITS
;ON.ERROR IS LIKE AN IF STATEMENT ON THE C-BIT
;ON.ERROR EXAMPLES
ON.ERROR
MOV A,B
ELSE
MOV C,B
END ;OF ON.ERROR
ON.NOERROR
MOV C,B
ELSE
MOV A,B
END ;OF ON.NOERROR

```

.MAIN. MACRO M1111 01-APR-79 16:41 PAGE 3-1

4200					
4201					
4202	75	000456			
4203	76				
4204	77	000466			
4205	78	000474	005237	000000	
4206	79	000500			
4207	80	000514			
4208	81	000516	005360	000000	
4209	82	000522			
4210	83	000534			

```

ON.ERROR THEN LET A :B= B
;FOR EXAMPLES
FOR I := -5 TO 23
INC A
END ;OF FOR I
FOR R0 := 0 TO 140 BY 4
DEC A(R0)
END ;OF FOR R0
FOR I := 133 DOWNT0 3 BY 2

```

```

4212      84 000542 063737 000000 000002
4213      85 000550
4214      86
4215      87 000566
4216      88 000566
4217      89 000570 116037 000000 000002
4218      90 000576
4219      91 000604
4220      92 000614
4221      93 000620
4222      94 000626
4223      95 000636
4224      96
4225      97 000636
4226      98 000640
4227      99 000644
4228     100
4229     101 000650 013700 000000
4230     102 000654
4231     103 000664 000000
4232     104 000666 000002
4233     105 000670 000004
4234     106 000672 000006
4235     107 000674 000010
4236     108 000676 000012
4237     109 000700
4238     110
4239     111      000001

```

```

      ADD A,B
      END ;OF FOR I
;BEGIN EXAMPLES
      BEGIN ALPHA
      FOR RO := 0 TO 167
      MOV B A(RO),B
      IF B LT 0 THEN LEAVE ALPHA
      END ;OF FOR RO
      FOR RO := 400 TO 567
      IF B GE 0 THEN LEAVE ALPHA
      END ;OF FOR RO
      END ALPHA
;SRETURN EXAMPLES
      SRETURN
      SRETURN ERROR
      SRETURN NOERROR
;CASE EXAMPLES
      MOV A,RO
      CASE RO
      A
      B
      C
      D
      E
      F
      END ;OF CASE RO
.END

```

7.4.3 SAMPLE LISTING FILE (WITH EXPANDED MACROS) - -
.MAIN. MACRO M1111 01-APR-79 16:10 PAGE 2

```

4244      1 000000
4245      2
4246      3
4247      4 000000
4248      5      000000
4249      6      000040
4250      7 000000 000000
4251      8 000002 000000
4252      9 000004 000000
4253     10 000006 000000
4254     11 000010 000000
4255     12 000012 000000
4256     13 000014 000000
4257     14 000016 000000
4258     15 000020 000000
4259     16 000022 000000

```

```

.ENABL ABS
.ENABL AMA
.MCALL .SUPER
.SUPER
LST$$=0
BITS=40
A: 0
B: 0
C: 0
D: 0
E: 0
F: 0
G: 0
H: 0
I: 0
J: 0

```

```

4260
4261
4262
4263

```

4265	18				
4266	19	000024	013700	000000	
4267		000024			
4268	20	000030			
4269		000030	013737	000004	000002
4270		000036	063737	000006	000002
4271	21	000044			
4272		000044	013737	000012	000010
4273		000052	005237	000010	
4274	22	000056			
4275		000056	013737	000016	000014
4276		000064	062737	000002	000014
4277	23	000072			
4278		000072	062737	000001	000022
4279	24	000100			
4280		000100	113737	000002	000000
4281	25				
4282	26	000106			
4283		000106	005737	000000	
4284		000112	001403		
4285	27	000114	012737	000023	000006
4286	28	000122			
4287		000122			
4288	29	000122			
4289		000122	005737	000002	
4290		000126	001003		
4291	30	000130	012737	000034	000010
4292	31	000136			
4293		000136			
4294	32	000136			
4295		000136	023737	000000	000002
4296		000144	001003		
4297		000146	013737	000006	000004
4298		000154			
4299	33	000154			
4300		000154	023737	000000	000002
4301		000162	002004		
4302	34	000164	013737	000004	000006
4303	35	000172			
4304		000172	000403		
4305		000174			
4306	36	000174	013737	000010	000006
4307	37	000202			
4308		000202			
4309	38	000202			
4310		000202	023737	000000	000002
4311		000210	001007		
4312		000212	023737	000004	000006
4313		000220	001403		
4314	39	000222	013737	000012	000014
4315	40	000230			

```

;LET EXAMPLES
LET R0 := A
MOV A,R0
LET B := C + D
MOV C,B
ADD D,B
LET E := F + 1
MOV F,E
INC E
LET G := H + 2
MOV H,G
ADD 2,G
LET J := J + 01
ADD 01,J
LET A := B
MOVB B,A

;IF EXAMPLES
IF A IS TRUE
TST A
BEQ L0
MOV 23,D
END ;OF IF A
L0:
IF B IS FALSE
TST B
BNE L1
MOV 34,E
END ;OF IF B
L1:
IF A EQ B THEN LET C := D
CMP A,B
BNE L2
MOV D,C
L2:
IF A LT B
CMP A,B
BGE L3
MOV C,D
ELSE
BR L4
L3:
MOV E,D
END ;OF IF A
L4:
IF A EQ B AND C NE D
CMP A,B
BNE L5
CMP C,D
BEQ L5
MOV F,G
END ;OF IF A

```


4317
4318
4319
4320
4321
4322
4323
4324
4325
4326
4327
4328
4329
4330
4331
4332
4333
4334
4335
4336
4337
4338
4339
4340
4341
4342
4343
4344
4345
4346
4347
4348
4349
4350
4351
4352
4353
4354
4355
4356
4357
4358
4359
4360
4361
4362
4363
4364
4365
4366
4367
4368
4369

```

000230
41 000230
000230 023737 000000 000002
000236 001404
000240 023737 000004 000006
000246 001403
    
```

L5:

```

IF A EQ B OR C NE D
CMP A,B
BEQ L6
CMP C,D
BEQ L7
    
```

.MAIN. MACRO M1111 01-APR-79 16:10 PAGE 3-1

```

000250
42 000250 013737 000012 000014
43 000256
000256
44 000256
000256 123737 000000 000002
000264 001010
000266 023727 000004 000001
000274 001004
45 000276 013737 000016 000022
46 000304
000304 000403
000306
47 000306 013737 000010 000022
48 000314
000314
49 000314
000314 123737 000000 000002
000322 001010
000324 123727 000004 000001
000332 001004
50 000334 013737 000016 000022
51 000342
000342 000403
000344
52 000344 013737 000010 000022
53 000352
000352
54 000352
000352 001003
55 000354 013737 000000 000002
56 000362
000362
57 000362
000362 032737 000040 000000
000370 001403
58 000372 013737 000002 000004
59 000400
000400
60 000400
000400 032737 000040 000000
    
```

L6:

```

MOV F,G
END ;OF IF A
    
```

L7:

```

IFB A EQ B AND C EQ 1
CMPB A,B
BNE L10
CMP C,1
BNE L10
MOV H,J
ELSE
BR L11
    
```

L10:

```

MOV E,J
END ;OF IFB A
    
```

L11:

```

IFB A EQ B ANDB C EQ 1
CMPB A,B
BNE L12
CMPB C,1
BNE L12
MOV H,J
ELSE
BR L13
    
```

L12:

```

MOV E,J
END ;OF IFB A
    
```

L13:

```

IF RESULT IS EQ
BNE L14
MOV A,B
END ;OF IF RESULT
    
```

L14:

```

IF BITS SET.IN A
BIT BITS,A
BEQ L15
    
```

L15:

```

MOV B,C
END ;OF IF BITS
IF BITS OFF.IN A
BIT BITS,A
    
```

4371	61	000410	013737	000004	000006	MOV C,D
4372						
4373	62	000416				END ;OF IF BITS
4374		000416				
4375	63					L16:
4376	64					;ON.ERROR IS LIKE AN IF STATEMENT ON THE C-BIT
4377	65	000416				;ON.ERROR EXAMPLES
4378		000416	103004			ON.ERROR
4379	66	000420	013737	000000	000002	BCC L17
4380	67	000426				MOV A,B
4381		000426	000403			ELSE
4382		000430				BR L20
4383						L17:
4384	68	000430	013737	000004	000002	MOV C,B
4385						
4386	69	000436				END ;OF ON.ERROR
4387		000436				L20:
4388	70	000436				ON.NOERROR

.MAIN. MACRO M1111 01-APR-79 16:10 PAGE 3-2

4390						
4391						
4392						
4393						
4394		000436	103404			BCS L21
4395	71	000440	013737	000004	000002	MOV C,B
4396	72	000446				ELSE
4397		000446	000403			BR L22
4398		000450				L21:
4399	73	000450	013737	000000	000002	MOV A,B
4400	74	000456				END ;OF ON.NOERROR
4401		000456				L22:
4402	75	000456				ON.ERROR THEN LET A :B= B
4403		000456	103003			BCC L23
4404		000460	113737	000002	000000	MOVB B,A
4405		000466				L23:
4406	76					;FOR EXAMPLES
4407	77	000466				FOR I := -5 TO 23
4408		000466	012737	177773	000020	MOV -5,I
4409		000474				B0:
4410	78	000474	005237	000000		INC A
4411	79	000500				END ;OF FOR I
4412		000500	005237	000020		INC I
4413		000504	023727	000020	000023	CMP I, 23
4414		000512	003770			BLE B0
4415		000514				E0:
4416	80	000514				FOR RO := 0 TO 140 BY 4
4417		000514	005000			CLR RO
4418		000516				B1:
4419	81	000516	005360	000000		DEC A(RO)
4420	82	000522				END ;OF FOR RO
4421		000522	062700	000004		ADD 4,RO
4422		000526	020027	000140		CMP RO, 140

4424		000532	003771						
4425		000534							
4426		000534							
4427		000534	012737	000133	000020				
4428		000542							
4429		000542	063737	000000	000002				
4430		000550							
4431		000550	162737	000002	000020				
4432		000556	023727	000020	000003				
4433		000564	002366						
4434		000566							
4435									
4436									
4437									
4438									
4439									
4440									
4441									
4442									
4443									
4444									
4445									
4446									
4447									
4448									
4449									
4450									
4451									
4452									
4453									
4454									
4455									
4456									
4457									
4458									
4459									
4460									
4461									
4462									
4463									
4464									
4465									
4466									
4467									
4468									
4469									
4470									
4471									
4472									
4473									
4474									
4475									
4476									

```

E1: BLE B1
    FOR I := 133 DOWNT0 3 BY 2
    MOV 133,I
B2:   ADD  A,B
    END ;OF FOR I
    SUB 2,I
    CMP I,3
    BGE B2
E2:   ;BEGIN EXAMPLES
    BEGIN ALPHA
B3:   FOR R0 := 0 TO 167
    CLR R0
B4:   MOV B A(R0),B
    IF B LT 0 THEN LEAVE ALPHA
    TST B
    BLT E3
    END ;OF FOR R0
    INC R0
    CMP R0, 167
    BLE B4
E4:   FOR R0 := 400 TO 567
    MOV 400,R0

```

.MAIN. MACRO M1111 01-APR-79 16:10 PAGE 3-3

4457		000620							
4458		000620							
4459		000620	005737	000002					
4460		000624	002004						
4461		000626							
4462		000626	005200						
4463		000630	020027	000567					
4464		000634	003771						
4465		000636							
4466		000636							
4467		000636							
4468									
4469									
4470									
4471									
4472									
4473									
4474									
4475									
4476									

```

B5:   IF B GE 0 THEN LEAVE ALPHA
    TST B
    BGE E3
    END ;OF FOR R0
    INC R0
    CMP R0, 567
    BLE B5
E5:   END ALPHA
E3:   ;$RETURN EXAMPLES
    $RETURN
    RTS PC
    $RETURN ERROR
    SEC
    RTS PC
    $RETURN NOERROR
    CLC
    RTS PC

```



```

4478      100
4479      101 000650 013700 000000
4480      102 000654
4481          000654 010046
4482          000656 006316
4483          000660 004737 000700
4484      103 000664 000000
4485      104 000666 000002
4486      105 000670 000004
4487      106 000672 000006
4488      107 000674 000010
4489      108 000676 000012
4490      109 000700
4491          000700
4492          000700 062616
4493          000702 013646
4494          000704 004736
4495      110
4496      111          000001
    
```

```

;CASE EXAMPLES
MOV      A,R0
CASE RO
MOV RO,-(SP)
ASL @SP
JSR PC,L24
A
B
C
D
E
F
END ;OF CASE RO
L24:
ADD (SP)+,@SP
MOV @ (SP)+,-(SP)
JSR PC,@(SP)+
.END
    
```

7.5 MEMORY MANAGEMENT MAPPING

7.5.1 MEMORY MANAGEMENT MAPPING FOR THE 11/44 -

PAR	SUPERVISOR	KERNEL	USER
0	PROGRAM	PROGRAM	DST BK/FST MEM
1	PROGRAM	PROGRAM	SRC BK/FST MEM
2	PROGRAM	PROGRAM	SRC BK/FST MEM
3	TEST AREA	PROGRAM	SRC BK/FST MEM
4	TEST AREA	PROGRAM	DST BK/FST MEM
5	TEST AREA	PROGRAM	DST BK/FST MEM
6	TEST AREA	MAP TO CSR'S	DST BK/FST MEM
7	PERIF PAGE	PERIF PAGE	DST BK/FST MEM

7.5.2 MEMORY MANAGEMENT MAPPING FOR UNIBUS-11'S WITH SUPERVISOR MODE (EG 11/45) -

PAR	SUPERVISOR	KERNEL	USER
0	PROGRAM	PROGRAM	DST BK
1	PROGRAM	PROGRAM	SRC BK
2	PROGRAM	PROGRAM	SRC BK
3	TEST AREA	PROGRAM	SRC BK
4	TEST AREA	PROGRAM	DST BK
5	TEST AREA	PROGRAM	DST BK
6	TEST AREA	MAP TO CSR'S	DST BK
7	PERIF PAGE	PERIF PAGE	DST BK

4500
4501
4502
4503
4504
4505
4506
4507
4508
4509
4510
4511
4512
4513
4514
4515
4516
4517
4518
4519
4520
4521
4522
4523
4524
4525
4526
4527

4529
4530
4531
4532
4533
4534
4535
4536
4537
4538
4539
4540
4541

7.5.3 MEMORY MANAGEMENT MAPPING FOR UNIBUS-11'S W/O SUPERVISOR MODE (EG 11/34) -

PAR	KERNEL	USER
---	-----	----
0	PROGRAM	PROGRAM/DST BK
1	PROGRAM	PROGRAM/SRC BK
2	PROGRAM	PROGRAM/SRC BK
3	PROGRAM	TEST AREA/SRC BK
4	PROGRAM	TEST AREA/DST BK
5	PROGRAM	TEST AREA/DST BK
6	MAP TO CSR'S	TEST AREA/DST BK
7	PERIF PAGE	PERIF PAGE/DST BK

4545
 4546 000000
 4547
 4548
 4549
 4550
 4551
 4552
 4553
 4554
 4555
 4556
 4557
 4558
 4559
 4560
 4561
 4562
 4563
 4564 163000
 4565 000001
 4566 000000

```
.LIST TOC
.ENABL ABS
.ENABL AMA
.DSABL GBL
:NOTE: CZMSDC.SML IS THE SUPER.MAC SOURCE AND IS RELEASED WITH
:THIS PROGRAM. ALL THESE .MCALL STATEMENTS REFERENCE THAT FILE.
.MCALL SMACIT,..PUSH,..POP,..TAG,..BRAN,..EMIT,..EMITN,..EMITL,..EMITR
.MCALL .IFOPR,..IS,..GENBR,..OPADD,..OPSUB,CLEAR,SET,CLEARB,SETB
.MCALL RNE,REQ,RLT,RGE,RGT,RLE,RPL,RMI,RHI,RLOS,RHIS,RLO,RCS,RCC
.MCALL IF,..OR,..IFARI,..LEAVE,..GOTO,OR,AND,THEN,ELSE,WHILE,CASE
.MCALL FOR,TO,DOWNTO,REPEAT,UNTIL,THRU,END,BEGIN
.MCALL $$SEND,LEAVE,JUMPTO,GOTO,PUSH,POP,LET
.MCALL .SIMPLE,..ARITH,ORB,ANDB,IFB,UNTILB,WHILEB,ON.ERROR,ON.NOERROR
.MCALL $CALL,$RETURN

.NLIST TTM
.LIST MC,SYM
.NLIST MD,CND,ME
LST$$= 0
$SWR= 163000
$TN= 1
SMACIT
```

```
:I WANT FAT PAPER!
:LIST MACRO CALLS, SYMBOL TABLE
:DON'T LIST MACRO DEFS & CONDITIONALS & EXPANSIONS
:DEFINED TO LIST SUPERMAC EXPANSIONS
:USE THESE SYSMAC SWITCHES
:FIRST TEST NUMBER TO ONE(1)
```


4569
4570
4571
4572
4573
4574
4575
4576
4577
4578
4579
4580
4581
4582
4583
4584
4585
4586
4587
4588
4589
4590
4591
4592
4593
4594
4595
4596
4597
4598
4599
4600
4601
4602
4603
4604
4605
4606
4607
4608
4609
4610
4611
4612
4613
4614
4615
4616
4617
4618
4619
4620
4621
4622
4623
4624
4625

```

.SBTTL DEFINE TRAPS
:ALL ENTRIES HERE MUST HAVE A CORRESPONDING ENTRY IN THE
:TRAP TABLE '$TRPAD' (NEAR END OF PROGRAM).
:*TRAP DEFINITIONS
:
:HERE IS HOW TRAPS WORK IN THIS PROGRAM
:
:ALL TRAPS EXECUTE A 'TRAP' INSTRUCTION WHICH TAKES THE PROGRAM
:TO SYMBOLIC LOCATION '$STRAP'
:
:AT $STRAP THE PROGRAM PICKS UP THE RIGHT BYTE OF THE TRAP INSTRUCTION
:AND INDEXES INTO A TABLE AT LOCATION '$TRPAD' WHICH SENDS THE PROGRAM TO
:THE SPECIFIC ROUTINE TO HANDLE THAT SPECIFIC TRAPS TASK.
:
:THE ULTIMATE DESTINATION OF A TRAP INSTRUCTION CAN BE GUESSED AT AS FOLLOWS
    
```

```

:EXAMPLE:      NOP
               NOP
               NOP
               KERNEL           :ENTER KERNEL MODE
               NOP
    
```

```

:      ADD A DOLLAR SIGN TO THE SYMBOLIC NAME AND CHECK THE CRF FOR SOMETHING CLOSE
:      IN THIS CASE THE CRF HAS $KERNE LISTED AS 032546
:      AT LOCATION 32546 YOU FIND THE ROUTINE $KERNEL
    
```

```

:      NOTE THAT CRF SYMBOLS ARE TRUCNATED TO 6 CHARACTERS
:      SYMBOLIC NAMES GREATER THAT 6 CHARACTERS ARE USED SO I CAN
:      REMEMBER WHAT THEY MEAN!
    
```

```

:      ALL GOOD TRAP ROUTINES RETURN VIA AN 'RTI' INSTRUCTION
TYPEIT= 104401      ;;TTY TYPEOUT ROUTINE
TYPOC= 104402      ;;TYPE OCTAL NUMBER (WITH LEADING ZEROS)
TYPOS= 104403      ;;TYPE OCTAL NUMBER (NO LEADING ZEROS)
:TYPON= 104404      ;;TYPE OCTAL NUMBER (AS PER LAST CALL)
TYPDS= 104405      ;;TYPE DECIMAL NUMBER (WITH SIGN)
:TYPBN= 104406      ;;TYPE BINARY (ASCII) NUMBER
    
```

```

GTSWR= 104407      ;;GET SOFT-SWR SETTING
CKSWR= 104410      ;;TEST FOR CHANGE IN SOFT-SWR
    
```

```

RDCHR= 104411      ;;TTY TYPEIN CHARACTER ROUTINE
RDLIN= 104412      ;;TTY TYPEIN STRING ROUTINE
RDOCT= 104413      ;;READ AN OCTAL NUMBER FROM TTY
RDDEC= 104414      ;;READ A DECIMAL NUMBER FROM TTY
    
```

```

SAVREG= 104415     ;;SAVE R0-R5 ROUTINE
RESREG= 104416     ;;RESTORE R0-R5 ROUTINE
    
```

```

KERNEL= 104417     ;ENTER KERNEL MODE
    
```

```

ENERGIZE=104420    ;TURN ON MEMORY MANAGEMENT & TRAPS
DEENERGIZE=104421 ;TURN OFF MEMORY MANAGEMENT & TRAPS
KMAP= 104422      ;MAP KERNEL 1 TO 1
    
```

```

CACHON= 104423    ;TURN ON CACHE
CACHOFF=104424   ;TURN OFF CACHE
    
```

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 110-1
 DEFINE TRAPS

4626			
4627	104425	LOADCSR=104425	:LOAD CORRECT CSR
4628	104426	READCSR=104426	:READ CORRECT CSR
4629			
4630	104427	PERR01= 104427	:PROGRAM DETECTED ERROR
4631	104430	PERR02= 104430	:PROGRAM DETECTED ERROR
4632	104431	PERR03= 104431	:PROGRAM DETECTED ERROR
4633	104432	PERR04= 104432	:PROGRAM DETECTED ERROR
4634	104433	PERR07= 104433	:PROGRAM DETECTED ERROR
4635	104434	PERR10= 104434	:PROGRAM DETECTED ERROR
4636	104435	PERR11= 104435	:PROGRAM DETECTED ERROR
4637	104436	PERR12= 104436	:PROGRAM DETECTED ERROR
4638	104437	PERR13= 104437	:PROGRAM DETECTED ERROR
4639	104440	PERR14= 104440	:PROGRAM DETECTED ERROR
4640	104441	PERR15= 104441	:PROGRAM DETECTED ERROR
4641	104442	PERR16= 104442	:PROGRAM DETECTED ERROR
4642	104443	PERR17= 104443	:PROGRAM DETECTED ERROR
4643	104444	PERR20= 104444	:PROGRAM DETECTED ERROR
4644	104445	PERR21= 104445	:PROGRAM DETECTED ERROR
4645	104446	PERR22= 104446	:PROGRAM DETECTED ERROR
4646	104447	PERR23= 104447	:PROGRAM DETECTED ERROR
4647	104450	PERR24= 104450	:PROGRAM DETECTED ERROR
4648	104451	PERR25= 104451	:PROGRAM DETECTED ERROR
4649	104452	PERR26= 104452	:PROGRAM DETECTED ERROR
4650	104453	PERR27= 104453	:PROGRAM DETECTED ERROR
4651	104454	PERR30= 104454	:PROGRAM DETECTED ERROR
4652	104455	PERR31= 104455	:PROGRAM DETECTED ERROR
4653	104456	PERR32= 104456	:PROGRAM DETECTED ERROR
4654	104457	PERR33= 104457	:PROGRAM DETECTED ERROR
4655	104460	PERR34= 104460	:PROGRAM DETECTED ERROR
4656	104461	PERR35= 104461	:PROGRAM DETECTED ERROR
4657	104462	PERR36= 104462	:PROGRAM DETECTED ERROR
4658	104463	PERR37= 104463	:PROGRAM DETECTED ERROR
4659	104464	PERR40= 104464	:PROGRAM DETECTED ERROR
4660	104465	PERR41= 104465	:PROGRAM DETECTED ERROR
4661	104466	PERR42= 104466	:PROGRAM DETECTED ERROR
4662	104467	PERR43= 104467	:PROGRAM DETECTED ERROR
4663			
4664	104470	ECCDIS= 104470	:DISABLE ECC ON ALL CSR'S
4665	104471	ECC1DIS=104471	:DISABLE ECC ON 1 SELECTED CSR
4666	104472	ECCINIT=104472	:INITIALIZE ALL ECC CSR'S
4667	104473	ECC1INIT=104473	:INITIALIZE 1 SELECTED ECC CSR
4668	104474	CBCSR= 104474	:WRITE GENERATED CHECKBITS IN ALL CSR'S
4669	104475	CB1CSR= 104475	:WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
4670	104476	WASSBE= 104476	:WAS THERE A SBE ON ANY CSR?
4671	104477	WAS1SBE=104477	:WAS THERE A SBE ON 1 SELECTED CSR?
4672	104500	WASDBE= 104500	:WAS THERE A DBE ON ANY CSR?
4673	104501	WAS1DBE=104501	:WAS THERE A DBE ON 1 SELECTED CSR?
4674	104502	CLRCR= 104502	:CLEAR ALL CSR'S
4675	104503	CLR1CSR=104503	:CLEAR 1 SELECTED CSR
4676	104504	CHKDIS= 104504	:DISABLE ECC & WRITE CHECKBITS FROM ALL CSR'S
4677	104505	CHK1DIS=104505	:DISABLE ECC & WRITE CHECKBITS FROM 1 SELECTED CSR
4678	104506	ENASBE= 104506	:ENABLE TRAPS ON SBE'S FROM ALL CSR'S
4679	104507	ENA1SBE=104507	:ENABLE TRAPS ON SBE'S FROM 1 SELECTED CSR
4680	104510	TSTREAD=104510	:TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
4681	104511	INVALID=104511	:INVALIDATE BACKGROUND PATTERN ON "BANK"
4682	104512	ERRGEN =104512	:CHECK ERROR ADDRESS

4683

104513

CBREG =104513

;ENABLES CHECK/SYNDROME BIT REGISTER

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 111
 DEFINE BASIC PDP11 STUFF

```

4685                                     .SBTTL DEFINE BASIC PDP11 STUFF
4686
4687                                     :*INITIAL ADDRESS OF THE STACK POINTER
4688         002000      STACK= 2000          ;;FIRST ADDRESS OF THE STACK
4689         002000      KERSTK= STACK        ;;KERNEL STACK
4690         000740      SUPSTK= 740         ;;SUPERVISOR STACK
4691         000700      USESTK= 700         ;;USER STACK
4692         104000      ERROR=EMT           ;;BASIC DEFINITION OF ERROR CALL
4693         000004      SCOPE=IOT           ;;BASIC DEFINITION OF SCOPE CALL
4694         177776      PSW= 177776         ;;PROCESSOR STATUS WORD
4695         :STKLMT=177774                   ;;STACK LIMIT REGISTER
4696         :PIRQ= 177772                   ;;PROGRAM INTERRUPT REQUEST REGISTER
4697         177570      DSWR= 177570        ;;HARDWARE SWITCH REGISTER
4698         177570      DDISP= 177570       ;;HARDWARE DISPLAY REGISTER
4699         177546      LKS= 177546         ;;LINE CLOCK (KW11-L) STATUS REGISTER
4700
4701                                     :*MISCELLANEOUS DEFINITIONS
4702         000011      HT= 11              ;;CODE FOR HORIZONTAL TAB
4703         000012      LF= 12              ;;CODE LINE FEED
4704         000015      CR= 15              ;;CODE CARRIAGE RETURN
4705         000200      CRLF= 200           ;;CODE FOR CARRIAGE RETURN-LINE FEED
4706         000007      MFPT= 7            ;;CODE FOR PROCESSOR TYPE INSTRUCTION
4707
4708                                     :*GENERAL PURPOSE REGISTER DEFINITIONS
4709         :SP=R6          ;;STACK POINTER
4710         :KSP=SP       ;;KERNEL STACK POINTER
4711         000006      SSP=SP              ;;SUPERVISOR STACK POINTER
4712         000006      USP=SP             ;;USER STACK POINTER
4713         :PC=R7        ;;PROGRAM COUNTER
4714
4715                                     :*"SWITCH REGISTER" SWITCH DEFINITIONS
4716         100000      SW15= 100000
4717         040000      SW14= 40000
4718         020000      SW13= 20000
4719         010000      SW12= 10000
4720         004000      SW11= 4000
4721         002000      SW10= 2000
4722         001000      SW9= 1000
4723         000400      SW8= 400
4724         000200      SW7= 200
4725         000100      SW6= 100
4726         000040      SW5= 40
4727         000020      SW4= 20
4728         000010      SW3= 10
4729         000004      SW2= 4
4730         000002      SW1= 2
4731         000001      SW0= 1
4732
4733                                     :*DATA BIT DEFINITIONS (BIT00 TO BIT15)
4734         100000      BIT15= 100000
4735         040000      BIT14= 40000
4736         020000      BIT13= 20000
4737         010000      BIT12= 10000
4738         004000      BIT11= 4000
4739         002000      BIT10= 2000
4740         001000      BIT9= 1000
4741         000400      BIT8= 400

```

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 111-1
 DEFINE BASIC PDP11 STUFF

```

4742      000200      BIT7= 200
4743      000100      BIT6= 100
4744      000040      BIT5= 40
4745      000020      BIT4= 20
4746      000010      BIT3= 10
4747      000004      BIT2= 4
4748      000002      BIT1= 2
4749      000001      BIT0= 1
4750
4751      : *BASIC "CPU" TRAP VECTOR ADDRESSES
4752      000004      ERRVEC= 4      :: TIME OUT AND OTHER ERRORS
4753      000010      RESVEC= 10     :: RESERVED AND ILLEGAL INSTRUCTIONS
4754      : TBITVEC=14      :: "T" BIT
4755      : TRTVEC= 14      :: TRACE TRAP
4756      : BPTVEC= 14      :: BREAKPOINT TRAP (BPT)
4757      000020      IOTVEC= 20     :: INPUT/OUTPUT TRAP (IOT) **SCOPE**
4758      000024      PWRVEC= 24     :: POWER FAIL
4759      000030      EMTVEC= 30     :: EMULATOR TRAP (EMT) **ERROR**
4760      000034      TRAPVEC=34     :: "TRAP" TRAP
4761      000060      TKVEC= 60      :: TTY KEYBOARD VECTOR
4762      : TPVEC= 64      :: TTY PRINTER VECTOR
4763      : LKVEC= 100     :: LINE CLOCK (KW11-L) VECTOR
4764      000114      CACHVEC=114    :: CACHE ERROR INTERRUPT VECTOR
4765      000114      PARVEC=CACHVEC
4766      : PIRQVEC=240    :: PROGRAM INTERRUPT REQUEST VECTOR
4767      000250      MMVEC= 250     :: MEMORY MANAGEMENT VECTOR
4768      : .SBTTL DEFINE CACHE REGISTERS
4769      : MEMERR = 177744    :: CACHE ERROR REGISTER
4770      177746      CONTRL = 177746 :: MEMORY CONTROL REGISTER
4771      177750      MAINT = 177750  :: MEMORY MAINTENANCE REGISTER
4772      : HITMIS = 177752  :: HIT MISS REGISTER "1" IMPLIES HIT IN CACHE
4773      177754      DATARG = 177754 :: DATA REGISTER
4774
4775      : .SBTTL DEFINE CPU REGISTERS
4776      177766      CPUERR = 177766  :: CPU ERROR REGISTER HOLDS CONDITION THAT CAUSED
4777
4778      : .SBTTL DEFINE MEMORY MANAGEMENT REGISTERS
4779      : *MEMORY MANAGEMENT STATUS REGISTER ADDRESSES
4780      177572      MMRO= 177572
4781      177574      MMR1= 177574
4782      177576      MMR2= 177576
4783      172516      MMR3= 172516
4784
4785      : *USER "I" PAGE DESCRIPTOR REGISTERS
4786      177600      UIPDR0= 177600
4787      : UIPDR1= 177602
4788      : UIPDR2= 177604
4789      : UIPDR3= 177606
4790      : UIPDR4= 177610
4791      : UIPDR5= 177612
4792      : UIPDR6= 177614
4793      : UIPDR7= 177616
4794
4795      : *USER "D" PAGE DESCRIPTOR REGISTORS
4796      : UDPDR0= 177620
4797      : UDPDR1= 177622
4798      : UDPDR2= 177624

```

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 111-2
 DEFINE MEMORY MANAGEMENT REGISTERS

```

4799          :UDPDR3=          177626
4800          :UDPDR4=          177630
4801          :UDPDR5=          177632
4802          :UDPDR6=          177634
4803          :UDPDR7=          177636
4804
4805
4806          177640          : *USER "I" PAGE ADDRESS REGISTERS
FASTCITY=UIPAR0
4807          177640          UIPAR0= 177640          :PATTERN PROGRAM SPACE
4808          177642          UIPAR1= 177642          :PATTERN PROGRAM SPACE
4809          177644          UIPAR2= 177644          :PATTERN PROGRAM SPACE
4810          177646          UIPAR3= 177646          :PATTERN PROGRAM SPACE
4811          177650          UIPAR4= 177650          :PATTERN PROGRAM SPACE
4812          177652          UIPAR5= 177652          :PATTERN PROGRAM SPACE
4813          177654          UIPAR6= 177654          :PATTERN PROGRAM SPACE
4814          :UIPAR7=          177656          :PATTERN PROGRAM SPACE
4815
4816          : *USER "D" PAGE ADDRESS REGISTERS
4817          177660          UDPAR0= 177660          :PATTERN PROGRAM SPACE
4818          :UDPAR1=          177662          :PATTERN PROGRAM SPACE
4819          :UDPAR2=          177664          :PATTERN PROGRAM SPACE
4820          :UDPAR3=          177666          :PATTERN PROGRAM SPACE
4821          :UDPAR4=          177670          :PATTERN PROGRAM SPACE
4822          :UDPAR5=          177672          :PATTERN PROGRAM SPACE
4823          :UDPAR6=          177674          :PATTERN PROGRAM SPACE
4824          177676          UDPAR7= 177676          :PATTERN PROGRAM SPACE
4825
4826          : *SUPERVISOR "I" PAGE DESCRIPTOR REGISTERS
4827          172200          SIPDR0= 172200
4828          :SIPDR1=          172202
4829          :SIPDR2=          172204
4830          :SIPDR3=          172206
4831          :SIPDR4=          172210
4832          :SIPDR5=          172212
4833          :SIPDR6=          172214
4834          :SIPDR7=          172216
4835
4836          : *SUPERVISOR "D" PAGE DESCRIPTOR REGISTERS
4837          :SDPDR0=          172220
4838          :SDPDR1=          172222
4839          :SDPDR2=          172224
4840          :SDPDR3=          172226
4841          :SDPDR4=          172230
4842          :SDPDR5=          172232
4843          :SDPDR6=          172234
4844          :SDPDR7=          172236
4845
4846          : *SUPERVISOR "I" PAGE ADDRESS REGISTERS
4847          172240          SIPAR0= 172240
4848          :SIPAR1=          172242
4849          :SIPAR2=          172244
4850          172246          SIPAR3= 172246          :TEST AREA
4851          :SIPAR4=          172250          :TEST AREA
4852          172252          SIPAR5= 172252          :TEST AREA
4853          172254          SIPAR6= 172254          :TEST AREA
4854          :SIPAR7=          172256
4855

```


CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 111-3
 DEFINE MEMORY MANAGEMENT REGISTERS

```

4856
4857      172260      SDPAR0= :*SUPERVISOR 'D' PAGE ADDRESS REGISTERS
4858      :SDPAR1=      172262
4859      :SDPAR2=      172264
4860      :SDPAR3=      172266
4861      :SDPAR4=      172270
4862      172272      SDPAR5= 172272
4863      172274      SDPAR6= 172274
4864      172276      SDPAR7= 172276
4865
4866      :*KERNEL 'I' PAGE DESCRIPTOR REGISTERS
4867      172300      KIPDR0= 172300
4868      :KIPDR1=      172302
4869      :KIPDR2=      172304
4870      :KIPDR3=      172306
4871      :KIPDR4=      172310
4872      :KIPDR5=      172312
4873      :KIPDR6=      172314
4874      :KIPDR7=      172316
4875
4876      :*KERNEL 'D' PAGE DESCRIPTOR REGISTERS
4877      :KDPDR0=      172320
4878      :KDPDR1=      172322
4879      :KDPDR2=      172324
4880      :KDPDR3=      172326
4881      :KDPDR4=      172330
4882      :KDPDR5=      172332
4883      :KDPDR6=      172334
4884      :KDPDR7=      172336
4885
4886      :*KERNEL 'I' PAGE ADDRESS REGISTERS
4887      172340      KIPAR0= 172340
4888      :KIPAR1=      172342
4889      :KIPAR2=      172344
4890      :KIPAR3=      172346
4891      172350      KIPAR4= 172350
4892      172352      KIPAR5= 172352
4893      172354      KIPAR6= 172354
4894      :KIPAR7=      172356
4895
4896      :*KERNEL 'D' PAGE ADDRESS REGISTERS
4897      172360      KDPAR0= 172360
4898      :KDPAR1=      172362
4899      :KDPAR2=      172364
4900      :KDPAR3=      172366
4901      :KDPAR4=      172370
4902      :KDPAR5=      172372
4903      172374      KDPAR6= 172374
4904      172376      KDPAR7= 172376
4905

```


CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 113-1
 DEFINE UNIBUS MAP REGISTERS

4965		:MAPL33 = 170354	
4966		:MAPH33 = 170356	
4967		:MAPL34 = 170360	
4968		:MAPH34 = 170362	
4969		:MAPL35 = 170364	
4970		:MAPH35 = 170366	
4971		:MAPL36 = 170370	
4972		:MAPH36 = 170372	
4973		:MAPL37 = 170374	
4974		:MAPH37 = 170376	
4975			
4976		.SBTTL DEFINE SOFTWARE SWITCH & DISPLAY REGISTERS	
4977	000174	DISPREG=174	
4978	000176	SWREG= 176	
4979			
4980		.SBTTL DEFINE CONTROL STATUS REGISTERS	
4981	172100	CSRADD=172100	
4982			
4983		.SBTTL DEFINE PARAMETERS	
4984	060000	FIRST=60000	:START OF THE 16K TEST PATTERN AREA
4985	157776	LAST=157776	:END OF THE 16K TEST PATTERN AREA
4986	040000	SIZE=40000	:SIZE OF THE 16K TEST PATTERN AREA (FOR SOB INSTRUCTIONS)

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 115
DEFINE PARAMETERS

4989
4990
4991
4992
4993
4994
4995
4996
4997
4998
4999
5000
5001
5002
5003
5004
5005
5006
5007
5008
5009
5010
5011
5012
5013
5014
5015
5016
5017
5018
5019
5020
5021
5022
5023
5024
5025

```

      .LIST MD ;BE NICE TO SEE MY DEFINITIONS
      .SBTTL MACRO FATAL
***** FATAL *****
: FATAL IS USED TO REPORT FATAL ERRORS (ERRORS THAT PREVENT
: THE PROGRAM FROM CONTINUING).
*****
      .MACRO FATAL ARG ;***MACRO***MACRO***MACRO***
      .NLIST
      .DSABL CRF
      .IIF DF LST$$ .LIST ME
      .ENABL CRF
      .LIST
      INC FATAL$ ;SET FATAL INDICATOR
      ERROR +ARG
      .DSABL CRF
      .IIF DF LST$$ .NLIST ME
      .ENABL CRF
      .ENDM FATAL

      .SBTTL MACRO TYPE
      .MACRO TYPE ARG
      .NLIST
      .DSABL CRF
      .IIF DF LST$$ .LIST ME
      .ENABL CRF
      .LIST
      .IF B ARG
      TYPEIT
      .IFF
      TYPEIT ,ARG
      .ENDC
      .DSABL CRF
      .IIF DF LST$$ .NLIST ME
      .ENABL CRF
      .ENDM TYPE

```

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 117
 MACRO NEWTST

5028
 5029
 5030
 5031
 5032
 5033
 5034
 5035
 5036
 5037
 5038
 5039
 5040
 5041
 5042
 5043
 5044
 5045
 5046
 5047
 5048
 5049
 5050
 5051
 5052
 5053
 5054
 5055
 5056
 5057
 5058
 5059
 5060
 5061
 5062
 5063
 5064
 5065
 5066
 5067
 5068
 5069
 5070
 5071
 5072
 5073
 5074
 5075

```

.SBTTL MACRO NEWTST
:***** NEWTST *****
:NEWTST IS USED AS THE FIRST INSTRUCTION OF A TEST.
:IT WILL:
:1) GENERATE A TEST NUMBER FOR THE LABEL OF THIS TEST
:2) PUT STARS BEFORE AND AFTER A MESSAGE
:ARGUMENTS
:1) ASCII -- THIS IS THE MESSAGE THAT WILL APPEAR
:           ON THE LISTING
:2) ICOUNT -- IF NON-BLANK AND BIT 11 OF $SWR = 1 IT WILL BE
:           THE NUMBER OF ITERATIONS TO MAKE ON THIS TEST
:3) RETURN -- IF NON-BLANK WILL BE THE ADDRESS TO
:           WHICH THE NEXT SCOPE STATEMENT WILL
:           LOOP BACK TO.
:4) COMAND -- IF NON-BLANK WILL BE THE FIRST
:           INSTRUCTION OF THE TEST
:           IF BLANK SCOPE WILL BE THE
:           FIRST INSTRUCTION
:*****
.MACRO NEWTST ASCII,ICOUNT,RETURN,COMAND
$STN=1
$NWTST=0
.NLIST MC
.IF B <COMAND>
$$NEWTST \STN,<ASCII>,SCOPE
.IFF
$$NEWTST \STN,<ASCII>,<COMAND>
.ENDC
.NLIST
.LIST ME
.LIST
.IF NE 4000&$SWR
.IF NB ICOUNT
.IF LE <ICOUNT-1>
MOV #1,$TIMES ;;DO 1 ITERATION
.IFF
MOV #ICOUNT,$TIMES ;;DO ICOUNT ITERATIONS
.ENDC
.ENDC
.IF NB RETURN,
MOV #RETURN,$LPADR ;;SET SCOPE LOOP ADDRESS
.ENDC
.ENDC
.NLIST
.LIST MC
.LIST
.NLIST ME
.ENDM NEWTST

```

5078
5079
5080
5081
5082
5083
5084
5085
5086
5087
5088
5089
5090
5091
5092
5093
5094
5095
5096
5097
5098
5099
5100
5101
5102
5103
5104
5105
5106
5107
5108
5109
5110
5111
5112
5113
5114
5115
5116
5117
5118
5119
5120
5121
5122
5123
5124
5125
5126
5127
5128
5129
5130

```

.SBTTL MACRO $$NEWTEST
.MACRO $$NEWTEST A,ASC,COMND
.IRP ASCI,<ASC>
.IF EQ $NWTST
$NWTST=1
.SBTTL T'A' ASCI
.NLIST
.LIST ME
.LIST
:*****
;*TEST A ASCI
.IFF
ASCI
.ENDC
.ENDM
:*****
TST'A: COMND
.NLIST ME
$TN=$TN+1
.ENDM $$NEWTEST

.SBTTL MACRO SUBTST
:***** SUBTST *****
:THIS MACRO WILL FORMAT A SUBTEST HEADING WITH STARS
:A .SBTTL WILL BE FORCED & .NLISTED FOR THE TABLE OF CONTENTS.
:ARGUMENT:
:1) TXT -- THIS IS THE MESSAGE THAT WILL APPEAR IN THE TABLE OF CONTENTS & LISTING.
:EXAMPLE: SUBTST <<THIS IS A FUN SUBTST>>
:*****

.MACRO SUBTST ASCII
.NLIST MC
$SUBTST <ASCII>
.LIST MC
.ENDM SUBTST

.SBTTL MACRO $$SUBTST
.MACRO $$SUBTST ASC
.IRP ASCI,<ASC>
.SBTTL ASCI
.NLIST
.LIST ME
.LIST
:*****
;*SUBTEST ASCI
.ENDM
:*****
.NLIST ME
.ENDM $$SUBTST

```


MACRO TYPOCT

5133
5134
5135
5136
5137
5138
5139
5140
5141
5142
5143
5144
5145
5146
5147
5148
5149
5150
5151
5152
5153
5154
5155
5156
5157
5158
5159
5160
5161
5162
5163
5164
5165
5166
5167
5168
5169
5170

```

.SBTTL MACRO TYPOCT
***** TYPOCT *****
:TYPOCT IS USED TO CHANGE A BINARY NUMBER
: TO A 6 DIGIT OCTAL NUMBER AND TYPE IT
:
:ARGUMENTS:
:1) NUM THE NUMBER TO BE TYPED
:2) REMARK ALLOWS A COMMENT TO BE MADE
:
:ROUTINES REQUIRED
:1) CONVERT BINARY TO OCTAL AND TYPE (.$TYPOCT)
:2) TYPE AN ASCIZ STRING (.$TYPE)
:
:EXAMPLES:
:1) TYPOCT HILMT,<TYPES THE CONTENTS OF HILMT>
:2) TYPOCT #5,<TYPES '000005'>
*****
.MACRO TYPOCT NUM,REMARK
.NLIST
.DSABL CRF
.IIF DF LST$$ .LIST ME
.ENABL CRF
.LIST
MOV NUM, -(SP) ;;SAVE NUM FOR TYPEOUT
.IIF NB <REMARK>, ;;REMARK
TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
.DSABL CRF
.IIF DF LST$$ .NLIST ME
.ENABL CRF
.ENDM TYPOCT

```

5173
5174
5175
5176
5177
5178
5179
5180
5181
5182
5183
5184
5185
5186
5187
5188
5189
5190
5191
5192
5193
5194
5195
5196
5197
5198
5199
5200
5201
5202
5203
5204
5205
5206
5207
5208
5209
5210
5211
5212
5213
5214
5215
5216
5217
5218
5219
5220
5221
5222
5223
5224
5225
5226

```

.SBTTL MACRO TYPOCS
***** TYPOCS *****
:TYPOCS IS USED TO CHANGE A BINARY NUMBER TO AN OCTAL
:NUMBER AND TYPE 1 TO 6 DIGITS
:WITH OR WITHOUT LEADING ZEROS.
:
:ARGUMENTS:
:1) NUM NUMBER TO BE TYPED
:2) REMARK ALLOWS A COMMENT TO BE MADE
:3) N NUMBER OF DIGITS (1 TO 6) TO BE TYPED
:4) Z BLANK=SUPPRESS LEADING ZEROS (TYPES SPACES)
:NON-BLANK=TYPE LEADING ZEROS
:
:ROUTINES REQUIRED
:1) CONVERT BINARY TO OCTAL AND TYPE (.$TYPOCT)
:2) TYPE AN ASCIZ STRING (.$TYPE)
:
:EXAMPLES:
:1) TYPOCS #12345,<TYPES '5'>,1
:2) TYPOCS #004,<TYPES '04'>,2,X
:3) TYPOCS #004,<TYPES '4'>,2
:
*****
.MACRO TYPOCS NUM,REMARK,N,Z
.NLIST
.DSABL CRF
.IIF DF LST$$ .LIST ME
.ENABL CRF
.LIST
MOV NUM, -(SP) ;;SAVE NUM FOR TYPEOUT
.IIF NB <REMARK>, ;;REMARK
TYPOS ;;GO TYPE--OCTAL ASCII
.IF NB N
.BYTE N ;;TYPE N DIGIT(S)
.IFF
.BYTE 6 ;;TYPE 6 DIGITS
.ENDC
.IF NB Z
.BYTE 1 ;;TYPE LEADING ZEROS
.IFF
.BYTE 0 ;;SUPPRESS LEADING ZEROS
.ENDC
.DSABL CRF
.IIF DF LST$$ .NLIST ME
.ENABL CRF
.ENDM TYPOCS

```

5229
5230
5231
5232
5233
5234
5235
5236
5237
5238
5239
5240
5241
5242
5243
5244
5245
5246
5247
5248
5249
5250
5251
5252
5253
5254
5255
5256
5257
5258
5259
5260
5261
5262
5263
5264
5265
5266
5267
5268
5269

```

.SBTTL MACRO TYPDEC
***** TYPDEC *****
TYPDEC IS USE TO CHANGE A BINARY NUMBER TO A SIGNED
DECIMAL NUMBER AND TYPE IT REPLACING LEADING ZERO
WITH SPACES.
NOTE: IF THE NUMBER IS NEGATIVE A
MINUS SIGN WILL BE TYPED.

ARGUMENTS:
1) NUM NUMBER TO BE TYPED
2) REMARK ALLOWS A COMMENT TO BE MADE

ROUTINES REQUIRED
1) CONVERT-BINARY TO DECIMAL AND TYPE (.$TYPDEC)
2) TYPE AN ASCIZ STRING (.$TYPE)

EXAMPLES
1) TYPDEC SIZE,<TYPE THE CONTENTS OF SIZE>
2) TYPDEC #-10.,<TYPE A MINUS TEN>
*****

.MACRO TYPDEC NUM,REMARK
.NLIST
.DSABL CRF
.IIF DF LST$$ .LIST ME
.ENABL CRF
.LIST
MOV NUM,-(SP) ;;SAVE NUM FOR TYPEOUT
.IIF NB <REMARK>, ;;REMARK
TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN
.DSABL CRF
.IIF DF LST$$ .NLIST ME
.ENABL CRF
.ENDM TYPDEC

```


MACRO BMOV

5271
5272
5273
5274
5275
5276
5277
5278
5279
5280
5281
5282
5283
5284
5285
5286
5287
5288
5289
5290
5291
5292
5293
5294
5295
5296
5297
5298
5299
5300
5301
5302
5303
5304
5305
5306
5307
5308
5309
5310
5311
5312
5313
5314
5315
5316
5317
5318
5319
5320
5321
5322
5323
5324
5325
5326
5327

```

.SBTTL MACRO BMOV
***** BMOV *****
: THIS MACRO MOVES A BLOCK OF DATA.
: ARGUMENTS:
: 1) FROMHERE THE FIRST ADDRESS OF THE SOURCE BLOCK.
: 2) TOHERE THE FIRST ADDRESS OF THE DESTINATION BLOCK.
: IF BLANK THE 1ST ADDRESS OF THE USER INSTRUCTION
: PAR'S IS USED (FASTCITY).
: 3) SIZE THE SIZE OF THE SOURCE BLOCK.
: IF BLANK A 16 WORD TRANSFER IS ASSUMED.
: 'WHY DEFAULT TO 16 WORDS?' YOU ASK!
: 'BECAUSE THAT'S HOW MANY WORDS TO THE USER PAR
: REGISTERS & THAT'S WHERE I INTEND TO MOVE LOTS
: OF STUFF.' I REPLY!
*****

```

```

.MACRO BMOV FROMHERE,TOHERE,SIZE
  .IF B TOHERE
    .NLIST
    .DSABL CRF
    .IIF DF LST$$ .LIST ME
    .ENABL CRF
    .LIST
    JSR R5,BLOCK1
    FROMHERE
    .DSABL CRF
    .IIF DF LST$$ .NLIST ME
    .ENABL CRF
    .MEXIT
  .ENDC
  .IF B SIZE
    .NLIST
    .DSABL CRF
    .IIF DF LST$$ .LIST ME
    .ENABL CRF
    .LIST
    JSR R5,BLOCK2
    TOHERE
    FROMHERE
    .DSABL CRF
    .IIF DF LST$$ .NLIST ME
    .ENABL CRF
    .MEXIT
  .IFF
    .NLIST
    .DSABL CRF
    .IIF DF LST$$ .LIST ME
    .ENABL CRF
    .LIST
    JSR R5,BLOCK3
  SIZE

```

5328
5329
5330
5331
5332
5333
5334

TOHERE
FROMHERE
.DSABL CRF
.IIF DF LST\$\$.NLIST ME
.ENABL CRF
.ENDC
.ENDM BMOV

5337
5338
5339
5340
5341
5342
5343
5344
5345
5346
5347
5348
5349
5350
5351
5352
5353
5354
5355
5356
5357
5358
5359
5360
5361
5362
5363
5364
5365
5366
5367
5368
5369
5370
5371
5372
5373

```

.SBTTL MACRO MAP
:***** MAP *****
: THIS MACRO MAPS A MEMORY BANK (16K) INTO THE
: TEST PATTERN AREA (SUPERVISOR VIRTUAL (60000-15777)).
: ARGUMENTS:
: 1) BANK THE BANK OF 16K WORDS TO BE MAPPED.
: THERE ARE 120 BANKS OF 16K WORDS
: EXAMPLES
: MAP LOC ;LOCATION 'LOC' CONTAINS THE # OF THE BANK TO MAP
: MAP #28. ;BANK 34 (OCTAL) WILL BE MAPPED
:*****

.MACRO MAP BANK
PUSH R3
.NLIST
.DSABL CRF
.IIF DF LST$$ .LIST ME
.ENABL CRF
.LIST
.IF B BANK
MOV #120.,R3
.IFF
MOV BANK,R3
.ENDC
CALL MAPPER
.DSABL CRF
.IIF DF LST$$ .NLIST ME
.ENABL CRF
POP R3
.ENDM MAP

```


5376
5377
5378
5379
5380
5381
5382
5383
5384
5385
5386
5387
5388
5389
5390
5391
5392
5393
5394
5395
5396
5397
5398
5399
5400
5401
5402
5403
5404
5405
5406
5407
5408
5409
5410
5411
5412
5413
5414
5415
5416
5417

```
.SBTTL MACRO SUPERVISOR
***** SUPERVISOR *****
: THIS MACRO SWITCHES TO SUPERVISOR MODE.
: ARGUMENTS: NONE.
*****
```

```
.MACRO SUPERVISOR
.NLIST
.DSABL CRF
.IIF DF LST$$ .LIST ME
.ENABL CRF
.LIST
BIS #BIT14,PSW ;GO TO SUPERVISOR MODE
.DSABL CRF
.IIF DF LST$$ .NLIST ME
.ENABL CRF
.ENDM SUPERVISOR
```

```
.SBTTL MACRO USER
***** USER *****
: THIS MACRO SWITCHES TO USER MODE.
: ARGUMENTS: NONE.
*****
```

```
.MACRO USER
.NLIST
.DSABL CRF
.IIF DF LST$$ .LIST ME
.ENABL CRF
.LIST
BIS #BIT15!BIT14,PSW ;GO TO USER MODE
.DSABL CRF
.IIF DF LST$$ .NLIST ME
.ENABL CRF
.ENDM USER
```

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 131
MACRO TESTAREA

5419
5420
5421
5422
5423
5424
5425
5426
5427
5428
5429
5430
5431
5432
5433
5434
5435
5436
5437
5438

```
.SBTTL MACRO TESTAREA
:***** TESTAREA *****
: THIS MACRO SWITCHES TO THE SPECIFIED TEST MODE.
: ARGUMENTS: NONE.
:*****
```

```
.MACRO TESTAREA
.NLIST
.DSABL CRF
.IIF DF LST$$ .LIST ME
.ENABL CRF
.LIST
BIS TESTMODE,PSW ;GC TO SYSTEM TEST MODE
.DSABL CRF
.IIF DF LST$$ .NLIST ME
.ENABL CRF
.ENDM TESTAREA
```

5441
5442
5443
5444
5445
5446
5447
5448
5449
5450
5451
5452
5453
5454
5455
5456
5457
5458
5459
5460
5461
5462
5463
5464
5465
5466
5467
5468
5469
5470
5471
5472
5473
5474
5475
5476
5477
5478
5479
5480
5481
5482
5483

```
.SBTTL MACRO SET4 & RES4
:***** SET4 & RES4 *****
: THESE MACROS SET & RESTORE VECTOR 4(TIMEOUT TRAP)
: IN IT'S RESTORED MODE TRAPS ARE REPORTED AS SUCH.
: ARGUMENTS: LOC ;THE LOCATION TO VECTOR TO (ONLY USED IN "SET4" NOT "RES4")
: I USE THE SET4 AND RES4 MACROS AROUND CODE THAT I EXPECT TO TRAP TO 4
: LIKE LOOKING FOR ALL POSSIBLE CSR'S AND ETC. WHENEVER CODE IS NOT
: SURROUNDED BY SET4 AND RES4 THEN ANY TRAPS TO 4 WILL CAUSE AN ERROR
: PRINTOUT THAT SAYS 'UNEXPECTED TRAP TO 4' AND ALL THE ASSOCIATED REGISTER JUNK
:*****
```

```
.MACRO SET4 ARG
.NLIST
.DSABL CRF
.IIF DF LST$$ .LIST ME
.ENABL CRF
.LIST
MOV ARG,4
.DSABL CRF
.IIF DF LST$$ .NLIST ME
.ENABL CRF
.ENDM SET4
```

```
.MACRO RES4
.NLIST
.DSABL CRF
.IIF DF LST$$ .LIST ME
.ENABL CRF
.LIST
MOV #TIMEOUT,4
CMP #1,PROTYP
BNE 101$
CLR CPUERR
```

101\$:

```
:IS THIS AN 11/44?
:BRANCH IF NOT
:CLEAR OUT THE CPU ERROR REGISTER BITS
:THAT A EXPECTED TRAP COULD HAVE SET
```

```
.DSABL CRF
.IIF DF LST$$ .NLIST ME
.ENABL CRF
.ENDM RES4
```


5486
5487
5488
5489
5490
5491
5492
5493
5494
5495
5496
5497
5498
5499
5500
5501
5502
5503
5504
5505
5506
5507

```

.SBTTL MACRO DLEFT
:***** DLEFT *****
: THIS MACRO DOES A DOUBLE WORD LEFT SHIFT
: ARGUMENTS: LOC ;THE LOCATION TO BE SHIFTED LEFT (CARRY TO LOC+2)
:*****

.MACRO DLEFT ARG
.NLIST
.DSABL CRF
.IIF DF LST$$ .LIST ME
.ENABL CRF
.LIST
ROL ARG
ROL ARG+2
.DSABL CRF
.IIF DF LST$$ .NLIST ME
.ENABL CRF
.ENDM DLEFT
.NLIST MD ;DON'T NEED TO SEE THEM ANY MORE

```

```

5510                                     .SBTTL TRAP CATCHER
5511                                     .=0
5512 000000 000000 000000             .WORD 0,0
5513                                     .REPT 177                ;.WORD .+2,HALT
5517
5518                                     .SBTTL ACT11 HOOKS
5519 ;*THE HOOKS REQUIRED BY ACT11 ARE DEFINED AND SETUP BELOW:
5520 ;*
5521 ;* DEFINITIONS:
5522 ;*
5523 ;* 1)LOC.46 'END-OF-PASS' HOOK
5524 ;* =ADDRESS OF END OF PASS ROUTINE
5525 ;* MODIFIED BY ACT11.
5526 ;* 2)LOC.52 PROGRAM NEEDS HOOK
5527 ;* BIT 15=1 PROGRAM SHOULD BE POWER
5528 ;* FAILED WHILE RUNNING
5529 ;* =0 NO POWER FAIL
5530 ;* BIT 14=1 PROGRAM MEMORY SIZE DEPENDENT
5531 ;* =0 NOT MEMORY SIZE DEPENDENT
5532 ;* BIT 13=1 PROGRAM REQUIRES MANUAL INTERVENTION
5533 ;* =0 MANUAL INTERVENTION NOT REQUIRED
5534 ;* BITS 12-0 MUST BE ZERO'S
5535 ;*
5536 ;*
5537 ;*
5538 ;*
5539 ;*
5540 ;*
5541 ;*
5542 ;*
5543 ;*
5544 ;*
5545 ;*
5546 ;*
5547 ;*
5548 ;*
5549 ;*
5550 ;*
5551 ;*
5552 ;*
5533 000046 000046
5534 000046 015234
5535 000052 000052
5536 000052 000020
5537
5538 000024 000024
5539 000024 000200
5540 000042 000042
5541 000042 002000
5542 000044 000044
5543 000044 066004
5544 000200 000200
5545 000200 000437
5546 000202 000442
5547 000300 000300
5548 000300 005037 002614
5549 000304 000137 003656
5550 000310 000137 003656
5551 000316 000137 003656
5552 002000 002000

.=46
$ENDAD
.=52
.WORD BIT4
.SBTTL APT11 HOOKS
.=24
200
.=42
STACK
.=44
$APTHDR
.=200
START3: BR START1
BR START2
.=300
START1: CLR RESTART
JMP START
START2: SET RESTART
JMP START
.=STACK
;SO RT11 CAN START WITH RUN COMMAND
;POINT TO APT INDIRECT ADDRESS PNTR.
;POINT TO APT HEADER BLOCK
;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
;2)SET LOC.52 TO INDICATE MEMORY SIZE DEPENDANT
;SET POWER FAIL TO POINT TO START OF PROGRAM
;FOR APT START UP
;'NORMAL' START
;RESTART (SAVE ERROR ACCOUNTING)

```

5555			.SBTTL VARIABLES	INITIALIZED TO ZERO
5556			:*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS	
5557			:*USED IN THE PROGRAM.	
5558	002000		\$CMTAG:	::START OF COMMON TAGS
5559	002000	000000	SELONLY:0	::SELECT ONLY BANKS MARKED BY FIELD SERVICE MODE FLAG
5560	002002	000000	DIAGFLAG:0	::SET FOR SHIFTING DIAGONAL TEST
5561	002004	000000	KAMIKAZE:0	::SET FOR KAMIKAZE MODE TESTING
5562	002006	000000	SKIPKAMI:0	::USED TO SKIP RESTORING KAMIKAZE MODE WHEN MODIFIED
5563			:NEXT TWO BYTES ARE DISPLAYED IN THE DISPLAY REGISTER	
5564	002010	000	\$PATMAR:.BYTE 0	::PATTERN NUMBER
5565	002011	000	\$BANK: .BYTE 0	::BANK & SIGN
5566	002012	000	\$ERFLG: .BYTE 0	::CONTAINS ERROR FLAG
5567	002013	000	\$ITEMB: .BYTE 0	::CONTAINS ITEM CONTROL BYTE
5568	002014	000000	LASTERROR:.WORD 0	::NUMBER OF ERRORS ON LAST PASS
5569	002016	000000	ERRPC: .WORD 0	::CONTAINS PC OF ERROR FOR TYPEOUT
5570	002020	000000	BADPC: .WORD 0	::CONTAINS PC OF ERROR
5571	002022	000000	ERRSP: .WORD 0	::CONTAINS SP OF ERROR FOR TYPEOUT
5572	002024	000000	BADSP: .WORD 0	::CONTAINS SP OF ERROR
5573	002026	000000	ERRPSW: .WORD 0	::CONTAINS PSW OF ERROR FOR TYPEOUT
5574	002030	000000	BADPSW: .WORD 0	::CONTAINS PSW OF ERROR
5575	002032	000000	ADDRESS:.WORD 0	::CONTAINS ADDRESS OF 'BAD' DATA
5576	002034	000000	PADDRESS:.WORD 0	::ADDRESS OF PARITY ERROR
5577	002036	000000 000000	PHYADD: .WORD 0.0	::22 BIT PHYSICAL ADDRESS
5578	002042	000000	GOOD: .WORD 0	::CONTAINS 'GOOD' DATA
5579	002044	000000	GOOD2: .WORD 0	::CONTAINS 'GOOD2' DATA
5580	002046	000000	GOOD3: .WORD 0	::CONTAINS 'GOOD3' DATA
5581	002050	000000	BAD: .WORD 0	::CONTAINS 'BAD' DATA
5582	002052	000000	BAD2: .WORD 0	::CONTAINS 'BAD2' DATA
5583	002054	000000	BAD3: .WORD 0	::CONTAINS 'BAD3' DATA
5584	002056	000000	BADXOR: .WORD 0	::XOR OF GOOD & BAD = BAD BITS!
5585	002060	000000	\$AUTO: .WORD 0	::AUTOMATIC MODE INDICATOR FOR APT,ACT, & XXDP
5586	002062	000000	FATALS: .WORD 0	::FATAL ERROR INDICATOR
5587	002064	000000	SKPERR: .WORD 0	::USED TO SKIP ERROR MESSAGE IN '\$ERRGEN'
5588	002066	000000	NEMCNT: 0	::NON-EXISTANT MEMORY COUNTER (HOLES)
5589	002070	000000	PARCNT: 0	::PARITY ERROR COUNTER
5590	002072	000000	PATERR: 0	::PATTERN ERROR COUNTER
5591	002074	000000	NOPAR: 0	::NO PARITY ERROR MODE INDICATOR
5592	002076	000000	NONEM: 0	::NO NON-EXISTANT MEMORY (HOLES) MODE INDICATOR
5593	002100	000000	BANK: 0	::MEMORY BANK UNDER TEST
5594	002102	000000	BANKINDEX:0	::USED TO INDEX INTO CONFIG TABLE
5595	002104	000000	CPUBIT: 0	::CONTAINS 1 BIT TO IDENTIFY CPU TO CONFIGURATION TABLE
5596	002106	000000	MUT: 0	::MEMORY UNDER TEST FLAG
5597	002110	000000	PATTERN:0	::PATTERN NUMBER UNDER TEST
5598	002112	000000	KPFLAG: .WORD 0	::BANK IS PROTECTED REGION OF ECC
5599	002114	000000	ACFLAG: .WORD 0	::BANK CAN BE ACCESSED BY THIS CPU
5600	002116	000000	MKFLAG: .WORD 0	::IF SET INDICATES MS11-M OR MF11S-K UNDER TEST
5601	002120	000000	PFLAG: .WORD 0	::BANK IS IN PROGRAM SPACE
5602	002122	000000	RRFLAG: .WORD 0	::BANK IS WHERE PROGRAM RELOCATION IS REQUIRED TO TEST
5603	002124	000000	RLFLAG: .WORD 0	::PROGRAM IS RELOCATED FLAG
5604	002126	000000	BMFLAG: .WORD 0	::'BANK IS IDENTIFIED AS BAD MEMORY' FLAG
5605	002130	000000	EUFLAG: .WORD 0	::'BANK HAS EUB MEMORY' FLAG
5606	002132	000000	TMFLAG: .WORD 0	::'TYPE OF MEMORY TO TEST' FLAG; 0 = PARITY, 1 = ECC
5607	002134	000000	INTFLAG: .WORD 0	::'BANK IS INTERLEAVED' FLAG
5608	002136	000000	INT64K: .WORD 0	::'BANK IS 64K INTERLEAVED' FLAG
5609	002140	000000	PMEMFLG: .WORD 0	::'MEMORY UNDER TEST IS A MS11-P' FLAG
5610	002142	000000	ABORTFLAG: .WORD 0	::'ABORT OCCURED' FLAG
5611	002144	000000	CTLKVEC: .WORD 0	::HOLDS OLD KERNAL STACK POINTER IN CASE OF CNTL/K

VARIABLES INITIALIZED TO ZERO

```

5612 002146 000000 CSR: .WORD 0 ;DATA TO OR FROM CSR
5613 002150 000000 CSRNO: 0 ;CSR ADDRESS NUMBER (4 LSB'S)
5614 002152 000000 SAVCSR: .WORD 0 ;LOCATION TO SAVE CSRNO DURING FS COMMAND
5615 002154 000000 OLDCSR: .WORD 0 ;OLD CSR NUMBER(USED IN INH PTR TEST)
5616 :THESE LOCATIONS STORE GPR'S DURING SUPERVISOR TESTS
5617 002156 000000 SUPDR0: 0
5618 002160 000000 SUPDR1: 0
5619 002162 000000 SUPDR2: 0
5620 002164 000000 SUPDR3: 0
5621 002166 000000 SUPDR4: 0
5622 002170 000000 SUPDR5: 0
5623 002172 000000 SUPDR6: 0
5624 002174 000000 DUMMY: 0 ;DUMMY LOCATION FOR ADDRESS PASSING
5625 :THESE LOCATIONS STORE GPR'S & PSW DURING DETAILED ERROR PRINTOUTS
5626 002176 000000 DETR0: 0
5627 002200 000000 DETR1: 0
5628 002202 000000 DETR2: 0
5629 002204 000000 DETR3: 0
5630 002206 000000 DETR4: 0
5631 002210 000000 DETR5: 0
5632 002212 000000 DETSP: 0
5633 002214 000000 DETPSW: 0
5634 002216 000000 DETFLAG:0 ;DETAILED REPORT FLAG
5635 002220 000000 CONTFLAG:0 ;CSR'S HAVE BEEN TESTED FLAG
5636 002222 000000 TOTCSRS:.WORD 0 ;1 BIT PER EXISTING CSR, EG-
5637 ;CSR 0 REPRESENTED BY BIT 15, ETC.
5638 002224 000000 CSRFIRST:.WORD 0 ;FIRST ADDRESS UNDER CONTROL OF THIS CSR
5639 002226 000000 CSRLAST: .WORD 0
5640 002230 000000 CSRFBANK:.WORD 0
5641 002232 000000 CSRLBANK:.WORD 0
5642 002234 000000 CSRINT: .WORD 0
5643 002236 000000 SPLTCSR: .WORD 0
5644 002240 000000 000000 DATBUF: .WORD 0,0 ;TWO WORD DATA BUFFER
5645 002244 000000 000000 TSTDAT: .WORD 0,0 ;TWO WORD TEST DATA
5646 002250 000000 000000 SBEMSK: .WORD 0,0 ;TWO WORD SINGLE BIT ERROR MASK
5647 002254 000000 000000 DBEMSK: .WORD 0,0 ;TWO WORD DOUBLE BIT ERROR MASK
5648 002260 000000 SUPDOADD:.WORD 0 ;ADDRESS OF SUBROUTINE TO EXECUTE IN SUPERVISOR MODE
5649 002262 000 PASFLG: .BYTE 0 ;LOCAL LOOP PASS CONTROL
5650 002263 000 UPPFLG: .BYTE 0 ;LOCAL LOOP PASS CONTROL
5651 002264 000000 PASSNO: .WORD 0 ;LOCAL LOOP PASS CONTROL
5652 002266 000000 SAV4: .WORD 0 ;USED TO SAVE KERNAL PAR 5 ;:I.L.C.:REV
5653 002270 000000 SAVPAR: .WORD 0 ;USED TO SAVE KERNAL PAR 5
5654 002272 000000 SAVMON: .WORD 0 ;XXDP MONITOR RETURN ADDRESS
5655 002274 000000 MONFLG: .WORD 0 ;RETURN TO MONITOR FLAG
5656 002276 000000 REALPAT:.WORD 0 ;REAL PATTERN UNDER TEST
5657 002300 000000 OLDCACHE:.WORD 0 ;BACKED UP VALUE OF CACHE CONTROL REGISTER
5658 002302 000000 PARTHERE:.WORD 0 ;PARITY TRAPS SOMETIMES GO TO ADDRESS STORED HERE
5659 002304 000000 FSSTACK:.WORD 0 ;STACK SAVED HERE IF IN FIELD SERVICE MODE
5660 002306 000000 NEWBANK:.WORD 0 ;USED FOR RELOCATION TO A NEW BANK
5661 002310 000000 SOURCE: .WORD 0 ;SOURCE OF DATA WORDS FOR CHECKBIT GENERATION SUBROUTINE
5662 002312 000000 CHECK: .WORD 0 ;CHECKBITS TO BE LOADED INTO CSR
5663 002314 000000 CBITS: .WORD 0 ;CHECK BITS TO BE WRITTEN
5664 002316 000000 MASK: .WORD 0 ;BIT MASK FOR CSR
5665 002320 000000 CSR1S: .WORD 0 ;CSR ALL 1'S PATTERN
5666 002322 000000 BITNO: .WORD 0 ;BIT POINTER
5667 002324 000000 PCBUMP: .WORD 0 ;VALUE TO BUMP THE PC BY TO RECOVER AFTER A PARITY TRAP
5668 002326 000000 CSRINC: .WORD 0 ;VALUE TO INCREMENT ADDRESS BY TO REMAIN IN THE SAME CSR

```

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 139-2
 VARIABLES INITIALIZED TO ZERO

5669	002330	000000			CSRLOOP: .WORD	0	: LOOP CONTROL FOR CSR TESTING
5670	002332	000000			SUCCESS: .WORD	0	: FLAG SET BY SUCCESSFULL TASK OR SUBROUTINE
5671	002334	000000			ZEROS: .WORD	0	: FOR AID IN 'MOV' INSTRUCTIONS
5672	002336	000000			TIME: .WORD	0	: SECONDS THAT BATTERIES SHOULD LAST
5673	002340	000000			SKIPMK: .WORD	0	: FLAG TO SKIP MKCONTROL SUBROUTINE
5674	002342	000000			NULLFLAG: .WORD	0	: SET WHEN RUNNING NULL PATTERNS
5675	002344	000000			QVFLAG: 0		: FLAGS QUICK VERIFY PASS UNDER APT, ACT, OR XXDP CHAIN MODE
5676	002346	000000			ACTFLAG: 0		: FLAGS ACT AUTOMATIC MODE PROGRAMMING RULES
5677	002350	000000			APTFLAG: 0		: FLAGS APT AUTOMATIC MODE PROGRAMMING RULES
5678	002352	000000			XXDPCHAIN: 0		: FLAGS XXDP CHAIN MODE PROGRAMMING RULES
5679					: NOTE: THESE TWO BYTES MUST STAY TOGETHER		
5680	002354	000			\$NULL: .BYTE	0	: CONTAINS NULL CHARACTER FOR FILLS
5681	002355	000			\$FILLS: .BYTE	0	: CONTAINS # OF FILL CHARACTERS
5682	002356	000			\$STPFLG: .BYTE	0	: 'TERMINAL NOT AVAILABLE' FLAG
5683					.EVEN		
5684	002360	000000			\$ESCAPE: 0		: ESCAPE ON ERROR ADDRESS
5685	002362	000000			EVEN: 0		: USED FOR ALTERNATE DATA PATTERNS
5686	002364	000000			STRIPES: 0		: COUNTS DIAGONAL STRIPES
5687	002366	000000			COUNT: 0		: BACKED UP COPY OF STRIPES
5688	002370	000000			NOTAB: 0		: NO TABLE BEING PRINTED - NOW
5689	002372	000000			Bsize: 0		: SIZE OF 11/45 MOS MEMORY IN K WORDS
5690	002374	000000			Ksize: 0		: SIZE OF MF11S-K MEMORY IN K WORDS
5691	002376	000000			Lsize: 0		: SIZE OF MS11-L MEMORY IN K WORDS
5692	002400	000000			Msize: 0		: SIZE OF MS11-M MEMORY IN K WORDS
5693	002402	000000			Psize: 0		: SIZE OF UNIBUS PARITY MEMORY IN K WORDS
5694	002404	000000			TOOMANY: 0		: FLAGS WHEN TOO MANY ERRORS HAVE BEEN PRINTED FOR A BANK
5695	002406	000000			READONLY: 0		: FLAG TO PATTERNS TO READ ONLY
5696	002410	000000	000000		TESTADD: 0,0		: THE ADDRESS TO RUN CSR TESTS ON
5697	002414	000000			UNITOP: 0		: HIGHEST ACCESSABLE BANK OF MEMORY THRU UNIBUS MAP
5698	002416	000000			STOPOK: 0		: FLAG TO ALLOW STOPPING WITH SWITCH REGISTER
5699	002420	000000			APTPAR: .WORD	0	: AMOUNT OF PARITY MEMORY ACCORDING TO APT
5700	002422	000000			APTECC: .WORD	0	: AMOUNT OF ECC MEMORY ACCORDING TO APT
5701	002424	000000			NOFSMODE: 0		: FLAG TO DISABLE FIELD SERVICE MODE
5702	002426	000000			NOERROR: 0		: 'THIS IS NOT AN ERROR' FLAG
5703	002430	000000			LOADBANK: 0		: BANK LOADERS ARE RELOCATED TO
5704	002432	000000			TEMP: 0		: USED FOR JUNK
5705	002434	000000			QUICK: 0		: QUICK STOP FLAG FOR APT POWER FAIL
5706	002436	000000			NOSCOPE: 0		: 'NO SCOPE LOOP ALLOWED' FLAG
5707	002440	000000			FSINFLAG: 0		: 'FIELD SERVICE - NO INTERNAL INTERLEAVE' FLAG
5708	002442	000000			APTSIZE: 0		: APT SIZING INFO FLAG
5709	002444	000000			FS7FLAG: 0		: TRUE WHEN IN FIELD SERVICE COMMAND 7
5710	002446	000000			CONFERROR: 0		: CONFIGURATION ERROR FLAG
5711	002450	000000			I: 0		: USED FOR GENERAL PURPOSE INDEXING
5712	002452	000000			NO22BIT: 0		: NO 22-BIT MODE FLAG
5713	002454	000000			NOSUPER: 0		: NO SUPERVISOR MODE FLAG
5714	002456	000000			ERRADD: .WORD	0	: HOLDS THE CSR'S ERROR ADDRESS
5715	002460	000000	000000	000000	CSRINFO: 0,0,0,0,0,0,0,0,0		: USED TO STORE INFORMATION ABOUT THE 16
	002466	000000	000000	000000			
	002474	000000	000000	000000			
5716	002500	000000	000000	000000		0,0,0,0,0,0,0,0,0	: POSSIBLE CSR'S
	002506	000000	000000	000000			
	002514	000000	000000	000000			
5717	002520	000000			LINK1: 0		: USED TO HOLD LINKS TO PATTERNS WHICH
5718	002522	000000			LINK2: 0		: CAN EXECUTE IN THE PAR/PDR'S OR NOT
5719	002524	000000			CSRHOLD: 0		: USED TO STORE CSR VALUES FOR CSR TESTS
5720	002526	000000			KFLAG: 0		: USED TO FLAG MF11S-K MEMORY TO TESTS
5721	002530	000000	000000		PGMCSR: .WORD	0,0	: POINTS TO PROGRAM CSR

VARIABLES INITIALIZED TO ZERO

5722 002534 000000
5723 002536 000000
5724 002540 000000
5725 002542

INHECC: .WORD 0
INHBANK: .WORD 0
FULLREL: .WORD 0
\$CMTGE: ;*END OF COMMON TAGS

:FLAGS INHIBIT ECC TESTS ON RELOCATION
:
:

VARIABLES INITIALIZED TO NON ZERO

5728
5729 002542 000001 000000
5730 002546 001415
5731 002550 040000
5732 002552 000012
5733 002554 000167
5734 002556 170000
5735 002560 000031
5736 002562 002000
5737 002564 000001
5738 002566 177777
5739 002570 176543
5740 002572 123456
5741 002574 176543
5742 002576 123456
5743 002600 177777
5744 002602 177777
5745 002604 000003
5746 002606 052525
5747 002610 000000
5748 002612 000000
5749 002614 000000
5750 002616 000000
5751
5752
5753 002620 000377
5754 002622 177400
5755
5756
5757 002624 177570
5758 002626 177570
5759 002630 177560
5760 002632 177562
5761 002634 177564
5762 002636 177566
5763 002640 012
5764 002641 207 377 377
5765 002645 077
5766 002646 015
5767 002647 012 000
5768

.SBTTL VARIABLES
CACHKN: 1,0
CACHKF: 1415
TESTMODE:40000
ERRMAX: 10.
LASTBANK:167
LASTBLOCK:170000
SOBK: 25.
KSTACK: STACK
LOADHOME:1
WORST: 177777
SEEDHI: 176543
SEEDLO: 123456
MSEEDH: 176543
MSEEDL: 123456
HEADER: 177777
ONES: 177777
FLIPLOC:3
SOFTPAT:52525
\$LPADR: .WORD 0
\$LPERR: .WORD 0
RESTART:0
\$ERTTL: .WORD 0
***** NOTE THESE TWO LOCATIONS MUST STAY TOGETHER *****
BAKPAT: .WORD 377
SWAPAT: .WORD 177400

SWR: .WORD DSWR
DISPLAY: .WORD DDISP
\$TKS: 177560
\$TKB: 177562
\$TPS: 177564
\$TPB: 177566
\$FILLC: .BYTE 12
\$BELL: .ASCIZ <207><377><377>
\$QUES: .ASCII /?/
\$CRLF: .ASCII <15>
\$LF: .ASCIZ <12>
.EVEN

INITIALIZED TO NON ZERO
: CACHE CONSTANT (MOVED TO CONTRL TO TURN ON CACHE)
: CACHE CONSTANT (MOVED TO CONTRL TO TURN OFF CACHE)
: USED TO SELECT THE PROPER TEST MODE FOR A PATTERN RUN
: MAX # OF ERRORS PER BANK WITH SW11
: HIGHEST BANK OF MEMORY
: HIGHEST BANK OF MEMORY+1 (IN PAR FORMAT)
: SOB CONSTANT
: STACK BEGINNING
: HOME BANK OF LOADERS
: SET IF TESTING BANKS IN WORST FIRST MODE(1ST PASS)
: WORKING SEED HI (USED FOR RANDOM NUMBER GENERATOR)
: WORKING SEED LO (USED FOR RANDOM NUMBER GENERATOR)
: MASTER SEED HI (USED FOR RANDOM NUMBER GENERATOR)
: MASTER SEED LO (USED FOR RANDOM NUMBER GENERATOR)
: USED TO PRINT HEADINGS ONLY ONCE
: FOR AID IN 'MOV' INSTRUCTIONS
: COUNTER FOR FLIPING DATA ON WORST CASE NOISE TEST
: PATTERN FOR SOFT ERROR BACKGROUND TESTS
: CONTAINS SCOPE LOOP ADDRESS
: CONTAINS SCOPE RETURN FOR ERRORS
: RESTART (START ADD 202) FLAG
: CONTAINS TOTAL ERRORS

***** NOTE THESE TWO LOCATIONS MUST STAY TOGETHER *****
: BACKGROUND PATTERN *
: SWAPPED BAKPAT *

: ADDRESS OF SWITCH REGISTER
: ADDRESS OF DISPLAY REGISTER
: TTY KBD STATUS
: TTY KBD BUFFER
: TTY PRINTER STATUS REG. ADDRESS
: TTY PRINTER BUFFER REG. ADDRESS
: INSERT FILL CHARS. AFTER A 'LINE FEED'
: CODE FOR BELL
: QUESTION MARK
: CARRIAGE RETURN
: LINE FEED

5771
5772
5773
5774
5775
5776
5777
5778
5779
5780
5781
5782
5783
5784
5785
5786
5787
5788
5789
5790
5791
5792
5793 002652 000201
5796 003656

```

.SBTTL CONFIGURATION TABLE
:CONFIG:FIRST 16K CONFIGURATION WORDS (2 EACH)
      2ND 16K CONFIGURATION WORDS (2 EACH)
      .....
      200TH 16K CONFIGURATION WORDS (2 EACH)
:CONFIGURATION WORDS:
      LOW: BIT 0 ERRORS PRESENT
           BIT 1 MEMORY SUCCESSFULLY ACCESSED
           BIT 2-4 RESERVED
           BIT 5 SKIP ECC LOGIC TESTS FLAG (1=SKIP)
           BIT 6 PROTECTED REGION OF ECC MEMORY
           BIT 7 PROTECTED (PROGRAM SPACE)
           BIT 8-11 CSR CODE
           BIT 12-15 INTERLEAVED CSR CODE
      HIGH: BIT 0-7 NUMBER OF ERRORS
           BIT 8-10 MEMORY TYPE
           BIT 11 INTERLEAVED BOARD TYPE (0=128K, 1=64K)
           BIT 12 INTERLEAVE ENABLED
           BIT 13 'BACKGROUND PATTERN VALID' FLAG
           BIT 14 BANK SELECTED FOR TEST BY FIELD SERVICE MODE
           BIT 15 LOADERS HOME BANK
:CONFIG: .REPT 201
:CONFIEND:

```

***** MAIN *****

5798
5799 003656

```
.SBTTL ***** MAIN *****
START: SUBTST <<INITIALIZE VARIABLES TO ZERO>>
:*****
:*SUBTEST INITIALIZE VARIABLES TO ZERO
:*****
```

5800 003656 105737 065724
5801 003662 001001
5802 003664 000005
5803 003666
5804 003672 010637 002272
5805 003676 013706 002562
5806 003702 012700 002000
5807 003706 005020
5808 003710 022700 002542
5809 003714 001374
5810 003716 012737 000167 002554
5811 003724

```
TSTB $ENV
BNE NORES
RESET
NORES: CLEAR MONFLG ;;CLEAR RETURN TO MONITOR FLAG
MOV SP, SAVMON ;;SAVE XXDP MONITOR RESTART ADDRESS
MOV KSTACK, SP ;;SETUP THE STACK POINTER
MOV #SCMTAG, RO ;;FIRST LOCATION TO BE CLEARED
1$: CLR (RO)+ ;;CLEAR MEMORY LOCATION
CMP #SCMTGE, RO ;;DONE?
BNE 1$ ;LOOP BACK IF NO
MOV #167, LASTBANK ;RESTORE LASTBANK (THIS MUST BE DONE PRIOR TO SYSTEM SIZING)
SUBTST <<CLEAR NON-PROGRAM SPACE>>
```

```
:*****
:*SUBTEST CLEAR NON-PROGRAM SPACE
:*****
```

5812
5813
5814
5815 003724 012737 000001 002074
5816 003732 005000
5817 003734 000241
5818 003736 005520
5819 003740 020027 160000
5820 003744 103773
5821 003746 005037 002074
5822

```
:THIS ATTEMPS TO GET RID OF ANY PARITY ERRORS BY WRITING INTO
:EVERY LOCATION THAT IS NOT LOADED INTO BY THE PROGRAM OR ALLOCATED
:TO THE XXDP LOADERS
MOV #1, NOPAR ;PARITY ACTION = COUNT & IGNORE
CLR RO
2$: CLC
ADC (RO)+
CMP RO, #160000
BLO 2$
CLR NOPAR ;RESTORE DEFAULT PARITY ACTION
```


5824 003752

SUBTST <<TYPE OF SYSTEM SIZER>>

```

*****
: *SUBTEST      TYPE OF SYSTEM SIZER
*****
5825 003752 000401          BR      SYSSIZ          ;SKIP OVER VARIABLE LOCATION
5826 003754 000000          PROTY: .WORD 0
5827 003756          SYSSIZ: SET4 #4$
5828 003764 005737 177746          TST     CONTRL          ;SEE IF CACHE REGISTER RESPONDS
5829 003770          SET4 #9$          ;YES - DO WE HAVE 11/44 TYPE CACHE
5830 003776 005737 177750          TST     MAINT           ;OR 11/60 TYPE CACHE?
5831 004002 000411          BR      5$             ;BRANCH IF 11/44 TYPE CACHE
5832 004004 012737 000014 002546 9$:  MOV     #14,CACHKF      ;TURN OFF CONSTANT FOR 11/60 CACHE
5833 004012 000405          BR      5$
5834 004014 005037 002542 4$:  CLR     CACHKN          ;NO CACHE ON SYSTEM
5835 004020 012737 002334 067374  MOV     #ZEROS,DT14     ;DO NOT PRINT CONTRL ERROR MESSAGES
5836 004026          SET4 #6$
5837 004034 005737 172516          TST     MMR3           ;DO WE HAVE AN MMR3?
5838 004040 005037 172516          CLR     MMR3           ;YES WE DO
5839 004044 052737 000020 172516  BIS     #BIT4,MMR3      ;SEE IF THERE IS 22-BIT MODE
5840 004052 032737 000020 172516  BIT     #BIT4,MMR3
5841 004060 001026          BNE    10$            ;BRANCH IF 22-BIT RELOCATION
5842 004062 000413          BR     7$             ;BRANCH IF MMR3 BUT NO 22-BIT RELOC.
5843          ;* 11/34 TYPE MACHINES ENTER HERE
5844 004064 012737 140000 002550 6$:  MOV     #140000,TESTMODE ;MAKE TESTMODE USER
5845 004072 005237 002454          INC     NOSUPER
5846 004076 005037 067244          CLR     DT5+10
5847 004102 005037 067404          CLR     DT14+10
5848 004106 005237 002452          INC     NO22BIT
5849          ;* 11/45 TYPE MACHINES ENTER HERE
5850 004112 005237 002452 7$:  INC     NO22BIT
5851 004116 012737 000007 002554  MOV     #7,LASTBANK
5852 004124 005037 067246          CLR     DT5+12
5853 004130 005037 067406          CLR     DT14+12
5854 004134 000417          BR     8$
5855 004136          10$:  SET4 #8$
5856 004144 000007          MFPT
5857          ;TYPE OF PROCESSOR TEST: THIS INSTRUCTION
5858          ; (AVAILABLE ON NEWER PROCESSORS ONLY) PLACES
5859          ; A CODE IN THE LOWER BYTE OF R0 THAT
5860          ; INDICATES THE PROCESSOR TYPE. 1=11/44
5861          ; 3=11/24
5861 004146 110037 003754          MOV     R0,PROTYP      ;MOV THE CODE TO PROTYP
5862 004152 022737 000003 003754  CMP     #3,PROTYP      ;IS THIS AN 11/24?
5863 004160 001005          BNE    8$             ;BRANCH IF NOT - WE HAVE AN 11/44
5864 004162 005237 002454          INC     NOSUPER
5865 004166 012737 140000 002550  MOV     #140000,TESTMODE ;MAKE TEST MODE USER
5866          ;
5867          8$:  SET4 #11$            ;TRAPS GO TO 11$ ;R-C
5868 004202 005037 061366          CLR     CPERRF         ;CLEAR THE FLAG ;R-C
5869 004206 005737 177766          TST     @#177766
5870 004212 012737 177777 061366  MOV     #-1,CPERRF     ;IS THERE A CPU ERROR REGISTER? ;R-C
5871 004220          11$:  RES4 ;YES-TRAPPED ;R-C

```

5874 004242

SUBTST <<INITIALIZE VARIABLES TO NON ZERO>>

: *SUBTEST INITIALIZE VARIABLES TO NON ZERO

5875 004242
5876 004250 012737 000003 002604
5877 004256
5878 004264 012737 176543 002574
5879 004272 012737 123456 002576
5880 004300 013737 002574 002570
5881 004306 013737 002576 002572
5882 004314 012737 000377 002620
5883 004322 012737 177400 002622
5884 004330

SET WORST
MOV #3,FLIPLOC
SET HEADER
MOV #176543,MSEEDH
MOV #123456,MSEEDL
MOV MSEEDH,SEEDHI ;PRIME THE RANDOM NUMBER GENERATOR
MOV MSEEDL,SEEDLO ;BOTH HIGH AND LOW WORDS
MOV #377,BAKPAT
MOV #177400,SWAPAT
SUBTST <<INITIALIZE VECTORS>>

: *SUBTEST INITIALIZE VECTORS

5885 004330 012737 060232 000020
5886 004336 012737 000340 000022
5887 004344 012737 060566 000030
5888 004352 012737 000340 000032
5889 004360 012737 066020 000034
5890 004366 012737 000340 000036
5891 004374 012737 054412 000024
5892 004402 012737 000340 000026
5893 004410 012737 042420 000114
5894 004416 012737 000340 000116
5895 004424 012737 042614 000010
5896 004432 012737 000340 000012
5897 004440 012737 042570 000004
5898 004446 012737 000340 000006
5899 004454 012737 042602 000250
5900 004462 012737 000340 000252
5901 004470 104423

MOV \$\$SCOPE,IOTVEC ;:IOT VECTOR FOR SCOPE ROUTINE
MOV #340,IOTVEC+2 ;:LEVEL 7
MOV \$ERROR,EMTVEC ;:EMT VECTOR FOR ERROR ROUTINE
MOV #340,EMTVEC+2 ;:LEVEL 7
MOV \$STRAP,TRAPVEC ;:TRAP VECTOR FOR TRAP CALLS
MOV #340,TRAPVEC+2;LEVEL 7
MOV \$PWRDN,PWRVEC ;:POWER FAILURE VECTOR
MOV #340,PWRVEC+2 ;:LEVEL 7
MOV \$PARITY,PARVEC;GET READY FOR PARITY ERRORS
MOV #340,PARVEC+2
MOV #PDP1105,RESVEC;RESERVED INSTRUCTION TRAP
MOV #340,RESVEC+2
MOV \$TIMEOUT,ERRVEC;SETUP TIMEOUT ERRORS
MOV #340,ERRVEC+2 ;:SET PRIORITY OF ERROR TRAPS
MOV \$MMTRAP,MMVEC ;VECTOR FOR MEMORY MANAGEMENT
MOV #340,MMVEC+2
CACHON ;TURN CACHE ON

5904 004472

SUBTST <<INITIALIZE PATTERNS>>

:SUBTEST INITIALIZE PATTERNS

5905
5906
5907
5908
5909 004472 012700 065770
5910 004476 012001
5911 004500 012703 020206
5912 004504 012702 000020
5913 004510 004737 004610
5914 004514 012001
5915 004516 012702 000010
5916 004522 004737 004610
5917 004526 012001
5918 004530 012703 020436
5919 004534 012702 000020
5920 004540 004737 004610
5921 004544 012001
5922 004546 012702 000010
5923 004552 004737 004610
5924 004556 012001
5925 004560 012703 020622
5926 004564 012702 000020
5927 004570 004737 004610
5928 004574 012001
5929 004576 012702 000010
5930 004602 004737 004610
5931 004606 000417
5932
5933 004610

:THE APT E-TABLE DETERMINES WHICH PATTERNS ARE GOING TO BE RUN.
:EACH BIT SET REPRESENTS A PATTERN TABLE ENTRY THAT IS TO BE LEFT
:ALONE (TO BE RUN). EACH BIT CLEARED REPRESENTS A PATTERN TABLE ENTRY
:THAT IS TO BE OVERLAYED WITH THE ADDRESS OF A NULL PATTERN.

```
MOV #SDDW0,R0
MOV (R0)+,R1
MOV #MKCSRT,R3
MOV #16.,R2
CALL PATPLUG
MOV (R0)+,R1
MOV #8.,R2
CALL PATPLUG
MOV (R0)+,R1
MOV #MKPAT,R3
MOV #16.,R2
CALL PATPLUG
MOV (R0)+,R1
MOV #8.,R2
CALL PATPLUG
MOV (R0)+,R1
MOV #MJPAT,R3
MOV #16.,R2
CALL PATPLUG
MOV (R0)+,R1
MOV #8.,R2
CALL PATPLUG
BR SUBAAA
```

PATPLUG:SUBTST <<SUBR PLUG IN NULL PATTERNS>>

:SUBTEST SUBR PLUG IN NULL PATTERNS

5934 004610
5935 004616 006001
5936 004620
5937 004622 012713 026764
5938 004626
5939 004626 062703 000002
5940 004632
5941 004644 000207

```
FOR I := #1 TO R2
ROR R1
ON.NOERROR ;IF CARRY CLEAR
MOV #MT0999,(R3)
END ;OF ON.ERROR
ADD #2,R3
END ;OF FOR
RETURN
```


5944 004646

SUBAAA: SUBTST <<CLEAR THE CONFIGURATION TABLE>>
:*****
:*SUBTEST CLEAR THE CONFIGURATION TABLE
:*****

5945

5946

5947

5948 004646

5949 004654 012700 002652

5950 004660 005020

5951 004662 022700 003656

5952 004666 001374

5953 004670

5954

5955 004670 012737 000002 002104

5956 004676

: THIS ZEROS (UNLESS WE STARTED AT ADDRESS 202) THE CONFIG TABLE
: WHICH IS FULLY DISCRIBED AT LOCATION 'CONFIG'.
.ENABLE LSB
IF RESTART IS FALSE
1\$: MOV #CONFIG,RO
CLR (RO)+
CMP #CONFIEND,RO
BNE 1\$
END ;OF IF RESTART
.DSABL LSB
MOV #DIT1,CPUBIT ;SET ID BIT
SUBTST <<SIZE FOR A HARDWARE SWITCH REGISTER>>

:*****
:*SUBTEST SIZE FOR A HARDWARE SWITCH REGISTER
:*****

5957

5958

5959

5960 004676

5961 004704 012737 177570 002624

5962 004712 012737 177570 002626

5963 004720

5964 004730 000403

5965 004732 012716 004740

5966 004736 000002

5967 004740

5968 004762 012737 000176 002624

5969 004770 012737 000174 002626

5970 004776

5971

:: IF NOT FOUND OR IT IS
:: EQUAL TO A '-1', SETUP FOR A SOFTWARE SWITCH REGISTER.
.ENABL LSB
SET4 #3\$;TRAPS TO 4 GOTO 3\$
MOV #DSWR,SWR ;:SETUP FOR A HARDWARE SWITCH REGISTER
MOV #DDISP,DISPLAY ;:AND A HARDWARE DISPLAY REGISTER
IF #-1 EQ @SWR ;:IF NO TRAP FROM REFERENCE TO @SWR AND @SWR = #-1
BR 2\$;:BRANCH IF NO TIMEOUT
3\$: MOV #2\$, (SP) ;:SET UP FOR TRAP RETURN
RTI
2\$: RES4 ;:RESET TRAPS TO 4 TO DEFAULT
MOV #SWREG,SWR ;:POINT TO SOFTWARE SWR
MOV #DISPREG,DISPLAY
END ;OF IF #-1
.DSABL LSB

5974 004776

```

SUBAAB: SUBTST <<SETUP ACT, APT, & XXDP>>
:*****
:*SUBTEST     SETUP ACT, APT, & XXDP
:*****
;THIS SETS UP A BUNCH OF FLAGS TO TELL THE PROGRAM EVERYTHING
;IT CARES TO KNOW ABOUT APT, ACT, & XXDP.
CLR     $PASS          ;CLEAR PASS COUNT
IFB #BIT5 SET.IN $ENVM
    SET  $STPFLG        ;INDICATE NO TERMINAL
END :OF IFB #BIT5
IFB #BIT7 SET.IN $ENVM
    SET  APTSIZE
END :OF IFB #BIT7
IFB $ENV EQ #1
    SET  APTFLAG,QVFLAG,$AUTO,QUICK
    MOV  #APTDOWN,PWRVEC
    MOV  #SSWREG,SWR    ;USE APT SWR
ELSE
    IF 42 NE #STACK AND 42 NE #0
        SET QVFLAG,$AUTO
        IF 42 EQ #SENDAD
            SET  ACTFLAG
        ELSE
            SET  XXDPCHAIN
        END :OF IF 42
    END :OF IF 42
END :OF IFB $ENV

```

```

5975
5976
5977 004776 005037 065712
5978 005002
5979 005012
5980 005020
5981 005020
5982 005030
5983 005036
5984 005036
5985 005046
5986 005076 012737 047760 000024
5987 005104 012737 065726 002624
5988 005112
5989 005114
5990 005132
5991 005146
5992 005156
5993 005164
5994 005166
5995 005174
5996 005174
5997 005174

```

5999 005174

```

SUBTST <<PROTECT PROGRAM & LOADERS>>
:*****
:*SUBTEST    PROTECT PROGRAM & LOADERS
:*****
BIS    #BIT7,CONFIG      ;PROTECT PROGRAM SPACE (BANK 0)
BIS    #BIT7,CONFIG+4    ;PROTECT LOADER SPACE (BANK 1)
IF #SENDAD NE 42        ;NOT ACT-11?
    IF NO22BIT NE #0
        SET    MONFLG
        ERROR +64
    ELSE
        TYPE    MSG000
        END
    END ;OF IF #SENDAD

```

6000 005174 052737 000200 002652
6001 005202 052737 000200 002656
6002 005210
6003 005220
6004 005226
6005 005234 104064
6006 005236
6007 005240
6008 005244
6009 005244
6010
6011 005244

```

SUBTST <<CHECK SYSTEM FOR CACHE>>
:*****
:*SUBTEST    CHECK SYSTEM FOR CACHE
:*****
:* THIS FIGURES OUT IF THERE IS A CACHE ON THE SYSTEM,
:* WHAT TYPE OF SYSTEM IT IS, AND WHETHER IT IS ENABLED
:* OR DISABLED.
SET4   #3$
TST    CONTRL          ;IS THERE A CONTROL REGISTER?
SET4   #2$
TST    MAINT           ;IS THERE A MAINTENANCE REGISTER?
SET4   #1$
TST    DATARG          ;IS THERE A DATA REGISTER?
TYPE   MSG117
BR     4$
1$:   SET    MONFLG
      ERROR +64
      ; 11/34
      ;
      ;PROCESSOR NOT SUPPORTED BY THIS DIAGNOSTIC
      ; 11/60
      ;
      ;PROCESSOR NOT SUPPORTED BY THIS DIAGNOSTIC
4$:   BIS    #BIT2!BIT3,CONTRL ;SET CACHE DISABLE BITS
      BIC    #BIT2!BIT3,CONTRL ;CLEAR CACHE DISABLE BITS
      BIT    #BIT2,CONTRL      ;IS THE BIT SET?
      BNE    7$
      BIT    #BIT3,CONTRL      ;IS THE BIT SET?
      BEQ    6$
      TYPE   MSG121
      ;CACHE BYPASSED
      ;
      ;SAVE INFO ABOUT CACHE
      MOV    CACHKN,CACHKN+2
      CLR    CACHKN
      BR     8$
      ;
      ;
3$:   TYPE   MSG119
6$:   TYPE   MSG120
      ;CACHE AVAILABLE

```

6012
6013
6014
6015 005244
6016 005252 005737 177746
6017 005256
6018 005264 005737 177750
6019 005270
6020 005276 005737 177754
6021 005302
6022 005306 000410
6023 005310
6024 005316 104064
6025
6026 005320
6027 005326 104064
6028
6029 005330 052737 000014 177746
6030 005336 042737 000014 177746
6031 005344 032737 000004 177746
6032 005352 001004
6033 005354 032737 000010 177746
6034 005362 001413
6035 005364
6036 005370 104424
6037 005372 013737 002542 002544
6038 005400 005037 002542
6039 005404 000404
6040 005406
6041 005412

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 155
CHECK SYSTEM FOR CACHE

6043 005416

SUBTST <<SETUP USER & SUPERVISOR STACK>>
:*****
:*SUBTEST SETUP USER & SUPERVISOR STACK
:*****

6044 005416 104421
6045 005420 005737 002454
6046 005424 001011
6047
6048
6049 005426 042737 030000 177776
6050 005434 052737 010000 177776
6051
6052 005442
6053 005446 006606
6054
6055
6056 005450 052737 030000 177776
6057
6058 005456
6059 005462 006606
6060
6061 005464

8\$: DEENERGIZE ;TURN OFF MEMORY MANAGEMENT
TST NOSUPER ;IS THERE A SUPERVISOR MODE?
BNE 5\$;NO-SKIP SUPERVISOR SETUP.

;SET PREVIOUS MODE TO SUPERVISOR
BIC #BIT13!BIT12,PSW
BIS #BIT12,PSW

PUSH #SUPSTK
MTP I SSP

5\$: ;SET PREVIOUS MODE TO USER
BIS #BIT13!BIT12,PSW

PUSH #USESTK
MTP I USP

SUBTST <<GET SOFTWARE SWITCH REGISTER IF NECESSARY>>
:*****
:*SUBTEST GET SOFTWARE SWITCH REGISTER IF NECESSARY
:*****

6062 005464
6063 005472
6064 005502 104407
6065 005504
6066 005504
6067
6068 005504

IF \$AUTO IS FALSE ;IF NOT(APT OR ACT)
IF SWR EQ #SWREG ;IF SOFTWARE SWITCH REG SELECTED
GTSWR ;:GET SOFT-SWR SETTINGS
END ;OF IF SWR
END ;OF IF \$AUTO

SUBTST <<GET MEMORY MANAGEMENT READY>>
:*****
:*SUBTEST GET MEMORY MANAGEMENT READY
:*****

6069 005504 104422
6070 005506
6071 005522 104420

KMAP ;MAP KERNEL SPACE 1 TO 1
MAP ;MAP SUPERVISOR SPACE (TEST AREA) 1 TO 1
ENERGIZE ;TURN ON MEMORY MANAGEMENT

6074 005524

NEWSTST <<BIT TEST OF ALL CSR'S>>

: *TEST 1 BIT TEST OF ALL CSR'S

TST1: SCOPE
: * THE FIRST PART OF THE CONFIGURATION ANALYSIS DOES THE FOLLOWING:
: * 1) FINDS WHICH CSR'S RESPOND, AND PUTS THEM INTO THE CSR INFORMATION
: * TABLE, AND STORES ANOTHER BIT FOR 'TOTCSRS'.
: * 2) TESTS THE CSR BITS COMMON TO ALL CSR'S.
: * 3) FIGURES OUT IF THE MODULE IS A ECC OR PARITY MEMORY
: * 4) TESTS THE BITS PARTICULAR TO THAT TYPE OF CSR.
: * 5) IF ANY BITS TEST BAD IN THE CSR UNDER TEST, THE CSR OK BIT IN THE
: * CSR INFORMATION TABLE IS CLEARED.

: * THE INFORMATION BITS ONE THROUGH THREE FORM A CODE WHICH GIVES THE TYPE
: * OF CSR:

TYPE	NOT USED BIT2	ECC TYPE BIT1	ECC BIT0	CODE TOTALS
MS11-L	0	0	0	0
MS11-M	0	0	1	1
MS11-P	0	1	1	3

: * THIS MEMORY CODE WILL BE USED IN THE SECOND PART OF THIS ANALYSIS

005524 000004

6075
6076
6077
6078
6079
6080
6081
6082
6083
6084
6085
6086
6087
6088
6089
6090
6091
6092
6093

6094 005526 005005
6095 005530 005000
6096 005532 012703 172100
6097 005536 012737 000001 002074
6098 005544
6099 005552
6100 005552 005713
6101 005554 052705 000001
6102 005560 005004
6103 005562 042760 000004 002460
6104 005570 052760 000030 002460
6105 005576 005013
6106 005600
6107 005604
6108 005612 004737 005716
6109 005616
6110 005616 005013
6111 005620 004737 006210
6112 005624
6113 005634 016037 002460 002050
6114 005642 104021
6115 005644
6116 005644 062700 000002
6117 005650 062703 000002
6118 005654 006305
6119 005656
6120 005660 005204
6121 005662
6122 005662
6123 005670 006005
6124 005672 005704
6125 005674
6126 005676 052705 100000

```

CLR R5 ;R5 IS THE TOTAL CSR NUMBER
CLR R0 ;R0 IS A TABLE INDEX
MOV #CSRADD,R3 ;R3 HAS THE CSR ADDRESS
MOV #1,NOPAR ;IGNORE PARITY ERRORS
SET4 #NXTCSR
REPEAT
TST (R3) ;DOES THIS CSR RESPOND???
BIS #1,R5 ;MARK IT IN CSR MAP
CLR R4 ;CLEAR THE LAST CSR INDICATOR
BIC #4,CSRINFO(R0) ;CLEAR UNUSED BITS
BIS #BIT4!BIT3,CSRINFO(R0) ;YES-MARK IT IN CSR INFORMATION TABLE
CLR (R3) ;CLEAR THE CSR UNDER TEST
LET (R3) := #BIT13 ;IS THIS AN ECC MEMORY???
IF #BIT13 SET IN (R3) ;IS BIT 13 SET FOR ECC MEMORY???
CALL ECCTYPE ;FIGURE OUT WHAT KIND OF ECC MEMORY WE HAVE
END
CLR (R3) ;CLEAR CSR UNDER TEST
CALL RWCSR ;BIT TEST OF ALL BITS IN CSR'S
IF CSRINFO(R0) MI #30 ;DO WE HAVE A LEGAL CONFIGURATION?
MOV CSRINFO(R0),BAD ;MOVE IN BAD DATA
ERROR +21
END
NXTCSR:
ADD #2,R0 ;GO TO NEXT CSR
ADD #2,R3 ;GO TO NEXT CSR
ASL R5 ;SHIFT CSR MAP
ON.ERROR ;IS THERE A CSR 0
INC R4 ;YES-SET CSR PRESENT FLAG
END
UNTIL RO EQ #40 ;UNTIL ALL CSR'S ARE DONE
ROR R5 ;RESYNC R5
TST R4 ;WAS THERE A CSR 0?
RNE 22$ ;BRANCH IF NOT EQUAL
BIS #BIT15,R5 ;YES SET IT IN CSR TABLE

```

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 157-1
T1 BIT TEST OF ALL CSR'S

```
6127 005702          22$: LET TOTCSRS := R5      ;STORE CSR MAP IN TOTCSRS
6128 005706 004737 005776      CALL CSRMAP      ;PRINT CSR MAP
6129 005712          JUMPTO CTEST          ;
```


6131 005716

SUBTST <<DETERMINE TYPE OF ECC MEMORY>>

: *SUBTEST DETERMINE TYPE OF ECC MEMORY

6132
6133
6134
6135

THIS ROUTINE WILL DETERMINE IF THE ECC MEMORY UNDER TEST IS
A MS11-M OR A MS11-P

6136 005716
6137 005716 052760 000001 002460
6138 005724
6139 005730
6140 005736 005013
6141 005740
6142 005744
6143 005750
6144 005756 052760 000002 002460
6145 005764
6146 005764
6147 005766 042760 000002 002460
6148 005774
6149 005774 000207

ECCTYPE:
BIS #BIT0,CSRINFO(R0) ;MARK IT IN THE TABLE AS BEING A ECC MEMORY
LET (R3) := #60004 ;IS THIS A MS11-P???
IF #BIT11 SET.IN (R3) ;IS BIT 11 SET???
CLR (R3) ;CLEAR CSR
LET (R3) := #20004 ;ENABLE CHECK/SYNDROME BIT REGISTER
LET (R3) := #27744 ;IT IS BUT MAKE SURE AGAIN
IF (R3) EQ #23744 ;DO WE HAVE 6 OR 7 CHECK BITS IN CSR
BIS #BIT1,CSRINFO(R0) ;6 CHECK BITS-MARK IT A MS11-P
END ;
ELSE ;IT IS A MS11-M
BIC #BIT1,CSRINFO(R0) ;MARK IT IN THE TABLE
END ;
RETURN ;

6151 005776

```

CSRMAP: SUBTST <<PRINT CSR REGISTER MAP>>
:*****
:*SUBTEST PRINT CSR REGISTER MAP
:*****
CLR R0 ;CLEAR CSR INFO POINTER
TYPE MSG008 ;PRINT TITLE
TYPE MSG016 ;PRINT CSR NUMBERS
CLR R1
REPEAT
MOV R1,R2
CMP #9.,R2
BPL 1$ ;JUMP AROUND NEXT INSTRUCTION
ADD #7,R2
ADD #60,R2 ;MAKE IT ASCII
MOVB R2,MSG015
TYPE MSG015
TYPE MSG014 ;TYPE SINGLE SPACE
INC R1
UNTIL R1 EQ #16.
TYPE MSG009 ;TYPE MEMTYPE
REPEAT
IF CSRINFO(R0) NE #0 ;IS CSR NONEXSISTANT???
IF #BIT0 SET.IN CSRINFO(R0)
IF #BIT1 SET.IN CSRINFO(R0)
MOVB #'P,MSG015 ;IT IS A MS11-P
ELSE
MOVB #'M,MSG015 ;IT IS A MS11-M
END
END
IF #BIT1!BIT0 OFF.IN CSRINFO(R0)
MOVB #'L,MSG015 ;IT IS A MS11-L
END
ELSE
MOVB #' ,MSG015
END
TYPE MSG015 ;TYPE MEMORY TYPE
TYPE MSG014 ;TYPE SPACE
NOP
ADD #2,R0 ;POINT TO NEXT ENTRY
UNTIL R0 EQ #40
TYPE MSG129
RETURN
TRACE: .WORD 0

```

```

6152 005776 005000
6153 006000
6154 006004
6155 006010 005001
6156 006012
6157 006012 010102
6158 006014 022702 000011
6159 006020 100002
6160 006022 062702 000007
6161 006026 062702 000060
6162 006032 110237 074750
6163 006036
6164 006042
6165 006046 005201
6166 006050
6167 006056
6168 006062
6169 006062
6170 006070
6171 006100
6172 006110 112737 000120 074750
6173 006116
6174 006120 112737 000115 074750
6175 006126
6176 006126
6177 006126
6178 006136 112737 000114 074750
6179 006144
6180 006144
6181 006146 112737 000040 074750
6182 006154
6183 006154
6184 006160
6185 006164 000240
6186 006166 062700 000002
6187 006172
6188 006200
6189 006204 000207
6190 006206 000000

```

6192 006210

6193
6194
6195
6196
6197
6198
6199
6200
6201
6202 006210
6203 006210
6204 006220
6205 006222 006205
6206 006224
6207 006230
6208 006234
6209 006244
6210 006250
6211 006252
6212 006256
6213 006256
6214 006264 040537 002320
6215 006270
6216 006272
6217 006274 040504
6218 006276
6219 006302
6220 006306
6221 006312 104035
6222 006314 042760 000010 002460
6223 006322
6224 006322
6225 006326
6226 006330 005013
6227 006332 040504
6228 006334
6229 006342
6230 006350
6231 006354 104010
6232 006356 042760 000010 002460
6233 006364
6234 006364
6235 006370
6236
6237 006370 005237 002262
6238 006374
6239 006402
6240 006412
6241 006416
6242 006420
6243 006424
6244 006424
6245 006424 005237 177640

```

SUBTST <<READ AND WRITE ALL CSR BITS>>
*****
*SUBTEST   READ AND WRITE ALL CSR BITS
*****
: THIS ROUTINE 'RWCSR' CHECK TO SEE THAT THE CSR CAN BEWRITTEN ON CORRECTLY
: BY WRITING AND CHECKING FOR THE FOLLOWING PATTERNS:
:
:     1-ZEROS
:     2-ONES
:     3-SHIFTING A ONE THROUGH A FIELD OF ZEROS
:     4-SHIFTING A ZEROS THROUGH A FIELD OF ONES
:
RWCSR:
PUSH R4,R5,UIPARO           ;SAVE R4,R5, AND UIPARO ON STACK
LET R5 := R0                 ;GET CSR NUMBER FOR POSSIBLE ERROR
ASR R5                       ;
LET CSRNO := R5              ;
LET ADDRESS := R3            ;GET ADDRESS FOR POSSIBLE ERROR
IF #BIT0 SET.IN CSRINFO(R0) ;WHAT KIND OF MEMORY IS THIS???      ;GET BIT MASKS FOR D
    LET R5 := #17740         ;MASK FOR MS11-M/P
ELSE
    LET R5 := #70032         ;IT IS A MS11-L
                                ;MASK FOR MS11-L
END
LET CSR1S := #177777         ;SET CSR1S TO ALL ONES
BIC R5,CSR1S                 ;CLEAR BITS FOR GOOD DATA
LET (R3) := #0                ;0----->CSR
LET R4 := (R3)                ;MASK OUT UNWANTED BITS
BIC R5,R4                     ;
IF R4 NE #0                   ;DO WE HAVE A CORRECT READ
    LET GOOD := #0            ;GOOD DATA=0'S
    LET CSR := R4             ;BAD DATA=CSR
    ERROR +35                 ;BIT SET ERROR
    BIC #BIT3,CSRINFO(R0)    ;CLEAR CSR OK BIT
END
LET (R3) := CSR1S             ;ONES--->(R3)
LET R4 := (R3)                ;MASK OUT CORRECT FIELD
CLR (R3)                       ;CLEAR OUT CSR
BIC R5,R4                     ;
IF R4 NE CSR1S                 ;WAS PATTERN WRITTEN CORRECTLY?
    LET GOOD := CSR1S        ;GOOD DATA = ALL LEGAL BITS SET IN CSR
    LET CSR := R4            ;BAD DATA=CSR
    ERROR +10                 ;BIT CLEAR ERROR
    BIC #BIT3,CSRINFO(R0)    ;CLEAR CSR OK BIT
END
LET PASFLG := #0              ;SET UP LOOP COUNTER
REPEAT                          ;REPEAT WITH A FIELD OF 1'S THROUGH 0'S
                                ;0'S THROUGH 1'S
    INC PASFLG                ;INCREMENT LOOP COUNTER
    LET UIPARO := #-1         ;USE USER PAR FOR BIT COUNTER
    IF PASFLG EQ #1           ;PASS 1
        LET R2 := #1         ;1----->FIELD OF ZEROS
    ELSE                       ;PASS 2
        LET R2 := #177776    ;0----->FIELD OF ONES
    END
REPEAT                          ;DO BITS 0-4 AND 13-15
    INC UIPARO                 ;INCREMENT BIT POINTER

```


CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 160-1
 READ AND WRITE ALL CSR BITS

6246	006430				IF PASFLG EQ #2 AND #BIT0 OFF. IN CSRINFO(R0) ;
6247	006450	042702	040004		BIC #BIT14!BIT2,R2 ;IF THIS IS PASS 2 ON A MS11-L, CLEAR EUB BIT AND WRITE
6248	006454				END
6249	006454				LET (R3) := R2 ;WRITE DATA
6250	006456				LET R1 := R2 ;GET GOOD DATA AND MASK IT OUT
6251	006460	040501			BIC R5,R1 ;GET GOOD DATA
6252	006462				LET R4 := (R3) ;GET DATA THAT IS READ
6253	006464	040504			BIC R5,R4 ;MASK OUT CSR BITS
6254	006466				IF R1 NE R4 ;IS DATA CORRECT???
6255	006472				LET BAD := R4 ;BAD DATA = CSR CONTENTS
6256	006476				LET CSR := R1 ;GET GOOD DATA
6257	006502				IF PASFLG EQ #1 ;SELECT ERROR DEPENDING ON PASS
6258	006512	104035			ERROR +35 ;BIT SET ERROR
6259	006514				ELSE ;PASS 2
6260	006516	104010			ERROR +10 ;BIT CLEAR ERROR
6261	006520				END
6262	006520	042760	000010	002460	BIC #BIT3,CSRINFO(R0) ;CLEAR CSR OK BIT
6263	006526				END
6264	006526				IF PASFLG EQ #1 ;GET DATA FOR NEXT LOOP
6265	006536	006302			ASL R2 ;SHIFT 1 ACROSS 0'S
6266	006540				ELSE ;
6267	006542	000261			SEC ;SET CARRY
6268	006544	006102			ROL R2 ;ROTATE A 0 ACROSS A FIELD OF ONES
6269	006546				END ;
6270	006546				UNTIL UIPARO EQ #15. ;UNTIL ALL BITS ARE DONE
6271	006556				UNTIL PASFLG EQ #2 ;DONE WITH 2 PASSES
6272	006566				IF #BIT0 SET. IN CSRINFO(R0) THEN JUMPTO DONE ;IF MS11-L DO ONE LAST WRITE
6273	006602				LET (R3) := #140005 ;WRITE ONES TO CSR WITH EUB BIT ENABLED
6274	006606				LET R2 := (R3) ;READ CSR FOR CORRECT BITS
6275	006610	042702	037772		BIC #37772,R2 ;CLEAR UNWANTED BITS
6276	006614				IF R2 NE #140005 ;WAS WRITE CORRECT
6277	006622				LET GOOD := #140005 ;GOOD DATA
6278	006630				LET CSR := R2 ;BAD DATA
6279	006634	104010			ERROR +10 ;BIT CLEAR ERROR
6280	006636	042760	000010	002460	BIC #BIT3,CSRINFO(R0) ;CLEAR CSR OK BIT!
6281	006644				END ;
6282	006644			DONE:	LET (R3) := #0 ;CLEAR OUT CSR
6283	006646				POP UIPARO,R5,R4 ;RESTORE UIPARO,R4, AND R5
6284	006656	000207			RETURN
6285					

```

6288                                     ;THE FOLLOWING ROUTINE DETERMINES WHICH CSR CONTROLS PROGRAM SPACE
6289                                     ;
6290 006660 104424                       ;TEST: CACHOFF
6291 006662 012737 177777 002530        MOV #177777,PGMCSR
6292 006670 012737 002000 172350        MOV #2000,KIPAR4 ;SET UP MAP REGISTER
6293 006676 012701 002410                MOV #TESTADD,R1
6294 006702 012737 100000 002410        MOV #100000,TESTADD
6295 006710 012737 100002 002412        MOV #100002,TESTADD+2
6296 006716 005000                        CLR R0 ;CLEAR CSR COUNTER
6297 006720 005037 002150                CLR CSRNO
6298 006724 013703 002222                MOV TOTCSRS,R3 ;OBTAIN CSR MAP
6299 006730 000240                        NOP ;DEBUG AID
6300 006732 006303 4$: ASL R3 ;PUT HIGH ORDER BIT INTO C BIT
6301 006734 103407                        BCS 2$ ;BRANCH IF CSR EXISTS
6302 006736 062700 000002 1$: ADD #2,R0 ;UPDATE CSR COUNTER
6303 006742 010037 002150                MOV R0,CSRNO
6304 006746 005703                        TST R3 ;IS MAP EMPTY?
6305 006750 001474                        BEQ 3$ ;BRANCH IF SO
6306 006752 000767                        BR 4$
6307 006754 000240 2$: NOP ;DEBUG AID
6308 006756 000241                        CLC ;CLEAR CARRY
6309 006760 032760 000003 002460        BIT #BIT1!BIT0,CSRINFO(R0) ;IS THIS PARITY MEMORY?
6310 006766 001014                        BNE 5$ ;BRACH IF NOT
6311 006770 052760 000004 172100        BIS #BIT2,CSRADD(R0) ;SET WRITE WRONG PARITY
6312 006776 012771 123456 000000        MOV #123456,@(R1) ;WRITE DATA
6313 007004 012771 123456 000002        MOV #123456,@2(R1)
6314 007012 005060 172100                CLR CSRADD(R0) ;RESTORE CSR
6315 007016 000414                        BR 6$
6316 007020 012760 000000 172100 5$: MOV #0,CSRADD(R0) ;CLEAR THE CSR UNDER TEST
6317 007026 012771 123456 000000        MOV #123456,@(R1) ;WRITE DATA
6318 007034 012771 123456 000002        MOV #123456,@2(R1)
6319 007042 012760 020006 172100        MOV #20006,CSRADD(R0) ;SET DIAG CHECK MODE
6320 007050 005771 000000 6$: TST @(R1) ;WRITE CHECKBITS TO CSR
6321 007054 016004 172100                MOV CSRADD(R0),R4 ;WRITE CSR TO R4
6322 007060 032760 000003 002460        BIT #BIT1!BIT0,CSRINFO(R0) ;PARITY MEMORY?
6323 007066 001003                        BNE 7$ ;BRANCH IF NOT
6324 007070 005704                        TST R4 ;PARITY ERROR?
6325 007072 100421                        BMI 8$ ;BRACH IF SO
6326 007074 000720                        BR 1$ ;TRY NEXT CSR
6327 007076 000240 7$: NOP ;DEBUG AID
6328 007100 072427 177773                ASH #-5,R4
6329 007104 042704 177600                BIC #^C177,R4
6330 007110 032760 000002 002460        BIT #BIT1,CSRINFO(R0) ;WHAT KIND OF ECC MEMORY IS THIS
6331 007116 001003                        BNE 10$ ;BRANCH IF MS11-P
6332 007120 012702 000157                MOV #157,R2 ;LOAD IN CORRECT CHECK BITS FOR MS11-M
6333 007124 000402                        BR 11$
6334 007126 012702 000040 10$: MOV #40,R2 ;CORRECT CHECK BITS FOR MS11-P
6335 007132 020204 11$: CMP R2,R4 ;CORRECT CHECKBITS?
6336 007134 001300                        BNE 1$ ;BRANCH IF NOT
6337 007136 010037 002530 8$: MOV R0,PGMCSR
6338 007142 000240 3$: NOP ;DEBUG AID
6339 007144 104502                        CLRCSR ;CLEAR ALL CSR'S
6340 007146 012771 000000 000000        MOV #0,@(R1) ;RESTORE TEST LOCATIONS
6341 007154 012771 000000 000002        MOV #0,@2(R1)
6342 007162 023727 002530 177777        CMP PGMCSR,#177777
6343 007170 001402                        BEQ FINT ;IF PROGRAM CSR NOT FOUND GO TO FINT
6344 007172 000137 007644                JMP CLRMEM ;GO TO SIZING ROUTINE IF FOUND

```


CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 163
 READ AND WRITE ALL CSR BITS

```

6346
6347
6348
6349
6350 007176
6351 007204 012771 123456 000000
6352 007212 012771 123456 000002
6353 007220 062737 010000 172350
6354 007226 000766
6355 007230 012700 177776
6356 007234 013703 002222
6357 007240 062700 000002
6358 007244 010037 002150
6359 007250 006303
6360 007252 103403
6361 007254 005703
6362 007256 001405
6363 007260 000767
6364 007262 012760 020006 172100
6365 007270 000763
6366 007272
6367 007300 012700 177776
6368 007304 012737 002000 172350
6369 007312 005771 000000
6370 007316 062700 000002
6371 007322 010037 002150
6372 007326 022700 000040
6373 007332 001535
6374 007334 032760 000002 002460
6375 007342 001003
6376 007344 012702 000157
6377 007350 000402
6378 007352 012702 000040
6379 007356 016004 172100
6380 007362 072427 177773
6381 007366 042704 177600
6382 007372 020204
6383 007374 001401
6384 007376 000747
6385 007400 110037 002530
6386 007404
6387 007412 012700 177776
6388 007416 013703 002222
6389 007422 062700 000002
6390 007426 010037 002150
6391 007432 006303
6392 007434 103403
6393 007436 005703
6394 007440 001405
6395 007442 000767
6396 007444 012760 020006 172100
6397 007452 000763
6398 007454 012700 177776
6399 007460 005771 000002
6400 007464 062700 000002
6401 007470 010037 002150
6402 007474 022700 000040

```

: IF PGMCSR WAS NOT FOUND BY THE PRECEEDING ROUTINE, THIS ROUTINE TRIES
 : TO FIND IT FOR INTERLEAVED MEMORIES

```

FINT: SET4 #2$
1$: MOV #123456,@(R1)
MOV #123456,@2(R1)
ADD #10000,KIPAR4
BR 1$
2$: MOV #-2,R0
MOV TOTCSRS,R3
3$: ADD #2,R0
MOV R0,CSRNO
ASL R3
BCS 4$
TST R3
BEQ 5$
BR 3$
4$: MOV #20006,CSRADD(R0)
BR 3$
5$: SET4 #6$
MOV #-2,R0
MOV #2000,KIPAR4
TST @(R1)
6$: ADD #2,R0
MOV R0,CSRNO
CMP #40,R0
BEQ 10$
BIT #BIT1,CSRINFO(R0)
BNE 55$
MOV #157,R2
BR 56$
55$: MOV #40,R2
56$: MOV CSRADD(R0),R4
ASH #-5,R4
BIC #^C177,R4
CMP R2,R4
BEQ 7$
BR 6$
7$: MOVB R0,PGMCSR
SET4 #8$
MOV #-2,R0
MOV TOTCSRS,R3
23$: ADD #2,R0
MOV R0,CSRNO
ASL R3
BCS 24$
TST R3
BEQ 25$
BR 23$
24$: MOV #20006,CSRADD(R0)
BR 23$
25$: MOV #-2,R0
TST @2(R1)
8$: ADD #2,R0
MOV R0,CSRNO
CMP #40,R0

```

:NE MEMORY TRAPS GO TO 2\$
 :WRITE DATA AT FIRST LOCATION OF BANK 2 IN BOARD
 :WRITE DATA AT SECOND LOCATION OF BANK 2 IN BOARD
 :UPDATE PAR4 TO POINT TO UPPER BOARDS
 :KEEP GOING TILL NO MORE MEMORY
 :PUT CSR MAP IN R3
 :UPDATE CSR COUNTER
 :UPDATE CSRNO
 :BRANCH IF CSR EXISTS
 :ANY CSR'S LEFT?
 :BRANCH IF NOT
 :LOOK FOR NEXT CSR
 :SET DIAGNOSTIC CHECK MODE IN CSR
 :LOOK FOR NEXT CSR
 :NE MEMORY TRAPS NOW GO TO 6\$
 :RESET CSR POINTER
 :REMAP PAR4 TO POINT TO BANK 2
 :TEST NONASSERTED LOCATIONS
 :UPDTAE CSR POINTER
 :NOT FOUND?
 :BRANCH IF NOT
 :GET TYPE OF ECC MEMORY
 :BRANCH IF MS11-P
 :MS11-M CHECK BITS
 :MS11-P CHECK BITS
 :GET CSR CONTENTS
 :CLEAR ALL BUT CHECKBITS
 :PROPER CHECKBITS?
 :BRANCH IF SO
 :TRY NEXT CSR IF NOT
 :WRITE NON-ASSERTED CSR # IN PGMCSR
 :NE TRAPS GO TO 8\$
 :PUT CSR MAP IN R3
 :UPDATE CSR COUNTER
 :UPDATE CSRNO
 :BRANCH IF CSR EXISTS
 :ANY CSR'S LEFT?
 :BRANCH IF NOT
 :LOOK FOR NEXT CSR
 :SET DIAGNOSTIC CHECK MODE IN CSR
 :LOOK FOR NEXT CSR
 :TEST ASSERTED LOCATIONS

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 163-1
 READ AND WRITE ALL CSR BITS

6403	007500	001452				BEQ	10\$	
6404	007502	032760	000002	002460		BIT	#BIT1,CSRINFO(R0)	:CHECK FOR TYPE OF ECC MEMORY
6405	007510	001003				BNE	76\$:BRANCH IF MS11-P
6406	007512	012702	000157			MOV	#157,R2	:CHECK BITS FOR MS11-M
6407	007516	000402				BR	77\$	
6408	007520	012702	000040		76\$:	MOV	#40,R2	:CHECK BITS FOR MS11-P
6409	007524	016004	172100		77\$:	MOV	CSRADD(R0),R4	
6410	007530	072427	177773			ASH	#-5,R4	
6411	007534	042704	177600			BIC	#^C177,R4	
6412	007540	020204				CMP	R2,R4	:PROPER CHECKBITS?
6413	007542	001401				BEQ	9\$:BRANCH IF SO
6414	007544	000747				BR	8\$:TRY NEXT CSR IF NOT
6415	007546	110037	002531		9\$:	MOVB	R0,PGMCSR+1	:WRITE ASSERTED CSR # IN PGMCSR
6416	007552	052737	100000	002530		BIS	#BIT15,PGMCSR	:SET INTERLEAVED INDICATOR IN PGMCSR
6417	007560	104502				CLRCR		
6418	007562	012737	002000	172350		MOV	#2000,KIPAR4	
6419	007570					SET4	#12\$:NE MEMORY TRAPS GO TO 12\$
6420	007576	012771	000000	000000	11\$:	MOV	#0,@(R1)	:WRITE DATA AT FIRST LOCATION OF BANK 2 IN BOARD
6421	007604	012771	000000	000002		MOV	#0,@2(R1)	:WRITE DATA AT SECOND LOCATION OF BANK 2 IN BOARD
6422	007612	062737	010000	172350		ADD	#10000,KIPAR4	:UPDATE PAR4 TO POINT TO UPPER BOARDS
6423	007620	000766				BR	11\$	
6424	007622	104423			12\$:	CACHON		
6425	007624	000407				BR	CLRMEM	
6426	007626	012737	001000	172350	10\$:	MOV	#1000,KIPAR4	
6427	007634					TYPE	MSG126	:ERROR - PROGRAM CSR NOT FOUND!
6428	007640	005037	002530			CLR	PGMCSR	:SET TO DEFAULT OF 0

6430 007644

SUBTST <<CLEAR ALL MEMORY SPACE FROM BANK 2 ON>>

:SUBTEST CLEAR ALL MEMORY SPACE FROM BANK 2 ON

6431
6432
6433
6434
6435
6436

: THIS ROUTINE CLEARS ALL MEMORY SPACE BEGINNING AT ADDRESS 200,000 AND
: CONTINUES UNTIL THERE IS NO MEMORY LEFT. IT SHOULD CLEAR ANY PARITY ERRORS
: CREATED BY THE LAST ROUTINE, AND CLEAN UP ANY JUNK LEFT HANGING AROUND IN
: HIGHER MEMORY.

6437 007644
6438 007652 005037 006206
6439 007656 012737 000001 002074
6440 007664 012737 002000 172350
6441 007672 012701 100000
6442 007676 020127 117776
6443 007702 001003
6444 007704 012737 177777 006206
6445 007712 005021
6446 007714 005737 006206
6447 007720 001001
6448 007722 000765
6449 007724 062737 000200 172350
6450 007732 022737 170000 172350
6451 007740 001405
6452 007742 005037 006206
6453 007746 012701 100000
6454 007752 000751
6455 007754 000240
6456 007756 005037 006206
6457 007762

CLRMEM: SET4 #CLREX ;NONEM TRAPS GO TO CLREX
CLR TRACE
MOV #1,NOPAR ;IGNORE PARITY ERRORS
MOV #200,KIPAR4 ;SET UP MAP TO START AT BANK 2
MOV #100000,R1 ;R1 MAPS TO KIPAR4
1\$: CMP R1,#117776 ;WHOLE 16K BANK DONE?
BNE 2\$;KEEP GOING IF NOT
MOV #-1,TRACE ;USE TRACE FLAG TO FLAG END OF BANK
2\$: CLR (R1)+ ;CLEAR CONTENTS & INCREMENT
TST TRACE ;EOB FLAG SET?
BNE 3\$;GO TO NEXT BANK IF SO
BR 1\$
3\$: ADD #200,KIPAR4 ;SET MAP FOR NEXT BANK
CMP #170000,KIPAR4 ;ARE WE AT THE PERIPHERAL PAGE
BEQ CLREX ;YES-GO ON
CLR TRACE ;RESET FLAG
MOV #100000,R1 ;RESET R1
BR 1\$;CLEAR NEXT BANK
CLREX: NOP
CLR TRACE
RES4

6460 010004

ANA2: SUBTST <<MATCH ALL CSR'S WITH MEMORY>>

: *SUBTEST MATCH ALL CSR'S WITH MEMORY

6461
6462
6463
6464
6465
6466
6467
6468
6469
6470
6471
6472
6473
6474
6475
6476
6477
6478
6479
6480
6481
6482
6483
6484
6485
6486
6487
6488
6489
6490

: * THE SECOND PART OF THE ANALYSIS MATCHES UP THE CSR'S WITH THE MEMORY, AND
: * INSTALLS ALL THE INFORMATION FOUND IN THE CONFIGURATION TABLE. FOR ECC,
: * THIS IS DONE BY TAKING EACH CSR FOUND IN THE PREVIOUS SECTION SEQUENTIALLY
: * AND CHECKING THROUGH ALL OF MEMORY, ONE BANK AT A TIME, TO SEE WHICH BANKS
: * RESPOND TO THE CSR IN QUESTION. THE FIRST DOUBLE WORD PAIR IN EACH BANK ARE
: * WRITTEN WITH DATA AND DIAGNOSTIC CHECK MODE SET IN THE CSR IN ORDER TO AC-
: * COMPLISH THIS. ALL POSSIBLE CONFIGURATIONS OF DOUBLE WORD PAIRS (NON-INTER-
: * LEAVED, 64K INTERLEAVED, OR 128K INTERLEAVED) ARE CHECKED FOR EACH BANK
: * THROUGH USE OF TESTADD AND KERNEL INSTRUCTION PAGE ADDRESS REGISTERS 4 AND
: * 5. IF WE GET THE #NOERR CHECKBITS BACK, WE HAVE A MATCH. IF NOT, THE ROUT-
: * INE CHECKS FOR SINGLE OR DOUBLE BIT ERRORS.
: * IF ONE OR THE OTHER IS FOUND, THE ERROR ADDRESS IS CHECKED
: * TO SEE IF IT IS THAT BANK. IF IT IS, WE HAVE A MATCH. AT THE END OF EACH
: * BANK PASS, FOR EACH CSR PASS, THE PROGRAM COMES UP WITH A NUMBER, STORED IN
: * "I", WHICH DENOTES THE FOLLOWING:

: *
: * I MEMORY DESCRIPTION
: * -
: * 0 NON-EXISTANT MEMORY
: * 1 NON-INTERLEAVED MEMORY MS11-L,MS11-P
: * 2 64K INTERLEAVED, A1 NOT ASSERTED MEMORY
: * 3 128K INTERLEAVED, A1 NOT ASSERTED MEMORY
: * 4 64K INTERLEAVED, A1 ASSERTED MEMORY
: * 5 128K INTERLEAVED, A1 ASSERTED MEMORY

: * NOTE - I=2 THROUGH I=5 CAN ONLY OCCUR WITH MS11-M MEMORY.

: * NOTE THAT PARITY MEMORY WRITES WRONG PARITY TO THE DOUBLE WORDS, THEN LOOKS
: * FOR THE PARITY ERROR BIT TO BE SET. IF THE BIT IS SET, WE HAVE A MATCH.

6491 010004
6492 010012 005037 002312
6493 010016 012701 002410
6494 010022 013703 002222
6495 010026 005000
6496 010030 005005
6497 010032 005737 002452
6498 010036 001403
6499 010040 005037 002556
6500 010044 000413
6501 010046 022737 000167 002554 7\$:
6502 010054 001407
6503 010056 013702 002554
6504 010062 005202
6505 010064 072227 000011
6506 010070 010237 002556
6507 010074 012702 000004 1\$:
6508 010100 012737 001000 172350
6509 010106 012737 001000 172352
6510 010114 006303 2\$:
6511 010116 103420
6512 010120 062700 000002
6513 010124 010037 002150

SET4 #100\$;NE MEMORY TRAPS GO TO 100\$
CLR CHECK ;CLEAR CHECK
MOV #TESTADD,R1 ;SET UP THE VIRTUAL ADDR. POINTER
MOV TOTCSRS,R3 ;MOVE CSR MAP INTO R3
CLR R0 ;CLEAR THE CSR POINTER
CLR R5 ;CLEAR THE PROGRAM CSR STATUS POINTER
TST NO22BIT ;IS THIS AN 11/44 OR 11/24?
BEQ 7\$;BRANCH IF IT IS
CLR LASTBLOCK ;ADJUST LASTBLOCK INDICATOR FOR 124K MACHINE
BR 1\$;BRANCH OVER NEXT PIECE OF CODE
CMP #167, LASTBANK ;IS THERE UNIBUS MEMORY ABOVE 17000000?
BEQ 1\$;BRANCH IF NOT
MOV LASTBANK,R2 ;SET UP A NEW LAST BLOCK INDICATOR
INC R2
ASH #9, R2
MOV R2, LASTBLOCK
MOV #4, R2 ;R2 IS INDEX FOR CONFIG TABLE
MOV #1000, KIPAR4 ;SET KIPAR4 FOR BANK 1
MOV #1000, KIPAR5 ;SET KIPAR5 FOR BANK 1
ASL R3 ;DOES THIS CSR EXIST?
BCS 3\$;BRANCH IF IT DOES EXIST
ADD #2, R0 ;INCREMENT THE CSR POINTER
MOV R0, CSRNO ;STORE IT IN CSRNO ALSO

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 166-1
MATCH ALL CSR'S WITH MEMORY

6514	010130	005703			TST	R3		:ARE THERE ANY MORE CSR'S TO DO?
6515	010132	001370			BNE	2\$:BRANCH IF ALL CSRS NOT DONE
6516	010134	012737	001000	172350	MOV	#1000,KIPAR4		:RESTORE KIPAR4
6517	010142	012737	001200	172352	MOV	#1200,KIPAR5		:RESTORE KIPAR5
6518	010150	013706	002562		MOV	KSTACK,SP		:RESTORE STACK
6519	010154	000137	011402		JMP	SUBAAS		:JUMP TO SUBAAS IF ALL CSR'S ARE DONE
6520	010160	010037	002150		MOV	RO,CSRNO		:MAKE SURE CSRNO IS UPDATED
6521	010164	104424			3\$:	CACHOFF		:TURN THE CACHE OFF
6522	010166	000240			13\$:	NOP		
6523	010170	012737	100000	002410	45\$:	MOV	#100000,TESTADD	:SET UP VIRTUAL ADDRESS TO KIPAR4
6524	010176	012737	120002	002412		MOV	#120002,TESTADD+2	:SET UP VIRTUAL ADDRESS TO KIPAR5
6525	010204	032762	000040	002652		BIT	#BIT5,CONFIG(R2)	:IS THIS A BANK TO SKIP?
6526	010212	001402			BEQ	43\$:NO - BRANCH AROUND NEXT INSTRUCTION
6527	010214	000137	011316		JMP	6\$:YES - GO TO END OF BANK
6528	010220	005037	002450		43\$:	CLR	I	:CLEAR THE MEMORY CONFIGURATION COUNTER
6529	010224	005771	000000		4\$:	TST	@(R1)	:TEST TO SEE THAT THERE IS MEMORY PRESENT
6530	010230	005237	002450			INC	I	
6531	010234					PUSH	@(R1),@(R1)	:SAVE THE LOCATIONS UNDER TEST
6532	010244	032760	000003	002460		BIT	#BIT1!BIT0,CSRINFO(R0)	:IS THIS PARITY MEMORY?
6533	010252	001014			BNE	34\$:NO - BRANCH
6534	010254	052760	000004	172100	BIS	#BIT2,CSRADD(R0)		:SET WRITE WRONG PARITY
6535	010262	012771	123456	000000	MOV	#123456,@(R1)		:SET THE FIRST LOCATION UNDER TEST
6536	010270	012771	123456	000002	MOV	#123456,@2(R1)		:SET THE SECOND LUT
6537	010276	005060			CLR	CSRADD(R0)		:CLEAR THE CSR
6538	010302	000411			BR	41\$:TEST LOCATIONS
6539	010304	012771	123456	000000	34\$:	MOV	#123456,@(R1)	:SET THE FIRST LOCATION UNDER TEST
6540	010312	012771	123456	000002		MOV	#123456,@2(R1)	:SET THE SECOND LUT
6541	010320	104503			CLR1CSR			:RESET CSR
6542	010322	104475			CB1CSR			:SET DIAG. CHECK MODE IN CSR UNDER TEST
6543	010324	000240			NOP			:DEBUG AID
6544	010326	005771	000000		41\$:	TST	@(R1)	:READ THE FIRST LUT TO WRITE CKBITS. INTO CSR
6545	010332	104426			READCSR			:READ THE CSR UNDER TEST
6546	010334	000240			NOP			:DEBUG AID
6547	010336	013704	002146		MOV	CSR,R4		:GET THE CHECKBITS FROM THE CSR
6548	010342	000240			NOP			:DEBUG AID
6549	010344	010437	002432		MOV	R4,TEMP		:SAVE IN TEMP FOR LATER
6550	010350	104503			CLR1CSR			:RESET CSR
6551	010352				POP	@2(R1),@(R1)		:RESTORE LOCATIONS UNDER TEST
6552	010362	032760	000003	002460	BIT	#BIT1!BIT0,CSRINFO(R0)		:IS THIS PARITY MEMORY?
6553	010370	001004			BNE	42\$:NO - BRANCH
6554	010372	005704			TST	R4		:DID WE GET A PARITY ERROR?
6555	010374	100431			BMI	25\$:YES - FILL IN CONFIG TABLE
6556	010376	000137	011316		JMP	6\$:NO - JUMP TO END OF BANK
6557	010402	072427	177773		42\$:	ASH	#-5,R4	:MANIPULATE THE CSR BITS
6558	010406	042704	177600		BIC	#C177,R4		:INTO A USABLE FORM.
6559	010412	032760	000002	002460	BIT	#BIT1,CSRINFO(R0)		:WHAT KIND OF ECC MEMORY IS THIS??
6560	010420	001005			BNE	76\$:BRANCH IF MS11-P
6561	010422	012737	000157	002314	MOV	#157,CBITS		:MS11-M CHECK BITS
6562	010430	000240			NOP			:DEBBUG AIDE
6563	010432	000404			BR	77\$		
6564	010434	012737	000040	002314	76\$:	MOV	#40,CBITS	:MS11-P CHECK BITS
6565	010442	000240			NOP			:DEBBUGING AIDE
6566	010444	023704	002314		77\$:	CMP	CBITS,R4	:DO THE CHECKBITS COMPARE TO WHAT WAS WRITTEN?
6567	010450	000240			NOP			:DEBBUG AIDE
6568	010452	001402			BEQ	25\$:BRANCH IF THERE IS A MATCH
6569	010454	000137	011004		JMP	22\$:ELSE BRANCH IF NOT THE SAME
6570					:*			

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 166-2
MATCH ALL CSR'S WITH MEMORY

```

6571
6572
6573 010460 010004      25$:  MOV     R0,R4           ;GET THE CSR NUMBER
6574 010462 000240      NOP
6575 010464 006204      ASR     R4             ;SET IT UP FOR USE IN THE
6576 010466 000304      SWAB   R4             ;CONFIGURATION TABLE.
6577 010470 042704 170377  BIC     #170377,R4    ;CLEAR OFF EXTRANEIOUS BITS
6578 010474 032737 000004 002450 BIT     #BIT2,I       ;INTERLEAVED A1 ASSERTED MEMORY FOUND?
6579 010502 001402      BEQ     15$           ;BRANCH IF NOT
6580 010504 072427 000004      ASH     #4,R4        ;PUT CSR NUMBER IN INTERLEAVED CSR SLOT
6581 010510 050462 002652      15$:  BIS     R4,CONFIG(R2) ;PUT CSR NUMBER IN CONFIG. TABLE
6582 010514 016004 002460      MOV     CSRINFO(R0),R4 ;GET MEMORY TYPE
6583 010520 042704 177770      BIC     #^C7,R4     ;CLEAR OFF THE EXTRANEIOUS BITS
6584 010524 000304      SWAB   R4             ;MOVE INTO PROPER POSITION
6585 010526 050462 002654      BIS     R4,CONFIG+2(R2) ;SET IT INTO THE CONFIG TABLE
6586 010532 022737 000001 002450 CMP     #1,I         ;WAS THIS NON-INTERLEAVED MEMORY?
6587 010540 001431      BEQ     24$           ;BRANCH IF IT WAS
6588 010542 052762 010000 002654 BIS     #BIT12,CONFIG+2(R2) ;SET THE INTERLEAVED BIT
6589 010550 010204      MOV     R2,R4        ;SAVE THE CURRENT BANK INDEX
6590 010552 032737 000001 002450 BIT     #BIT0,I       ;WAS THIS 128K INTERLEAVED?
6591 010560 001006      BNE     5$           ;BRANCH IF TRUE
6592 010562 052762 004000 002654 BIS     #BIT11,CONFIG+2(R2) ;SET 64K INTERLEAVED FLAG IN CONFIG
6593 010570 062704 000020      ADD     #20,R4       ;SET NEW BANK POINTER TO 4 BANKS AHEAD
6594 010574 000402      BR      16$          ;JUMP OVER NEXT INSTRUCTION
6595 010576 062704 000040      5$:  ADD     #40,R4       ;SET NEW BANK POINTER 8 BANKS AHEAD
6596 010602 052764 000040 002652 16$:  BIS     #BIT5,CONFIG(R4) ;SET SKIP ECC LOGIC TESTS FLAG
6597 010610 056264 002652 002652 BIS     CONFIG(R2),CONFIG(R4) ;SET OTHER INFO INTO THAT BANK
6598 010616 056264 002654 002654 BIS     CONFIG+2(R2),CONFIG+2(R4)
6599
6600
6601
6602 010624 022737 001000 172350 24$:  CMP     #1000,KIPAR4 ;IS THIS BANK 1 ?
6603 010632 001402      BEQ     30$           ;BRANCH IF TRUE
6604 010634 000137 011156      JMP     33$           ;ELSE JUMP TO END OF THIS BANK
6605 010640 032737 100020 002432 30$:  BIT     #BIT15!BIT4,TEMP ;WAS THERE A SBE OR DBE?
6606 010646 001417      BEQ     10$           ;BRANCH IF NOT
6607 010650 013704 002432      MOV     TEMP,R4      ;GET CSR CONTENTS
6608 010654 072427 177767      ASH     #-9,R4       ;MAKE ERROR ADDRESS INTO BANK #
6609 010660 022704 000001      CMP     #1,R4        ;ERROR IN BANKS 0 OR 1?
6610 010664 003010      BGT     10$           ;BRANCH IF NOT
6611 010666 052762 000001 002652 BIS     #BIT0,CONFIG(R2) ;SET ERROR FLAG IN CONFIG TABLE
6612 010674 105262 002654      INCB   CONFIG+2(R2) ;ADD ONE TO BANK ERROR COUNT
6613 010700      SET     CFGERROR     ;PRINT CONFIG TABLE
6614 010706 053737 002656 002652 10$:  BIS     CONFIG+4,CONFIG ;SET UP INFORMATION IN BANK ZERO
6615 010714 053737 002660 002654 BIS     CONFIG+6,CONFIG+2
6616 010722 000240      NOP
6617 010724 022737 000001 002450 CMP     #1,I         ;DEBUG AID
6618 010732 001002      BNE     46$          ;WAS THIS NON-INTERLEAVED MEMORY
6619 010734 000137 011316      JMP     6$           ;NO - BRANCH OVER NEXT STMT.
6620 010740 012704 000020      46$:  MOV     #20,R4       ;YES - JUMP TO END OF THIS BANK
6621 010744 032737 000001 002450 BIT     #BIT0,I       ;SET UP COUNTER FOR 64K INTERLEAVED
6622 010752 001402      BEQ     26$           ;WAS IT 128K INTERLEAVED?
6623 010754 062704 000020      ADD     #20,R4       ;BRANCH IF NOT
6624 010760 053764 002652 002652 26$:  BIS     CONFIG,CONFIG(R4) ;SET UP COUNTER FOR 128K INTERLEAVED
6625 010766 053764 002654 002654 BIS     CONFIG+2,CONFIG+2(R4) ;SET OTHER BANK WITH SAME INFORMATION
6626 010774 052764 000040 002652 BIS     #BIT5,CONFIG(R4) ;AS IN BANK 0
6627 011002 000465      BR      33$          ;SET SKIP ECC LOGIC TESTS FLAG
                          ;BRANCH

```


CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 166-3
MATCH ALL CSR'S WITH MEMORY

```

6628
6629
6630
6631 011004 032737 100020 002146 22$: BIT #BIT15!BIT4,CSR ;SBE OR DBE FLAGS SET?
6632 011012 001001 BNE 8$ ;BRANCH IF TRUE
6633 011014 000460 BR 33$ ;CHECK TO SEE IF IT IS MS11-M
6634 011016 013704 002150 8$: MOV CSRNO,R4 ;GET CSRNO
6635 011022 042764 000006 172100 BIC #6,CSRADD(R4) ;TURN OFF DIAG CHECK & ECC DISABLE
6636 011030 PUSH RC,R1 ;SAVE R0 & R1
6637 011034 016401 172100 MOV CSRADD(R4),R1 ;GET CSR INFORMATION
6638 011040 072127 177773 ASH #-5,R1 ;SET UP ERROR ADDRESS
6639 011044 042701 177600 BIC #^C177,R1
6640 011050 052764 040000 172100 BIS #BIT14,CSRADD(R4) ;GET EXTENDED ERROR ADDRESS BITS
6641 011056 016400 172100 MOV CSRADD(R4),R0 ;READ FROM CSR
6642 011062 042764 040000 172100 BIC #BIT14,CSRADD(R4) ;TURN OFF EUB BIT
6643 011070 042700 177037 BIC #^C740,R0 ;SET UP EXTENDED BITS
6644 011074 006300 ASL R0
6645 011076 006300 ASL R0
6646 011100 060001 ADD R0,R1 ;SET UP TOTAL ERROR ADDRESS
6647 011102 010104 27$: MOV R1,R4 ;SAVE IN R4
6648 011104 POP R1,R0 ;RESTORE R0 & R1
6649 011110 072427 000005 ASH #5,R4 ;SET ERROR ADDRESS UP IN PAR NOTATION
6650 011114 020437 172350 CMP R4,KIPAR4 ;DOES IT EQUAL KIPAR4?
6651 011120 001001 BNE 28$ ;BRANCH IF FALSE
6652 011122 000403 BR 35$ ;YES - MARK INFO IN CONFIG TABLE
6653 011124 020437 172352 28$: CMP R4,KIPAR5 ;DOES IT EQUAL KIPAR5?
6654 011130 001012 BNE 33$ ;BRANCH IF FALSE
6655 011132 052762 000001 002652 35$: BIS #BIT0,CONFIG(R2) ;SET BANK ERROR FLAG
6656 011140 105262 002654 INCB CONFIG+2(R2) ;INCREMENT BANK ERROR COUNTER
6657 011144 SET CFGERRR ;PRINT CONFIG TABLE
6658 011152 000137 010460 JMP 25$ ;YES - MARK INFO IN CONFIG TABLE
6659
6660
6661
6662
6663 011156 032760 000001 002460 33$: BIT #BIT0,CSRINFO(R0) ;IS THIS MS11-M MEMORY?
6664 011164 001454 BEQ 6$ ;NO - GO TO END OF BANK
6665 011166 032760 000002 002460 BIT #BIT1,CSRINFO(R0)
6666 011174 001050 BNE 6$
6667 011176 022737 000001 002450 CMP #1,I ;IS THIS 1ST TIME THROUGH?
6668 011204 103410 BLO 18$ ;BRANCH IF NOT
6669 011206 162737 000002 002412 SUB #2,TESTADD+2 ;TRY AS 64K INTERLEAVED
6670 011214 062737 004000 172352 ADD #4000,KIPAR5 ;A1 NON-ASSERTED MEMORY
6671 011222 000137 010224 JMP 4$ ;TRY TO MATCH AGAIN
6672 011226 022737 000004 002450 18$: CMP #4,I ;4TH TIME THROUGH?
6673 011234 001404 BEQ 20$ ;YES - BRANCH
6674 011236 022737 000002 002450 CMP #2,I ;2ND TIME THROUGH
6675 011244 103405 BLO 12$ ;NO - BRANCH
6676 011246 062737 004000 172352 20$: ADD #4000,KIPAR5 ;TRY AS 128K INTERLEAVED
6677 011254 000137 010224 JMP 4$ ;TRY TO MATCH AGAIN
6678 011260 022737 000003 002450 12$: CMP #3,I ;THIRD TIME THROUGH?
6679 011266 103413 BLO 6$ ;NO - BRANCH
6680 011270 062737 000002 002410 ADD #2,TESTADD ;TRY TESTING THE BANK
6681 011276 062737 000002 002412 ADD #2,TESTADD+2 ;AS A1 ASSERTED
6682 011304 162737 004000 172352 SUB #4000,KIPAR5 ;64K INTERLEAVED MEMORY
6683 011312 000137 010224 JMP 4$ ;TRY TO MATCH AGAIN
6684

```


CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 166-4
 MATCH ALL CSR'S WITH MEMORY

6685						:*END OF BANK ROUTINE		
6686						:*		
6687	011316	104503			6\$:	CLR1CSR		:CLEAR THE CSR UNDER TEST
6688	011320	062702	000004			ADD #4,R2		:UPDATE CONFIGURATION POINTER
6689	011324	062737	001000	172350		ADD #1000,KIPAR4		:UPDATE KIPAR4 TO NEXT BANK
6690	011332	013737	172350	172352		MOV KIPAR4,KIPAR5		:AND UPDATE KIPAR5
6691	011340	000240				NOP		:DEBUG AID
6692	011342	023737	002556	172350		CMP LASTBLOCK,KIPAR4		:HAVE WE DONE THE WHOLE MEMORY SPACE?
6693	011350	101402				BLOS 19\$:BRANCH IF DONE ;R-C
6694	011352	000137	010170			JMP 45\$:JUMP IF NOT DONE
6695	011356	062700	000002		19\$:	ADD #2,R0		:INCREMENT CSR POINTER
6696	011362	000240				NOP		:DEBUG AID
6697	011364	104423				CACHON		:TURN ON THE CACHE
6698	011366	000137	010074			JMP 1\$:JUMP TO TRY NEXT CSR
6699								
6700	011372	062706	000004		100\$:	ADD #4,SP		:RESTORE STACK ;R-C
6701	011376	000137	011316			JMP 6\$:GO TO END OF BANK ROUTINE ;R-C

6703 011402 104423
6704 011404 104472
6705 011406

SUBAAS: CACHON ;MAKE SURE THE CACHE IS ON
ECCINIT ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
NEWTST <<TEST BANK 0 ACCESSES>>

:*TEST 2 TEST BANK 0 ACCESSES

011406 000004

6706
6707
6708
6709
6710
6711
6712 011410 005037 002070
6713 011414 012737 000001 002074
6714 011422 005037 002066
6715 011426 012737 000001 002076
6716 011434
6717 011442 005000
6718 011444 012701 040000
6719 011450 104424
6720 011452 005720
6721 011454 077102
6722 011456 104423
6723
6724 011460 005737 002070
6725 011464 001403
6726 011466
6727 011474 005737 002066
6728 011500 001406
6729 011502 162737 000002 002032
6730 011510
6731 011516 053737 002104 002652
6732 011524
6733
6734 011546

TST2: SCOPE
;THIS DOES A "TST" INSTRUCTION ON EVERY LOCATION IN BANK #0 TO SEE
;IF IT GETS ANY PARITY TRAPS.
;SINCE EVERY LOCATION IS EITHER LOADED OR WRITTEN INTO BY THE PROGRAM
;PRIOR TO THIS POINT - THEN A PARITY ERROR IMPLIES THAT THERE IS A
;HARDWARE FAILURE IN THE MEMORY.
;THESE ERRORS ARE COUNTED AND A FATAL ACTION IS TAKEN
CLR PARCNT ;CLEAR PARITY ERROR COUNTER
MOV #1,NOPAR ;SET THE NO PARITY ERROR FLAG
CLR NEMCNT ;CLEAR NON-EXISTANT MEMORY ERROR COUNTER
MOV #1,NONEM ;SET THE NON-EXISTANT MEMORY ERROR MODE TO COUNT
SET4 #NONEXIST ;TRAPS TO 4 GOTO NONEXIST
CLR R0
MOV #SIZE,R1
CACHOFF ;TURN CACHE OFF
1\$: TST (R0)+ ;SEE IF I CAN DO A READ ACCESS WITHOUT A PARITY TRAP
SOB R1,1\$
CACHON ;TURN CACHE ON
;SEE IF ANY FAILURES
TST PARCNT ;ANY PARITY ERRORS?
BEQ 2\$;NO - SKIP
FATAL 3
2\$: TST NEMCNT ;ANY NON-EXISTANT MEMORY (HOLES)?
BEQ 3\$;SKIP IF EQUAL
SUB #2,ADDRESS ;UPDATE 1ST ADDRESS FAILURE FROM AUTO INCREMENT #
FATAL 4
3\$: BIS CPUBIT,CONFIG ;SET CORRECT ACCESSED BIT ON BANK 0
RES4 ;RESET TRAPS TO 4 TO DEFAULT

SUBTST <<ENABLE ECC FOR CORRECT TRAPS>>

:*SUBTEST ENABLE ECC FOR CORRECT TRAPS

6735 011546
6736 011564 104506
6737 011566
6738 011570 104472
6739 011572

IF #SWO SET.IN @SWR OR ACTFLAG IS TRUE
ENASBE ;TRAP ON SINGLE BIT ERRORS
ELSE
ECCINIT ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
END ;OF IF #SWO

6742 011572

NEWTST <<TEST BANKS 1-200 (OCTAL) FOR ZEROS & ONES>>

:*TEST 3 TEST BANKS 1-200 (OCTAL) FOR ZEROS & ONES

6743				
6744				
6745				
6746				
6747				
6748				
6749				
6750	011574	005037	002100	
6751	011600	012737	000001	002074
6752	011606	012737	000002	002076
6753	011614			
6754	011622	022737	000001	003754
6755	011630	001407		
6756	011632	012737	012430	002520
6757	011640	012737	012432	002522
6758	011646	000411		
6759	011650			
6760	011656	012737	177644	002520
6761	011664	012737	177646	002522
6762	011672	005237	002100	
6763	011676	023737	002554	002100
6764	011704	103457		
6765	011706	013701	002100	
6766	011712	006301		
6767	011714	006301		
6768	011716	010137	002102	
6769	011722	005037	002072	
6770	011726	005037	002070	
6771	011732	005037	002066	
6772	011736			
6773	011752	105761	002652	
6774	011756	100555		
6775	011760	012777	000207	170532
6776	011766	012700	060000	
6777	011772	010004		
6778	011774	012701	040000	
6779	012000	010103		
6780	012002	005002		
6781	012004	104424		
6782	012006			
6783	012014	022737	000001	003754
6784	012022	001403		
6785	012024	004737	012424	
6786	012030	000402		
6787	012032	004737	177640	
6788	012036	104417		
6789	012040	104423		
6790	012042	000416		
6791	012044	005037	002100	
6792	012050			
6793	012072	005037	002074	
6794	012076	000564		

```

TST3:  SCOPE
        :EACH BANK IS TESTED FOR EXISTANCE AND IF IT EXISTS
        :THEN IS IS TESTED FOR ZEROS & ONES.
        :EXCEPT -
        :
        :   PROTECTED BANKS (WHERE THE PROGRAM IS) ARE ONLY TESTED BY
        :   "TST" INSTRUCTIONS LIKE BANK #0
        :ANY BAD BANKS ARE LOGGED IN THE CONFIGURATION TABLE.
        :THIS ROUTINE IS ONLY DOING A SMART SIZE - NOT ACTUAL TESTING!
        CLR   BANK
        MOV   #1,NOPAR           :SET NO PARITY ERROR FLAG
        MOV   #2,NONEM          :SET NON-EXISTANT MEMORY MODE TO EXIT TEST LOOP
        SET4  #NONEXIST         :TRAPS TO 4 GOTO NONEXIST
        CMP   #1,PROTYP         :IS THIS AN 11/44?
        BEQ   1$                :BRANCH IF TRUE
        MOV   #MTST3+4,LINK1    :SET UP LINKS
        MOV   #MTST3+6,LINK2
        BR    TAG9$
1$:     BMOV  MTST3              :PUT IN FAST MEMORY
        MOV   #UIPAR2,LINK1     :SET UP LINKS
        MOV   #UIPAR3,LINK2
TAG9$:  INC   BANK
        CMP   LASTBANK,BANK     :DONE?
        BLO  TAG2$              :YES - SKIP TO NEXT TEST
        MOV   BANK,R1
        ASL  R1
        ASL  R1                  :BANK * 4
        MOV  R1,BANKINDEX
        CLR  PATERR              :CLEAR PATTERN ERROR COUNTER
        CLR  PARCNT              :CLEAR PARITY ERROR COUNTER
        CLR  NEMCNT              :CLEAR NON-EXISTANT MEMORY COUNTER (HOLES)
        MAP  BANK                :MAP SUPERVISOR SPACE (TEST AREA) TO BANK
        TSTB CONFIG(R1)         :IS THIS BANK PROTECTED?
        BMI  TSTBANK             :YES - GO TEST BANK SPECIAL
        MOV  #207,@LINK1        :PUT "RETURN" INSTRUCTION AFTER WRITE ROUTINE
        MOV  #FIRST,R0
        MOV  R0,R4
        MOV  #SIZE,R1
        MOV  R1,R3
        CLR  R2
        CACHOFF                 :DATA IS ZEROS
        TESTAREA                 :TURN CACHE OFF
        CMP  #1,PROTYP           :ENTER SUPERVISOR MODE
        BEQ  1$                  :IS THIS AN 11/44?
        CALL MTST3               :BRANCH IF TRUE
        BR   2$
1$:     CALL  FASTCITY           :CALL TO THE USER INSTRUCTION PAR'S
2$:     KERNEL
        CACHON                   :ENTER KERNEL MODE
        BR   TAG3$               :TURN CACHE ON
        CLR  BANK                :SKIP NEXT INSTRUCTION
        RES4
        CLR  NOPAR               :RESET TRAPS TO 4 TO DEFAULT
        BR   SUBAAI              :INDICATE DEFAULT PARITY ACTION

```


CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 169-2
 T3 TEST BANKS 1-200 (OCTAL) FOR ZEROS & ONES

6852	012404				END ;OF IF PARCNT	
6853	012404				IF NEMCNT EQ #0	
6854	012412	053761	002104	002652	BIS CPUBIT,CONFIG(R1)	:ACCESSED BANK
6855	012420				END ;OF IF NEMCNT	
6856	012420	000137	011672		JMP TAG9\$	
6857	012424	010220			MTST3: MOV R2,(R0)+	:V177640
6858	012426	077102			SOB R1,MTST3	:V177642
6859	012430	000240			NOP	:V177644
6860	012432	012401			2\$: MOV (R4)+,R1	:V177646
6861	012434	020102			CMP R1,R2	:V177650
6862	012436	001402			BEQ 3\$:V177652
6863	012440	005237	002072		INC PATERR	:V177654
6864	012444	077306			3\$: SOB R3,2\$:V177660
6865	012446	000207			RETURN	:V177662

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 170
 T3 TEST BANKS 1-200 (OCTAL) FOR ZEROS & ONES

6867 012450

```
SUBAAI: SUBTST <<FIND SHADOW INHIBIT MODE POINTERS>>
:*****
:*SUBTEST      FIND SHADOW INHIBIT MODE POINTERS
:*****
```

```
6868
6869
6870
6871 012450 005037 002100
6872 012454 004737 047032
6873 012460 013700 002102
6874 012464
6875 012500
6876 012506 062700 000020
6877 012512 062737 000010 002100
6878 012520
6879 012522 062702 000040
6880 012526 062737 000020 002100
6881 012534
6882 012534 052760 000200 002652
6883 012542
6884 012544 005237 002100
6885 012550
6886 012550 023737 002554 002100
6887 012556 002336
```

```
* THIS SECTION LOOKS FOR INTERLEAVED MS11-M MEMORIES AND FIGURES OUT
* WHERE THE SHADOW INHIBIT MODE POINTERS ARE LOCATED. THESE AREAS
* ARE THEN MARKED AS PROGRAM SPACE.
CLR      BANK      ;RESET BANK TO ZERO
SHADL1: CALL EXBANK ;SET BANK PARAMETERS
MOV      BANKINDEX,R0
IF ACFLAG IS TRUE AND INTFLAG IS TRUE
    IF INT64K IS TRUE
        ADD #20,R0      ;POINT TO BANKINDEX + 4
        ADD #10,BANK    ;POINT TO BANK + 8
    ELSE
        ADD #40,R2      ;POINT TO BANKINDEX + 8
        ADD #20,BANK    ;POINT TO BANK + 16
    END; OF IF INT64K
    BIS   #BIT7,CONFIG(R0) ;MAKE NEW BANK PROGRAM SPACE
ELSE
    INC   BANK          ;GO TO NEXT BANK
END; OF IF ACFLAG
CMP     LASTBANK,BANK  ;HAVE WE DONE ALL THE BANKS?
BGE     SHADL1         ;BRANCH IF NOT
```


6890 012560

NEWSTST <<ECC INHIBIT MODE POINTER TEST>>

:*TEST 4 ECC INHIBIT MODE POINTER TEST

012560 000004

TST4: SCOPE
:THE MS11-M OR MF11S-K INHIBIT ECC DISABLE AND DIAGNOSTIC CHECK MODE
:ON THE BOTTOM FIRST OR SECOND 16K WORDS CONTROLLED BY A CSR. THIS
:IS CONSIDERED TO BE A PROTECTED BANK BY THE PROGRAM. IT MAY BE
:QUITE COMPLEX TO DETERMINE ON A GIVEN SYSTEM CONFIGURATION WHICH
:BANKS CAN BE PROTECTED;
:SO
:THIS ROUTINE ATTEMPS TO CREATE A DOUBLE BIT ERROR IN ADDRESS 0 & 2
:OF EVERY ECC BANK. ECC HARDWARE WILL PREVENT THIS FROM HAPPENING
:IN PROTECTED BANKS WHICH SHOULD ALWAYS INCLUDE BANK ZERO - WHERE
:THE PROGRAM IS.

6891
6892
6893
6894
6895
6896
6897
6898
6899
6900
6901
6902
6903
6904
6905
6906
6907
6908 012562 104424
6909 012564 012737 177777 002154
6910 012572
6911 012576 012701 060000
6912 012602 004737 047032
6913 012606 013700 002102
6914 012612
6915 012620
6916 012626
6917 012634
6918 012642 012703 040000
6919 012646 012737 000001 002236
6920 012654
6921 012656 012703 000002
6922 012662
6923 012662 116002 002653
6924 012666 006302
6925 012670 042702 177741
6926 012674 010237 002150
6927 012700
6928 012710 013737 002150 002154
6929 012716
6930 012724 052760 000100 002652
6931 012732
6932 012732 004737 013066

:WARNING:!!!!!!!!!!!!
: IN CASE OF HARDWARE FAILURE IT IS COMMON THAT A DOUBLE BIT ERROR
: WILL BE CREATED ON THE KERNEL STACK & "CRASH" THE DIAGNOSTIC
: DURING THIS ROUTINE. YOUR ONLY CLUE IS THAT YOU CAN GET AS FAR AS
: THIS ROUTINE BUT NOT PAST IT!
CACHOFF ;TURN CACHE OFF
MOV #-1,OLDCSR
FOR BANK := #0 TO LASTBANK
MOV #FIRST,R1 ;SET UP VIRT ADDR POINTER
CALL EXBANK
MOV BANKINDEX,R0
IF ACFLAG IS TRUE
IF MKFLAG IS TRUE
IF SKIPMK IS FALSE
IF INTFLAG IS TRUE
MOV #40000,R3 ;SET INDEX COUNTER
MOV #1,SPLTCSR ;MAP AS INTERLEAVED BANK
ELSE
MOV #2,R3 ;SET INDEX COUNTER
END; OF IF INTFLAG
MOVB CONFIG+1(R0),R2
ASL R2
BIC #^C36,R2
MOV R2,CSRNO
IF CSRNO NE OLDCSR
MOV CSRNO,OLDCSR
IF PFLAG IS FALSE
BIS #BIT6,CONFIG(R0)
END; OF IF PFLAG
CALL IMPTEST

CZMSP80 MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 173
 T4 ECC INHIBIT MODE POINTER TEST

6934	012736			IF INTFLAG IS TRUE	
6935	012744	116002	002653	MOVB CONFIG+1(R0),R2	
6936	012750	072227	177775	ASH #-3,R2	
6937	012754	042702	177741	BIC #^C36,R2	
6938	012760	010237	002150	MOV R2,CSRNO	
6939	012764	062701	000002	ADD #2,R1	;FIX POINTER FOR A1 ASSERTED HALF
6940	012770	004737	013066	CALL IMPTEST	
6941	012774	005037	002236	CLR SPLTCSR	
6942	013000			END; OF IF INTFLAG	
6943	013000			END; OF IF CSRNO	
6944	013000			END; OF IF SKIPMK	
6945	013000			END; OF IF MKFLAG	
6946	013000			END; OF IF ACFLAG	
6947	013000			END; OF FOR BANK	
6948	013014			MAP	;MAP TEST SPACE TO BANK 0
6949	013030	005037	002100	CLR BANK	
6950	013034			IF #SWO SET.IN @SWR OR ACTFLAG IS TRUE	
6951	013052	104506		ENASBE	;TRAP ON SINGLE BIT ERRORS
6952	013054			ELSE	
6953	013056	104472		ECCINIT	;TRAP ON DOUBLE BIT ERRORS (NORMAL)
6954	013060			END; OF IF #SWO	
6955	013060	104423		CACHON	;TURN THE CACHE BACK ON
6956	013062	000137	013374	JMP SUBAAR	;JUMP OVER THE SUBROUTINE

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 174
 T4 ECC INHIBIT MODE POINTER TEST

```

6958 013066 005004          IMPTEST:CLR      R4
6959 013070          MAP BANK                ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
6960 013104 005005          CLR      R5
6961 013106 012737 020000 002146      MOV      #BIT13,CSR
6962 013114          TESTAREA
6963 013122          PUSH      (R1)                ;ENTER TEST MODE
6964 013124 060301          ADD      R3,R1                ;SAVE TEST LOCATION
6965 013126          PUSH      (R1)                ;INDEX TO NEXT LOCATION
6966 013130 104505          CHK1DIS                ;SAVE TEST LOCATION
6967 013132 010411          MOV      R4,(R1)          ;DISABLE ECC & WRITE CHECKBITS FOR 1 CSR
6968 013134 160301          SUB      R3,R1          ;WRITE CHECKBITS (ALL ZEROS)
6969 013136 010411          MOV      R4,(R1)
6970 013140 104503          CLR1CSR                ;CLEAR CSR
6971 013142 005711          TST      (R1)           ;READ CHECKBITS INTO REAL CSR
6972 013144 104501          WAS1DBE                ;WAS THERE A DOUBLE BIT ERROR
6973
6974
6975          ;THIS MAKES SURE THAT SBE'S DON'T LOOK LIKE PROTECTED AREAS
6976 013146
6977 013150 012737 020000 002146      ON.NOERROR :1
6978 013156 104505          MOV      #BIT13,CSR
6979 013160 013711 002602          CHK1DIS                ;DISABLE ECC & WRITE CHECKBITS FOR 1 CSR
6980 013164 060301          MOV      ONES,(R1)
6981 013166 013711 002602          ADD      R3,R1
6982 013172 160301          MOV      ONES,(R1)
6983 013174 104503          SUB      R3,R1
6984 013176 005711          CLR1CSR                ;CLEAR CSR
6985 013200 104501          TST      (R1)           ;WAS THERE A DOUBLE BIT ERROR
6986 013202          WAS1DBE                ;WAS THERE A DOUBLE BIT ERROR
6987 013204          ON.NOERROR :2
6988 013214 104513          IF #BIT9 SET.IN CONFIG+2(R0) ;IS THIS A MS11-P
6989 013216 012737 023140 002146      CBREG                ;ENABLE CHECK/SYNDROME BIT REGISTER
6990 013224          MOV      #23140,CSR    ;WRITE DBE'S IN CSR
6991 013226 012737 027400 002146      ELSE                ;OR A MS11-M
6992 013234          MOV      #27400,CSR    ;WRITE DBE'S IN CSR
6993 013234 104505          END
6994 013236 010411          CHK1DIS                ;DISABLE ECC & WRITE CHECKBITS FOR 1 CSR
6995 013240 060301          MOV      R4,(R1)
6996 013242 010411          ADD      R3,R1          ;ADD INDEX TO GET TO SECOND WORD
6997 013244 160301          MOV      R4,(R1)
6998 013246 104503          SUB      R3,R1          ;SUBTRACT INDEX TO FIRST WORD
6999 013250 005711          CLR1CSR                ;CLEAR CSR
7000 013252 104501          TST      (R1)           ;WAS THERE A DOUBLE BIT ERROR
7001 013254          WAS1DBE                ;WAS THERE A DOUBLE BIT ERROR
7002 013256          ON.NOERROR :3
7003 013266 104513          IF #BIT9 SET.IN CONFIG+2(R0) ;IS THIS A MS11-P
7004 013270 012737 023604 002146      CBREG                ;ENABLE CHECK/SYNDROME BIT REGISTER
7005 013276          MOV      #23604,CSR    ;WRITE DBE'S IN CSR
7006 013300 012737 074000 002146      ELSE                ;IS IT A MS11-P
7007 013306          MOV      #74000,CSR    ;WRITE DBE'S IN CSR
7008 013306 104505          END
7009 013310 010411          CHK1DIS                ;DISABLE ECC & WRITE CHECKBITS FOR 1 CSR
7010 013312 060301          MOV      R4,(R1)
7011 013314 010411          ADD      R3,R1          ;INDEX TO SECOND WORD
7012 013316 104503          MOV      R4,(R1)
7013 013320 160301          CLR1CSR                ;CLEAR CSR
7014 013322 005711          SUB      R3,R1          ;GO BACK TO FIRST WORD
          TST      (R1)

```


CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 174-1
 T4 ECC INHIBIT MODE POINTER TEST

7015	013324	104501		WAS1DBE					
7016	013326			END ;OF ON.NOERROR ;3					;WAS THERE A DOUBLE BIT ERROR
7017	013326			END ;OF ON.NOERROR ;2					
7018	013326			END ;OF ON.NOERROR ;1					
7019	013326			ON.ERROR					
7020	013330	005205		INC R5					;IDENTIFY AS BAD BANK
7021	013332			END ;OF ON.ERROR					
7022	013332	104471		ECC1DIS					;DISABLE ERROR CORRECTION
7023	013334	010411		MOV R4,(R1)					;CLEAR OUT DOUBLE BIT ERROR!
7024	013336	060301		ADD R3,R1					;INDEX TO SECOND WORD
7025	013340	010411		MOV R4,(R1)					;CLEAR OUT DOUBLE BIT ERROR!
7026	013342	104503		CLR1CSR					
7027	013344	005705		TST R5					
7028	013346	001405		BEQ 1\$					
7029	013350	050560	002652	BIS R5,CONFIG(R0)					
7030	013354	105260	002654	INCB CONFIG+2(R0)					
7031	013360	104036		ERROR +36					
7032	013362			POP (R1)					
7033	013364	160301		SUB R3,R1					;RESTORE TEST LOCATION (2ND WORD)
7034	013366			POP (R1)					;GO BACK TO FIRST WORD
7035	013370	104417		KERNEL					;RESTORE TEST LOCATION (1ST WORD)
7036	013372	000207		RETURN					
7037									
7038	013374			SUBAAR: SET	STOPOK				;PROGRAM CAN NOW BE HALTED

7047 013402

```

7048 013402 012700 000020
7049 013406 012701 002460
7050 013412 005021
7051 013414 077002
7052 013416
7053 013422 004737 047032
7054 013426 013700 002102
7055
7056 013432
7057 013440 116003 002653
7058 013444 042703 177760
7059 013450 006303
7060 013452 005263 002460
7061 013456
7062
7063 013464
7064 013464
7065 013472 116003 002653
7066 013476 010304
7067 013500 042703 177760
7068 013504 072427 177774
7069 013510 042704 177760
7070 013514
7071 013520 042760 014000 002654
7072 013526 042760 170000 002652
7073 013534
7074 013536
7075 013536
7076 013540
7077 013542
7078 013542
7079 013550
7080 013550
7081 013550
7082 013550
7083 013564
7084 013570 005000
7085 013572 005001
7086 013574 005005
7087 013576 005037 014004
7088 013602 022761 000040 002460 2$:
7089 013610 002043
7090 013612 022761 000010 002460
7091 013620 002003
7092 013622 004737 014126
7093 013626 000434
7094 013630 016005 002652 3$:
7095 013634 032705 000002
7096 013640 001415
7097 013642 042705 170377
7098 013646 072527 177771
7099 013652 020501
7100 013654 001007
    
```

```

SUBTST <<LEGAL CONFIGURATION CHECK>>
:*****
:*SUBTEST      LEGAL CONFIGURATION CHECK
:*****
1$:
MOV      #16,R0
MOV      #CSRINFO,R1
CLR      (R1)+
SOB     R0,1$
        FOR BANK := #0 TO LASTBANK
        CALL      EXBANK
        MOV BANKINDEX,R0

        IF ACFLAG IS TRUE
        MOV      CONFIG+1(R0),R3
        BIC      #^C17,R3
        ASL      R3
        INC      CSRINFO(R3)
        IF MKFLAG IS TRUE
        :MAKE SURE THAT EACH BANK HAS NO MORE THAN 2 CSRS
        BEGIN    LEGALCSR
        IF INTFLAG IS TRUE
        MOV      CONFIG+1(R0),R3
        MOV      R3,R4
        BIC      #^C17,R3
        ASH      #-4,R4
        BIC      #^C17,R4
        IF R3 EQ R4
        BIC      #BIT11!BIT12,CONFIG+2(R0)
        BIC      #170000,CONFIG(R0)
        LEAVE    LEGALCSR
        END; OF IF R3
        ELSE
        LEAVE    LEGALCSR
        END; OF IF INTFLAG
        SET      CONFGERROR
        END      LEGALCSR
        END ;OF IF MKFLAG
        END ;OF IF ACFLAG
        END; OF FOR BANK
PUSH     R5,R0
CLR      R0
CLR      R1
CLR      R5
CLR      MBERR
        :SAVE CONTENTS OF R5, R0
        :CLEAR REGISTERS
        :CLEAR ERROR INDICATOR
        :IS CURRENT CSR <= 40
        BGE     5$
        :BRANCH IF SO
        :IS CURRENT CSR < 10
        BGE     3$
        :BRANCH IF SO
        :CALL ERROR ROUTINE
        BR      5$
        :TRY NEXT CSR
        :MOVE LOW WORD TO R5
        :DOES MEMORY EXIST HERE?
        BEQ     4$
        :BRANCH IF NOT
        BIC      #^C7400,R5
        :ISOLATE CSR NUMBER IN
        ASH      #-7,R5
        :REGISTER 5
        CMP     R5,R1
        :IS IT THE CURRENT CSR?
        BNE     4$
        :TRY NEXT WORD OF CONFIG IF NOT
    
```


CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 182-1
LEGAL CONFIGURATION CHECK

```

7101 013656 032760 010000 002654 BIT #BIT12,CONFIG+2(R0) ;IS IT INTERLEAVED?
7102 013664 001003 BNE 4$ ;BRANCH IF SO
7103 013666 012737 000001 014004 MOV #1,MBERR ;SET ERROR INDICATOR
7104 013674 062700 000004 4$: ADD #4,R0 ;SET ERROR INDICATOR
7105 013700 022700 000340 CMP #340,R0 ;UPDATE CONFIG COUNTER
7106 013704 001351 BNE 3$ ;CONFIG TABLE ALL DONE?
7107 013706 005737 014004 TST MBERR ;BRANCH IF NOT
7108 013712 001402 BEQ 5$ ;ERRORS FOUND?
7109 013714 004737 014126 CALL ILLCSR ;TRY NEXT CSR IF NOT
7110 013720 005000 5$: CLR R0 ;CALL ERROR ROUTINE
7111 013722 005037 014004 CLR MBERR ;REINITIALIZE CONFIG COUNTER
7112 013726 062701 000002 ADD #2,R1 ;CLEAR ERROR INDICATOR
7113 013732 022701 000040 CMP #40,R1 ;UPDATE CSR COUNTER
7114 013736 001321 BNE 2$ ;ALL CSR'S DONE?
7115 013740 POP R0,R5 ;BRANCH IF NOT
7116 013744 005037 014004 CLR MBERR ;RESTORE REGISTERS
7117 013750 012700 000734 MOV #734,R0 ;RESET ERROR INDICATOR
7118 013754 032760 000002 002652 6$: BIT #BIT1,CONFIG(R0) ;INDEX TO TOP OF CONFIG TABLE ;R-C
7119 013762 001003 BNE 7$ ;MEMORY PRESENT? ;R-C
7120 013764 162700 000004 SUB #4,R0 ;BRANCH IF SO ;R-C
7121 013770 000771 BR 6$ ;TRY NEXT LOWER ENTRY IN CONFIG TABLE ;R-C
7122 013772 006200 7$: ASR R0 ;R-C
7123 013774 006200 ASR R0 ;R-C
7124 013776 010037 002554 MOV R0,LASTBANK ;DIVIDE INDEX BY 4 TO GET BANK # ;R-C
7125 014002 000402 BR SKUJ ;STORE IN LASTBANK ;R-C
7126 014004 000000 MBERR: .WORD 0 ;SAVE SPACE FOR ERROR INDICATOR
7127 014006 000000 PHEBE: .WORD 0 ;SAVE SPACE FOR ODD BOUNDARY INTERLEAVED INDICATOR
7128 014010 005000 SKUJ: CLR R0 ;CLEAR CONFIG COUNTER
7129 014012 005037 014006 CLR PHEBE ;CLEAR COUNTER
7130 014016 032760 000002 002652 1$: BIT #BIT1,CONFIG(R0) ;IS THERE MEMORY PRESENT?
7131 014024 001431 BEQ 3$ ;BRANCH IF NOT
7132 014026 032760 010000 002654 BIT #BIT12,CONFIG+2(R0) ;IS IT INTERLEAVED?
7133 014034 001005 BNE 2$ ;BRANCH IF SO
7134 014036 005237 014006 INC PHEBE ;INCREMENT COUNTER
7135 014042 062700 000004 ADD #4,R0 ;INCREMENT CONFIG COUNTER
7136 014046 000763 BR 1$ ;TRY NEXT BANK
7137 014050 023727 014006 000010 2$: CMP PHEBE,#10 ;IS THE COUNTER EQUAL TO...
7138 014056 001417 BEQ 4$ ;ONE OF THE SPECIAL VALUES.
7139 014060 023727 014006 000030 CMP PHEBE,#30 ;IF IT IS...
7140 014066 001413 BEQ 4$ ;BRANCH TO 4$
7141 014070 023727 014006 000050 CMP PHEBE,#50
7142 014076 001407 BEQ 4$
7143 014100 023727 014006 000070 CMP PHEBE,#70
7144 014106 001403 BEQ 4$
7145 014110 005037 014006 3$: CLR PHEBE ;CLEAR INDICATOR
7146 014114 000403 BR 5$
7147 014116 012737 000001 014006 4$: MOV #1,PHEBE ;SET INDICATOR
7148 014124 000421 5$: BR SUBAAP ;BRANCH TO NEXT SUBTEST
7149 014126 010102 ILLCSR: MOV R1,R2 ;R2 HAS CSR NUMBER
7150 014130 006202 ASR R2 ;MAKE ACCEPTABLE FOR PRINTING
7151 014132 022702 000012 CMP #10,,R2
7152 014136 100002 BPL 1$
7153 014140 062702 000007 ADD #7,R2
7154 014144 062702 000060 1$: ADD #60,R2 ;PUT NUMBER INTO ERROR MESSAGE
7155 014150 110237 100060 MOVB R2,MSG122
7156 014154 TYPE MSG122
7157 014160 SET CONFERROR

```


7158 014166 000207

RETURN

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 184
LEGAL CONFIGURATION CHECK

7161 014170

SUBAAP: SUBTST <<PRINT CONFIGURATION DETAILS>>

: *SUBTEST PRINT CONFIGURATION DETAILS
: *****

7162 014170
7163 014204 013702 002554
7164 014210 006302
7165 014212 006302
7166 014214
7167 014216
7168 014226
7169 014236
7170 014246
7171 014252
7172 014254
7173 014260
7174 014260
7175 014262
7176 014266
7177 014266
7178 014266

CLEAR LSIZE,MSIZE,PSIZE
MOV LASTBANK,R2
ASL R2
ASL R2
FOR R1 := #0 TO R2 BY #4
IF CPUBIT SET.IN CONFIG(R1)
IF #BIT8 SET.IN CONFIG+2(R1)
IF #BIT9 SET.IN CONFIG+2(R1)
LET PSIZE := PSIZE + #1
ELSE
LET MSIZE := MSIZE + #1
END;IF BIT9
ELSE
LET LSIZE := LSIZE + #1
END;IF BIT8
END; OF IF CPUBIT
END ;OF FOR ALL BANKS IN TABLE

7179
7180 014276 005037 002450
7181 014302
7182 014304 006361 002372
7183 014310 006361 002372
7184 014314 006361 002372
7185 014320 006361 002372
7186 014324 066137 002372 002450
7187 014332
7188 014344
7189 014346
7190 014356
7191 014362
7192 014362
7193 014374 006337 002414
7194 014400 006337 002414
7195 014404 006337 002414
7196 014410 006337 002414
7197 014414
7198 014432
7199 014436 005737 002376
7200 014442 001405
7201 014444
7202 014452
7203 014456 005737 002400
7204 014462 001405
7205 014464
7206 014472
7207 014476 005737 002402
7208 014502 001405
7209 014504
7210 014512
7211 014516
7212 014524
7213 014530
7214 014540 004737 041364

CLR I
FOR R1 := #0 TO #10 BY #2
ASL BSIZE(R1)
ASL BSIZE(R1)
ASL BSIZE(R1)
ASL BSIZE(R1)
ADD BSIZE(R1),I ;BSIZE(R1) := BSIZE(R1) * 16.
;I <- I + BSIZE(R1)
END; FOR R1
FOR R1 := #0 TO #200 BY #4
IF CPUBIT SET.IN CONFIG(R1)
LET UNITOP := UNITOP + #1
END; OF IF CPUBIT
END; OF FOR R1
ASL UNITOP
ASL UNITOP
ASL UNITOP
ASL UNITOP ;UNITOP := UNITOP * 16.
IF I LT UNITOP THEN LET I := UNITOP
TYPE \$CRLF
2\$: TST LSIZE
BEQ 3\$
TYPDEC LSIZE
TYPE MSG112
3\$: TST MSIZE
BEQ 4\$
TYPDEC MSIZE
TYPE MSG113
4\$: TST PSIZE
BEQ 5\$
TYPDEC PSIZE
TYPE MSG114
5\$: TYPDEC I
TYPE MSG070
IF #SW6 OFF.IN @SWR
CALL PCONFIG

7215 014544

END; OF IF #SW6

7218 014544

SUBTST <<CHECK APT SIZING>>

: *SUBTEST CHECK APT SIZING

7219	014544		
7220	014560	005037	002432
7221	014564	012700	065734
7222	014570		
7223	014572		
7224	014600	111001	
7225	014602	042701	177400
7226	014606		
7227	014614	000261	
7228	014616		
7229	014620	000241	
7230	014622		
7231	014622	006101	
7232	014624	005201	
7233	014626	006301	
7234	014630	006301	
7235	014632	006301	
7236	014634	006301	
7237	014636	163701	002432
7238	014642	010137	002432
7239	014646		
7240	014656	060137	002420
7241	014662		
7242	014662		
7243	014672	060137	002422
7244	014676		
7245	014676	062700	000004
7246	014702		
7247	014702		
7248	014712		
7249	014742	104046	
7250	014744		
7251	014744		

IF APTFLAG IS TRUE AND APTSIZE IS TRUE

```

CLR TEMP
MOV #SMAMS1,R0
FOR R2 := #0 TO #4
  IFB 1(R0) NE #0
    MOVB (R0),R1
    BIC #177400,R1
    IF 2(R0) LT #0
      SEC
    ELSE
      CLC
    END ;OF IF 2(R0)
  ROL R1
  INC R1
  ASL R1
  ASL R1
  ASL R1
  ASL R1
  SUB TEMP,R1
  MOV R1,TEMP
  IFB 1(R0) EQ #3
    ADD R1,APTPAR
  END ;OF IFB 1(R0)
  IFB 1(R0) EQ #4
    ADD R1,APTECC
  END ;OF IFB 1(R0)
  ADD #4,R0
END ;OF IFB 1(R0)
END ;OF FOR R2

```

;TO COMPENSATE FOR 4 BANKS BEING (0-3)

```

IF APTPAR NE LSIZE OR APTECC NE MSIZE OR APTECC NE PSIZE
  ERROR +46
END ;OF IF APTPAR
END ;OF IF APTFLAG

```

7253 014744

LOOP: NEWTST <<DIAGNOSTIC MODE DISPATCH ROUTINE>>
:*****
:*TEST 5 DIAGNOSTIC MODE DISPATCH ROUTINE
:*****

014744 000004
7254 014746 005037 002220
7255 014752 017700 165646
7256 014756 042700 177761
7257 014762 004770 014772
7258 014766 000137 015012
7259 014772 015454
7260 014774 015562
7261 014776 015670
7262 015000 016020
7263 015002 016150
7264 015004 016300
7265 015006 016452
7266 015010 016602
7267
7268 015012 004737 015354
7269
7270 015016

TST5: SCOPE
CLR CONTFLAG
MOV @SWR,RO ;GET SWITCHES
BIC #^C16,RO ;MASK TO ONLY MODE BITS
CALL @DISPTBL(RO) ;DISPATCH TO ROUTINE THROUGH NEXT TABLE
JMP MEMDONE ;GO TO NEXT TEST
DISPTBL:BAFPAF ;MODE 0;BANKS FORWARD, PATTERNS FORWARD
BAFPAR ;MODE 1;BANKS FORWARD, PATTERNS REVERSE
BAWPAF ;MODE 2;BANKS WORST FIRST, PATTERNS FORWARD
BAWPAR ;MODE 3;BANKS WORST FIRST, PATTERNS REVERSE
PAFBAF ;MODE 4;PATTERNS FORWARD, BANKS FORWARD
PAFBAW ;MODE 5;PATTERNS FORWARD, BANKS WORST FIRST
PARBAF ;MODE 6;PATTERNS REVERSE, BANKS FORWARD
PARBAW ;MODE 7;PATTERNS REVERSE, BANKS WORST FIRST
MEMDONE:CALL DOBACK ;CHECK BACKGROUND PATTERN

NEWTST<<UNIQUE BANK TEST>>

:*****
:*TEST 6 UNIQUE BANK TEST
:*****

015016 000004
7271
7272
7273 015020
7274 015026
7275 015042 004737 024104
7276 015046
7277 015054 005037 002106
7278 015060
7279 015060 004737 015354
7283
7284 015064

TST6: SCOPE
:MAKE SURE THAT EACH BANK CAN HAVE UNIQUE DATA
:WRITE AND READ THE BANK NUMBER IN EACH BANK (EXCEPT WHERE THE PROGRAM IS)
IF SELONLY IS FALSE
SET HEADER,MUT
CALL MT0027
SET HEADER
CLR MUT
END ;OF IF SELONLY
CALL DOBACK ;RESTORE BACKGROUND PATTERN

FLUSH: SUBTST <<FLUSH OUT DBE'S>>

:*****
:*SUBTEST FLUSH OUT DBE'S
:*****
CALL MT0030

7285 015064 004737 024570


```

7288
7289
7290
7291
7292
7293
7294
7295 015070 005037 002440
7296 015074 012700 002654
7297 015100 042710 020000
7298 015104 062700 000004
7299 015110 020027 003620
7300 015114 003771
7301 015116 013737 002616 002014
7302 015124 005237 065712
7303 015130 042737 100000 065712
7304 015136
7305 015142
7306 015170
7307 015174 005037 002344
7308 015200
7309 015200
7310 015206 013700 000042
7311 015212 001456
7312 015214 022700 002000
7313 015220 001453
7314
7315 015222
7316 015224 004737 047676
7317 015230
7318 015232 000005
7319 015234 004710
7320 015236 000240
7321 015240 000240
7322 015242 000240
7323 015244
7324
7325
7326
7327
7328 015244 013706 002562
7329 015250 005737 002452
7330 015254 001003
7331 015256 052737 000060 172516
7332 015264 104420
7333 015266 013700 002564
7334 015272 012701 000001
7335 015276 004737 046422
7336 015302
7337 015310
7338 015320 012701 000050
7339 015324 077001
7340 015326 062737 000001 065714
7341 015334 005537 065716
7342 015340 077107
7343 015342 005237 065712
7344 015346 000764

```

```

.SBTTL END OF PASS ROUTINE
:*****
:*INCREMENT THE PASS NUMBER ($PASS)
:*INDICATE END-OF-PROGRAM AFTER EACH PASSES THRU THE PROGRAM
:*TYPE 'END PASS #XXXXX' (WHERE XXXXX IS A DECIMAL NUMBER)
:*IF THERES A MONITOR GO TO IT
:*IF THERE ISN'T JUMP TO LOOP
$EOP: CLR FSINFLAG
MOV #CONFIG+2,RO ;MOVE 2ND WORD OF CONFIG TO RO
1$: BIC #BIT13,(R0) ;CLEAR BACKGROUND VALID BIT
ADD #4,RO ;INCREMENT TO NEXT BANK
CMP RO,#3620 ;DONE?
BLE 1$ ;NO - BRANCH
MOV $ERTTL,LASTERROR
INC $PASS ;:INCREMENT THE PASS NUMBER
BIC #100000,$PASS ;:DON'T ALLOW A NEG. NUMBER
TYPE MSG077 ;:TYPE 'END PASS #'
IF #SW11 SET.IN @SWR OR QVFLAG IS TRUE OR $PASS EQ #1
TYPE MSG035 ;:QV
CLR QVFLAG
END :OF IF SW11
TYPDEC $PASS
MOV 42,RO ;:GET MONITOR ADDRESS
BEQ $DOAGAIN ;:BRANCH IF NO MONITOR
$ZAP42: CMP #STACK,RO ;:ARE WE UNDER RT11
BEQ $DOAGAIN ;:YES - BRANCH
;WE ARE UNDER (HEAVEN HELP US) XXDP!
PUSH RO
CALL SHUTUP
POP RO
RESET ;:CLEAR THE WORLD
$ENDAD: CALL (R0) ;:GO TO MONITOR
NOP ;:SAVE ROOM
NOP ;:FOR
NOP ;:ACT11
$DOAGN: ;UNDO SHUTUP STUFF
; RESTORE STACK
; ENERGIZE UNIBUS MAP & 22 BIT ADDRESSING
; ENERGIZE MEMORY MANAGEMENT
; PUT LOADERS BACK HOME
MOV KSTACK,SP
TST NO22BIT ;:IS THIS AN 11/44 OR 11/24?
BNE 1$
BIS #BITS!BIT4,MMR3
1$: ENERGIZE ;:TURN ON MEMORY MANAGEMENT
MOV LOADHOME,RO ;:DESTINATION BANK
MOV #1,R1 ;:SOURCE BANK
CALL BANKMOV
IF APTFLAG IS TRUE
IF $USWR EQ $PASS
APTHANG: MOV #50,R1
2$: SOB RO,2$
ADD #1,$DEVCT
ADC $UNIT
SOB R1,2$
INC $PASS
BR APTHANG

```


CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 189-1
END OF PASS ROUTINE

7345 015350
7346 015350
7347 015350 000137 014744

END :OF IF \$USWR
END :OF IF APTFLAG
\$DOAGAIN: JMP LOOP ;RETURN

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 191
END OF PASS ROUTINE

7350 015354

DOBACK: SUBTST <<WRITE BACKGROUND PATTERNS>>
:*****
:*SUBTEST WRITE BACKGROUND PATTERNS
:*****

7351 015354 005037 002110
7352 015360
7353 015364 004737 047032
7354 015370
7355 015404
7356 015420 004737 020276
7357 015424 005037 002106
7358 015430
7359 015436
7360 015436
7361 015452 000207

CLR PATTERN
FOR BANK := #0 TO LASTBANK
CALL EXBANK
IF ACFLAG IS TRUE AND RRFLAG IS FALSE
SET HEADER,MUT
CALL MKTEST ;CALL MJTEST WOULD ALSO WORK
CLR MUT
SET HEADER
END ;OF IF ACFLAG
END ;OF FOR BANK
RETURN

```

7364
7365
7366 015454

7367 015454 005037 002100
7368
7369 015460 004737 047032
7370 015464 005737 002114
7371 015470 001412
7372 015472 005737 002122
7373 015476 001007
7374 015500 005037 002110
7375
7376 015504 004737 016754
7377
7378 015510 004737 047476
7379 015514 001373
7380
7381 015516 005037 002220
7382 015522 004737 047522
7383 015526 002354
7384
7385 015530 005737 002124
7386 015534 001401
7387 015536 000207
7388 015540 004737 045204
7389 015544
7390
7391 015550 004737 015454
7392 015554 004737 046070
7393 015560 000207

```

```

.SBTTL MTEST MODES
BAFPAF: SUBTST <<BANKS FORWARD,PATTERNS FORWARD **RECURSIVE**>>
:*****
:*SUBTEST BANKS FORWARD,PATTERNS FORWARD **RECURSIVE**
:*****
CLR BANK ;SET BANK TO 0
:START OF BANK LOOP
1$: CALL EXBANK ;EXAMINE BANK
TST ACFLAG ;CAN WE ACCESS THIS BANK?
BEQ 4$ ;NO - GO TO BANK LOOP TERMINATION
TST RRFLAG ;RELOCATION REQUIRED?
BNE 4$ ;YES - GO TO BANK LOOP TERMINATION
CLR PATTERN ;SET PATTERN TO 0
:START OF PATTERN LOOP
2$: CALL MTEST ;GO TEST CORRECT MEMORY
:TERMINATION OF PATTERN LOOP
CALL INCPAT ;GO SEE IF THIS IS THE LAST PATTERN
BNE 2$ ;NO - LOOP ON THIS PATTERN
:TERMINATION OF BANK LOOP
4$: CLR CONTFLAG
CALL INCBNK ;NEXT HIGHER BANK
BGE 1$ ;IF NOT DONE - LOOP ON THIS BANK
:END OF LOOPS
TST RLFLAG ;HAVE WE BEEN RELOCATED?
BEQ 5$ ;NO - SKIP
RETURN ;YES - RETURN
5$: CALL RELOCATE ;MOVE & MAP PROGRAM
ON.ERROR THEN $RETURN
:**NOTE** RECURSIVE CALL
CALL BAFPAF ;CALL SELF
CALL UNRELOCATE ;UNMOVE & UNMAP PROGRAM
RETURN

```


7396 015562

BAFPAR: SUBTST <<BANKS FORWARD, PATTERNS REVERSE **RECURSIVE**>>
:*****
:*SUBTEST BANKS FORWARD, PATTERNS REVERSE **RECURSIVE**
:*****

7397 015562 005037 002100
7398
7399 015566 004737 047032
7400 015572 005737 002114
7401 015576 001412
7402 015600 005737 002122
7403 015604 001007
7404 015606 004737 047512
7405
7406 015612 004737 016754
7407
7408 015616 005337 002110
7409 015622 100373
7410
7411 015624 005037 002220
7412 015630 004737 047522
7413 015634 002354
7414
7415 015636 005737 002124
7416 015642 001401
7417 015644 000207
7418 015646 004737 045204
7419 015652
7420
7421 015656 004737 015562
7422 015662 004737 046070
7423 015666 000207

CLR BANK ;SET BANK TO 0
:START OF BANK LOOP
1\$: CALL EXBANK ;EXAMINE BANK
TST ACFLAG ;CAN WE ACCESS THIS BANK?
BEQ 4\$;NO - GO TO BANK LOOP TERMINATION
TST RRFLAG ;RELOCATION REQUIRED?
BNE 4\$;YES - GO TO BANK LOOP TERMINATION
CALL SETPAT ;SET HIGH PATTERN FOR CORRECT MEMORY
:START OF PATTERN LOOP
2\$: CALL MTEST ;GO TEST CORRECT MEMORY
:TERMINATION OF PATTERN LOOP
DEC PATTERN ;IS THIS THE LAST PATTERN?
BPL 2\$;NO - LOOP ON THIS PATTERN
:TERMINATION OF BANK LOOP
4\$: CLR CONTFLAG
CALL INCBNK ;NEXT HIGHER BANK
BGE 1\$;IF NOT DONE - LOOP ON THIS BANK
:END OF LOOPS
TST RLFLAG ;HAVE WE BEEN RELOCATED?
BEQ 5\$;NO - SKIP
RETURN ;YES - RETURN
5\$: CALL RELOCATE ;MOVE & MAP PROGRAM
ON.ERROR THEN \$RETURN
:**NOTE** RECURSIVE CALL
CALL BAFPAR ;CALL SELF
CALL UNRELOCATE ;UNMOVE & UNMAP PROGRAM
RETURN

7426 015670

```

BAWPAF: SUBTST <<BANKS WORST FIRST,PATTERNS FORWARD **RECURSIVE**>>
:*****
:*SUBTEST BANKS WORST FIRST,PATTERNS FORWARD **RECURSIVE**
:*****
CLR BANK ;SET BANK TO 0
:START OF BANK LOOP
1$: CALL EXBANK ;EXAMINE BANK
TST ACFLAG ;CAN WE ACCESS THIS BANK?
BEQ 4$ ;NO - GO TO BANK LOOP TERMINATION
TST BMFLAG ;IS THIS BAD MEMORY (WORST FIRST)?
BEQ 4$ ;NO - GO TO BANK LOOP TERMINATION
TST RRFLAG ;RELOCATION REQUIRED?
BNE 4$ ;YES - GO TO BANK LOOP TERMINATION
CLR PATTERN ;SET PATTERN TO 0
:START OF PATTERN LOOP
2$: CALL MTEST ;GO TEST CORRECT MEMORY
;TERMINATION OF PATTERN LOOP
CALL INCPAT ;GO SEE IF THIS IS THE LAST PATTERN
BNE 2$ ;NO - LOOP ON THIS PATTERN
;TERMINATION OF BANK LOOP
4$: CLR CONTFLAG
CALL INCBNK ;NEXT HIGHER BANK
BGE 1$ ;IF NOT DONE - LOOP ON THIS BANK
;END OF LOOPS
COM WORST ;IS THIS AN EVEN NUMBERED PASS?
BNE 5$ ;YES - SKIP
;***NOTE** RECURSIVE CALL
CALL BAWPAF ;CALL SELF
RETURN
5$: TST RLFLAG ;HAVE WE BEEN RELOCATED?
BEQ 6$ ;NO - SKIP
RETURN ;YES - RETURN
6$: CALL RELOCATE ;MOVE & MAP PROGRAM
ON.ERROR THEN $RETURN
;***NOTE** RECURSIVE CALL
CALL BAWPAF ;CALL SELF
CALL UNRELOCATE ;UNMOVE & UNMAP PROGRAM
RETURN

```

```

7427 015670 005037 002100
7428
7429 015674 004737 047032
7430 015700 005737 002114
7431 015704 001415
7432 015706 005737 002126
7433 015712 001412
7434 015714 005737 002122
7435 015720 001007
7436 015722 005037 002110
7437
7438 015726 004737 016754
7439
7440 015732 004737 047476
7441 015736 001373
7442
7443 015740 005037 002220
7444 015744 004737 047522
7445 015750 002351
7446
7447 015752 005137 002566
7448 015756 001003
7449
7450 015760 004737 015670
7451 015764 000207
7452 015766 005737 002124
7453 015772 001401
7454 015774 000207
7455 015776 004737 045204
7456 016002
7457
7458 016006 004737 015670
7459 016012 004737 046070
7460 016016 000207

```


CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 199
BANKS WORST FIRST,PATTERNS FORWARD **RECURSIVE**

7463 016020

```

BAWPAR: SUBST <<BANKS WORST FIRST,PATTERNS REVERSE **RECURSIVE**>>
:*****
:*SUBTEST      BANKS WORST FIRST,PATTERNS REVERSE **RECURSIVE**
:*****
CLR          BANK          ;SET BANK TO 0
;START OF BANK LOOP
1$: CALL      EXBANK       ;EXAMINE BANK
TST         ACFLAG       ;CAN WE ACCESS THIS BANK?
BEQ         4$           ;NO - GO TO BANK LOOP TERMINATION
TST         BMFLAG       ;IS THIS BAD MEMORY (WORST FIRST)
BEQ         4$           ;NO - GO TO BANK LOOP TERMINATION
TST         RRFLAG       ;RELOCATION REQUIRED?
BNE         4$           ;YES - GO TO BANK LOOP TERMINATION
CALL        SETPAT       ;SET HIGH PATTERN FOR CORRECT MEMORY
;START OF PATTERN LOOP
2$: CALL      MTEST        ;GO TEST CORRECT MEMORY
;TERMINATION OF PATTERN LOOP
DEC         PATTERN       ;IS THIS THE LAST PATTERN?
BPL         2$           ;NO - LOOP ON THIS PATTERN
;TERMINATION OF BANK LOOP
4$: CLR       CONTFLAG
CALL        INCBNK       ;NEXT HIGHER BANK
BGE         1$           ;IF NOT DONE - LOOP ON THIS BANK
;END OF LOOPS
COM         WORST        ;IS THIS AN EVEN NUMBERED PASS?
BNE         5$           ;YES - SKIP
;***NOTE** RECURSIVE CALL
CALL        BAWPAR       ;CALL SELF
RETURN
5$: TST       RLFLAG       ;HAVE WE BEEN RELOCATED?
BEQ         6$           ;NO - SKIP
RETURN      ;YES - RETURN
6$: CALL      RELOCATE     ;MOVE & MAP PROGRAM
ON.ERROR THEN $RETURN
;***NOTE** RECURSIVE CALL
CALL        BAWPAR       ;CALL SELF
CALL        UNRELOCATE   ;UNMOVE & UNMAP PROGRAM
RETURN

```

```

7464 016020 005037 002100
7465
7466 016024 004737 047032
7467 016030 005737 002114
7468 016034 001415
7469 016036 005737 002126
7470 016042 001412
7471 016044 005737 002122
7472 016050 001007
7473 016052 004737 047512
7474
7475 016056 004737 016754
7476
7477 016062 005337 002110
7478 016066 100373
7479
7480 016070 005037 002220
7481 016074 004737 047522
7482 016100 002351
7483
7484 016102 005137 002566
7485 016106 001003
7486
7487 016110 004737 016020
7488 016114 000207
7489 016116 005737 002124
7490 016122 001401
7491 016124 000207
7492 016126 004737 045204
7493 016132
7494
7495 016136 004737 016020
7496 016142 004737 046070
7497 016146 000207

```


CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 201
 BANKS WORST FIRST,PATTERNS REVERSE **RECURSIVE**

7500 016150

```

PAFBAF: SUBTST <<PATTERNS FORWARD,BANKS FORWARD **RECURSIVE**>>
:*****
:*SUBTEST PATTERNS FORWARD,BANKS FORWARD **RECURSIVE**
:*****
7501 016150 005037 002110 CLR PATTERN ;SET PATTERN TO 0
7502 ;START OF PATTERN LOOP
7503 016154 005037 002100 1$: CLR BANK ;SET BANK TO 0
7504 ;START OF BANK LOOP
7505 016160 004737 047032 2$: CALL EXBANK ;EXAMINE BANK
7506 016164 004737 047460 CALL BANKOK ;CORRECT MEMORY FOR THIS BANK?
7507 016170 001010 BNE 4$ ;NO - GO TO BANK LOOP TERMINATOR
7508 016172 005737 002114 TST ACFLAG ;CAN WE ACCESS THIS BANK?
7509 016176 001405 BEQ 4$ ;NO - GO TO BANK LOOP TERMINATION
7510 016200 005737 002122 TST RRFLAG ;RELOCATION REQUIRED?
7511 016204 001002 BNE 4$ ;YES - GO TO BANK LOOP TERMINATION
7512 016206 004737 016754 CALL MTEST ;GO TEST CORRECT MEMORY
7513 ;TERMINATION OF BANK LOOP
7514 016212 005037 002220 4$: CLR CONTFLAG
7515 016216 004737 047522 CALL INCBNK ;NEXT HIGHER BANK
7516 016222 002356 BGE 2$ ;IF NOT DONE - LOOP ON THIS BANK
7517 ;TERMINATION OF PATTERN LOOP
7518 016224 004737 047476 CALL INCRPT ;NEXT HIGHER PATTERN
7519 016230 001351 BNE 1$ ;OK - LOOP; ELSE CONTINUE
7520 ;END OF LOOPS
7521 016232 005137 002132 COM TMFLAG ;COMPLEMENT TYPE OF MEMORY
7522 ;IS THIS AN EVEN NUMBER PASS?
7523 016236 001403 BEQ 5$ ;YES - SKIP
7524 ;**NOTE** RECURSIVE CALL
7525 016240 004737 016150 CALL PAFBAF ;CALL SELF
7526 016244 000207 RETURN
7527 016246 005737 002124 5$: TST RLFLAG ;HAVE WE BEEN RELOCATED?
7528 016252 001401 BEQ 6$ ;NO - SKIP
7529 016254 000207 RETURN ;YES - RETURN
7530 016256 004737 045204 6$: CALL RELOCATE ;MOVE & MAP PROGRAM
7531 016262 ON.ERROR THEN $RETURN
7532 ;**NOTE** RECURSIVE CALL
7533 016266 004737 016150 CALL PAFBAF ;CALL SELF
7534 016272 004737 046070 CALL UNRELOCATE ;UNMOVE & UNMAP PROGRAM
7535 016276 000207 RETURN

```

7538 016300

```

PAFBAW: SUBTST <<PATTERNS FORWARD,BANKS WORST FIRST **RECURSIVE**>>
:*****
:*SUBTEST PATTERNS FORWARD,BANKS WORST FIRST **RECURSIVE**
:*****
7539 016300 005037 002110 CLR PATTERN ;SET PATTERN TO 0
7540 :START OF PATTERN LOOP
7541 016304 005037 002100 1$: CLR BANK ;SET BANK TO 0
7542 :START OF BANK LOOP
7543 016310 004737 047032 2$: CALL EXBANK ;EXAMINE BANK
7544 016314 004737 047460 CALL BANKOK ;CORRECT MEMORY FOR THIS BANK?
7545 016320 001013 BNE 4$ ;NO - GO TO BANK LOOP TERMINATOR
7546 016322 005737 002114 TST ACFLAG ;CAN WE ACCESS THIS BANK?
7547 016326 001410 BEQ 4$ ;NO - GO TO BANK LOOP TERMINATION
7548 016330 005737 002126 TST BMFLAG ;IS THIS BAD MEMORY (WORST FIRST)
7549 016334 001405 BEQ 4$ ;NO - GO TO BANK LOOP TERMINATION
7550 016336 005737 002122 TST RRFLAG ;RELOCATION REQUIRED?
7551 016342 001002 BNE 4$ ;YES - GO TO BANK LOOP TERMINATION
7552 016344 004737 016754 CALL MTEST ;GO TEST CORRECT MEMORY
7553 :TERMINATION OF BANK LOOP
7554 016350 005037 002220 4$: CLR CONTFLAG
7555 016354 004737 047522 CALL INCBNK ;NEXT HIGHER BANK
7556 016360 002353 BGE 2$ ;IF NOT DONE - LOOP ON THIS BANK
7557 :TERMINATION OF PATTERN LOOP
7558 016362 004737 047476 CALL INCRPT ;NEXT HIGHER PATTERN
7559 016366 001346 BNE 1$ ;OK - LOOP; ELSE CONTINUE
7560 :END OF LOOPS
7561 016370 005137 002132 COM TMFLAG ;COMPLEMENT TYPE OF MEMORY
7562 :IS THIS AN EVEN NUMBER PASS?
7563 016374 001403 BEQ 5$ ;YES - SKIP
7564 :**NOTE** RECURSIVE CALL
7565 016376 004737 016300 CALL PAFBAW ;CALL SELF
7566 016402 000207 RETURN
7567 016404 005137 002566 5$: COM WORST ;4TH PASS?
7568 016410 001003 BNE 6$ ;YES - SKIP
7569 :**NOTE** RECURSIVE CALL
7570 016412 004737 016300 CALL PAFBAW ;CALL SELF
7571 016416 000207 RETURN
7572 016420 005737 002124 6$: TST RLFLAG ;HAVE WE BEEN RELOCATED?
7573 016424 001401 BEQ 7$ ;NO - SKIP
7574 016426 000207 RETURN ;YES - RETURN
7575 016430 004737 045204 7$: CALL RELOCATE ;MOVE & MAP PROGRAM
7576 016434 ON.ERROR THEN $RETURN
7577 :**NOTE** RECURSIVE CALL
7578 016440 004737 016300 CALL PAFBAW ;CALL SELF
7579 016444 004737 046070 CALL UNRELOCATE ;UNMOVE & UNMAP PROGRAM
7580 016450 000207 RETURN

```


7583 016452

```

PARBAF: SUBTST <<PATTERNS REVERSE,BANKS FORWARD **RECURSIVE**>>
:*****
:*SUBTEST PATTERNS REVERSE,BANKS FORWARD **RECURSIVE**
:*****
7584 016452 004737 047512 CALL HIPAT :SET HIGHEST PATTERNS
7585 :START OF PATTERN LOOP
7586 016456 005037 002100 1$: CLR BANK :SET BANK TO 0
7587 :START OF BANK LOOP
7588 016462 004737 047032 2$: CALL EXBANK :EXAMINE BANK
7589 016466 004737 047460 CALL BANKOK :CORRECT MEMORY FOR THIS BANK?
7590 016472 001010 BNE 4$ :NO - GO TO BANK LOOP TERMINATOR
7591 016474 005737 002114 TST ACFLAG :CAN WE ACCESS THIS BANK?
7592 016500 001405 BEQ 4$ :NO - GO TO BANK LOOP TERMINATION
7593 016502 005737 002122 TST RRFLAG :RELOCATION REQUIRED?
7594 016506 001002 BNE 4$ :YES - GO TO BANK LOOP TERMINATION
7595 016510 004737 016754 CALL MTEST :GO TEST CORRECT MEMORY
7596 :TERMINATION OF BANK LOOP
7597 016514 005037 002220 4$: CLR CONTFLAG
7598 016520 004737 047522 CALL INCBNK :NEXT HIGHER BANK
7599 016524 002356 BGE 2$ :IF NOT DONE - LOOP ON THIS BANK
7600 :TERMINATION OF PATTERN LOOP
7601 016526 005337 002110 DEC PATTERN :NEXT LOWER PATTERN
7602 016532 100351 BPL 1$ :OK - LOOP; ELSE CONTINUE
7603 :END OF LOOPS
7604 016534 005137 002132 COM TMFLAG :COMPLEMENT TYPE OF MEMORY
7605 :IS THIS AN EVEN NUMBER PASS?
7606 016540 001403 BEQ 5$ :YES - SKIP
7607 :**NOTE** RECURSIVE CALL
7608 016542 004737 016452 CALL PARBAF :CALL SELF
7609 016546 000207 RETURN
7610 016550 005737 002124 5$: TST RLFLAG :HAVE WE BEEN RELOCATED?
7611 016554 001401 BEQ 6$ :NO - SKIP
7612 016556 000207 RETURN :YES - RETURN
7613 016560 004737 045204 6$: CALL RELOCATE :MOVE & MAP PROGRAM
7614 016564 ON.ERROR THEN $RETURN
7615 :**NOTE** RECURSIVE CALL
7616 016570 004737 016452 CALL PARBAF :CALL SELF
7617 016574 004737 046070 CALL UNRELOCATE :UNMOVE & UNMAP PROGRAM
7618 016600 000207 RETURN

```


7621 016602

```

PARBAW: SUBTST <<PATTERNS REVERSE,BANKS WORST FIRST **RECURSIVE**>>
:*****
:*SUBTEST PATTERNS REVERSE,BANKS WORST FIRST **RECURSIVE**
:*****
CALL HIPAT ;SET HIGHEST PATTERN
:START OF PATTERN LOOP
1$: CLR BANK ;SET BANK TO 0
:START OF BANK LOOP
2$: CALL EXBANK ;EXAMINE BANK
CALL BANKOK ;CORRECT MEMORY FOR THIS BANK?
BNE 4$ ;NO - GO TO BANK LOOP TERMINATOR
TST ACFLAG ;CAN WE ACCESS THIS BANK?
BEQ 4$ ;NO - GO TO BANK LOOP TERMINATION
TST BMFLAG ;IS THIS BAD MEMORY (WORST FIRST)
BEQ 4$ ;NO - GO TO BANK LOOP TERMINATION
TST RRFLAG ;RELOCATION REQUIRED?
BNE 4$ ;YES - GO TO BANK LOOP TERMINATION
CALL MTEST ;GO TEST CORRECT MEMORY
:TERMINATION OF BANK LOOP
4$: CLR CONTFLAG
CALL INCBNK ;NEXT HIGHER BANK
BGE 2$ ;IF NOT DONE - LOOP ON THIS BANK
:TERMINATION OF PATTERN LOOP
DEC PATTERN ;NEXT LOWER PATTERN
BPL 1$ ;OK - LOOP; ELSE CONTINUE
:END OF LOOPS
COM TMFLAG ;COMPLEMENT TYPE OF MEMORY
BEQ 5$ ;IS THIS AN EVEN NUMBER PASS?
;YES - SKIP
**NOTE** RECURSIVE CALL
CALL PARBAW ;CALL SELF
RETURN
5$: COM WORST ;4TH PASS?
BNE 6$ ;YES - SKIP
**NOTE** RECURSIVE CALL
CALL PARBAW ;CALL SELF
RETURN
6$: TST RLFLAG ;HAVE WE BEEN RELOCATED?
BEQ 7$ ;NO - SKIP
RETURN ;YES - RETURN
7$: CALL RELOCATE ;MOVE & MAP PROGRAM
ON.ERROR THEN $RETURN
**NOTE** RECURSIVE CALL
CALL PARBAW ;CALL SELF
CALL UNRELOCATE ;UNMOVE & UNMAP PROGRAM
RETURN
    
```

7622 016602 004737 047512
 7623
 7624 016606 005037 002100
 7625
 7626 016612 004737 047032
 7627 016616 004737 047460
 7628 016622 001013
 7629 016624 005737 002114
 7630 016630 001410
 7631 016632 005737 002126
 7632 016636 001405
 7633 016640 005737 002122
 7634 016644 001002
 7635 016646 004737 016754
 7636
 7637 016652 005037 002220
 7638 016656 004737 047522
 7639 016662 002353
 7640
 7641 016664 005337 002110
 7642 016670 100346
 7643
 7644 016672 005137 002132
 7645
 7646 016676 001403
 7647
 7648 016700 004737 016602
 7649 016704 000207
 7650 016706 005137 002566
 7651 016712 001003
 7652
 7653 016714 004737 016602
 7654 016720 000207
 7655 016722 005737 002124
 7656 016726 001401
 7657 016730 000207
 7658 016732 004737 045204
 7659 016736
 7660
 7661 016742 004737 016602
 7662 016746 004737 046070
 7663 016752 000207

7666 016754

```

MTEST: SUBTST <<SUBR SETUP MEMORY TEST>>
;*****
;*SUBTEST      SUBR      SETUP MEMORY TEST
;*****
          SET      HEADER      ;INITIALIZE HEADER MESSAGE TIMEOUT
          SET      MUT          ;INDICATE THERE IS A MEMORY UNDER TEST
          CLR      PASFLG
          TST      MKFLAG      ;ECC?
          BEQ      MT1          ;NO - SKIP
          BEGIN   HOLDLOOP
          IF CONTFLAG IS TRUE THEN LEAVE HOLDLOOP
          IF SKIPMK IS FALSE
            CALL   MKCONTROL
          END; OF IF SKIPMK
          END     HOLDLOOP
          CALL   MKTEST      ;YES - DO ECC TESTS
          BR     MT2
MT1:     CALL   MJTEST      ;DO PARITY TESTS
MT2:     CLR    MUT          ;NOW - NO MEMORY UNDER TEST
          SET    HEADER      ;ALLOW HEADERS NORMAL
          RETURN

```

```

7667 016754
7668 016762
7669 016770 005037 002262
7670 016774 005737 002116
7671 017000 001413
7672 017002
7673 017002
7674 017010
7675 017016 004737 017050
7676 017022
7677 017022
7678 017022 004737 020276
7679 017026 000402
7680 017030 004737 020516
7681 017034 005037 002106
7682 017040
7683 017046 000207

```


7686 017050

MKCONTROL:SUBTST <<SUBR TEST ECC CSR LOGIC DISPATCH>>
:*****
:*SUBTEST SUBR TEST ECC CSR LOGIC DISPATCH
:*****

7687
7688
7689
7690 017050
7691 017060
7692 017070
7693 017104 012737 060000 002230
7694 017112 012737 157776 002232
7695 017120 005037 002234
7696 017124 005037 002236
7697 017130 005037 002330
7698 017134 013700 002102
7699 017140 016001 002652
7700 017144 000301
7701 017146 042701 177760
7702 017152 006301
7703 017154 010137 002524
7704 017160 005737 002134
7705 017164 001421
7706 017166 005237 002236
7707 017172 012737 120000 002232
7708 017200 005237 002330
7709 017204 005237 002234
7710 017210 016001 002652
7711 017214 072127 177775
7712 017220 042701 160777
7713 017224 050137 002524
7714 017230 005003
7715 017232 116337 002524 002150
7716 017240 042737 177741 002150
7717 017246
7718 017252
7719 017260
7720 017274 104511
7721 017276
7722 017276 005037 002332
7723 017302
7724 017316 013737 002224 002226
7725 017324 062737 004000 002226
7726 017332
7727 017340 013737 002410 002412
7728 017346 005737 002236
7729 017352 001404
7730 017354 062737 040000 002412
7731 017362 000403
7732 017364 062737 000002 002412
7733 017372 004737 017672
7734 017376
7735 017400 104424
7736 017402 005037 002074
7737 017406
7738 017412
7739 017420 005037 002262

:THE NEXT TWO MODULES SOLVE THE PROBLEM OF
:HOW TO RUN THE CSR TESTS ON EACH ECC MEMORY
:
:IF SELONLY IS TRUE THEN \$RETURN
:IF INHECC IS TRUE THEN \$RETURN
PUSH BANK,R0,R1,R2,R3
MOV #FIRST,CSRFBANK ;SET FIRST TEST ADDRESS TO FIRST ADDR.
MOV #LAST,CSRLBANK
CLR CSRINT
CLR SPLTCSR
CLR CSRLOOP ; AND ZERO THE LOOP COUNTER
MOV BANKINDEX,R0 ;GET THE BANK INDEX
MOV CONFIG(R0),R1 ;GET CSR NUMBER
SWAB R1
BIC #C17,R1
ASL R1
MOV R1,CSRHOLD ;STORE IN THE LOW BYTE
TST INTFLAG ;IS THIS BANK INTERLEAVED?
BEQ 1\$;BRANCH IF NOT INTERLEAVED
INC SPLTCSR
MOV #120000,CSRLBANK ;WE MUST LOOP TWICE FOR AN INTERLEAVED BANK
INC CSRLOOP
INC CSRINT
MOV CONFIG(R0),R1 ;GET THE INTERLEAVE CSR NUMBER
ASH #-3,R1
BIC #C17000,R1
BIS R1,CSRHOLD ;STORE IT IN CSRHOLD'S UPPER BYTE
1\$: CLR R3
MKLOOP: MOVB CSRHOLD(R3),CSRNO
BIC #C36,CSRNO ;CLEAR ANY UNNECESSARY BITS
FOR MKCNT := #0 TO CSRINT
FOR CSRFIRST := CSRFBANK TO CSRLBANK BY #4000
MAP BANK ;MAP TEST SPACE TO BANK
INVALIDATE ;INVALIDATE BACKGROUND PATTERN
BEGIN CSRSTUFF
CLR SUCCESS
IF ACFLAG IS TRUE AND RRFLAG IS FALSE
MOV CSRFIRST,CSRLAST
ADD #4000,CSRLAST
FOR TESTADD := CSRFIRST TO CSRLAST BY #4
MOV TESTADD,TESTADD+2
TST SPLTCSR
BEQ 1\$
ADD #40000,TESTADD+2
BR 2\$
1\$: ADD #2,TESTADD+2
2\$: CALL SBTEST
ON.NOERROR
CACHOFF ;TURN CACHE OFF
CLR NOPAR ;INDICATE PARITY ACTION
FOR I := #0 TO #27
SET HEADER
CLR PASFLG

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 211-1
 SUBR TEST ECC CSR LOGIC DISPATCH

7740	017424				LET R0 := I	
7741	017430				PUSH R3	;SAVE LOOP COUNTER
7742	017432	010637	002144		MOV SP,CTLKVEC	;SAVE VECTOR IN CSR OF *K
7743	017436	162737	000002	002144	SUB #2,CTLKVEC	
7744	017444	004737	020176		CALL CSRCASE	
7745	017450				POP R3	;RESTORE LOOP COUNTER
7746	017452				END ;OF FOR I	
7747	017466	104423			CACHON	;TURN CACHE ON
7748	017470				SET SUCCESS	
7749	017476				LEAVE CSRSTUFF	
7750	017500				END ;OF ON.NOERROR	
7751	017500				END ;OF FOR TESTADD	
7752	017516				END ;OF IF	
7753	017516				END CSRSTUFF	
7754	017516				IF SUCCESS IS FALSE	
7755	017524				TYPE MSGA34	
7756	017530				TYPOCS BANK,<TYPES BANK NUMBER>,3	
7757	017540				TYPE MSGB34	
7758	017544	004737	057532		CALL PERBNK	
7759	017550				END ;OF IF SUCCESS	
7760	017550				END ;OF FOR CSRFIRST	
7761	017566	005237	002236		INC SPLTCSR	
7762	017572				END ;OF FOR MKCNT	
7763	017606	062737	000002	002230	ADD #2,CSRFBANK	
7764	017614	012737	000001	002236	MOV #1,SPLTCSR	
7765	017622	005203			INC R3	
7766	017624	020337	002330		CMP R3,CSRLOOP	
7767	017630	003002			BGT 1\$	
7768	017632	000137	017232		JMP MKLOOP	
7769	017636	104472			1\$: ECCINIT	;TRAP ON DOUBLE BIT ERRORS (NORMAL)
7770	017640				SET CONTFLAG	
7771	017646	005037	002236		CLR SPLTCSR	
7772	017652				POP R3,R2,R1,R0,BANK	
7773	017666	000207			RETURN	
7774	017670	000000			MKCNT: .WORD 0	;COUNTER FOR MKLOOP

7777 017672

```

SBETEST:SUBTST <<CHECK FOR SBE FREE LOCATIONS>>
:*****
:*SUBTEST CHECK FOR SBE FREE LOCATIONS
:*****
:IN ORDER TO DETERMINE IF A LOCATION IS SBE FREE I DO THIS
:
:WRITE ZEROS WITH ECC DISABLE
:READ ZEROS BACK
:IF NOT ZEROS THEN RETURN ERROR
:
:WRITE ZEROS WITH ECC ENABLED BUT TRAPS DISABLED
:READ ZEROS BACK
:IF NOT ZEROS THEN RETURN ERROR
:
:TEST THE LOCATION FROM THE PAR'S (WITH NO PROGRAM FETCHES)
:IF THERE WERE ANY SBE'S OR DBE'S THEN RETURN ERROR
:
:COMPLIMENT ZEROS TO ONES WITH ECC DISABLE
:READ ONES BACK
:IF NOT ONES THEN RETURN ERROR
:
:WRITE 100,100000,00000 (CHECKBITS COMPLIMENT OF BEFORE)
: WITH ECC ENABLED BUT TRAPS DISABLED
:TEST THE LOCATION FROM THE PAR'S (WITH NO PROGRAM FETCHES)
:IF THERE WERE ANY SBE'S OR DBE'S THEN RETURN ERROR
:
:IF NONE OF THE ABOVE FORCES A RETURN ERROR THEN RETURN NO.ERROR
.ENABL LSB
PUSH R0,R1,R4 ;PUSH R0,R1,R4 ONTO STACK
MOV TESTADD,R1
MOV TESTADD+2,R4
TESTAREA ;ENTER TEST MODE
CACHOFF ;TURN CACHE OFF
ECC1DIS ;DISABLE ECC ON 1 SELECTED CSR
CLEAR (R1),(R4)
TST (R1)
BNE SBENT
TST (R4)
BNE SBENT

CLR1CSR ;CLEAR 1 SELECTED CSR
CLEAR (R1),(R4)
TST (R1)
BNE SBENT
TST (R4)
BNE SBENT

TSTREAD ;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
IF #BIT15!BIT4 SET.IN CSR
SET SKPERR ;DISABLE ERRGEN'S ERROR PRINTOUT
ERRGEN
MOV ERRADD,R0
ASH #-4,R0
BIC #^C177,R0
IF BANK EQ R0 THEN GOTO SBENT
END: OF IF #BIT15
ECC1DIS ;DISABLE ECC ON 1 SELECTED CSR
    
```

7778
 7779
 7780
 7781
 7782
 7783
 7784
 7785
 7786
 7787
 7788
 7789
 7790
 7791
 7792
 7793
 7794
 7795
 7796
 7797
 7798
 7799
 7800
 7801
 7802 017672
 7803 017700 013701 002410
 7804 017704 013704 002412
 7805 017710
 7806 017716 104424
 7807 017720 104471
 7808 017722
 7809 017726 005711
 7810 017730 001107
 7811 017732 005714
 7812 017734 001105
 7813
 7814 017736 104503
 7815 017740
 7816 017744 005711
 7817 017746 001100
 7818 017750 005714
 7819 017752 001076
 7820
 7821 017754 104510
 7822 017756
 7823 017766
 7824 017774 104512
 7825 017776 013700 002456
 7826 020002 072027 177774
 7827 020006 042700 177600
 7828 020012
 7829 020020
 7830 020020 104471

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 213-1
CHECK FOR SBE FREE LOCATIONS

7831	020022	005111		COM	(R1)	
7832	020024	005114		COM	(R4)	
7833	020026	023711	002602	CMP	ONES, (R1)	
7834	020032	001046		BNE	SBENT	
7835	020034	023714	002602	CMP	ONES, (R4)	
7836	020040	001043		BNE	SBENT	
7837						
7838	020042	104503		CLR1CSR		:CLEAR 1 SELECTED CSR
7839	020044	005011		CLR	(R1)	
7840	020046	012714	100000	MOV	#BIT15, (R4)	
7841	020052	005711		TST	(R1)	
7842	020054	001035		BNE	SBENT	
7843	020056	022714	100000	CMP	#BIT15, (R4)	
7844	020062	001032		BNE	SBENT	
7845						
7846	020064	104510		TSTREAD		:TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
7847	020066			IF #BIT15!BIT4 SET.IN CSR		
7848	020076			SET SKPERR		:DISABLE ERRGEN'S ERROR PRINTOUT
7849	020104	104512		ERRGEN		
7850	020106	013700	002456	MOV	ERRADD, R0	
7851	020112	072027	177774	ASH	#-4, R0	
7852	020116	042700	177600	BIC	#^C177, R0	
7853	020122			IF BANK EQ R0 THEN GOTO SBENT		
7854	020130			END: OF IF #BIT15		
7855						
7856	020130	104417		KERNEL		:ENTER KERNEL MODE
7857	020132	104473		ECC1INIT		:INITIALIZE 1 SELECTED CSR
7858	020134	104423		CACHON		:TURN CACHE ON
7859	020136			POP	R4, R1, R0	:POP R0, R1 & R4 FROM STACK
7860	020144			\$RETURN	NOERROR	
7861						
7862	020150	104503		SBENT: CLR1CSR		:CLEAR 1 SELECTED CSR
7863	020152			CLEAR	(R1), (R4)	
7864	020156	104417		KERNEL		:ENTER KERNEL MODE
7865	020160	104473		ECC1INIT		:INITIALIZE 1 SELECTED CSR
7866	020162	104423		CACHON		:TURN CACHE ON
7867	020164			POP	R4, R1, R0	:POP R0, R1 & R4 FROM STACK
7868	020172			\$RETURN	ERROR	
7869				.DSABL	LSB	

7973 020516

MJTEST: SUBTST <<SUBR PARITY TEST DISPATCH>>

:SUBTEST SUBR PARITY TEST DISPATCH

7974 020516 012737 000002 002074
7975 020524 012737 000002 002324
7976 020532 012737 060000 002410
7977 020540 012737 060002 002412
7978 020546 013700 002110
7979 020552 006300
7980 020554
7981 020574 104511
7982 020576
7983 020576 010637 002144
7984 020602 162737 000002 002144
7985 020610 004770 020622
7986 020614 005037 002074
7987 020620 000207
7988
7989
7990
7991
7992
7993 020622
7994 020622 026002
7995 020624 021752
7996 020626 022576
7997 020630 022006
7998 020632 020762
7999 020634 021102
8000 020636 021242
8001 020640 021474
8002 020642 021616
8003 020644 022710
8004 020646 026154
8005 020650 023162
8006 020652 023214
8007 020654 023602
8008 020656 023260
8009 020660 025072
8010 020662 025262
8011 020664 025614
8012 020666 026002
8013
8014 020670 026764
8015 020672 026764
8016 020674 026764
8017 020676 026764
8018 020700 026764

MOV #2,NOPAR ;INDICATE PARITY ACTION
MOV #2,PCBUMP ;TRAPS ADD 2 TO PC
MOV #FIRST,TESTADD
MOV #FIRST+2,TESTADD+2
MOV PATTERN,R0 ;GET PATTERN NUMBER
ASL R0 ;MAKE IT A WORD ADDRESS
IF MJPAT(R0) NE #MT0034 AND MJPAT(R0) NE #MT0999
INVALIDATE ;INVALIDATE BACKGROUND PATTERN ON 'BANK'
END :OF IF MJPAT(R0)
MOV SP,CTLKVEC ;SAVE VECTOR IN CASE OF ^K
SUB #2,CTLKVEC
CALL @MJPAT(R0) ;INDEX OFF TABLE
CLR NOPAR ;INDICATE PARITY ACTION
RETURN

:WARNING IF YOU CHANGE THIS TABLE ALSO
:CHANGE '\$DDW0' - '\$DDW5' (THE PATTERN BIT MAP)

MJPAT: :PAT TIME DISCRPTION
:NOTE MT0034 MUST BE FIRST & LAST
MT0034 :<1 SEC :SOFT ERROR - BACKGROUND PATTERN TEST
MT0006 :<1 SEC :INITIAL DATA TEST
MT0017 :<1 SEC :HOLDING 1'S & 0'S TEST
MT0007 :<1 SEC :ADDRESS BIT TEST
MT0001 :<1 SEC :ADDRESS TEST
MT0002 :<1 SEC :COMPLEMENT ADDRESS TEST
MT0003 : 1 SEC :3 XOR 9 WORST CASE NOISE TEST
MT0004 : 1 SEC :ROTATING ZEROS TEST
MT0005 : 1 SEC :ROTATING ONES TEST
MT0021 : 1 SEC :MARCHING 0'S & 1'S TEST
MT0035 :<1 SEC :WORSE CASE NOISE PARITY TEST
MT0022 :10 SEC :REFRESH TEST
MT0023 :10 SEC :SHIFTING DIAGONAL TEST
MT0026 :<1 SEC :RANDOM DATA TEST
MT0024 :20 SEC :FAST GALLOPING PATTERN TEST
MT0031 : 3 SEC :SOB-A-LONG TEST
MT0032 :<1 SEC :WRITE RECOVERY TEST
MT0033 :35 SEC :BRANCH GOBBLE TEST
MT0034 :<1 SEC :SOFT ERROR - BACKGROUND PATTERN TEST
:NOTE MT0034 MUST BE FIRST & LAST
MT0999 : 0 SEC :NULL TEST
MT0999 : 0 SEC :NULL TEST
MT0999 : 0 SEC :NULL TEST
MT0999 : 0 SEC :NULL TEST
MT0999 : 0 SEC :NULL TEST

8020
8021
8022
8023 020702

.SBTTL PATTERNS

```

.SBTTL MEMORY TEST SETUP ROUTINES
MT0000: SUBTST <<MT0000      SETUP DATA PATTERN TEST>>
:*****
:*SUBTEST      MT0000  SETUP DATA PATTERN TEST
:*****
      CLR      REALPAT          ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
      MOV      #FIRST,R0
      MOV      #SIZE,R1
      CALL     REGCOPY
      CMP      #1,PROTYP        ;ARE WE ON AN 11/44?
      BEQ      1$              ;BRANCH IF YES
      MOV      #000,SUPDOADD    ;ELSE DO PATTERN IN MAIN MEMORY
      CALL     SUPDO1
      RETURN
1$:   BMOV     ....,000
      CALL     SUPDO1          ;DO IT IN SUPERVISOR MODE
      RETURN

```

8024 020702 005037 002276
8025 020706 012700 060000
8026 020712 012701 040000
8027 020716 004737 041076
8028 020722 022737 000001 003754
8029 020730 001406
8030 020732 012737 027404 002260
8031 020740 004737 027212
8032 020744 000207
8033 020746
8034 020754 004737 027034
8035 020760 000207
8036 020762

```

MT0001: SUBTST <<MT0001      SETUP ADDRESS TEST>>
:*****
:*SUBTEST      MT0001  SETUP ADDRESS TEST
:*****
      MOV      #1,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
      MOV      #FIRST,R0
      MOV      #SIZE,R1
      TST      NOSUPER
      BNE      2$
      CMP      SIPAR5,SIPAR6
      BNE      4$
      BR       3$
2$:   CMP      UIPAR5,UIPAR6
      BNE      4$
3$:   MOV      #30000,R1
4$:   CLR      R2
      CALL     REGCOPY
      CMP      #1,PROTYP        ;IS THIS AN 11/44?
      BEQ      1$              ;BRANCH IF IT IS
      MOV      #MTP001,SUPDOADD ;SET UP CALLING ADDRESS
      CALL     SUPDO3
      RETURN
1$:   BMOV     MTP001
      CALL     SUPDO1          ;DO IT IN SUPERVISOR MODE
      RETURN

```

8037 020762 012737 000001 002276
8038 020770 012700 060000
8039 020774 012701 040000
8040 021000 005737 002454
8041 021004 001005
8042 021006 023737 172252 172254
8043 021014 001007
8044 021016 000404
8045 021020 023737 177652 177654 2\$:
8046 021026 001002
8047 021030 012701 030000 3\$:
8048 021034 005002 4\$:
8049 021036 004737 041076
8050 021042 022737 000001 003754
8051 021050 001406
8052 021052 012737 027430 002260
8053 021060 004737 027212
8054 021064 000207
8055 021066
8056 021074 004737 027034
8057 021100 000207
8058 021102

```

MT0002: SUBTST <<MT0002      SETUP COMPLEMENT ADDRESS TEST>>
:*****
:*SUBTEST      MT0002  SETUP COMPLEMENT ADDRESS TEST
:*****
      MOV      #2,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
      MOV      #LAST+2,R0
      MOV      #SIZE,R1
      MOV      #FIRST,R4
      MOV      #100001,R5
      TST      NOSUPER
      BNE      2$
      CMP      SIPAR5,SIPAR6
      BNE      4$

```

8059 021102 012737 000002 002276
8060 021110 012700 160000
8061 021114 012701 040000
8062 021120 012704 060000
8063 021124 012705 100001
8064 021130 005737 002454
8065 021134 001005
8066 021136 023737 172252 172254
8067 021144 001013

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 220-1
 MT0002 SETUP COMPLEMENT ADDRESS TEST

8068	021146	000404				BR	3\$	
8069	021150	023737	177652	177654	2\$:	CMP	UIPAR5,UIPAR6	
8070	021156	001006				BNE	4\$	
8071	021160	012701	030000		3\$:	MOV	#30000,R1	
8072	021164	012700	140000			MOV	#140000,R0	
8073	021170	012705	120001			MOV	#120001,R5	
8074	021174	012702	000001		4\$:	MOV	#1,R2	
8075	021200	010103				MOV	R1,R3	
8076	021202	022737	000001	003754		CMP	#1,PROTYP	;IS THIS AN 11/44?
8077	021210	001406				BEQ	1\$;BRANCH IF TRUE
8078	021212	012737	027462	002260		MOV	#MTP002,SUPDOADD	;SET UP CALLING ADDRESS
8079	021220	004737	027212			CALL	SUPDO3	
8080	021224	000207				RETURN		
8081	021226				1\$:	BMOV	MTP002	
8082	021234	004737	027034			CALL	SUPDO1	
8083	021240	000207				RETURN		

8086 021242

MT0003: SUBTST <<MT0003 SETUP 3 XOR 9 WORST CASE NOISE TEST>>

:SUBTEST MT0003 SETUP 3 XOR 9 WORST CASE NOISE TEST

8087 021242
8088 021252 012737 000003 002276
8089 021260 005037 002324
8090 021264 004737 041106
8091 021270 012701 060000
8092 021274 012703 020000
8093 021300 072327 177770
8094 021304 012702 000004
8095 021310 012705 000100
8096 021314 022737 000001 003754
8097 021322 001415
8098 021324 104415
8099 021326 012737 027514 002260
8100 021334 004737 027212
8101 021340 104416
8102 021342 012737 027554 002260
8103 021350 004737 027226
8104 021354 000442
8105 021356
8106 021364 104415
8107 021366 004737 027034
8108 021372
8109 021400
8110 021412
8111 021424 012737 172360 177642
8112 021432 012737 172260 172374
8113 021440 012737 177644 172276
8114 021446 012737 001032 172272
8115 021454 104416
8116 021456 004737 027050
8117 021462 022737 000003 002604
8118
8119 021470 001275
8120 021472 000207
8121
8122 021474

IF EUFLAG IS TRUE THEN \$RETURN
MOV #3,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
CLR PCBUMP ;TRAPS DO NOT ADD TO PC
1\$: CALL FLIPWARN ;SETUP WARNING CONSTANTS & R2
2\$: MOV #FIRST,R1 ;R1 <-- STARTING ADDRESS
MOV #20000,R3
ASH #-8.,R3 ;R3 <-- R3 / 256.
MOV #4,R2 ;SMALL LOOP SIZE
MOV #64.,R5 ;MEDIUM LOOP SIZE
CMP #1,PROTYP ;IS THIS AN 11/44?
BEQ 3\$;BRANCH IF IT IS
SAVREG
MOV #MTPA03,SUPDOADD
CALL SUPDO3 ;DO IT IN MAIN MEMORY
RESREG
MOV #MTPB03,SUPDOADD
CALL SUPDO4
BR 4\$
3\$: BMOV MTPA03
SAVREG
CALL SUPDO1
BMOV MTPB03
BMOV MTPC03,KDPAR0,8.
BMOV MTPD03,SDPAR0,8.
MOV #KDPAR0,UIPAR1 ;SET UP PAR LINKS
MOV #SDPAR0,KDPAR6
MOV #UIPAR2,SDPAR7
MOV #1032,SDPAR5 ;CHANGE INST TO BR .+56 (BR TO KDPAR1)
RESREG
CALL SUPDO2
4\$: CMP #3,FLIPLC ;DONE WITH 4 PATTERNS
;[(0,177777):(177777,0):(401,177777):(177777,401)]?
BNE 1\$;NO - LOOP
RETURN

MT0004: SUBTST <<MT0004 SETUP ROTATING ZEROS TEST>>

:SUBTEST MT0004 SETUP ROTATING ZEROS TEST

8123 021474 012737 000004 002276
8124 021502 012737 000004 002324
8125 021510 013702 002602
8126 021514 004737 041236
8127 021520 012700 060000
8128 021524 012701 040000
8129 021530 022737 000001 003754
8130 021536 001406
8131 021540 012737 027652 002260
8132 021546 004737 027226
8133 021552 000207
8134 021554
8135 021562
8136 021574 012737 172360 177652

MOV #4,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
MOV #4,PCBUMP ;TRAPS ADD 4 TO PC
MOV ONES,R2
CALL BACKGND ;WRITE BACKGROUND OF ONES
MOV #FIRST,R0
MOV #SIZE,R1
CMP #1,PROTYP ;IS THIS AN 11/44?
BEQ 1\$;BRANCH IF IT IS
MOV #MTPA04,SUPDOADD ;SET UP LINKS
CALL SUPDO4
RETURN
1\$: BMOV MTPA04
BMOV MTPB04,KDPAR0,8.
MOV #KDPAR0,UIPAR5

CZMSPB0 MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 222-1
MT0004 SETUP ROTATING ZEROS TEST

8137 021602 012737 177654 172376
8138 021610 004737 027050
8139 021614 000207
8140 021616

MOV #UIPAR6,KDPA7
CALL SUPD02
RETURN

MT0005: SUBTST <<MT0005 SETUP ROTATING ONES TEST>>

: *SUBTEST MT0005 SETUP ROTATING ONES TEST

8141 021616 012737 000005 002276
8142 021624 012737 000004 002324
8143 021632 005002
8144 021634 004737 041236
8145 021640 012700 060000
8146 021644 012701 040000
8147 021650 022737 000001 003754
8148 021656 001414
8149 021660 012737 027726 002260
8150 021666 012737 027742 027724
8151 021674 004737 027226
8152 021700 012737 027666 027724
8153 021706 000207
8154 021710
8155 021716
8156 021730 012737 172360 177652
8157 021736 012737 177654 172376
8158 021744 004737 027050
8159 021750 000207

MOV #5,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
MOV #4,PCBUMP ;TRAPS ADD 4 TO PC
CLR R2
CALL BACKGND ;WRITE BACKGROUND OF ZEROS
MOV #FIRST,R0
MOV #SIZE,R1
CMP #1,PROTYP ;IS THIS AN 11/44?
BEQ 1\$;BRANCH IF IT IS
MOV #MTP005,SUPD0ADD ;SET UP LINKS
MOV #MTP005+14,MTPB04+16
CALL SUPD04
MOV #MTPA04+14,MTPB04+16 ;RESET TEST'S ORIGINAL VALUE
RETURN
1\$: BMOV MTP005
BMOV MTPB04,KDPA0,8.
MOV #KDPA0,UIPA5
MOV #UIPAR6,KDPA7
CALL SUPD02
RETURN

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 224
MT0005 SETUP ROTATING ONES TEST

8162 021752

MT0006: SUBTST <<MT0006 SETUP INITIAL DATA TEST>>
:*****
:*SUBTEST MT0006 SETUP INITIAL DATA TEST
:*****

8163 021752 012737 000006 002276
8164 021760 012737 000004 002324
8165 021766 012701 002410
8166 021772 012737 027762 002260
8167 022000 004737 027212
8168 022004 000207
8169 022006

MOV #6,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
MOV #4,PCBUMP ;TRAPS ADD 4 TO PC
MOV #TESTADD,R1
MOV #MTP006,SUPDOADD
CALL SUPDO3 ;DO IT IN SUPERVISOR MODE
RETURN

MT0007: SUBTST <<MT0007 SETUP ADDRESS BIT TEST>>
:*****
:*SUBTEST MT0007 SETUP ADDRESS BIT TEST
:*****

8170 022006 012737 000007 002276
8171 022014 005002
8172 022016 004737 041236
8173 022022 012701 060000
8174 022026 012702 000001
8175 022032 050201
8176 022034 012737 030162 002260
8177 022042 004737 027212
8178 022046 000207
8179 022050

MOV #7,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
CLR R2
CALL BACKGND ;OF ZEROS
MOV #FIRST,R1
MOV #1,R2
BIS R2,R1
MOV #MTP007,SUPDOADD
CALL SUPDO3 ;DO IT IN SUPERVISOR MODE
RETURN

MT0010: SUBTST <<MT0010 SETUP BYTE ADDRESSING TEST>>
:*****
:*SUBTEST MT0010 SETUP BYTE ADDRESSING TEST
:*****

8180 022050 012737 000010 002276
8181 022056 012737 000004 002324
8182 022064 013704 002410
8183 022070 012737 030262 002260
8184 022076 004737 027212
8185 022102 000207

MOV #10,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
MOV #4,PCBUMP ;TRAPS ADD 4 TO PC
MOV TESTADD,R4
MOV #MTP010,SUPDOADD
CALL SUPDO3 ;DO IT IN SUPERVISOR MODE
RETURN

CZMSPB0 MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 226
MT0010 SETUP BYTE ADDRESSING TEST

8188 022104

```

MT0011: SUBTST <<MT0011      SETUP CREATE SINGLE BIT ERROR TEST>>
:*****
:*SUBTEST      MT0011      SETUP CREATE SINGLE BIT ERROR TEST
:*****
      IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
      IF $PASS NE #0 THEN $RETURN
      END: OF IF ACTFLAG
      IF PMEMFLG IS TRUE THEN $RETURN ;EXIT IF NOT MS11-M
      MOV      #11,REALPAT           ;SETUP PATTERN NUMBER FOR TYPEOJT & DISPLAY
      MOV      #MTP011,SUPDOADD
      CALL     SUPDO3                ;DO IT IN SUPERVISOR MODE
      RETURN

```

8189 022104
8190 022120
8191 022130
8192 022130
8193 022140
8194 022146
8195 022154
8196 022160
8197 022162

012737 000011 002276
012737 030370 002260
004737 027212
000207

```

MT0012: SUBTST <<MT0012      SETUP WRITE BYTE CLEARS SBE TEST>>
:*****
:*SUBTEST      MT0012      SETUP WRITE BYTE CLEARS SBE TEST
:*****
      IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
      IF $PASS NE #0 THEN $RETURN
      END: OF IF ACTFLAG
      IF PMEMFLG IS TRUE THEN $RETURN ;IS THIS A MS11-M?
      MOV      #12,REALPAT           ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
      MOV      BANKINDEX,R0
      IF #BIT12 SET.IN CONFIG+2(R0)
      MOV      #40000,R5
      ELSE
      MOV      #2,R5
      END: OF IF #BIT12
      MOV      #MTP012,SUPDOADD
      CALL     SUPDO3                ;DO IT IN SUPERVISOR MODE
      RETURN

```

8198 022162
8199 022176
8200 022206
8201 022206
8202 022216
8203 022224
8204 022230
8205 022240
8206 022244
8207 022246
8208 022252
8209 022252
8210 022260
8211 022264
8212 022266

012737 000012 002276
013700 002102
012705 040000
012705 000002
012737 031166 002260
004737 027212
000207

```

MT0013: SUBTST <<MT0013      SETUP CREATE DOUBLE BIT ERROR TEST>>
:*****
:*SUBTEST      MT0013      SETUP CREATE DOUBLE BIT ERROR TEST
:*****
      IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
      IF $PASS NE #0 THEN $RETURN
      END: OF IF ACTFLAG
      IF PMEMFLG IS TRUE THEN $RETURN ;EXIT IF NOT MS11-M
      MOV      #13,REALPAT           ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
      MOV      #MTP013,SUPDOADD
      MOV      #3,NOPAR              ;INDICATE PARITY ACRION
      CALL     SUPDO3                ;DO IT IN SUPERVISOR MODE
      RETURN

```

8213 022266
8214 022302
8215 022312
8216 022312
8217 022322
8218 022330
8219 022336
8220 022344
8221 022350

012737 000013 002276
012737 031554 002260
012737 000003 002074
004737 027212
000207

C7MSPB0 MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 227
M:0013 SETUP CREATE DOUBLE BIT ERROR TEST

8223 022352

MT0014: SUBTST <<MT0014 SETUP BASIC DOUBLE BIT ERROR TEST>>
:*****
:*SUBTEST MT0014 SETUP BASIC DOUBLE BIT ERROR TEST
:*****

8224 022352

8225 022366

8226 022376

8227 022376

8228 022406 012737 000014 002276

8229 022414 004737 046720

8230 022420

8231 022424 004737 041336

8232 022430 004737 032270

8233 022434 004737 047006

8234 022440 000207

IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
IF \$PASS NE #0 THEN \$RETURN
END; OF IF ACTFLAG
IF PMEMFLG IS FALSE THEN \$RETURN ;EXIT IF NOT MS11-P
MOV #14,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
CALL MAPKERNAL ;MAP KERNAL SPACE
LET R1 := #100000 ;SETUP TEST ADDRESS
CALL GETCSR ;GET CSR INFO FROM CONFIGURATION TABLE
CALL MTP014 ;DO BASIC DOUBLE BIT ERROR TEST
CALL UNMAP ;UNMAP KERNAL SPACE
RETURN

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 229
 MT0014 SETUP BASIC DOUBLE BIT ERROR TEST

8237 022442

MT0015: SUBTST <<MT0015 SETUP WRITE INHIBIT OF BYTE WITH DBE>>

 ;*SUBTEST MT0015 SETUP WRITE INHIBIT OF BYTE WITH DBE
 ;*****

8238 022442

IF ACTFLAG IS TRUE OR APTFLAG IS TRUE

8239 022456

IF \$PASS NE #0 THEN \$RETURN

8240 022466

END ;OF IF ACTFLAG

8241 022466

IF PMEMFLG IS TRUE THEN \$RETURN ;EXIT IF NOT MS11-M

8242 022476

012737

000015

002276

MOV #15,REALPAT

;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY

8243 022504

012737

032514

002260

MOV #MTP015,SUPDOADD

8244 022512

004737

027212

CALL SUPDO3

;DO IT IN SUPERVISOR MODE

8245 022516

000207

RETURN

8246 022520

MT0016: SUBTST <<MT0016 SETUP WRITE INHIBIT OF WORD WITH DBE>>

 ;*SUBTEST MT0016 SETUP WRITE INHIBIT OF WORD WITH DBE
 ;*****

8247 022520

IF PMEMFLG IS TRUE THEN \$RETURN ;EXIT IF NOT MS11-M

8248 022530

IF ACTFLAG IS TRUE OR APTFLAG IS TRUE

8249 022544

IF \$PASS NE #0 THEN \$RETURN

8250 022554

END ;OF IF ACTFLAG

8251 022554

012737

000016

002276

MOV #16,REALPAT

;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY

8252 022562

012737

033260

002260

MOV #MTP016,SUPDOADD

8253 022570

004737

027212

CALL SUPDO3

;DO IT IN SUPERVISOR MODE

8254 022574

000207

RETURN

8255 022576

MT0017: SUBTST <<MT0017 SETUP HOLDING 1'S & 0'S>>

 ;*SUBTEST MT0017 SETUP HOLDING 1'S & 0'S
 ;*****

8256 022576

012737

000017

002276

MOV #17,REALPAT

;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY

8257 022604

012737

034042

002260

MOV #MTP017,SUPDOADD

8258 022612

004737

027212

CALL SUPDO3

;DO IT IN SUPERVISOR MODE

8259 022616

000207

RETURN

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 231
MT0017 SETUP HOLDING 1'S & 0'S

8262 022620

MT0020: SUBTST <<MT0020 SETUP SYNDROMES TO CSR ON SINGLE BIT ERROR>>

;*SUBTEST MT0020 SETUP SYNDROMES TO CSR ON SINGLE BIT ERROR

8263 022620

IF ACTFLAG IS TRUE OR APTFLAG IS TRUE

8264 022634

IF \$PASS NE #0 THEN \$RETURN

8265 022644

END; OF IF ACTFLAG

8266 022644 012737 000020 002276

MOV #20,REALPAT

;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY

8267 022652

IF PMEMFLG IS FALSE THEN \$RETURN

;EXIT IF NOT MS11-P

8268 022662 004737 046720

CALL MAPKERNAL

;MAP KERNAL SPACE

8269 022666

LET R1 := #100000

;SETUP TEST ADDRESS

8270 022672 004737 041336

CALL GETCSR

;GET CSR INFO FROM CONFIGURATION TABLE

8271 022676 004737 034120

CALL MTP020

;DO SYNDROMES TO CSR ON SINGLE ERROR TEST

8272 022702 004737 047006

CALL UNMAP

;UNMAP KERNAL SPACE

8273 022706 000207

RETURN

CZMSPB0 MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 232
MT0020 SETUP SYNDROMES TO CSR ON SINGLE BIT ERROR

8275 022710

MT0021: SUBST <<MT0021 SETUP MARCHING 0'S & 1'S TEST>>

:SUBTEST MT0021 SETUP MARCHING 0'S & 1'S TEST

```

8276 022710          SET NOSCOPE
8277 022716 012737 000021 002276  MOV #21,REALPAT          ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
8278 022724 013702 002620          MOV BAKPAT,R2
8279 022730 004737 041236          CALL BACKGND
8280 022734 010203          MOV R2,R3
8281 022736 000303          SWAB R3
8282 022740 012701 160000          MOV #LAST+2,R1
8283 022744 010105          MOV R1,R5
8284 022746 012704 060000          MOV #FIRST,R4
8285 022752 022737 000001 003754  CMP #1,PROTYP          ;IS THIS AN 11/44?
8286 022760 001441          BEQ 1$                 ;BRANCH IF IT IS
8287 022762 022737 000003 003754  CMP #3,PROTYP          ;IS THIS AN 11/24?
8288 022770 001407          BEQ 3$                 ;BRANCH IF SO
8289 022772 022737 000007 002100  CMP #7,BANK
8290 023000 001003          BNE 3$
8291 023002 012701 140000          MOV #140000,R1
8292 023006 010105          MOV R1,R5
8293 023010 012737 034434 002260 3$:  MOV #MTPA21,SUPDOADD
8294 023016 004737 027212          CALL SUPD03
8295 023022 012737 034464 002260  MOV #MTPB21,SUPDOADD
8296 023030 004737 027226          CALL SUPD04
8297 023034 010401          MOV R4,R1
8298 023036 012737 034520 002260  MOV #MTPC21,SUPDOADD
8299 023044 004737 027226          CALL SUPD04
8300 023050 012737 034554 002260  MOV #MTPD21,SUPDOADD
8301 023056 004737 027226          CALL SUPD04
8302 023062 000434          BR 2$
8303 023064 022737 000177 002100 1$:  CMP #177,BANK
8304 023072 001003          BNE 4$
8305 023074 012701 140000          MOV #140000,R1
8306 023100 010105          MOV R1,R5
8307 023102          4$:  BMOV MTPA21
8308 023110 004737 027034          CALL SUPD01
8309
8310 023114          BMOV MTPB21
8311 023122 004737 027050          CALL SUPD02
8312
8313 023126 010401          MOV R4,R1
8314 023130          BMOV MTPC21
8315 023136 004737 027050          CALL SUPD02
8316
8317 023142          BMOV MTPD21
8318 023150 004737 027050          CALL SUPD02
8319 023154 005037 002436 2$:  CLR NOSCOPE
8320 023160 000207          RETURN

```

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 233
 MT0021 SETUP MARCHING 0'S & 1'S TEST

8322 023162

MT0022: SUBTST <<MT0022 SETUP REFRESH & SHIFTING DIAGONAL TEST>>

 ;*SUBTEST MT0022 SETUP REFRESH & SHIFTING DIAGONAL TEST

8323 023162 004737 027000

CALL KAMITEST ;CHECK FOR KAMIKAZE MODE

8324 023166

ON.ERROR THEN \$RETURN ;IF NOT IN KAMIKAZE MODE RETURN

8325 023172 012737 000022 002276

MOV #22,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY

8326 023200 012737 034604 002260

MOV #MTP022,SUPDOADD

8327 023206 004737 027212

CALL SUPDO3 ;DO IT IN SUPERVISOR MODE

8328 023212 000207

RETURN

8329

8330 023214

MT0023: SUBTST <<MT0023 SHIFTING DIAGONAL TEST>>

 ;*SUBTEST MT0023 SHIFTING DIAGONAL TEST

8331 023214 004737 027000

CALL KAMITEST ;CHECK FOR KAMIKAZE MODE

8332 023220

ON.ERROR THEN \$RETURN ;IF NOT IN KAMIKAZE MODE RETURN

8333 023224 012737 000023 002276

MOV #23,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY

8334 023232 012737 034604 002260

MOV #MTP022,SUPDOADD

8335 023240

SET DIAGFLAG ;IDENTIFY DIAGONAL TEST TO MTP022

8336 023246 004737 027212

CALL SUPDO3 ;DO IT IN SUPERVISOR MODE

8337 023252 005037 002002

CLR DIAGFLAG

8338 023256 000207

RETURN

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 234
MT0023 SHIFTING DIAGONAL TEST

8340 023260

MT0024: SUBTST <<MT0024 SETUP FAST GALLOPING PATTERN TEST>>

:SUBTEST MT0024 SETUP FAST GALLOPING PATTERN TEST

8341 023260 004737 027000
8342 023264
8343 023270
8344 023276 012737 000024 002276
8345 023304 013702 002620
8346 023310 004737 041236
8347 023314 010203
8348 023316 010304
8349 023320 000304
8350 023322 012701 060000
8351 023326 012705 157776
8352 023332 022737 000001 003754
8353 023340 001417
8354 023342 022737 000003 003754
8355 023350 001406
8356 023352 022737 000007 002100
8357 023360 001002
8358 023362 012705 137776
8359 023366 104415
8360 023370 012737 035320 002260
8361 023376 000440
8362 023400 022737 000177 002554
8363 023406 001002
8364 023410 012705 137776
8365 023414 104415
8366 023416
8367 023424
8368 023436
8369 023450 012737 172260 002260
8370 023456 012737 172260 177676
8371 023464 012737 172360 172272
8372 023472 012737 177660 172374
8373 023500 004737 027226
8374
8375
8376 023504 104416
8377 023506 000302
8378 023510 000303
8379 023512 004737 027226
8380 023516 005037 002436
8381 023522 000207
8382 023524

CALL KAMITEST ;CHECK FOR KAMIKAZE MODE
ON.ERROR THEN \$RETURN ;IF NOT IN KAMIKAZE MODE RETURN
SET NOSCOPE
MOV #24,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
MOV BAKPAT,R2
CALL BACKGND
MOV R2,R3
MOV R3,R4
SWAB R4
MOV #FIRST,R1
MOV #LAST,R5
CMP #1,PROTYP
BEQ 1\$
CMP #3,PROTYP
BEQ 3\$
CMP #7,BANK
BNE 3\$
MOV #137776,R5
3\$: SAVREG
MOV #MTPB24,SUPDOADD
BR 2\$
1\$: CMP #177,LASTBANK
BNE 4\$
MOV #137776,R5
4\$: SAVREG
BMOV MTPA24
BMOV MTPB24,SDPAR0,8.
BMOV MTPC24,KDPAR0,8.
MOV #SDPAR0,SUPDOADD
MOV #SDPAR0,UDPAR7 ;SET UP PAR LINKS
MOV #KDPAR0,SDPAR5
MOV #UDPAR0,KDPAR6
2\$: CALL SUPD04

;DO IT AGAIN FOR COMPLEMENT DATA
RESREG
SWAB R2
SWAB R3
CALL SUPD04
CLR NOSCOPE
RETURN

MT0025: SUBTST <<MT0025 SETUP INTERRUPT ENABLE TEST>>

:SUBTEST MT0025 SETUP INTERRUPT ENABLE TEST

8383 023524
8384 023540
8385 023550
8386 023550
8387 023560 012737 000025 002276
8388 023566 012737 035352 002260
8389 023574 004737 027212
8390 023600 000207

IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
IF \$PASS NE #0 THEN \$RETURN
END ;OF IF ACTFLAG
IF PMEMFLG IS TRUE THEN \$RETURN ;EXIT IF NOT MS11-P
MOV #25,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
MOV #MTP025,SUPDOADD
CALL SUPD03 ;DO IT IN SUPERVISOR MODE
RETURN

CZMSPB0 MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 236
MT0025 SETUP INTERRUPT ENABLE TEST

8393 023602

MT0026: SUBTST <<MT0026 SETUP RANDOM DATA TEST>>

: *SUBTEST MT0026 SETUP RANDOM DATA TEST

8394	023602	012737	000026	002276	MOV	#26,REALPAT	
8395	023610	005037	002324		CLR	PCBUMP	:TRAPS DO NOT ADD TO THE PC
8396	023614	013703	002572		MOV	SEEDLO,R3	:INITIALIZE RANDOM NUMBERS
8397	023620	013702	002570		MOV	SEEDHI,R2	
8398	023624	010305			MOV	R3,R5	
8399	023626	010204			MOV	R2,R4	
8400	023630	012701	060000		MOV	#FIRST,R1	
8401	023634	012700	020000		MOV	#SIZE/2,R0	
8402	023640	022737	000001	003754	CMP	#1,PROTYP	:DO WE HAVE AN 11/44?
8403	023646	001437			BEQ	1\$:BRANCH IF WE DO
8404	023650	022737	000003	003754	CMP	#3,PROTYP	:11/24?
8405	023656	001406			BEQ	3\$:BRANCH IF SO
8406	023660	022737	000007	002100	CMP	#7,BANK	
8407	023666	001002			BNE	3\$	
8408	023670	012700	014000		MOV	#14000,R0	
8409	023674	104415			3\$: SAVREG		
8410	023676	012737	036024	036124	MOV	#MTPA26+4,MTPD26+14	
8411	023704	012737	036020	002260	MOV	#MTPA26,SUPDOADD	
8412	023712	004737	027212		CALL	SUPDO3	
8413	023716	005037	036050		CLR	RANODD	:FOR ERROR REPORTING
8414	023722	012737	036040	036124	MOV	#MTPB26+4,MTPD26+14	:SET UP NEXT LINK
8415	023730	012737	036034	002260	MOV	#MTPB26,SUPDOADD	
8416	023736	104416			RESREG		
8417	023740	004737	027212		CALL	SUPDO3	
8418	023744	000452			BR	2\$	
8419	023746	022737	000177	002100	1\$: CMP	#177,BANK	
8420	023754	001002			BNE	4\$	
8421	023756	012700	014000		MOV	#14000,R0	
8422	023762	104415			4\$: SAVREG		
8423	023764				BMOV	MTPA26	:WRITE ROUTINE TO FAST MEMORY
8424	023772				BMOV	MTPC26,KDPAR0,8.	:RANDOM SUBPROGRAM TO FAST MEMORY
8425	024004	012737	000730	172376	MOV	#730,KDPAR7	:WRITES 'BR .-116' IN (BR SDPAR0)
8426	024012				BMOV	MTPD26,SDPAR0,8.	:RANDOM SUBSUBPROGRAM TO FAST MEMORY
8427	024024	012737	172360	177642	MOV	#KDPAR0,UIPAR1	
8428	024032	012737	177644	172274	MOV	#UIPAR2,SDPAR6	
8429	024040	004737	027034		CALL	SUPDO1	:WRITE RANDOM DATA
8430	024044	005037	036050		CLR	RANODD	:FOR ERROR REPORTING
8431	024050				BMOV	MTPB26	:READ ROUTINE TO FAST MEMORY
8432	024056	012737	172360	177642	MOV	#KDPAR0,UIPAR1	:SET UP PAR LINK
8433	024064	104416			RESREG		
8434	024066	004737	027034		CALL	SUPDO1	:READ RANDOM DATA
8435	024072	010337	002572		2\$: MOV	R3,SEEDLO	:UPDATE FOR NEW RANDOM NUMBERS
8436	024076	010237	002570		MOV	R2,SEEDHI	
8437	024102	000207			RETURN		

8440 024104

MT0027: SUBTST <<MT0027 UNIQUE BANK TEST>>

:*SUBTEST MT0027 UNIQUE BANK TEST

8441
8442
8443 024104 012737 000027 002276
8444 024112 104502
8445 024114 022737 000001 003754
8446 024122 001404
8447 024124 012737 027212 002520
8448 024132 000414
8449 024134
8450 024142 012737 177646 002260
8451 024150 012737 027034 002520
8452 024156
8453 024164
8454 024172
8455 024176 004737 047032
8456 024202
8457 024216 104511
8458 024220
8459 024224 012700 060000
8460 024230 010004
8461 024232 012701 040000
8462 024236 010103
8463 024240
8464 024250 022737 000001 003754
8465 024256 001403
8466 024260 012737 036344 002260
8467 024266 004777 156226
8468 024272
8469 024272
8470 024302 022737 000001 003754
8471 024310 001403
8472 024312 012737 036352 002260
8473 024320 004737 027212
8474 024324
8475 024324
8476 024324
8477 024340
8478 024354
8479 024362 005037 002424
8480 024366 000207
8481 024370
8482 024370
8483 024376
8484 024404 004737 047032
8485 024410
8486 024424
8487 024430 005102
8488 024432 012700 060000
8489 024436 010004
8490 024440 012701 040000
8491 024444 010103
8492 024446
8493 024456 022737 000001 003754

```

;MAKE SURE THAT EACH BANK CAN HAVE UNIQUE DATA
;WRITE AND READ THE BANK NUMBER IN EACH BANK (EXCEPT WHERE THE PROGRAM IS)
MOV #27,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
CLRCSR ;CLEAR CSRS
CMP #1,PROTYP ;IS THIS AN 11/44?
BEQ 1$ ;BRANCH IF TRUE
MOV #SUPDO3,LINK1 ;SET UP LINK
BR STAR27 ;BRANCH TO RUN
1$: BMOV MTP034
WARN7: MOV #UIPAR3,SUPDOADD
MOV #SUPDO1,LINK1 ;SET UP LINK
SET NOFSMODE
STAR27: FOR I := #1 TO #2
FOR BANK := #0 TO LASTBANK
CALL EXBANK
IF ACFLAG IS TRUE AND RRFLAG IS FALSE
INVALIDATE ;INVALIDATE BACKGROUND PATTERN ON 'BANK'
LET R2 := BANK
MOV #FIRST,R0
MOV R0,R4
MOV #SIZE,R1
MOV R1,R3
IF I EQ #1
CMP #1,PROTYP
BEQ 2$
MOV #MTP034,SUPDOADD
2$: CALL @LINK1
END ;OF IF
IF I EQ #2
CMP #1,PROTYP
BEQ 3$
MOV #MTP034+6,SUPDOADD
3$: CALL SUPDO3
END ;OF IF
END ;OF IF
END ;OF FOR BANK
END ;OF FOR I
IF FS7FLAG IS TRUE
CLR NOFSMODE
RETURN
END ;OF IF FS7FLAG
FOR I := #1 TO #2
FOR BANK := LASTBANK DOWNT0 #0
CALL EXBANK
IF ACFLAG IS TRUE AND RRFLAG IS FALSE
LET R2 := BANK
COM R2
MOV #FIRST,R0
MOV R0,R4
MOV #SIZE,R1
MOV R1,R3
IF I EQ #1
CMP #1,PROTYP

```

CZMSPB0 MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 238-1
 MTO027 UNIQUE BANK TEST

8494	024464	001403			BEQ	4\$
8495	024466	012737	036344	002260	MOV	#MTP034,SUPDOADD
8496	024474	004777	156020		4\$: CALL	@LINK1
8497	024500				END ;OF IF	
8498	024500				IF I EQ #2	
8499	024510	022737	000001	003754	CMP	#1,PROTYP
8500	024516	001403			BEQ	5\$
8501	024520	012737	036352	002260	MOV	#MTP034+6,SUPDOADD
8502	024526	004737	027212		5\$: CALL	SUPD03
8503	024532				END ;OF IF	
8504	024532				END ;OF IF	
8505	024532				END ;OF FOR BANK	
8506	024546				END ;OF FOR I	
8507	024562	005037	002424		CLR	NOFSMODE
8508	024566	000207			RETURN	

8511 024570

MT0030: SUBTST <<MT0030 SETUP FLUSH OUT DBE'S TEST>>

: *SUBTEST MT0030 SETUP FLUSH OUT DBE'S TEST

8512 024570 005037 002262

CLR PASFLG

8513 024574

SET FULLREL

8514 024602 012737 000030 002276

MTA030: MOV #30,REALPAT

:SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY

8515 024610 012737 000001 002074

MOV #1,NOPAR

:INDICATE COUNT PARITY ERRORS

8516 024616 022737 000001 003754

CMP #1,PROTYP

8517 024624 001007

BNE 4\$

8518 024626

BMOV MTP030

8519 024634 012737 027034 002520

MOV #SUPD01,LINK1

8520 024642 000406

BR 1\$

8521 024644 012737 027212 002520 4\$:

MOV #SUPD03,LINK1

8522 024652 012737 036126 002260

MOV #MTP030,SUPDOADD

8523 024660 104470 1\$:

ECCDIS ;DISABLE ERROR CORRECTION

8524 024662

SET NOFSMODE,NOSCOPE

8525 024676

FOR BANK := #0 TO LASTBANK

8526 024702 004737 047032

CALL EXBANK

8527 024706

IF MKFLAG IS TRUE

8528 024714

IF ACFLAG IS TRUE AND RRFLAG IS FALSE

8529 024730 012701 040000

MOV #SIZE,R1

8530 024734 012700 060000

MOV #FIRST,R0

8531 024740 004777 155554

CALL @LINK1

8532 024744

END ;OF IF ACFLAG

8533 024744

END ;OF IF MKFLAG

8534 024744

END ;OF FOR

8535 024760

IF PASFLG IS FALSE

8536 024766

SET PASFLG

8537 024774 104502

CLRCR

:CLEAR CSRS

8538 024776 004737 045204

CALL RELOCATE

8539 025002

ON.ERROR

8540 025004 104472

ECCINIT

:TRAP ON DOUBLE BIT ERRORS (NORMAL)

8541 025006

CLEAR NOFSMODE,NOSCOPE,FULLREL

8542 025022 000207

RETURN

8543 025024

END ;OF ON.ERROR

8544 025024 013737 002306 002100

MOV NEWBANK,BANK

8545 025032 004737 047032

CALL EXBANK

8546 025036 004737 024602

CALL MTA030

8547 025042 104472

ECCINIT

:TRAP ON DOUBLE BIT ERRORS (NORMAL)

8548 025044 004737 046070

CALL UNRELOCATE

8549 025050 000207

RETURN

8550 025052

END ;OF IF PASFLG

8551 025052 104472

ECCINIT

:TRAP ON DOUBLE BIT ERRORS (NORMAL)

8552 025054

CLEAR NOFSMODE,NOSCOPE,FULLREL

8553 025070 000207

RETURN

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 242
 MT0030 SETUP FLUSH OUT DBE'S TEST

8556 025072

MT0031: SUBTST <<MT0031 SETUP SOB-A-LONG TEST>>

```

*****
:SUBTEST      MT0031  SETUP SOB-A-LONG TEST
*****
8557 025072  004737  027000          CALL    KAMITEST          ;CHECK FOR KAMIKAZE MODE
8558 025076          ON.ERROR THEN $RETURN      ;IF NOT IN KAMIKAZE MODE RETURN
8559 025102          SET      NOSCOPE
8560 025110  012737  000031  002276  MOV     #31,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
8561 025116  005037  002074          CLR     NOPAR           ;SETUP PARITY ACTION
8562 025122          MAP     BANK             ;MAP FIRST SO BLOCK MOVE WORKS
8563 025136          TESTAREA
8564 025144          BMOV    MTP031,FIRST,SOBLENGTH/2 ;ENTER TEST MODE
8565 025156  104417          KERNEL
8566 025160  013702  002560          MOV     SOBK,R2         ;ENTER KERNEL MODE
8567 025164  010200          MOV     R2,R0
8568 025166  012701  100776          MOV     #100776,R1
8569 025172  012705  060056          MOV     #FIRST+SOBLENGTH,R5 ;COMPLEMENT OF INSTRUCTION "SOB R0,DOT"
8570 025176  012737  060002  002260  MOV     #FIRST+2,SUPDOADD
8571 025204  012737  160000  002520  MOV     #LAST+2,LINK1
8572 025212  005737  002454          TST     NOSUPER
8573 025216  001005          BNE     1$
8574 025220  023737  172252  172254  CMP     SIPAR5,SIPAR6
8575 025226  001405          BEQ     2$
8576 025230  000407          BR      3$
8577 025232  023737  177652  177654  1$:    CMP     UIPAR5,UIPAR6
8578 025240  001003          BNE     3$
8579 025242  012737  140000  002520  2$:    MOV     #140000,LINK1
8580 025250  004737  027226          3$:    CALL   SUPD04
8581 025254  005037  002436          CLR     NOSCOPE
8582 025260  000207          RETURN
    
```

8585 025262

MT0032: SUBTST <<MT0032 SETUP WRITE RECOVERY TEST>>

:SUBTEST MT0032 SETUP WRITE RECOVERY TEST

8586	025262	004737	027000		CALL	KAMITEST			:CHECK FOR KAMIKAZE MODE
8587	025266				ON.ERROR	THEN \$RETURN			:IF NOT IN KAMIKAZE MODE RETURN
8588	025272				SET	NOSCOPE			
8589	025300	012737	000032	002276	MOV	#32,REALPAT			:SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
8590	025306	005037	002074		CLR	NOPAR			:SETUP PARITY ACTION
8591	025312				MAP	BANK			:MAP FIRST SO THAT THE BLOCK MOVE WORKS
8592	025326	012700	010247		MOV	#10247,R0			:OP CODE OF INSTRUCTION 'MOV R2,-(PC)'
8593	025332	012701	177667		MOV	#177667,R1			:OP CODE OF COMPLEMENT OF INSTRUCTION 'JMP (R0)'
8594	025336	012702	020000		MOV	#SIZE/2,R2			:USED FOR 1/2 BANK LOOP
8595	025342	010237	002520		MOV	R2,LINK1			
8596	025346	012703	060000		MOV	#FIRST,R3			
8597	025352	012704	160000		MOV	#LAST+2,R4			
8598	025356	005037	002522		CLR	LINK2			
8599	025362	005737	002454		TST	NOSUPER			
8600	025366	001005			BNE	1\$			
8601	025370	023737	172252	172254	CMP	SIPAR5,SIPAR6			
8602	025376	001405			BEQ	2\$			
8603	025400	000415			BR	3\$			
8604	025402	023737	177652	177654	1\$:	CMP	UIPAR5,UIPAR6		
8605	025410	001011			BNE	3\$			
8606	025412	012704	140000		2\$:	MOV	#140000,R4		
8607	025416	012702	014000			MOV	#14000,R2		
8608	025422	010237	002520			MOV	R2,LINK1		
8609	025426	012737	000001	002522		MOV	#1,LINK2		
8610									
8611	025434				3\$:	TESTAREA			:ENTER TEST MODE
8612						:MOVE	TEST TO MEMORY UNDER TEST		
8613	025442	010023			4\$:	MOV	R0,(R3)+		
8614	025444	010144				MOV	R1,-(R4)		
8615	025446	077203				SOB	R2,4\$		
8616									
8617	025450	022737	000001	003754		CMP	#1,PROTYP		
8618	025456	001003				BNE	5\$		
8619						:MOVE	LAST PART OF TEST TO FASTCITY		
8620	025460					BMOV	MTP032		
8621	025466	104417			5\$:	KERNEL			:ENTER KERNEL MODE
8622									
8623	025470	012702	005141			MOV	#5141,R2		:OP CODE OF INSTRUCTION 'COM -(R1)'
8624	025474	012700	025612			MOV	#10\$,R0		:ADDRESS TO RETURN TO IN R0
8625	025500	012701	160000			MOV	#LAST+2,R1		:TOP OF BANK
8626	025504	012737	060000	002260		MOV	#FIRST,SUPDOADD		
8627	025512	005737	002522			TST	LINK2		
8628	025516	001402				BEQ	6\$		
8629	025520	012701	140000			MOV	#140000,R1		
8630	025524	004737	027226		6\$:	CALL	SUPD04		
8631	025530	012703	020000			MOV	#SIZE/2,R3		
8632	025534	012705	000110			MOV	#110,R5		
8633	025540	012704	060000			MOV	#FIRST,R4		
8634	025544	005737	002522			TST	LINK2		
8635	025550	001402				BEQ	7\$		
8636	025552	012703	014000			MOV	#14000,R3		
8637	025556	022737	000001	003754	7\$:	CMP	#1,PROTYP		
8638	025564	001406				BEQ	8\$		

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 244-1
MT0032 SETUP WRITE RECOVERY TEST

8639	025566	012737	036214	002260	MOV	#MTP032,SUPDOADD
8640	025574	004737	027226		CALL	SUPD04
8641	025600	000402			BR	9\$
8642	025602	004737	027050	8\$:	CALL	SUPD02
8643	025606	005037	002436	9\$:	CLR	NOSCOPE
8644	025612	000207		10\$:	RETURN	
8645						
8646						

;THIS RETURN ACTS AS A NORMAL RETURN FROM MT0032
;ALSO A RETURN FROM THE "CALL SUPD04" ABOVE

8649 025614

MT0033: SUBTST <<MT0033 SETUP BRANCH GOBBLE TEST>>

: *SUBTEST MT0033 SETUP BRANCH GOBBLE TEST
: *****

8650 025614 004737 027000
8651 025620
8652 025624
8653 025632 012737 000033 002276
8654 025640 005037 002074
8655 025644
8656
8657 025660
8658 025666
8659 025700 104417
8660
8661 025702 012705 060076
8662 025706 012737 060004 002260
8663 025714 012701 060002
8664 025720 012702 060003
8665 025724 012737 160000 002520
8666 025732 005737 002454
8667 025736 001005
8668 025740 023737 172252 172254
8669 025746 001405
8670 025750 000407
8671 025752 023737 177652 177654 1\$:
8672 025760 001003
8673 025762 012737 140000 002520 2\$:
8674
8675 025770 004737 027226 3\$:
8676 025774 005037 002436
8677 026000 000207
8678
8679 026002

CALL KAMITEST ;CHECK FOR KAMIKAZE MODE
ON.ERROR THEN \$RETURN ;IF NOT IN KAMIKAZE MODE RETURN
SET NOSCOPE
MOV #33,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
CLR NOPAR ;SETUP PARITY ACTION
MAP BANK ;MAP FIRST SO THAT BLOCK MOVE WORKS

TESTAREA ;ENTER TEST MODE
BMOV MTP033,FIRST,GBLENGTH/2
KERNEL ;ENTER KERNEL MODE

MOV #FIRST+GBLENGTH,R5
MOV #FIRST+4,SUPDOADD
MOV #FIRST+2,R1
MOV #FIRST+3,R2
MOV #LAST+2,LINK1
TST NOSUPER
BNE 1\$
CMP SIPAR5,SIPAR6
BEQ 2\$
BR 3\$
CMP UIPAR5,UIPAR6
BNE 3\$
MOV #140000,LINK1

3\$: CALL SUPD04
CLR NOSCOPE
RETURN

MT0034: SUBTST <<MT0034 SOFT ERROR - BACKGROUND PATTERN TEST>>

: *SUBTEST MT0034 SOFT ERROR - BACKGROUND PATTERN TEST
: *****

8680 026002 012737 000034 002276
8681 026010 012700 060000
8682 026014 012701 040000
8683 026020 013702 002606
8684 026024 010103
8685 026026 013705 002102
8686 026032 010004
8687 026034 022737 000001 003754
8688 026042 001006
8689 026044
8690 026052 012737 177646 002260
8691 026060
8692
8693 026070 022737 000001 003754
8694 026076 001403
8695 026100 012737 036352 002260
8696 026106 004737 027212
8697 026112
8698
8699 026114 022737 000001 003754

MOV #34,REALPAT
MOV #FIRST,R0
MOV #SIZE,R1
MOV SOFTPAT,R2
MOV R1,R3
MOV BANKINDEX,R5
MOV R0,R4
CMP #1,PROTYP ;IS THIS AN 11/44?
BNE 1\$;BRANCH IF NOT
BMOV MTP034
MOV #UIPAR3,SUPDOADD
1\$: IF #BIT13 SET.IN CONFIG+2(R5)
;BACKGROUND PATTERN IS VALID
CMP #1,PROTYP
BEQ 2\$
MOV #MTP034+6,SUPDOADD
2\$: CALL SUPD03 ;READ IT
ELSE
;BACKGROUND PATTERN HAS BEEN INVALIDATED
CMP #1,PROTYP

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 246-1
MT0034 SOFT ERROR - BACKGROUND PATTERN TEST

8700	026122	001406				BEQ	3\$		
8701	026124	012737	036344	002260		MOV	#MTP034,SUPDOADD		
8702	026132	004737	027212			CALL	SUPDO3		
8703	026136	000402				BR	4\$		
8704	026140	004737	027034		3\$:	CALL	SUPDO1		:WRITE IT
8705	026144	052765	020000	002654	4\$:	BIS	#BIT13,CONFIG+2(R5)		:VALIDATE IT
8706	026152					END	:OF IF #BIT13		
8707	026152	000207				RETURN			
8708									
8709	026154								

MT0035: SUBTST <<MT0035 SETUP WORST CASE NOISE PARITY TEST>>

```

:*****
:*SUBTEST      MT0035  SETUP WORST CASE NOISE PARITY TEST
:*****

```

8710	026154	012737	000035	002276		MOV	#35,REALPAT		:SET UP TEST NUMBER FOR DISPLAY
8711	026162	013703	002102			MOV	BANKINDEX,R3		
8712	026166	016301	002652			MOV	CONFIG(R3),R1		
8713	026172	000301				SWAB	R1		
8714	026174	042701	177760			BIC	#^C17,R1		
8715	026200	006301				ASL	R1		
8716	026202	010137	002150			MOV	R1,CSRNO		
8717	026206	023737	002150	002530		CMP	CSRNO,PGMCSR		
8718	026214	001001				BNE	1\$		
8719	026216	000207				RETURN			
8720	026220	012702	052524		1\$:	MOV	#52524,R2		
8721	026224	004737	041236			CALL	BACKGND		:WRITE BACKGROUND OF ALMOST ALT. 1'S AND 0'S
8722	026230	012737	036370	002260		MOV	#MTP035,SUPDOADD		
8723	026236	004737	027212			CALL	SUPDO3		
8724	026242					IF	QVFLAG IS TRUE THEN \$RETURN		
8725	026252	005102				COM	R2		
8726	026254	004737	041236			CALL	BACKGND		:WRITE COMPLEMENT PATTERN INTO MUT
8727	026260	004737	027226			CALL	SUPDO4		
8728	026264	000207				RETURN			

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 247
MT0035 SETUP WORST CASE NOISE PARITY TEST

8730 026266

MT0036: SUBTST <<MT0036 SETUP CORRECTION CODE TEST>>

:SUBTEST MT0036 SETUP CORRECTION CODE TEST

8731 026266
8732 026276 012737 000036 002276
8733 026304 004737 041336
8734 026310 005037 002262
8735 026314 005000
8736 026316 012701 100000
8737 026322 004737 046720
8738 026326 004737 036532
8739 026332 004737 047006
8740 026336 000207

IF PMEMFLG IS FALSE THEN \$RETURN :IF NOT MS11-P THEN EXIT
MOV #36,REALPAT :SET UP TEST NUMBER FOR TYPEOUT AND DISPLAY
CALL GETCSR :GET CSR INFO FROM CONFIG TABLE
CLR PASFLG :CLEAR LOOP COUNTER
CLR R0 :GET TEST DATA
MOV #100000,R1 :GET FIRST ADDRESS IN BANK
CALL MAPKERNAL :MAP KIPARS AND 6 TO BANK
CALL MTP036 :EXECUTE TEST
CALL UNMAP :REMAP KERNAL SPACE
RETURN

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 248
MT0036 SETUP CORRECTION CODE TEST

8742 026340

MT0037: SUBTST <<MT0037 SETUP ECC DISABLE TEST>>

:SUBTEST MT0037 SETUP ECC DISABLE TEST

8743 026340

IF PMEMFLG IS FALSE THEN \$RETURN ;RETURN IF NOT A MS11-P
MOV #37,REALPAT ;SETUP PATTERN AND NUMBER FOR TYPEOUT AND DISPLAY
MOV #100000,R1 ;SET UP TEST ADDRESS
CLR R0 ;CLEAR DATA TO BE WRITTEN
CALL MAPKERNAL ;MAP THIS TEST TO KERNEL SPACE
CALL GETCSR ;GET CSRINFO FROM CONFIG TABLE
CALL MTP037 ;CHECK ECC DISABLE
CALL UNMAP ;REMAP KERNEL SPACE
RETURN

8744 026350

012737 000037 002276

8745 026356

012701 100000

8746 026362

005000

8747 026364

004737 046720

8748 026370

004737 041336

8749 026374

004737 036756

8750 026400

004737 047006

8751 026404

000207

8752

8753

8754 026406

MT0040: SUBTST <<MT0040 >>

:SUBTEST MT0040

8755 026406

000207

RETURN ;

8757 026410

MT0041: SUBTST <<MT0041 SETUP ADDRESS TO CSR ON DOUBLE BIT ERROR TEST>>

;*SUBTEST MT0041 SETUP ADDRESS TO CSR ON DOUBLE BIT ERROR TEST

8758 026410

IF PMEMFLG IS FALSE THEN \$RETURN :EXIT IF NOT MS11-P

8759 026420 012737 000041 002276

MOV #41,REALPAT ;SETUP PATTERN AND NUMBER FOR TYPEOUT AND DISPLAY

8760 026426 004737 041336

CALL GETCSR ;GET CSR NUMBER AND ADDRESS FROM CONFIGURATION TABLE

8761 026432

LET SUPDOADD := #MTP041 ;SET UP TEST ADDRESS

8762 026440

LET R1 := #FIRST ;SET UP FIRST ADDRESS

8763 026444 004737 027212

CALL SUPDO3 ;EXECUTE ADDRESS TO CSR TEST IN SUPVISIOR MODE

8764 026450 000207

RETURN ;

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 250
MT0041 SETUP ADDRESS TO CSR ON DOUBLE BIT ERROR TEST

8766 026452

MT0042: SUBTST <<MT0042 SETUP EXTENDED UNIBUS ADDRESS TO CSR TEST>>

:*SUBTEST MT0042 SETUP EXTENDED UNIBUS ADDRESS TO CSR TEST

8767 026452
8768 026462
8769 026470
8770 026474
8771 026500
8772 026504
8773 026510
8774 026514

012737 000042 002276
012701 100000
004737 046720
004737 041336
004737 037202
004737 047006
000207

IF P^MMEMFLG IS FALSE THEN \$RETURN ;EXIT IF NOT MS11-P
MOV #42,REALPAT ;SETUP PATTERN AND NUMBER FOR TYPEOUT AND DISPLAY
MOV #100000,R1 ;SET UP TEST ADDRESS
CALL MAPKERNAL ;MAP TO KERNEL SPACE
CALL GETCSR ;SET UP CSRINFO FROM CONFIGURATION TABLE
CALL MTPJ42 ;CHECK EXTENDED UNIBUS ADDRESS TO CSR
CALL UNMAP ;REMAP KERNEL SPACE
RETURN

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 251
MT0042 SETUP EXTENDED UNIBUS ADDRESS TO CSR TEST

8776 026516

MT0043: SUBTST <<MT0043 SETUP WRITE BYTE CLEARS SBE TEST>>

:SUBTEST MT0043 SETUP WRITE BYTE CLEARS SBE TEST

8777 026516
8778 026526 012737 000043 002276
8779 026534 004737 046720
8780 026540
8781 026544 004737 037436
8782 026550 004737 047006
8783 026554 000207
8784 026556

IF PMEMFLG IS FALSE THEN \$RETURN ;EXIT IF NOT MS11-P
MOV #43,REALPAT ;SET UP TEST NUMBER FOR TYPEOUT AND DISPLAY
CALL MAPKERNAL ;MAP TO KERNEL SPACE
LET R1 := #100000 ;SET UP TEST ADDRESS
CALL MTP043 ;PERFORM WRITE BYTE TEST
CALL UNMAP ;REMAP KERNEL SPACE
RETURN

MT0044: SUBTST <<MT0044 SETUP SHIFTING 1/0'S THROUGH THE CHECK BITS TEST>>

:SUBTEST MT0044 SETUP SHIFTING 1/0'S THROUGH THE CHECK BITS TEST

8785 026556
8786 026566 012737 000044 002276
8787 026574 004737 041336
8788 026600
8789 026606
8790 026612 004737 027212
8791 026616 000207
8792 026620

IF PMEMFLG IS FALSE THEN \$RETURN ;EXIT IF NOT MS11-P
MOV #44,REALPAT ;SET UP TEST NUMBER FOR TYPEOUT AND DISPLAY
CALL GETCSR ;GET CSR NUMBER AND ADDRESS FROM CONFIGURATION TABLE
LET SUPDOADD := #MTP044 ;SET UP TEST ADDRESS
LET R1 := #FIRST ;SET UP FIRST ADDRESS
CALL SUPDO3 ;EXECUTE ADDRESS TO CSR TEST IN SUPVISOR MODE
RETURN

MT0045: SUBTST <<MT0045 SETUP SYNDROMES TO CSR ON DOUBLE BIT ERROR>>

:SUBTEST MT0045 SETUP SYNDROMES TO CSR ON DOUBLE BIT ERROR

8793 026620
8794 026630 012737 000045 002276
8795 026636 004737 046720
8796 026642
8797 026646 004737 040154
8798 026652 004737 047006
8799 026656 000207
8800 026660

IF PMEMFLG IS FALSE THEN \$RETURN ;EXIT IF NOT MS11-P
MOV #45,REALPAT ;SET UP TEST NUMBER FOR TYPEOUT AND DISPLAY
CALL MAPKERNAL ;MAP TO KERNEL SPACE
LET R1 := #100000 ;SET UP TEST ADDRESS
CALL MTP045 ;PERFORM SYNDROMES TO CSR ON DOUBLE BIT ERROR
CALL UNMAP ;REMAP KERNEL SPACE
RETURN

MT0046: SUBTST <<MT0046 SETUP CHECK SINGLE BIT ERRORS WITH ECC DISABLED TET>>

:SUBTEST MT0046 SETUP CHECK SINGLE BIT ERRORS WITH ECC DISABLED TET

8801 026660
8802 026670 012737 000046 002276
8803 026676 004737 046720
8804 026702
8805 026706 004737 040342
8806 026712 004737 047006
8807 026716 000207
8808 026720

IF PMEMFLG IS FALSE THEN \$RETURN ;EXIT IF NOT MS11-P
MOV #46,REALPAT ;SET UP TEST NUMBER FOR TYPEOUT AND DISPLAY
CALL MAPKERNAL ;MAP TO KERNEL SPACE
LET R1 := #100000 ;SET UP TEST ADDRESS
CALL MTP046 ;PERFORM TRAPS DETECTED ON SBE WITH ECC DISABLED TE
CALL UNMAP ;REMAP KERNEL SPACE
RETURN

MT0047: SUBTST <<MT0047 SETUP NO CSR UPDATE ON SBE WITH EXSISTING DBE TEST>>

:SUBTEST MT0047 SETUP NO CSR UPDATE ON SBE WITH EXSISTING DBE TEST

8809 026720
8810 026730 012737 000047 002276
8811 026736 004737 046720
8812 026742
8813 026746
8814 026752 004737 040702
8815 026756 004737 047006
8816 026762 000207

IF PMEMFLG IS FALSE THEN \$RETURN ;EXIT IF NOT MS11-P
MOV #47,REALPAT ;SET UP TEST NUMBER FOR TYPEOUT AND DISPLAY
CALL MAPKERNAL ;MAP TO KERNEL SPACE
LET R1 := #100000 ;SET UP TEST ADDRESS
LET R2 := #120000 ;" " SECOND TEST ADDRESS
CALL MTP047 ;PERFORM NO UPDATE TO CSR ON SBE WITH DBE
CALL UNMAP ;REMAP KERNEL SPACE
RETURN

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 253
MT0047 SETUP NO CSR UPDATE ON SBE WITH EXSISTING DBE TEST

8819 026764

MT0999: SUBTST <<MT0999 SETUP NULL TEST>>

: *SUBTEST MT0999 SETUP NULL TEST

8820 026764 005037 002276

CLR REALPAT
SET NULLFLAG
RETURN

8821 026770

8822 026776 000207

8823

8824 027000

KAMITEST: SUBTST <<CHECK FOR KAMIKAZE MODE>>

: *SUBTEST CHECK FOR KAMIKAZE MODE

8825 027000

IF KAMIKAZE IS TRUE OR ACTFLAG IS TRUE OR APTFLAG IS TRUE

8826 027022

\$RETURN NOERROR ;RUN THE TEST

8827 027026

ELSE

8828 027030

\$RETURN ERROR ;DON'T RUN THE TEST

8829 027034

END ;OF IF KAMIKAZE

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 255
CHECK FOR KAMIKAZE MODE

8832 027034

SUPD01: SUBTST <<SUBR EXECUTE PATTERN IN SUPERVISOR>>
:*****
:*SUBTEST SUBR EXECUTE PATTERN IN SUPERVISOR
:*****

8833 027034
8834 027050 004737 060514
8835 027051
8836 027064 010037 002156
8837 027070 012700 002160
8838 027074 010120
8839 027076 010220
8840 027100 010320
8841 027102 010420
8842 027104 010520
8843 027106 010620
8844 027110 013700 002156
8845 027114 012737 027130 002610
8846 027122 013737 002610 002612
8847 027130 012700 002174
8848 027134 014006
8849 027136 014005
8850 027140 014004
8851 027142 014003
8852 027144 014002
8853 027146 014001
8854 027150 014000
8855 027152
8856 027160 012706 000740
8857 027164 104424
8858 027166 004737 177640
8859 027172 104423
8860 027174 104417
8861 027176 000004
8862 027200
8863 027210 000207

SUPD02: MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
CALL GETDIS
PUSH \$LPERR,\$LPADR
MOV R0,SUPDRO
MOV #SUPDR1,R0
MOV R1,(R0)+
MOV R2,(R0)+
MOV R3,(R0)+
MOV R4,(R0)+
MOV R5,(R0)+
MOV SP,(R0)+
MOV SUPDRO,R0
MOV #TAG4\$,\$LPADR
MOV \$LPADR,\$LPERR
TAG4\$: MOV #SUPDR6+2,R0
MOV -(R0),SP
MOV -(R0),R5
MOV -(R0),R4
MOV -(R0),R3
MOV -(R0),R2
MOV -(R0),R1
MOV -(R0),R0
SUPERVISOR ;ENTER SUPERVISOR MODE
MOV #SUPSTK,SSP
CACHOFF ;TURN CACHE OFF
CALL FASTCITY ;CALL TO THE USER INSTRUCTION PAR'S
CACHON ;TURN CACHE ON
KERNEL ;ENTER KERNEL MODE
SCOPE
POP \$LPADR,\$LPERR
RETURN

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 257
 SUBR EXECUTE PATTERN IN SUPERVISOR

8866	027212			SUPD03:	MAP	BANK	
8867	027226	004737	060514	SUPD04:	CALL	GETDIS	:MAP SUPERVISOR SPACE (TEST AREA) TO BANK
8868	027232				PUSH	\$LPERR,\$LPADR	
8869	027242	010037	002156		MOV	R0,SUPDR0	
8870	027246	012700	002160		MOV	#SUPDR1,R0	
8871	027252	010120			MOV	R1,(R0)+	
8872	027254	010220			MOV	R2,(R0)+	
8873	027256	010320			MOV	R3,(R0)+	
8874	027260	010420			MOV	R4,(R0)+	
8875	027262	010520			MOV	R5,(R0)+	
8876	027264	010620			MOV	SP,(R0)+	
8877	027266	013700	002156		MOV	SUPDR0,R0	
8878	027272	012737	027306	002610	MOV	#TBG4\$,\$LPADR	
8879	027300	013737	002610	002612	MOV	\$LPADR,\$LPERR	
8880	027306	012700	002174	TBG4\$:	MOV	#SUPDR6+2,R0	
8881	027312	014006			MOV	-(R0),SP	
8882	027314	014005			MOV	-(R0),R5	
8883	027316	014004			MOV	-(R0),R4	
8884	027320	014003			MOV	-(R0),R3	
8885	027322	014002			MOV	-(R0),R2	
8886	027324	014001			MOV	-(R0),R1	
8887	027326	014000			MOV	-(R0),R0	
8888	027330				TESTAREA		:ENTER SUPERVISOR MODE
8889	027336	005737	002454		TST	NOSUPER	
8890	027342	001403			BEQ	1\$	
8891	027344	012706	000700		MOV	#USESTK,USP	
8892	027350	000402			BR	2\$	
8893	027352	012706	000740	1\$:	MOV	#SUPSTK,SSP	
8894	027356	104424		2\$:	CACHOFF		:TURN CACHE OFF
8895	027360	004777	152674		CALL	@SUPDOADD	
8896	027364	104423			CACHON		:TURN CACHE ON
8897	027366	104417			KERNEL		:ENTER KERNEL MODE
8898	027370	000004			SCOPE		
8899	027372				POP	\$LPADR,\$LPERR	
8900	027402	000207			RETURN		

```

8903
8904
8905
8906
8907
8908
8909
8910
8911
8912
8913 027404

8914 027404 010220
8915 027406 077102
8916 027410 000240
8917 027412 012401
8918 027414 020102
8919 027416 001402
8920 027420 104430
8921 027422 000240
8922 027424 077306
8923 027426 000207
8924 027430

8925 027430 010220
8926 027432 062702 000002
8927 027436 077104
8928 027440 000240
8929 027442 012400
8930 027444 020005
8931 027446 001401
8932 027450 104427
8933 027452 062705 000002
8934 027456 077307
8935 027460 000207
8936 027462

8937 027462 010540
8938 027464 062705 000002
8939 027470 077104
8940 027472 000240
8941 027474 162702 000002
8942 027500 012401
8943 027502 020102
8944 027504 001401
8945 027506 104430
8946 027510 077307
8947 027512 000207

```

```

.SBTTL MEMORY TEST PATTERN ROUTINES
*****
PATTERN REGISTER CONVENTIONS
R0 FIRST ADDRESS OF PATTERN (FIRST, LAST+2, ETC)
R1 NUMBER OF ADDRESSES IN PATTERN (SIZE)
R2 DATA FOR PATTERN (ONES, 52525, ETC)
R3 COPY OF R1 (IF NECESSARY)
R4 COPY OF R0 (IF NECESSARY)
R5 COPY OF R2 (IF NECESSARY)
*****
MTP000: SUBTST <<MTP000 BASIC DATA TEST>>
*****
*SUBTEST MTP000 BASIC DATA TEST
*****
1$: MOV R2, (R0)+ :V177640
SOB R1, MTP000 :V177642
NOP :V177644
2$: MOV (R4)+, R1 :V177646
CMP R1, R2 :V177650
BEQ 3$ :V177652
PERR02 :V177654
NOP :V177656
3$: SOB R3, 2$ :V177660
RETURN :V177662
MTP001: SUBTST <<MTP001 ADDRESS TEST>>
*****
*SUBTEST MTP001 ADDRESS TEST
*****
3$: MOV R2, (R0)+ :V177640
ADD #2, R2 :V177642
SOB R1, 3$ :V177646
NOP :V177650
1$: MOV (R4)+, R0 :V177652
CMP R0, R5 :V177654
BEQ 2$ :V177656
PERR01 :V177660
2$: ADD #2, R5 :V177662
SOB R3, 1$ :V177666
RETURN :V177672
MTP002: SUBTST <<MTP002 COMPLEMENT ADDRESS TEST (WRITE DOWN, READ UP)>>
*****
*SUBTEST MTP002 COMPLEMENT ADDRESS TEST (WRITE DOWN, READ UP)
*****
3$: MOV R5, -(R0) :V177640
ADD #2, R5 :V177642
SOB R1, 3$ :V177646
NOP :V177650
1$: SUB #2, R2 :V177652
MOV (R4)+, R1 :V177656
CMP R1, R2 :V177660
BEQ 2$ :V177662
PERR02 :V177664
2$: SOB R3, 1$ :V177666
RETURN :V177670

```


CZMSPB0 MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 261
MTP002 COMPLEMENT ADDRESS TEST (WRITE DOWN, READ UP)

8950 027514

MTPA03: SUBTST <<MTPA03 3 XOR 9 WORST CASE NOISE TEST (WRITE)>>

:SUBTEST MTPA03 3 XOR 9 WORST CASE NOISE TEST (WRITE)

8951
8952
8953
8954
8955
8956
8957 027514 010421
8958 027516 010421
8959 027520 077203
8960 027522 005104
8961 027524 052704
8962 027526 000401 000004
8963 027530 012702
8964 027534 077511
8965 027536 005104
8966 027540 052704
8967 027542 000401 000100
8968 027544 012705
8969 027550 077317
8970 027552 000207
8971
8972
8973 027554

:R1 = ADDRESS
:R2 = SMALL LOOP CONSTANT
:R3 = NUM OF ADD TO TEST (LARGE LOOP)
:R4 = GOOD DATA
:R5 = MEDIUM LOOP CONSTANT
.ENABL LSB
1\$: MOV R4,(R1)+ :V177640
MOV R4,(R1)+ :V177642
SOB R2,1\$:V177644
COM R4 :V177646
BIS (PC)+,R4 :V177650
WARN2: 401 :V177652 WARNING LOCATION IS MODIFIED BEFORE LOADING
MOV #4,R2 :V177654
SOB R5,1\$:V177660
COM R4 :V177662
BIS (PC)+,R4 :V177664
WARN3: 401 :V177666 WARNING LOCATION IS MODIFIED BEFORE LOADING
MOV #64,R5 :V177670
SOB R3,1\$:V177674
RETURN :V177676
.DSABL LSB

MTPB03: SUBTST <<MTPB03 3 XOR 9 WORST CASE NOISE TEST (READ)>>

:SUBTEST MTPB03 3 XOR 9 WORST CASE NOISE TEST (READ)

8974
8975 027554 000137 027614
8976 027560 077203
8977 027562 005104
8978 027564 052704
8979 027566 000401 000004
8980 027570 012702
8981 027574 077511
8982 027576 005104
8983 027600 052704
8984 027602 000401 000100
8985 027604 012705
8986 027610 077317
8987 027612 000207
8988

.ENABL LSB
1\$: JMP @MTPC03 :V177640 GO TO V172360
SOB R2,1\$:V177644
COM R4 :V177646
BIS (PC)+,R4 :V177650
WARN4: 401 :V177652 WARNING LOCATION IS MODIFIED BEFORE LOADING
MOV #4,R2 :V177654
SOB R5,1\$:V177660
COM R4 :V177662
BIS (PC)+,R4 :V177664
WARN5: 401 :V177666 WARNING LOCATION IS MODIFIED BEFORE LOADING
MOV #64,R5 :V177670
SOB R3,1\$:V177674
RETURN :V177676
.DSABL LSB

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 263
MTPB03 3 XOR 9 WORST CASE NOISE TEST (READ)

8991 027614

MTPC03: SUBTST <<MTPC03 TEST DATA SUBPROGRAM>>
:*****
:*SUBTEST MTPC03 TEST DATA SUBPROGRAM
:*****

8992 027614 020421
8993 027616 001401
8994 027620 104431
8995 027622 005141
8996 027624 005111
8997 027626 000137 027632
8998
8999 027632

1\$: CMP R4,(R1)+ :V172360
BEQ 1\$:V172362
PERR03 :V172364
COM -(R1) :V172366
COM (R1) :V172370
JMP @#MTPD03 :V172372 GO TO V172260

MTPD03: SUBTST <<MTPD03 TEST DATA SUBSUBPROGRAM>>
:*****
:*SUBTEST MTPD03 TEST DATA SUBSUBPROGRAM
:*****

9000 027632 020421
9001 027634 001401
9002 027636 104431
9003 027640 005127
9004 027642 000000
9005 027644 001363
9006 027646 000137 027560

1\$: CMP R4,(R1)+ :V172260
BEQ 1\$:V172262
PERR03 :V172264
COM (PC)+ :V172266
0 :V172270
BNE MTPC03 :V172272 GO TO V172360
JMP @#MTPB03+4 :V172274 GO TO V177644

9009 027652

MTPA04: SUBTST <<MTPA04 ROTATING ZEROS TEST>>
:*****
:*SUBTEST MTPA04 ROTATING ZEROS TEST
:*****

9010 027652 012705 000010
9011 027656 010504
9012 027660 000241
9013 027662 000137 027706
9014 027666 016004 177776
9015 027672 103402
9016 027674 020204
9017 027676 001401
9018 027700 104432
9019 027702 077115
9020 027704 000207
9021
9022 027706

1\$: MOV #8,R5 :V177640
MOV R5,R4 :V177644
CLC :V177646
JMP @#MTPB04 :V177650
MOV -2(R0),R4 :V177654
BCS 2\$:V177660
CMP R2,R4 :V177662
BEQ 3\$:V177664
2\$: PERR04 :V177666
3\$: SOB R1,1\$:V177670
RETURN :V177672

MTPB04: SUBTST <<MTPB04 SUBR ROTATING BIT>>
:*****
:*SUBTEST MTPB04 SUBR ROTATING BIT
:*****

9023 027706 106110
9024 027710 077502
9025 027712 106120
9026 027714 106110
9027 027716 077402
9028 027720 106120
9029 027722 000137 027666
9030
9031 027726

1\$: ROLB (R0) :V172360
SOB R5,1\$:V172362
ROLB (R0)+ :V172364
2\$: ROLB (R0) :V172366
SOB R4,2\$:V172370
ROLB (R0)+ :V172372
JMP @#MTPA04+14 :V172374

MTP005: SUBTST <<MTP005 ROTATION ONES TEST>>
:*****
:*SUBTEST MTP005 ROTATION ONES TEST
:*****

9032 027726 012705 000010
9033 027732 010504
9034 027734 000261
9035 027736 000137 027706
9036 027742 016004 177776
9037 027746 103002
9038 027750 020204
9039 027752 001401
9040 027754 104432
9041 027756 077115
9042 027760 000207

1\$: MOV #8,R5 :V177640
MOV R5,R4 :V177644
SEC :V177646
JMP @#MTPB04 :V177650
MOV -2(R0),R4 :V177654
BCC 2\$:V177660
CMP R2,R4 :V177662
BEQ 3\$:V177664
2\$: PERR04 :V177666
3\$: SOB R1,1\$:V177670
RETURN :V177672

IF THIS HAPPENS THE GOOD & BAD MATCH

9045 027762

```
MTP006: SUBTST <<MTP006 INITIAL DATA TEST>>
:*****
:SUBTEST MTP006 INITIAL DATA TEST
:*****
```

```

9046
9047
9048 027762 012737 000001 002240      MOV #1,DATBUF      :SET THE FIRST TEST BIT
9049 027770 005037 002242              CLR DATBUF+2      :CLEAR 2ND WORD
9050 027774 013771 002240 000000 1$:  MOV DATBUF,@(R1)  :WRITE TEST WORD 1
9051 030002 013771 002242 000002      MOV DATBUF+2,@2(R1) :AND TEST WORD 2
9052 030010 017102 000000              MOV @2(R1),R2
9053 030014 023702 002240              CMP DATBUF,R2      :NOW READ THEM
9054 030020 001401                      BEQ 2$             :BR IF FIRST 16 OK
9055 030022 104433                      PERR07            :ERROR TRAP
9056
9057 030024 017102 000002 2$:  MOV @2(R1),R2
9058 030030 023702 002242              CMP DATBUF+2,R2   :NOW READ SECOND WORD
9059 030034 001401                      BEQ 3$             :BR IF OK
9060 030036 104434                      PERR10            :ERROR TRAP
9061
9062 030040 005737 002242 3$:  TST DATBUF+2      :HAS LAST BIT BEEN TESTED ?
9063 030044 100405                      BMI 4$             :MINUS MEANS BIT 31
9064 030046                      DLEFT DATBUF      :NO, SHIFT TEST BIT LEFT
9065 030056 000746                      BR 1$              :GO WRITE NEW TEST DATA
9066
9067 030060 012737 177776 002240 4$:  MOV #177776,DATBUF :PUT A 0 IN BIT 0
9068 030066 012737 177777 002242      MOV #-1,DATBUF+2  :AND 1'S IN ALL OTHERS
9069 030074 013771 002240 000000 5$:  MOV DATBUF,@(R1)  :WRITE THE DATA
9070 030102 013771 002242 000002      MOV DATBUF+2,@2(R1) :2 WORDS WORTH
9071 030110 017102 000000              MOV @2(R1),R2
9072 030114 023702 002240              CMP DATBUF,R2     :NOW READ FIRST WORD
9073 030120 001401                      BEQ 6$             :BR IF OK
9074 030122 104433                      PERR07
9075
9076 030124 017102 000002 6$:  MOV @2(R1),R2
9077 030130 023702 002242              CMP DATBUF+2,R2   :NOW, READ SECOND WORD
9078 030134 001401                      BEQ 7$             :BR IF OK
9079 030136 104434                      PERR10
9080
9081 030140 005737 002242 7$:  TST DATBUF+2      :TESTED BIT 31 YET?
9082 030144 100005                      BPL 8$             :BR IF YES, WE'RE DONE
9083 030146                      DLEFT DATBUF
9084 030156 000746                      BR 5$              :KEEP GOING
9085 030160 000207 8$:  RETURN
```

9088 030162

```
MTP007: SUBTST <<MTP007 ADDRESS BIT TEST>>
:*****
:*SUBTEST MTP007 ADDRESS BIT TEST
:*****
```

```

9089
9090
9091
9092
9093 030162 111100
9094 030164 105700
9095 030166 001401
9096 030170 104435
9097
9098 030172 105111
9099 030174 111100
9100 030176 105700
9101 030200 001001
9102 030202 104436
9103
9104 030204 040201
9105 030206 006302
9106 030210 050201
9107 030212 011100
9108 030214 005700
9109 030216 001401
9110 030220 104437
9111
9112 030222 005111
9113 030224 011100
9114 030226 005700
9115 030230 001001
9116 030232 104440
9117
9118 030234 022702 100000
9119 030240 001407
9120 030242 022702 010000
9121 030246 001356
9122 030250 006302
9123 030252 012701 160000
9124 030256 000752
9125 030260 000207

:
:
:
:
MOV (R1),R0
TSTB R0
BEQ 1$
PERR11
:
:
:
:
1$: COMB (R1)
MOV (R1),R0
TSTB R0
BNE 2$
PERR12
:
:
:
:
2$: BIC R2,R1
ASL R2
BIS R2,R1
MOV (R1),R0
TST R0
BEQ 3$
PERR13
:
:
:
:
3$: COM (R1)
MOV (R1),R0
TST R0
BNE 4$
PERR14
:
:
:
:
4$: CMP #100000,R2
BEQ 5$
CMP #10000,R2
BNE 2$
ASL R2
MOV #160000,R1
BR 2$
:
:
:
:
5$: RETURN

THIS TEST CHECKS TO SEE THAT EACH ADDRESS
BIT IN EACH 16K BANK CAN BE ASSERTED UNIQUELY.
IT CHECKS FOR ADDRESS BITS THAT MAY BE STUCK
HIGH, STUCK LOW OR STUCK TOGETHER.

;READ AND COMPARE FOR ZEROS
;BR IF OK

;COMPLEMENT THE BYTE

;READ FOR NON ZEROS
;BR IF OK

;MASK OFF THE ASSERTED BIT
;SHIFT R2 FOR NEXT BIT
;SET THE NEW BIT INTO R1

;READ THE NEW ADDRESS
;READ FOR ZEROS

;COMPL THE WORD

;READ IT AGAIN

;CHECK FOR MSB IN 4K BANK
;NOT LAST BIT, BRANCH
```

9128 030262

MTP010: SUBTST <<MTP010 BYTE ADDRESSING TEST>>

:SUBTEST MTP010 BYTE ADDRESSING TEST

9129

9130

9131 030262 010402

9132 030264 010403

9133 030266 062702 000004

9134 030272 012713 177777

9135 030276 012763 177777 000002

9136 030304 105013

9137 030306 010401

9138 030310 020201

9139 030312 001420

9140 030314 020301

9141 030316 001007

9142 030320 111100

9143

9144 030322 022700 000000

9145 030326 001401

9146 030330 104435

9147

9148 030332 005201

9149 030334 000765

9150 030336 111100

9151 030340 122700 177777

9152 030344 001401

9153 030346 104436

9154

9155 030350 005201

9156 030352 000756

9157 030354 112713 177777

9158 030360 005203

9159 030362 020302

9160 030364 001347

9161 030366 000207

```

:TEST 3 THIS TEST CHECKS FOR PROPER
:BYTE ADDRESSING WITH ECC DISABLED
MOV R4,R2 ;R4 HAS LOWEST ADDRESS
MOV R4,R3 ;PUT IT IN R3 ALSO
ADD #4,R2 ;POINT R2 TO LAST BYTE +1
MOV #-1,(R3) ;WRITE ALL ONES IN
MOV #-1,2(R3) ;THE 4 TEST BYTES
1$: CLR (R3) ;CLEAR A BYTE
MOV R4,R1 ;INITIALIZE R1 FOR EACH PASS
2$: CMP R2,R1 ;IF EQUAL, JUST READ LAST BYTE
BEQ 6$ ;BR IF EQUAL
CMP R3,R1 ;IS THIS THE BYTE OF ZEROS
BNE 4$ ;BR IF NOT
MOVB (R1),R0
;WARNING IF YOU OPTIMIZE CHANGE THE PCBUMP FOR THIS ERROR INCASE OF TRAPS
CMP #0,R0 ;IT IS, COMPARE FOR ZEROS
BEQ 3$
PERR11
3$: INC R1 ;NEXT BYTE
BR 2$ ;RETURN
4$: MOVB (R1),R0
CMPB #-1,R0 ;ITS NOT THE BYTE OF 0'S, READ 1'S
BEQ 5$
PERR12
5$: INC R1 ;MOVE TO NEXT BYTE
BR 2$
6$: MOVB #-1,(R3) ;RESTORE 1'S TO BYTE JUST TESTED
INC R3 ;INC TO NEXT BYTE
CMP R3,R2 ;WAS THAT JUST THE LAST ONE?
BNE 1$ ;BR IF NO
RETURN

```


9164 030370

MTP011: SUBTST <<MTP011 SINGLE BIT ERROR TEST>>
:*****

:*SUBTEST MTP011 SINGLE BIT ERROR TEST
:*****

9165
9166
9167
9168
9169
9170
9171
9172
9173
9174
9175
9176
9177
9178
9179
9180
9181
9182
9183
9184
9185
9186
9187
9188
9189
9190
9191
9192
9193
9194
9195
9196
9197
9198
9199
9200
9201
9202
9203
9204
9205
9206
9207
9208
9209
9210
9211
9212
9213
9214
9215
9216
9217

030370 104503
030372 005737 014006
030376 001407
030400 013702 172246
030404 013737 172252 172246
030412 010237 172252
030416 012737 000001 002240
030424 005037 002242
030430 012737 000001 002250
030436 005037 002252
030442 013737 002240 002244
030450 013737 002242 002246
030456 105737 002262
030462 001404
030464 005137 002244
030470 005137 002246
030474 013702 002244
030500 013703 002246
030504 012737 002244 002310
030512 004737 044400
030516 013701 002250
030522 074137 002244
030526 013701 002252
030532 074137 002246
030536 013701 002410
030542 013705 002412
030546 104471
030550 013711 002244
030554 104475
030556 013715 002246

: (1) CREATE A SINGLE BIT ERROR
: (2) READ BACK SBE UNCORRECTED (WITH ECC DISABLE)
: (3) ENABLE ECC & READ CORRECTED DATA
: (4) CHECK THAT THE SBE FLAG WAS SET FROM THE LAST READ
: (5) DO (1-4) FOR DATA CONSISTING OF 1 BIT SET IN EACH OF 32
: POSITIONS OF A DOUBLE WORD
: THEN DO IT AGAIN FOR 1 BIT CLEARED IN EACH OF 32 POSITIONS OF
: A DOUBLE WORD
: IE (64 TIMES)
: (6) DO (1-5) FOR A SBE IN EACH OF 32 BIT POSITIONS
: IE (RUN TEST 64 * 32 = 2048 TIMES)
CLR1CSR :CLEAR 1 SELECTED CSR
TST PHEBE :TEST SPECIAL CASE INDICATOR
BEQ MTLA11 :BRANCH IF NOT SET
MOV SIPAR3,R2 :SAVE CONTENTS OF SIPAR #3
MOV SIPAR5,@SIPAR3 :COPY CONTENTS OF #5 INTO #3
MOV R2,@SIPAR5 :COPY CONTENTS OF #3 INTO #5
:BIG LOOP
MTLA11: MOV #1,DATBUF :INITIAL DATA
CLR DATBUF+2 :32 BITS WORTH
:MEDIUM LOOP
MTLB11: MOV #1,SBEMSK :INITIAL ERROR MASK
CLR SBEMSK+2 :32 BITS WORTH
:LITTLE LOOP
MTLC11: MOV DATBUF,TSTDAT :
MOV DATBUF+2,TSTDAT+2 :TO SAVE ORIG DATA
TSTB PASFLG :COMP DATA ON SECOND PASS ONLY
BEQ 4\$:BR IF FIRST PASS
COM TSTDAT :SECOND PASS, COMP BOTH WORDS
COM TSTDAT+2
4\$: MOV TSTDAT,R2
MOV TSTDAT+2,R3
MOV #TSTDAT,SOURCE :SET UP ADDRESS FOR CHKGEN
CALL CHKGEN :GEN CHECKBITS ON TSTDAT
:*****
:** CREATE A SINGLE BIT ERROR **
:*****
MOV SBEMSK,R1
XOR R1,TSTDAT
MOV SBEMSK+2,R1
XOR R1,TSTDAT+2
MTLD11: MOV TESTADD,R1 :FIRST TEST ADDRESS
MOV TESTADD+2,R5 :SECOND TEST ADDRESS
ECC1DIS :DISABLE ECC ON 1 SELECTED CSR
MOV TSTDAT,(R1) :WRITE FIRST 16 BITS
CB1CSR :WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
MOV TSTDAT+2,(R5) :WRITE SECOND 16 BITS AND
:CHECK BITS. WE NOW HAVE CHECKBITS

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 273-2
 MTP011 SINGLE BIT ERROR TEST

9275	031022					IF OVFLAG IS TRUE THEN GOTO 13\$
9276	031030	005737	002242			TST DATBUF+2 ;LAST DATA BIT ?
9277	031034	100406				BMI 13\$;WHICH IS BIT 31
9278	031036					DLEFT DATBUF
9279	031046	000137	030430			JMP MTLB11
9280	031052	105737	002262	13\$:		TSTB PASFLG ;FIRST OR SECOND PASS ?
9281	031056	001004				BNE 14\$;NON ZERO MEANS WE'RE DONE
9282	031060	105237	002262			INCB PASFLG ;NOT DONE, GO DO SECOND PASS
9283	031064	000137	030416			JMP MTLA11
9284	031070	052737	000200	002262	14\$:	BIS #BIT7,PASFLG
9285	031076	005002				CLR R2
9286	031100	005003				CLR R3
9287	031102	005037	002244			CLR TSTDAT
9288	031106	005037	002246			CLR TSTDAT+2
9289	031112	012704	000040			MOV #40,R4
9290	031116	012737	003740	002312	15\$:	MOV #3740,CHECK
9291	031124	074437	002312			XOR R4,CHECK
9292	031130	006304				ASL R4
9293	031132	032704	020000			BIT #BIT13,R4
9294	031136	001002				BNE 16\$
9295	031140	000137	030536			JMP MTLD11
9296						;CLEAR OUT ANY DBE'S OR SBE'S
9297	031144	104471		16\$:		ECC1DIS ;DISABLE ECC ON 1 SELECTED CSR
9298	031146	013701	002410			MOV TESTADD,R1
9299	031152	013705	002412			MOV TESTADD+2,R5
9300	031156					CLEAR (R1),(R5)
9301	031162	104503				CLR1CSR ;CLEAR 1 SELECTED CSR
9302	031164	000207				RETURN

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 275
MTP011 SINGLE BIT ERROR TEST

9305 031166

MTP012: SUBTST <<MTP012 WRITE BYTE CLEARS SBE TEST>>
:*****
: *SUBTEST MTP012 WRITE BYTE CLEARS SBE TEST
:*****

```

9306                                     :SINGLE BIT ERROR TEST TO INSURE THAT A WRITE
9307                                     :BYTE CLEARS SINGLE BIT ERRORS.
9308 031166 104503 CLR1CSR                               :CLEAR 1 SELECTED CSR
9309 031170 012737 000001 002240 MOV #1,DATBUF           :INITIAL DATA
9310 031176 005037 002242 CLR DATBUF+2           :32 BITS WORTH
9311 031202 012737 000001 002250 1$: MOV #1,SBEMSK      :INITIAL ERROR MASK
9312 031210 005037 002252 CLR SBEMSK+2         :32 BITS WORTH
9313 031214 013737 002240 002244 2$: MOV DATBUF,TSTDAT  :SAVE ORIGINAL DATA
9314 031222 013737 002242 002246 MOV DATBUF+2,TSTDAT+2 :BOTH WORDS
9315 031230 012737 002244 002310 MOV #TSTDAT,SOURCE  :NEED ADDRESS FOR CHKGEN
9316 031236 004737 044400 CPL CHKGEN           :GENERATE CHECK BITS
9317 031242 013701 002250 MOV SBEMSK,R1
9318 031246 074137 002244 XOR R1,TSTDAT
9319 031252 013701 002252 MOV SBEMSK+2,R1
9320 031256 074137 002246 XOR R1,TSTDAT+2
9321 031262 013704 002410 MOV TESTADD,R4      :FIRST TEST ADDRESS
9322 031266 010401 MOV R4,R1                   :PUT IT IN R1 ALSO
9323 031270 104471 ECC1DIS                       :DISABLE ECC ON 1 SELECTED CSR
9324 031272 013711 002244 MOV TSTDAT,(R1)       :WRITE 16 BITS
9325 031276 104475 CB1CSR                       :WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
9326 031300 060501 ADD R5,R1                   :INDEX UP TO SECOND WORD
9327 031302 013711 002246 MOV TSTDAT+2,(R1)     :WRITE HIGH WORD+CHECKBITS
9328 031306 104503 CLR1CSR                       :CLEAR 1 SELECTED CSR
9329                                     :IT'S DANGEROUS IF WE DON'T
9330 031310 012702 002250 MOV #SBEMSK,R2        :ADDRESS OF ERROR MASK
9331 031314 160501 SUB R5,R1                   :RETURN TO FIRST WORD
9332 031316 112711 177777 3$: MOVB #-1,(R1)      :WRITE A BYTE OF 1'S
9333 031322 005737 002526 TST KFLAG          :IS THIS MF11S-K
9334 031326 001403 BEQ 4$                       :BRANCH IF NOT - IT'S MS11-M
9335 031330 132712 177777 BITB #-1,(R2)        :DID THIS BYTE HAVE THE BAD BIT IN IT?
9336 031334 001420 BEQ 6$                       :NO - BRANCH
9337 031336 104510 4$: TSTREAD                 :TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
9338 031340 103011 BCC 5$                       :NO - SKIP
9339 031342 SET HEADER                          :ENABLE PRINTING OF ERROR HEADER INFO
9340 031350 010137 002032 MOV R1,ADDRESS
9341 031354 104017 ERROR +17
9342 031356 SET HEADER                          :ENABLE PRINTING OF ERROR HEADER INFO
9343
9344 031364 111100 5$: MOVB (R1),R0
9345 031366 122700 177777 CMPB #-1,R0          :CHECK DATA
9346 031372 001414 BEQ 7$                       :BR IF OK
9347 031374 104457 PERR33
9348
9349 031376 104510 6$: TSTREAD                 :TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
9350                                     :READ THE BYTE
9351                                     :SBE ERROR BIT ONLY SET ?
9352 031400 103771 BCS 5$                       :SHOULD BE SET, BR IF OK
9353 031402 SET HEADER                          :ENABLE PRINTING OF ERROR HEADER INFO
9354 031410 010137 002032 MOV R1,ADDRESS
9355 031414 104460 PERR34
9356 031416 SET HEADER                          :ENABLE PRINTING OF ERROR HEADER INFO
9357
9358 031424 132712 177777 7$: BITB #-1,(R2)    :CHECK FOR LAST BYTE

```

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 275-1
 MTP012 WRITE BYTE CLEARS SBE TEST

9359	031430	001012		BNE	8\$:
9360	031432	005202		INC	R2	
9361	031434	005201		INC	R1	:MOVE TO NEXT BYTE
9362	031436	013704	002410	MOV	TESTADD,R4	:FIRST TEST ADDRESS
9363	031442	032701	000002	BIT	#2,R1	:TEST FOR LOWER WORD
9364	031446	001723		BEQ	3\$:BR IF IT'S LOW 16 BITS
9365	031450	062704	000002	ADD	#2,R4	:ADJUST POINTER FOR ERROR REPT.
9366	031454	000720		BR	3\$	
9367	031456	005737	002252	8\$: TST	SBEMSK+2	:LAST ERROR BIT ?
9368	031462	100405		BMI	9\$:MINUS MEANS BIT 31
9369	031464			DLEFT	SBEMSK	
9370	031474	000647		BR	2\$	
9371	031476			9\$: IF #SW11 SET. IN @SWR THEN GOTO 10\$		
9372	031506			IF QVFLAG IS TRUE THEN GOTO 10\$		
9373	031514	005737	002242	TST	DATBUF+2	:LAST DATA BIT?
9374	031520	100405		BMI	10\$:MINUS = BIT 31
9375	031522			DLEFT	DATBUF	
9376	031532	000623		BR	1\$	
9377				:CLEAR OUT ANY DBE'S OR SBE'S		
9378	031534	104471		10\$: ECC1DIS		:DISABLE ECC ON 1 SELECTED CSR
9379	031536	013701	002410	MOV	TESTADD,R1	
9380	031542	005011		CLR	(R1)	
9381	031544	060501		ADD	R5,R1	
9382	031546	005011		CLR	(R1)	
9383	031550	104503		CLR1CSR		:CLEAR 1 SELECTED CSR
9384	031552	000207		RETURN		

9387 031554

```

MTP013: SUBTST <<MTP013          CREATE DOUBLE BIT ERROR TEST>>
:*****
:*SUBTEST      MTP013  CREATE DOUBLE BIT ERROR TEST
:*****
;DOUBLE BIT ERROR FORCE TO CHECK DOUBLE ERROR LOGIC
CLR1CSR          ;CLEAR 1 SELECTED CSR
MOV #TESTADD,R1
CLR DATBUF       ;MAKE INITIAL DATA
CLR DATBUF+2     ;ALL ZEROS
MOV #1,SBEMSK    ;INITIAL SINGLE ERROR MASK
CLR SBEMSK+2     ;SECOND WORD
MOV #1,DBEMSK    ;INITIAL DOUBLE ERROR MASK
CLR DBEMSK+2     ;32 BITS HERE ALSO
MOV DATBUF,TSTDAT
MOV DATBUF+2,TSTDAT+2
TSTB PASFLG     ;NO COMPLEMENTING FIRST PASS
BEQ 5$
COM TSTDAT      ;COMP FIRST WORD
COM TSTDAT+2    ;SECOND WORD
CLR1CSR          ;CLEAR 1 SELECTED CSR
CMP SBEMSK,DBEMSK ;CAN'T HAVE THE SAME ERROR BIT SET
BNE 6$         ;IN BOTH MASKS
CMP SBEMSK+2,DBEMSK+2 ;COULD BE EQUAL IN SECOND WORD
BEQ 13$        ;GO MAKE THEM NOT EQUAL
MOV #TSTDAT,SOURCE ;SOURCE ADDRESS FOR CHKGEN
CALL CHKGEN     ;GO GENERATE CHECK BITS
MOV SBEMSK,R2
XOR R2,TSTDAT
MOV SBEMSK+2,R2
XOR R2,TSTDAT+2
MOV DBEMSK,R2
XOR R2,TSTDAT
MOV DBEMSK+2,R2
XOR R2,TSTDAT+2
16$: ECC1DIS     ;DISABLE ECC ON 1 SELECTED CSR
MOV TSTDAT,@(R1)+ ;WRITE 16 BITS
CB1CSR          ;WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
MOV TSTDAT+2,@(R1) ;WRITE HIGH WORD
CLR1CSR          ;CLEAR 1 SELECTED CSR
SUB #2,R1       ;ADJUST TEST ADDRESS
TST @(R1)       ;READ THE LOCATION
WAS1DBE        ;WAS THERE ANY DOUBLE BIT ERRORS ON 1 SELECTED CSR
BCS 9$         ;IT SHOULD BE SET
SET HEADER
MOV (R1),ADDRESS
ERROR +30
SET HEADER
    
```

9388
9389 031554 104503 002410
9390 031556 012701 002240
9391 031562 005037 002240
9392 031566 005037 002242
9393 031572 012737 000001 002250
9394 031600 005037 002252
9395 031604 012737 000001 002254
9396 031612 005037 002256
9397 031616 013737 002240 002244
9398 031624 013737 002242 002246
9399 031632 105737 002262
9400 031636 001404
9401 031640 005137 002244
9402 031644 005137 002246
9403 031650 104503
9404 031652 023737 002250 002254
9405 031660 001004
9406 031662 023737 002252 002256
9407 031670 001460
9408 031672 012737 002244 002310
9409 031700 004737 044400
9410 031704 013702 002250
9411 031710 074237 002244
9412 031714 013702 002252
9413 031720 074237 002246
9414 031724 013702 002254
9415 031730 074237 002244
9416 031734 013702 002256
9417 031740 074237 002246
9418 031744 104471
9419 031746 013731 002244
9420 031752 104475
9421 031754 013771 002246 000000
9422 031762 104503
9423 031764 162701 000002
9424 031770 005771 000000
9425 031774 104501
9426 031776 103411
9427 032000
9428 032006 011137 002032
9429 032012 104030
9430 032014

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 279
 MTP013 CREATE DOUBLE BIT ERROR TEST

```

9433 032022 104512          9$:  ERRGEN
9434 032024 105737 002262    TSTB  PASFLG
9435 032030 100452          BMI    14$
9436 032032 005737 002256    13$:  TST  DBEMSK+2      ;CHECK MASK FOR LAST BIT
9437 032036 100405          BMI    10$           ;MINUS = BIT31
9438 032040                DLEFT  DBEMSK
9439 032050 000662          BR     4$
9440 032052                10$:  IF #SW11 SET.IN @SWR THEN GOTO 11$
9441 032062                IF QVFLAG IS TRUE THEN GOTO 11$
9442 032070 005737 002252    TST  SBEMSK+2      ;CHECK SINGLE ERROR MASK TOO
9443 032074 100405          BMI    11$           ;BR IF DONE
9444 032076                DLEFT  SBEMSK
9445 032106 000636          BR     3$
9446 032110 105737 002262    11$:  TSTB  PASFLG ;FIRST PASS
9447 032114 001003          BNE    12$           ;NON ZERO MEANS WE'RE DONE
9448 032116 105237 002262    INCB  PASFLG ;FIRST PASS, NOT DONE
9449                ;CLEAR OUT ANY DBE'S OR SBE'S
9450 032122 000617          BR     1$           ;KEEP GOING
9451 032124 052737 000200 002262 12$:  BIS   #BIT7,PASFLG ;SET UP FOR CHECK BIT TEST
9452 032132 005037 002244    CLR   TSTDAT
9453 032136 005037 002246    CLR   TSTDAT+2
9454 032142 012737 000040 002250  MOV   #40,SBEMSK
9455 032150 012737 000100 002254  MOV   #100,DBEMSK
9456 032156 012737 003740 002312 14$:  MOV   #3740,CHECK
9457 032164 013702 002250    MOV   SBEMSK,R2
9458 032170 074237 002312    XOR   R2,CHECK
9459 032174 013702 002254    MOV   DBEMSK,R2
9460 032200 074237 002312    XOR   R2,CHECK
9461 032204 006337 002254    ASL   DBEMSK
9462 032210 032737 020000 002254  BIT   #BIT13,DBEMSK
9463 032216 001652          BEQ   16$
9464 032220 006337 002250    ASL   SBEMSK
9465 032224 032737 004000 002250  BIT   #BIT11,SBEMSK
9466 032232 001006          BNE   15$
9467 032234 013737 002250 002254  MOV   SBEMSK,DBEMSK
9468 032242 006337 002254    ASL   DBEMSK
9469 032246 000743          BR     14$
9470 032250 104471          15$:  ECC1DIS ;DISABLE ECC ON 1 SELECTED CSR
9471 032252 012701 002410    MOV   #TESTADD,R1
9472 032256                CLEAR  @R1+,@R1
9473 032264 104503          CLR1CSR ;CLEAR 1 SELECTED CSR
9474 032266 000207          RETURN

```

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 282
 MTP013 CREATE DOUBLE BIT ERROR TEST

9478 032270

MTP014: SUBTST <<MTP014 BASIC DOUBLE BIT ERROR TEST>>

9479

9480

9481

9482

9483

9484 032270 104424

9485 032272

9486 032276

9487 032304

9488 032312 104513

9489 032314

9490 032322 104425

9491 032324

9492 032332

9493 032334 005711

9494 032336

9495 032346 104055

9496 032350

9497 032350 104426

9498 032352 042737 020000 002146

9499 032360

9500 032370

9501 032376

9502 032404 104065

9503 032406

9504 032406 104473

9505 032410 005037 002264

9506 032414

9507 032414 104473

9508 032416 005237 002264

9509 032422 005037 002070

9510 032426

9511 032432 105711

9512 032434

9513 032444

9514 032452

9515 032456

9516 032464 104056

9517 032466

9518 032466 005201

9519 032470

9520 032500 005041

9521 032502 104503

9522 032504 005037 002070

9523 032510 104423

9524 032512 000207

```
*****
:SUBTEST      MTP014  BASIC DOUBLE BIT ERROR TEST
*****
```

```
: THIS TEST CHECKS THAT A DOUBLE ERROR WILL BE DETECTED
: A BYTE WRITE WITH A DOUBLE ERROR ON A MS11-P
: WILL BE ABORTED.
```

```
CACHOFF          :TURN OFF CACHE
LET PARCNT := #0  :CLEAR PARCNT
LET NOPAR := #1   :SET PARITY ACTION
LET ADDRESS := #1:RST :SET ADDRESS FOR ERROR REPORT
CBREG            :ENABLE CHECK/SYNDROME BIT REGISTER
LET CSR := #3145  :DBE CHECK BITS FOR CSR
LOADCSR         :WRITE DBE CHECK BITS TO CSR
LET GOOD := #103145 :GOOD DATA
LET (R1) := #0    :WRITE ZEROS AND DBL ERROR CHK BITS A=0
TST (R1)         :READ A=0 TO GET DOUBLE BIT ERROR
IF PARCNT NE #1  :WAS BUSPBL ASSERTED???
    ERROR +55     :ERROR CALL ; MISSED EXPECTED TRAP
END              :
READCSR          :READ CSR FOR CORRECT CHECK BITS AND DBE INDICATOR
BIC #BIT13,CSR   :CLEAR INHIBIT MODE POINTER FROM DATA IF IT EXISTS1
IF CSR NE GOOD THEN :CHECK IF DOUBLE ERROR BIT IS SET
    SET HEADER    :
    LET BAD := CSR :BAD DATA
    ERROR +65     :
END              :
ECC1INIT         :ENABLE BUSPBL
CLR PASSNO       :CLEAR LOOP COUNTER
REPEAT          :
    ECC1INIT     :ENABLE BUSPBL
    INC PASSNO   :INCREMENT LOOP COUNTER
    CLR PARCNT   :CLEAR PARITY ACTION COUNTER
    LET (R1) := #377 :WRITE BYTE SHOULD BE ABORTED
    TSTB (R1)    :READ R1 TO SEE IF IT IS STILL 0
    IF PARCNT NE #1 :WAS WRITE ABORTED???
        SET HEADER :
        LET GOOD := #0 :GOOD DATA
        LET BAD := #377 :BAD DATA
        ERROR +56     :
    END          :
    INC R1       :AND REPEAT ON HIGH BYTE
UNTIL PASSNO EQ #2 :
CLR -(R1)        :CLEAR LUT
CLR1CSR          :CLEAR CSR
CLR PARCNT       :CLEAR PARITY TRAP COUNTER
CACHON           :TURN ON CACHE
RETURN           :
```

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 283
 MTP014 BASIC DOUBLE BIT ERROR TEST

9526 032514

```

MTP015: SUBTST <<MTP015 WRITE INHIBIT OF BYTE WITH DBE>>
:*****
:*SUBTEST MTP015 WRITE INHIBIT OF BYTE WITH DBE
:*****
:CHECK FOR WRITE INHIBIT DURING A WRITE BYTE.
:CHECKS FOR UNCORRECTED DATA.
1$: CLR DATBUF ;INITIAL DATA
CLR DATBUF+2 ;32 BITS WORTH
2$: MOV #1,SBEMSK ;SINGLE ERROR MASK
CLR SBEMSK+2 ;
3$: MOV #1,DBEMSK ;DOUBLE ERROR MASK
CLR DBEMSK+2 ;
4$: MOV DATBUF,TSTDAT ;PRESERVE ORIG DATA
MOV DATBUF+2,TSTDAT+2 ;
TSTB PASFLG ;WHICH PASS ?
BEQ 5$ ;FIRST PASS, NO COMPLEMENTING
COM TSTDAT
COM TSTDAT+2 ;SECOND PASS, COMPLEMENT TSTDAT
5$: CLR1CSR ;CLEAR 1 SELECTED CSR
CMP SBEMSK,DBEMSK ;CHECK FOR SAME MASKS
BNE 6$ ;BR IF NOT EQUAL
CMP SBEMSK+2,DBEMSK+2 ;SECOND WORD ALSO
BEQ 11$ ;BR TO MAKE THEM NOT EQUAL
6$: MOV #TSTDAT,SOURCE ;ADDRESS FOR CHKGEN
CALL CHKGEN ;GO GENERATE CHECK BITS
MOV SBEMSK,R1
XOR R1,TSTDAT
MOV SBEMSK+2,R1
XOR R1,TSTDAT+2
MOV DBEMSK,R1
XOR R1,TSTDAT
MOV DBEMSK+2,R1
XOR R1,TSTDAT+2
7$: MOV #TESTADD,R1 ;TEST LOCATION
ECC1DIS ;DISABLE ECC ON 1 SELECTED CSR
MOV TSTDAT,@(R1)+ ;WRITE FIRST 16 BITS
;LOAD CSR WITH IMAGE FROM R2
CB1CSR ;WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
MOV TSTDAT+2,@(R1) ;WRITE SECOND 16 BITS + CHECKBITS
CLR1CSR ;CLEAR 1 SELECTED CSR
MOV TESTADD,R2 ;GET ADDRESS OF TEST LOC
MOV R2,R3 ;R2 DESIGNATES FIRST BYTE
ADD #3,R3 ;R3 DESIGNATES LAST BYTE
8$: MOVB #360,(R2)+ ;TRY WRITING A BYTE
MOV #TESTADD,R1
MOV @(R1),R0
CMP TSTDAT,R0 ;CHECK FOR UNCHANGED DATA
BEQ 9$ ;BR IF OK
MOV @(R1),ADDRESS
PERR31
9$: MOV @2(R1),R0
CMP TSTDAT+2,R0 ;READ SECOND WORD
BEQ 10$ ;BR IF UNCHANGED
MOV @2(R1),ADDRESS
PERR31

```

9527

9528

9529 032514 005037 002240

9530 032520 005037 002242

9531 032524 012737 000001 002250

9532 032532 005037 002252

9533 032536 012737 000001 002254

9534 032544 005037 002256

9535 032550 013737 002240 002244

9536 032556 013737 002242 002246

9537 032564 105737 002262

9538 032570 001404

9539 032572 005137 002244

9540 032576 005137 002246

9541 032602 104503

9542 032604 023737 002250 002254

9543 032612 001004

9544 032614 023737 002252 002256

9545 032622 001474

9546 032624 012737 002244 002310

9547 032632 004737 044400

9548 032636 013701 002250

9549 032642 074137 002244

9550 032646 013701 002252

9551 032652 074137 002246

9552 032656 013701 002254

9553 032662 074137 002244

9554 032666 013701 002256

9555 032672 074137 002246

9556 032676 012701 002410

9557 032702 104471

9558 032704 013731 002244

9559

9560 032710 104475

9561 032712 013771 002246 000000

9562 032720 104503

9563 032722 013702 002410

9564 032726 010203

9565 032730 062703 000003

9566 032734 112722 000360

9567 032740 012701 002410

9568 032744 017100 000000

9569 032750 023700 002244

9570 032754 001404

9571 032756 017137 000000 002032

9572 032764 104455

9573

9574 032766 017100 000002

9575 032772 023700 002246

9576 032776 001404

9577 033000 017137 000002 002032

9578 033006 104455

9579

CZMSPB0 MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 283-1
 MTP015 WRITE INHIBIT OF BYTE WITH DBE

```

9580 033010 020203          10$:  CMP      R2,R3          ;TESTED LAST BYTE ?
9581 033012 001350          BNE      8$              ;BR IF NO
9582 033014 105737 002262    11$:  TSTB    PASFLG
9583 033020 100452          BMI      15$            ;BRANCH IF TESTING CHECK BITS
9584 033022 005737 002256    TST     DBEMSK+2        ;CHECKING FOR LAST ERROR BIT
9585 033026 100405          BMI      12$            ;BR IF DONE HERE
9586 033030
9587 033040 000643          DLEFT   DBEMSK
9588 033042          BR       4$
9589 033052          12$:  IF #SW11 SET.IN @SWR THEN GOTO 13$
9590 033060 005737 002252    IF QVFLAG IS TRUE THEN GOTO 13$
9591 033064 100405          TST     SBEMSK+2        ;LAST SBE MASK
9592 033066          BMI      13$            ;BR IF DONE WITH THIS PASS
9593 033076 000617          DLEFT   SBEMSK
9594 033100 105737 002262    BR       3$
9595 033104 001003          13$:  TSTB    PASFLG ;TEST PASS FLAG
9596 033106 105237 002262    BNE     14$            ;NON ZERO MEANS WE'RE DONE
9597 033112 000600          INCB    PASFLG ;NOT DONR
9598 033114 052737 000200 002262 14$:  BR       1$
9599 033122 005037 002244    BIS     #BIT7,PASFLG
9600 033126 005037 002246    CLR     TSTDAT
9601 033132 012737 000040 002250  CLR     TSTDAT+2
9602 033140 012737 000100 002254  MOV     #40,SBEMSK
9603 033146 012737 003740 002312 15$:  MOV     #100,DBEMSK
9604 033154 013702 002250    MOV     #3740,CHECK
9605 033160 074237 002312    MOV     SBEMSK,R2
9606 033164 013702 002254    XOR     R2,CHECK
9607 033170 074237 002312    MOV     DBEMSK,R2
9608 033174 006337 002254    XOR     R2,CHECK
9609 033200 032737 020000 002254  ASL     DBEMSK
9610 033206 001633          BIT     #BIT13,DBEMSK
9611 033210 006337 002250    BEQ     7$
9612 033214 032737 004000 002250  ASL     SBEMSK
9613 033222 001006          BIT     #BIT11,SBEMSK
9614 033224 013737 002250 002254  BNE     16$
9615 033232 006337 002254    MOV     SBEMSK,DBEMSK
9616 033236 000743          ASL     DBEMSK
9617 033240 104471          BR       15$
9618 033242 012701 002410    16$:  ECC1DIS ;DISABLE ECC ON 1 SELECTED CSR
9619 033246          MOV     #TESTADD,R1 ;TEST LOCATION
9620          CLEAR @ (R1)+,@ (R1) ;TO ERASE ANY DBE'S FROM TESTING
9621 033254 104503          ;RESTORE CSR
9622 033256 000207    CLR1CSR ;CLEAR 1 SELECTED CSR
          RETURN

```

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 285
MTP015 WRITE INHIBIT OF BYTE WITH DBE

9625 033260

MTP016: SUBTST <<MTP016 WRITE INHIBIT OF WORD WITH DBE>>
:*****
: *SUBTEST MTP016 WRITE INHIBIT OF WORD WITH DBE
:*****

```

9626
9627
9628
9629
9630 033260 005037 002240
9631 033264 005037 002242
9632 033270 012737 000001 002250
9633 033276 005037 002252
9634 033302 012737 000001 002254
9635 033310 005037 002256
9636 033314 013737 002240 002244
9637 033322 013737 002242 002246
9638 033330 105737 002262
9639 033334 001404
9640 033336 005137 002244
9641 033342 005137 002246
9642 033346 023737 002250 002254
9643 033354 001004
9644 033356 023737 002252 002256
9645 033364 001502
9646 033366 012737 002244 002310
9647 033374 004737 044400
9648 033400 013701 002250
9649 033404 074137 002244
9650 033410 013701 002252
9651 033414 074137 002246
9652 033420 013701 002254
9653 033424 074137 002244
9654 033430 013701 002256
9655 033434 074137 002246
9656 033440 012701 002410
9657 033444 104471
9658 033446 013731 002244
9659 033452 104475
9660 033454 013771 002246 000000
9661 033462 105037 002263
9662 033466 162701 000002
9663 033472 104503
9664 033474 012771 177400 000000
9665 033502 012701 002410
9666 033506 017100 000000
9667 033512 023700 002244
9668 033516 001404
9669 033520 017137 000000 002032
9670 033526 104455
9671
9672 033530 062701 000002
9673 033534 017100 000000
9674 033540 023700 002246
9675 033544 001404
9676 033546 017137 000000 002032
9677 033554 104455

```

```

T12A: CLR DATBUF ;BACKGROUND FOR DOUBLE ERRORS
CLR DATBUF+2 ;2 WORDS WORTH
MOV #1,SBEMSK ;SINGLE ERROR MASK
CLR SBEMSK+2
T12B: MOV #1,DBEMSK ;DOUBLE ERROR MASK
CLR DBEMSK+2
1$: MOV DATBUF,TSTDAT ;DATA FOR TEST
MOV DATBUF+2,TSTDAT+2 ;BOTH WORDS
TSTB PASFLG ;COMP DATA ON SECOND PASS ONLY
BEQ 2$ ;BR IF FIRST PASS
COM TSTDAT ;COMP FIRST WORD
COM TSTDAT+2 ;NOW SECOND WORD
2$: CMP SBEMSK,DBEMSK ;CHECK FOR IDENTICAL MASKS
BNE 3$ ;BR IF DIFFERENT
CMP SBEMSK+2,DBEMSK+2 ;UPPER WORD TOO
BEQ 8$ ;BR TO MAKE THEM NOT EQUAL
3$: MOV #TSTDAT,SOURCE ;NEED ADDR OF DATA FOR CHKGEN
CALL CHKGEN ;GO GENERATE CHECK BITS
MOV SBEMSK,R1
XOR R1,TSTDAT
MOV SBEMSK+2,R1
XOR R1,TSTDAT+2
MOV DBEMSK,R1
XOR R1,TSTDAT
MOV DBEMSK+2,R1
XOR R1,TSTDAT+2
4$: MOV #TESTADD,R1 ;FIRST TEST ADDRESS
ECC1DIS ;DISABLE ECC ON 1 SELECTED CSR
MOV TSTDAT,@(R1)+ ;WRITE FIRST 16 BITS
CB1CSR ;WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
MOV TSTDAT+2,@(R1) ;WRITE SECOND 16 BITS + CHECKBITS
CLRB UPPFLG ;SET FOR 2 LOOPS
SUB #2,R1 ;POINT TO LOW WORD
5$: CLR1CSR ;CLEAR 1 SELECTED CSR
MOV #177400,@(R1) ;TRY WRITING LOCATION
MOV #TESTADD,R1
MOV @(R1),R0
CMP TSTDAT,R0 ;CHECK FOR ORIGINAL DATA
BEQ 6$ ;SHOULD BE UNCHANGED
MOV @(R1),ADDRESS
PERR31
6$: ADD #2,R1
MOV @(R1),R0
CMP TSTDAT+2,R0 ;THIS SHOULD BE UNCHANGED ALSO
BEQ 7$
MOV @(R1),ADDRESS
PERR31

```

CZMSP80 MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 287
 MTP016 WRITE INHIBIT OF WORD WITH DBE

```

9680 033556 105737 002263      7$:  TSTB  UPPFLG      :WHICH LOOP ?
9681 033562 001003              BNE    8$         :SECOND, BR OUT
9682 033564 105237 002263      INCB  UPPFLG      :FIRST, KEEP GOING
9683 033570 000740              BR    5$
9684 033572 105737 002262      8$:  TSTB  PASFLG
9685 033576 100454              BMI   12$
9686 033600 005737 002256      TST  DBEMSK+2    :LAST BIT ?
9687 033604 100405              BMI   9$         :MINUS = BIT 31
9688 033606
9689 033616 000636              DLEFT DBEMSK
9690 033620
9691 033630
9692 033636 005737 002252      9$:  BR    1$
          IF #SW11 SET.IN @SWR THEN GOTO 10$
          IF QVFLAG IS TRUE THEN GOTO 10$
9693 033642 100406              TST  SBEMSK+2    :LAST BIT IN THIS MASK ?
9694 033644              BMI   10$       :BR IF LAST BIT
9695 033654 000137 033302      DLEFT SBEMSK
9696 033660 105737 002262      10$: JMP    T12B
          TSTB  PASFLG ;FIRST PASS ?
9697 033664 001004              BNE   11$       :BR IF SECOND
9698 033666 105237 002262      INCB  PASFLG ;INDICATE SECOND PASS COMING
9699 033672 000137 033260      JMP    T12A
9700 033676 052737 000200 002262 11$:  BIS   #BIT7,PASFLG
9701 033704 005037 002244      CLR   TSTDAT
9702 033710 005037 002246      CLR   TSTDAT+2
9703 033714 012737 000040 002250      MOV   #40,SBEMSK
9704 033722 012737 000100 002254      MOV   #100,DBEMSK
9705 033730 012737 003740 002312 12$:  MOV   #3740,CHECK
9706 033736 013702 002250      MOV   SBEMSK,R2
9707 033742 074237 002312      XOR   R2,CHECK
9708 033746 013702 002254      MOV   DBEMSK,R2
9709 033752 074237 002312      XOR   R2,CHECK
9710 033756 006337 002254      ASL   DBEMSK
9711 033762 032737 020000 002254      BIT   #BIT13,DBEMSK
9712 033770 001623              BEQ   4$
9713 033772 006337 002250      ASL   SBEMSK
9714 033776 032737 004000 002250      BIT   #BIT11,SBEMSK
9715 034004 001006              BNE   13$
9716 034006 013737 002250 002254      MOV   SBEMSK,DBEMSK
9717 034014 006337 002254      ASL   DBEMSK
9718 034020 000743              BR    12$
9719 034022 104471              13$:  ECC1DIS
9720 034024 012701 002410      MOV   #TESTADD,R1 ;DISABLE ECC ON 1 SELECTED CSR
9721 034030 005031              CLR   @R1+       ;RESTORE TEST ADDRESS
9722 034032 005071 000000      CLR   @R1       ;CLEAR ANY DBE'S FROM TEST
9723 034036 104503              CLR1CSR ;CLEAR 1 SELECTED MK11 CSR
9724 034040 000207      RETURN

```


CZMSP80 MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 289
 MTP016 WRITE INHIBIT OF WORD WITH DBE

9727 034042

MTP017: SUBTST <<MTP017 HOLDING 1'S & 0'S TEST>>

```
*****
:*SUBTEST      MTP017  HOLDING 1'S & 0'S TEST
*****
```

```

9728          :*(1)  THIS TEST CHECKS THE MEMORY FOR THE CAPABILITY
9729          :*    OF HOLDING 1'S AND 0'S BY WRITING A BACKGROUND
9730          :*    OF 000377 AND READING IT
9731          :*(2)  MEMORY IS WRITTEN USING A BYTE AT A TIME
9732          :*(3)  STEPS 1 & 2 ARE REPEATED WITH A SWAPPED BACKGROUND PATTERN
9733          ;NOTE: THIS TEST WRITES BYTES & READS WORDS
9734 034042 012701 060000  MOV    #FIRST,R1
9735 034046 010104        MOV    R1,R4
9736 034050 012705 160000  MOV    #LAST+2,R5
9737 034054 012700 000377  MOV    #377,R0          ;GET THE PATTERN INTO R0
9738 034060 010003        MOV    R0,R3
9739 034062 000303        SWAB   R3
9740 034064 110021 1$:    MOVB   R0,(R1)+      ;WRITE A BYTE
9741 034066 110321        MOVB   R3,(R1)+      ;WRITE THE MEMORY WITH THE BYTE STORED IN BAKPAT+1
9742 034070 020105        CMP    R1,R5        ;COMPARE TEST LOC TO TOP + 2
9743 034072 103774        BLO   1$          ;BRANCH IF LOWER
9744
9745 034074 014102 2$:    MOV    -(R1),R2
9746 034076 020002        CMP    R0,R2        ;TEST THE MEMORY TO SEE IF IT CONTAINS
9747                                ;THE WORD STORED IN BAKPAT
9748 034100 001401        BEQ   3$
9749 034102 104446        PERR22
9750
9751 034104 020104 3$:    CMP    R1,R4        ;KEEP ON TESTING THE MEMORY UNTIL
9752 034106 101372        BHI   2$          ;R1 EQUALS THE LOWEST ADDRESS
9753 034110 000303        SWAB   R3          ;CHANGE THE DATA PATTERN
9754 034112 000300        SWAB   R0
9755 034114 001763        BEQ   1$          ;IF THE DATA PATTERN DOES NOT HAVE LOW
9756                                ; BYTE =0 THEN FALL THRU
9757 034116 000207        RETURN
```

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 291
MTP017 HOLDING 1'S & 0'S TEST

9761 034120

MTP020: SUBTST <<MTP020 SYNDROMES TO CSR ON SINGLE BIT ERROR TEST>>

```
*****
:SUBTEST      MTP020 SYNDROMES TO CSR ON SINGLE BIT ERROR TEST
*****
```

9762

9763

9764

9765

9766

9767 034120 104424

9768 034122 005000

9769 034124 105037 002262

9770 034130 104513

9771 034132

9772 034132

9773 034136

9774 034142

9775 034146

9776 034156

9777 034162

9778 034164

9779 034170

9780 034170

9781 034170 005237 002322

9782 034174

9783 034176

9784 034202 072227 000005

9785 034206 052702 000004

9786 034212

9787 034216 104425

9788 034220

9789 034222 104503

9790 034224 005711

9791 034226 104426

9792 034230 042737 177757 002146

9793 034236

9794 034246

9795 034254

9796 034262 104060

9797 034264

9798 034264 104513

9799 034266 104426

9800 034270 042737 174033 002146

9801 034276

9802 034302 072327 000005

9803 034306 052703 000004

9804 034312

9805 034320

9806 034326

9807 034332

9808 034340 104042

9809 034342

9810 034342 005011

9811 034344 006305

9812 034354 006305

9813 034356

9814 034360 000261

```

: THIS TEST CHECKS TO SEE IF THE SINGLE BIT ERRORS CAUSE THE SBE
: BIT IN THE CSR TO BE SET AND CORRECT SYNDROME BITS ARE GENERATED FOR
: ALL 16 DATA BITS.
:
: CACHOFF          :TURN OFF CACHE
CLR R0             :CLEAR DATA
CLRB PASFLG       :CLEAR PASFLG
CBREG             :ENABLE CHECK/SYNDROME BIT REGISTER
REPEAT           :
  LET PASFLG :B= PASFLG + #1 :INCREMENT LOOP COUNTER
  LET R4 := #-1 :INDEX TO SINGLE BIT ERROR TABLE
  LET BITNO := #0 :CLEAR INNER LOOP COUNTER
  IFB PASFLG EQ #1 :SELECT DATA TO BE CORRECTED BY PASSNO
    LET R5 := #1 :DATA=0;BIT TO BE CORRECTED IS A ONE
  ELSE :
    LET R5 := #177776 :DATA=177776;BIT TO BE CORRECTED IS A ZERO
  END
REPEAT           :
  INC BITNO :INCREMENT BIT POINTER
  LET R4 := R4 + #1 :POINT TO NEXT SET OF CHECK BITS
  LET R2 :B= PTABLE(R4) :GET NEXT SET OF CHECK BITS
  ASH #5,R2 :SHIFT TO LINE UP IN CSR
  BIS #BIT2,R2 :ENABLE DIAG MODE
  LET CSR := R2 :GET CHECK BITS TO BE WRITTEN
  LOADCSR :LOAD CSR WITH DATA
  LET (R1) := R0 :WRITE DATA TO TEST ADDRESS
  CLR1CSR :CLEAR CSR
  TST (R1) :CORRECT SBE
  READCSR :READ CSR FOR CORRECT SBE BIT AND SYNDROMES
  BIC #^C20,CSR :CLEAR ALL BUT SBE INDICATOR
  IF CSR NE #20 :WAS DATA CORRECTED???
    LET GOOD := #20
    LET BAD := CSR
    ERRJR +60
  END :NO ERROR
CBREG :ENABLE SYNDROME BIT REGISTER
READCSR :GET SYNDROMES FROM CSR
BIC #^C3744,CSR :MASK SYNDROME BITS
LET R3 :B= SBESYN(R4) :GET GOOD SYNDROMES
ASH #5,R3 :SHIFT INTO POSITION
BIS #BIT2,R3 :SET DIAG MODE IN DATA
IF R3 NE CSR :DO SYNDROME BITS AGREE
  SET HEADER
  LET GOOD := R3
  LET BAD := CSR
  ERROR +42
END
CLR (R1) :CLEAR LUT
IFB PASFLG EQ #1 :SHIFT NEW DATA DEPENDING ON PASFLG
  ASL R5 :SHIFT BITNO TO THE LEFT
ELSE :
  SEC :SET CARRY BIT AND.....

```


CZMSPB0 MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 292
 MTP020 SYNDROMES TO CSR ON SINGLE BIT ERROR TEST

9829 034434

MTPA21: SUBST <<MTPA21 MARCHING 1'S & 0'S PATTERN TEST>>

 :SUBTEST MTPA21 MARCHING 1'S & 0'S PATTERN TEST

9830

:READ,BYTESWAP-MODIFY,READ,DOWN

9831 034434 014100

1\$: MOV -(R1),R0 :V177640

9832 034436 020200

CMP R2,R0 :V177642

9833 034440 001401

BEQ 2\$:V177644

9834 034442 104443

PERR17 :V177646

9835

9836 034444 000311

2\$: SWAB (R1) :V177650

9837 034446 011100

MOV (R1),R0 :V177652

9838 034450 020300

CMP R3,R0 :V177654

9839 034452 001401

BEQ 3\$:V177656

9840 034454 104444

PERR20 :V177660

9841

9842 034456 020401

3\$: CMP R4,R1 :V177662

:DONE?

9843 034460 001365

BNE 1\$:V177664

:NO - LOOP

9844 034462 000207

RETURN :V177666

:YES - RETURN

9845

9846 034464

MTPB21: :READ,BYTESWAP-MODIFY,READ,UP

9847 034464 011100

1\$: MOV (R1),R0 :V177640

9848 034466 020300

CMP R3,R0 :V177642

9849 034470 001401

BEQ 2\$:V177644

9850 034472 104444

PERR20 :V177646

9851

9852 034474 000311

2\$: SWAB (R1) :V177650

9853 034476 011100

MOV (R1),R0 :V177652

9854 034500 020200

CMP R2,R0 :V177654

9855 034502 001401

BEQ 3\$:V177656

9856 034504 104443

PERR17 :V177660

9857

9858 034506 062701 000002

3\$: ADD #2,R1 :V177662

9859 034512 020501

CMP R5,R1 :V177666

:DONE?

9860 034514 001363

BNE 1\$:V177670

:NO - LOOP

9861 034516 000207

RETURN :V177672

:YES - RETURN

9862

9863 034520

MTPC21: :READ,BYTESWAP-MODIFY,READ,UP

9864 034520 011100

1\$: MOV (R1),R0 :V177640

9865 034522 020200

CMP R2,R0 :V177642

9866 034524 001401

BEQ 2\$:V177644

9867 034526 104443

PERR17 :V177646

9868

9869 034530 000311

2\$: SWAB (R1) :V177650

9870 034532 011100

MOV (R1),R0 :V177652

9871 034534 020300

CMP R3,R0 :V177654

9872 034536 001401

BEQ 3\$:V177656

9873 034540 104444

PERR20 :V177660

9874

9875 034542 062701 000002

3\$: ADD #2,R1 :V177662

9876 034546 020501

CMP R5,R1 :V177666

:DONE?

9877 034550 001363

BNE 1\$:V177670

:NO - LOOP

9878 034552 000207

RETURN :V177672

:YES - RETURN

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 294
 MTPA21 MARCHING 1'S & 0'S PATTERN TEST

9881	034554		MTPD21:	;READ,BYTESWAP-MODIFY,READ,DOWN		
9882	034554	014100	1\$:	MOV	-(R1),R0	:V177640
9883	034556	020300		CMP	R3,R0	:V177642
9884	034560	001401		BEQ	2\$:V177644
9885	034562	104444		PERR20		:V177646
9886						
9887	034564	000311	2\$:	SWAB	(R1)	:V177650
9888	034566	011100		MOV	(R1),R0	:V177652
9889	034570	020200		CMP	R2,R0	:V177654
9890	034572	001401		BEQ	3\$:V177656
9891	034574	104443		PERR17		:V177660
9892						
9893	034576	020401	3\$:	CMP	R4,R1	:V177662
9894	034600	001365		BNE	1\$:V177664
9895	034602	000207		RETURN		:V177666
9896						

;DONE?
;NO - LOOP
;YES - RETURN

CZMSPBO MS11-L/M/P MEMG.V DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 296
 MTPA21 MARCHING 1'S & 0'S PATTERN TEST

9899 034604

```

MTP022: SUBTST <<MTP022 REFRESH & SHIFTING DIAGONAL TEST>>
:*****
:*SUBTEST MTP022 REFRESH & SHIFTING DIAGONAL TEST
:*****
: (1) WE WRITE A DIAGONAL PATTERN IN MEMORY (WITH CACHE ON).
: (2) IF A REFRESH TEST WE DISTURB ALL ROWS FOR > 2 MS (WITH CACHE ON).
: (3) WE READ & CHECK FOR CORRECTNESS THE DIAGONAL PATTERN
: (WITH CACHE OFF).
KDIAG=8. ;HOW OFTEN A DIAGONAL STRIPE OCCURS (MUST BE A POWER OF 2)
FOR EVEN := #1 TO #2 ;FOR DATA & COMPLEMENT DATA
IF EVEN EQ #1
LET R2 := ZEROS
LET R3 := ONES
ELSE
LET R2 := ONES
LET R3 := ZEROS
END ;OF IF EVEN
FOR STRIPES := #0 TO #KDIAG-1 ;FOR THE NUMBER OF STRIPES

;WRITE LOOP
CACHON ;TURN CACHE ON
LET COUNT := STRIPES
LET R1 := #FIRST
WHILE R1 LOS #LAST
IF COUNT LT #0 THEN LET COUNT := #KDIAG-1
IF #374 OFF. IN R1 THEN LET COUNT := COUNT - #1
IF COUNT NE #0
LET (R1) := R2
LET 2(R1) := R2
ELSE
LET (R1) := R3
LET 2(R1) := R3
END ;OF IF COUNT
LET COUNT := COUNT - #1
LET R1 := R1 + #4
END ;OF WHILE
;END OF WRITE LOOP

IF DIAGFLAG IS FALSE THEN SCALL REFRESH
;READ LOOP
LET COUNT := STRIPES
LET R1 := #FIRST
CACHOFF ;TURN CACHE OFF

```

9900

9901

9902

9903

9904

000010

9905 034604

9906 034612

9907 034622

9908 034626

9909 034632

9910 034634

9911 034640

9912 034644

9913 034644

9914

9915

9916 034650

104423

9917 034652

9918 034660

9919 034664

9920 034672

9921 034706

9922 034720

9923 034726

9924 034730

9925 034734

9926 034736

9927 034740

9928 034744

9929 034744

9930 034750

9931 034754

9932

9933

9934 034756

9935

9936 034770

9937 034776

9938 035002

104424


```

9940 035004
9941 035012
9942 035026
9943 035040
9944 035046
9945 035050
9946 035054 104443
9947 035056
9948 035056
9949 035062
9950 035066 104443
9951 035070
9952 035070
9953 035072
9954 035074
9955 035100 104444
9956 035102
9957 035102
9958 035106
9959 035112 104444
9960 035114
9961 035114
9962 035114
9963 035120
9964 035124
9965
9966
9967 035126
9968 035142
9969 035156 000207
9970
9971 035160

9972
9973 035160
9974 035164 004737 035230
9975 035170
9976 035202
9977 035206
9978 035214 004737 035230
9979 035220
9980 035224
9981 035226 000207
9982 035230 012704 000640
9983 035234 062700 000002
9984 035240 005140
9985 035242 005120
9986 035244 005110
9987 035246 005110
9988 035250 077405
9989 035252 162700 000002
9990 035256 000207

```

```

WHILE R1 LOS #LAST
IF COUNT LT #0 THEN LET COUNT := #KDIAG-1
IF #374 OFF.IN R1 THEN LET COUNT := COUNT - #1
IF COUNT NE #0
LET R0 := (R1)
IF R2 NE R0
PERR17
END ;OF IF R2
LET R0 := 2(R1)
IF R2 NE R0
PERR17
END ;OF IF R2
ELSE
LET R0 := (R1)
IF R3 NE R0
PERR20
END ;OF IF R3
LET R0 := 2(R1)
IF R3 NE R0
PERR20
END ;OF IF R3
END ;OF IF COUNT
LET COUNT := COUNT - #1
LET R1 := R1 + #4
END ;OF WHILE
;END OF READ LOOP

END ;OF FOR STRIPES
END ;OF FOR EVEN
RETURN

REFRESH:SUBTST <<SUBR REFRESH DELAY>>
:*****
:*SUBTEST SUBR REFRESH DELAY
:*****
;DISTURB EACH ROW FOR > 3.2 MS
FOR RO := #FIRST TO #FIRST+374 BY #4
CALL REFSUB
END ;OF FOR RO
LET RO := #FIRST+BIT14
WHILE RO LOS #LAST+BIT14+374
CALL REFSUB
LET RO := RO + #4
END ;OF WHILE
RETURN

REFSUB: MOV #640,R4 ;TIME FOR A > 3.2 MS LOOP
ADD #2,R0
1$: COM -(R0)
COM (R0)+
COM (R0)
COM (R0)
SOB R4,1$
SUB #2,R0
RETURN

```

9993 035260

MTPA24: SUBTST <<MTPA24 FAST GALLOPING PATTERN TEST>>
:*****
:*SUBTEST MTPA24 FAST GALLOPING PATTERN TEST
:*****

9994
9995
9996
9997
9998
9999
10000
10001
10002
10003
10004
10005
10006
10007
10008
10009
10010
10011
10012
10013
10014
10015
10016
10017
10018
10019 035260 011100
10020 035262 020004
10021 035264 001401
10022 035266 104447
10023
10024 035270 011200
10025 035272 020003
10026 035274 001401
10027 035276 104450
10028
10029 035300 062702 000400
10030 035304 020205
10031 035306 101764
10032
10033 035310 062701 000002
10034 035314 000137 035320

:THE TOTAL TEST (INCLUDING SETUP) IS AS FOLLOWS
:*(1) THIS TEST WRITES THE MEMORY WITH A BACK GROUND PATTERN
:* STORED AT LOCATION BAKPAT
:*(2) TEST BEGINS AT LOWEST LOCATION BEING TESTED
:* (LETS NAME IT 'A')
:*(3) LETS NAME THE 1ST LOCATION IN THE ROW/COLUMN UNDER TEST AS 'B'.
:*(4) SWAPS BYTES FOR LOCATION 'A'.
:*(5) READS 'A', READS 'B'
:*(6) 'B' = 'B'+400 (ADDS 64 DOUBLE WORDS TO 'B')
:*(7) REPEATS STEPS 5 AND 6 UNTIL 'B' IS GREATER THAN THE
:*(8) END OF THE BANK A+2
:*(9) REPEATS STEPS 3-8 UNTILL 'A' REACHES THE END OF THE BANK
:*(10) AFTER EXECUTING THE TEST DATA IS COMPLEMENTED
:* AND STEPS 1-9 ARE REPEATED
:REGISTERS ARE USED AS FOLLOWS
:R0 TEST DATA
:R1 'A'
:R2 'B'
:R3 BAKPAT
:R4 SWAPAT
:R5 LAST

:NOTE THE PATTERN STARTS AT MTPB24!!!!!!!!!!!!!!!!!!!!

:UIPAR'S
1\$: MOV (R1),R0 :V177640 :READ 'A'
CMP R0,R4 :V177642 :CHECK 'A'
BEQ 2\$:V177644 :BR IF OK
PERR23 :V177646 :REPORT ERROR
2\$: MOV (R2),R0 :V177650 :READ 'B'
CMP R0,R3 :V177652 :CHECK 'B'
BEQ 3\$:V177654 :BR IF OK
PERR24 :V177656 :REPORT ERROR
3\$: ADD #400,R2 :V177660 :BUMP 'B'
CMP R2,R5 :V177664 :AT END YET?
BLOS 1\$:V177666 :BR IF NO
ADD #2,R1 :V177670 :BUMP 'A'
JMP @#MTPB24 :V177674 :GOTO V177260

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 301
MTPA24 FAST GALLOPING PATTERN TEST

10037 035320

MTPB24: SUBTST <<MTPB24 FAST GALLOP PART B>>
:*****
:*SUBTEST MTPB24 FAST GALLOP PART B
:*****

10038

10039 035320 010411
10040 035322 020105
10041 035324 001001
10042 035326 000207
10043 035330 000137 035334
10044
10045 035334

:SDPAR'S
MOV R4,(R1) :V172260 :WRITE 'A'
CMP R1,R5 :V172262 :DONE?
BNE 1\$:V172264 :BR IF NO
RETURN :V172266 :YES - RETURN
1\$: JMP @#MTPC24 :V172270 :GOTO V172360

MTPC24: SUBTST <<MTPC24 FAST GALLOP PART C>>
:*****
:*SUBTEST MTPC24 FAST GALLOP PART C
:*****

10046

10047 035334 010102
10048 035336 011100
10049 035340 020004
10050 035342 001401
10051 035344 104447
10052 035346 000137 035300

:KDPAR'S
MOV R1,R2 :V172360 :RESET 'B' <--- 'A'
MOV (R1),R0 :V172362 :READ 'A'
CMP R0,R4 :V172364 :CHECK 'A'
BEQ 1\$:V172366 :BR IF OK
PERR23 :V172370 :REPORT ERROR
1\$: JMP @#MTPA24+20 :V172372 :GOTO V177660

10055 035352

MTP025: SUBTST <<MTP025 INTERRUPT ENABLE TEST>>
:*****
:SUBTEST MTP025 INTERRUPT ENABLE TEST
:*****

10056	035352	005037	002244		CLR	TSTDAT	:GENERATE CHECKBITS ON 0,,0
10057	035356	005037	002246		CLR	TSTDAT+2	
10058	035362	012737	002244	002310	MOV	#TSTDAT,SOURCE	
10059	035370	004737	044400		CALL	CHKGEN	
10060	035374	012737	000003	002074	MOV	#3,NOPAR	:SETUP PARITY ACTION
10061	035402	012701	002410		MOV	#TESTADD,R1	:FIRST TEST ADDRESS
10062	035406	012737	035446	002302	MOV	#1\$,PARTHERE	:SETUP TRAP DESTINATION
10063	035414	004737	035670		CALL	MTPA25	:WRITE DATA & CHECKBITS
10064	035420	104473			ECC1INIT		:INITIALIZE 1 SELECTED MK11 CSR
10065	035422	005771	000000		TST	@(R1)	:ACCESS LOCATIONS FOR DBE TRAPS
10066	035426	005771	000002		TST	@2(R1)	
10067							:NONE - GOOD - ACCESS FOR SBE TRAPS
10068	035432	104507			ENA1SBE		:DISABLE TRAPS ON SBE'S FROM 1 SELECTED CSR
10069	035434	005771	000000		TST	@(R1)	
10070	035440	005771	000002		TST	@2(R1)	
10071	035444	000404			BR	2\$:NONE - GOOD - SKIP
10072	035446	104426			1\$:	READCSR	
10073	035450				FATAL	27	
10074	035456	005237	002244		2\$:	INC	TSTDAT
10075	035462	004737	035616		CALL	MTPD25	:CHECK FOR CORRECT ACTION ON SBE'S
10076	035466	012737	000400	002244	MOV	#400,TSTDAT	:IN ALL 4 BYTES
10077	035474	004737	035616		CALL	MTPD25	
10078	035500	005037	002244		CLR	TSTDAT	
10079	035504	005237	002246		INC	TSTDAT+2	
10080	035510	004737	035616		CALL	MTPD25	
10081	035514	012737	000400	002246	MOV	#400,TSTDAT+2	
10082	035522	004737	035616		CALL	MTPD25	
10083							
10084	035526	005037	002246		CLR	TSTDAT+2	:CHECK FOR CORRECT ACTION ON DBE'S
10085	035532	012737	000003	002244	MOV	#3,TSTDAT	:IN ALL 4 BYTES
10086	035540	004737	035640		CALL	MTPE25	
10087	035544	012737	001400	002244	MOV	#1400,TSTDAT	
10088	035552	004737	035640		CALL	MTPE25	
10089	035556	005037	002244		CLR	TSTDAT	
10090	035562	012737	000003	002246	MOV	#3,TSTDAT+2	
10091	035570	004737	035640		CALL	MTPE25	
10092	035574	012737	001400	002246	MOV	#1400,TSTDAT+2	
10093	035602	004737	035640		CALL	MTPE25	
10094	035606	104503			CLR1CSR		:CLEAR 1 SELECTED MK11 CSR
10095	035610	005037	002074		CLR	NOPAR	:INDICATE PARITY ACTION
10096	035614	000207			RETURN		
10097							
10098	035616	004737	035670		MTPD25:	CALL	MTPA25
10099	035622	104471			ECC1DIS		:WRITE DATA & CHECKBITS
10100	035624	004737	035712		CALL	MTPB25	:DISABLE ECC ON 1 SELECTED CSR
10101	035630	104507			ENA1SBE		:CHECK FOR NO TRAPS
10102	035632	004737	035752		CALL	MTPC25	:DISABLE TRAPS ON SBE'S FROM 1 SELECTED CSR
10103	035636	000207			RETURN		:CHECK FOR EXPECTED TRAP

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 305
 MTP025 INTERRUPT ENABLE TEST

```

10106 035640 004737 035670          MTPE25: CALL      MTPA25          :WRITE DATA & CHECKBITS
10107 035644 104471                   ECC1DIS          :DISABLE ECC ON 1 SELECTED CSR
10108 035646 004737 035712          CALL      MTPB25          :CHECK FOR NO TRAPS
10109                                     :ENABLE DBE TRAPS
10110 035652 104473                   ECC1INIT          :INITIALIZE 1 SELECTED MK11 CSR
10111 035654 004737 035752          CALL      MTPC25          :CHECK FOR EXPECTED TRAP
10112 035660 104507                   ENA1SBE          :DISABLE TRAPS ON SBE'S FROM 1 SELECTED CSR
10113 035662 004737 035752          CALL      MTPC25          :CHECK FOR EXPECTED TRAP
10114 035666 000207                   RETURN
10115
10116                                     :WRITE TSTDAT & TSTDAT+2 & CHECKBITS
10117 035670 104471          MTPA25: ECC1DIS          :DISABLE ECC ON 1 SELECTED CSR
10118 035672 013771 002244 000000    MOV      TSTDAT,@(R1)    :WRITE FIRST 16 BITS
10119 035700 104475          CB1CSR          :WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
10120 035702 013771 002246 000002    MOV      TSTDAT+2,@2(R1) :WRITE 2ND 16 BITS & CHECKBITS
10121 035710 000207          RETURN
10122
10123                                     :CHECK FOR NO TRAP OCCURING CONDITION
10124 035712 012737 035732 002302    MTPB25: MOV      #1$,PARTHERE :SETUP TRAP DESTINATION
10125 035720 005771 000000          TST      @(R1)          :ACCESS LOCATIONS
10126 035724 005771 000002          TST      @2(R1)
10127 035730 000207          RETURN                :NO TRAP - GOOD - RETURN
10128
10129 035732 104426          1$:  READCSR
10130 035734 011137 002032          MOV      (R1),ADDRESS  :SAVE VIRTUAL ADDRESS
10131 035740 104024          ERROR   +24
10132 035742          SET     HEADER
10133 035750 000207          RETURN
10134
10135                                     :TRAP SHOULD OCCURE TEST
10136 035752 012737 035766 002302    MTPC25: MOV      #1$,PARTHERE :SETUP TRAP DESTINATION
10137 035760 005771 000000          TST      @(R1)          :ACCESS 1ST LOCATION
10138 035764 000405          BR      2$
10139 035766 012737 036016 002302    1$:  MOV      #3$,PARTHERE :SETUP TRAP DESTINATION
10140 035774 005771 000002          TST      @2(R1)          :ACCESS 2ND LOCATION
10141 036000 104426          2$:  READCSR
10142 036002 011137 002032          MOV      (R1),ADDRESS  :SAVE VIRTUAL ADDRESS
10143 036006 104025          ERROR   +25
10144 036010          SET     HEADER
10145 036016 000207          3$:  RETURN
10146

```

10149 036020

```

MTPA26: SUBTST <<MTPA26      RANDOM DATA (WRITE)>>
:*****
:*SUBTEST      MTPA26  RANDOM DATA (WRITE)
:*****
1$:      JMP      @#MTPC26      :V177640      GOTO V172360
          MOV      R2,(R1)+      :V177644
          MOV      R3,(R1)+      :V177646
          SOB      R0,1$         :V177650
          RETURN                     :V177652

```

10150 036020 000137 036070
 10151 036024 010221
 10152 036026 010321
 10153 036030 077005
 10154 036032 000207
 10155
 10156 036034

```

MTPB26: SUBTST <<MTPB26      RANDOM DATA (READ)>>
:*****
:*SUBTEST      MTPB26  RANDOM DATA (READ)
:*****
          .DSABL  AMA
          .ENABL  LSB
1$:      JMP      @#MTPC26      :V177640      GOTO V172360
          CMP      R2,(R1)+      :V177644
          BEQ      2$           :V177646
          PERR25                     :V177650
2$:      COM      (PC)+          :V177652
          RANODD: 0              :V177654      FOR ERROR REPORTING
          CMP      R3,(R1)+      :V177656
          BEQ      3$           :V177660
          PERR25                     :V177662
3$:      COM      RANODD         :V177664
          SOB      R0,1$         :V177670
          RETURN                     :V177672
          .DSABL  LSB
          .ENABL  AMA

```

10157
 10158
 10159 036034 000137 036070
 10160 036040 020221
 10161 036042 001401
 10162 036044 104451
 10163 036046 005127
 10164 036050 000000
 10165 036052 020321
 10166 036054 001401
 10167 036056 104451
 10168 036060 005167 177764
 10169 036064 077015
 10170 036066 000207
 10171
 10172
 10173
 10174 036070

```

MTPC26: SUBTST <<RANDOM NUMBER SUBPROGRAM>>
:*****
:*SUBTEST      RANDOM NUMBER SUBPROGRAM
:*****
          ;CALLER MUST SETUP
          :      MOV      SEEDLO,R3
          :      MOV      SEEDHI,R2
          :      MOV      R3,R5
          :      MOV      R2,R4
          ASHC   #7,R4          :V172360
          ADD    R3,R5          :V172364
          ADC    R4            :V172366
          ADD    R2,R4          :V172370
          ADD    #1057,R5      :V172372
          NOP                    :V172376      GOTO V172260

```

10175
 10176
 10177
 10178
 10179
 10180 036070 073427 000007
 10181 036074 060305
 10182 036076 005504
 10183 036100 060204
 10184 036102 062705 001057
 10185 036106 000240
 10186
 10187 036110

```

MTPD26: SUBTST <<RANDOM NUMBER SUBSUBPROGRAM>>
:*****
:*SUBTEST      RANDOM NUMBER SUBSUBPROGRAM
:*****
          ADC    R4            :V172260
          ADD    #47401,R4     :V172262
          MOV    R5,R3         :V172266
          MOV    R4,R2         :V172270
          JMP    @#MTPA26+4    :V172272      GOTO V177644

```

10188 036110 005504
 10189 036112 062704 047401
 10190 036116 010503
 10191 036120 010402
 10192 036122 000137 036024

10195 036126

```

MTP030: SUBTST <<MTP030      FLUSH OUT DBE'S>>
:*****
:*SUBTEST      MTP030  FLUSH OUT DBE'S
:*****
1$:  MOV      (R0),R2      :V177640
      MOV      R2,(R0)+    :V177642
      SOB      R1,1$      :V177644
      RETURN                      :V177646
    
```

10196 036126 011002
10197 036130 010220
10198 036132 077103
10199 036134 000207
10200
10201 036136

```

MTP031: SUBTST <<MTP031      SOB-A-LONG TEST>>
:*****
:*SUBTEST      MTP031  SOB-A-LONG TEST
:*****
    
```

10202
10203 036136 000000
10204 036140 077001
10205 036142 005167 177772
10206 036146 020167 177766
10207 036152 001403
10208 036154 104454
10209 036156 010167 177756
10210 036162 005167 177752
10211 036166 010200
10212
10213 036170 010503
10214 036172 005725
10215 036174 010504
10216 036176 020537 002520
10217 036202 001001
10218 036204 000207
10219
10220 036206 014344
10221 036210 001376
10222 036212 000752
10223 000056
10224

```

      .DSABL  AMA
0      :MOVE TERMINATOR
1$:  SOB      R0,1$      :SOB TILL R0 UNDERFLOWS
      COM      1$      :WRITE COMPLEMENT OF SOB
      CMP      R1,1$    :READ & CHECK FOR NOT 'SOB R0,DOT'
      BEQ      2$      :OK - SKIP
      PERR30
      MOV      R1,1$
2$:  COM      1$      :CORRECT SOB INSTRUCTION
      MOV      R2,R0    :REINITIALIZE SOB CONSTANT
      :UPDATE MOVE REGISTERS
      MOV      R5,R3
      TST      (R5)+    :BUMP (SAFELY) BY 2
      MOV      R5,R4
      CMP      R5,@LINK1 :DONE?
      BNE      3$      :NO - SKIP
      RETURN                      :YES
3$:  MOV      -(R3),-(R4)
      BNE      3$
      BR      1$
SOBLENGTH=-MTP031
      .ENABL  AMA
    
```

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 311
MTP031 SOB-A-LONG TEST

10252 036214

MTP032: SUBTST <<MTP032 WRITE RECOVERY TEST>>
:*****
:SUBTEST MTP032 WRITE RECOVERY TEST
:*****

10253
10254
10255
10256
10257
10258 036214 012401
10259 036216 020102
10260 036220 001401
10261 036222 104430
10262 036224 077305
10263 036226 013703 002520
10264 036232 012400
10265 036234 020005
10266 036236 001401
10267 036240 104427
10268 036242 077305
10269 036244 000207

:THE TEST ACTUALLY EXECUTED ALREADY IN THE MEMORY UNDER TEST.
:THIS CODE INSURES THAT IT CHANGED MEMORY TO HAVE
:1/2 BANK OF #5141 WHICH IS A "COM -(R1)" INSTRUCTION AND
:1/2 BANK OF #110 WHICH IS A "JMP (R0)" INSTRUCTION.
1\$: MOV (R4)+,R1 ;V177640 ;GET DATA FROM LOWER 1/2 BANK
CMP R1,R2 ;V177642 ;IS IT #5141?
BEQ 2\$;V177644 ;YES - SKIP
PERR02 ;V177646 ;NO - TAKE ERROR TRAP
2\$: SOB R3,1\$;V177650 ;LOOP FOR 1/2 BANK
MOV @#LINK1,R3 ;V177652 ;RESTORE LOOP SIZE
3\$: MOV (R4)+,R0 ;V177656 ;GET DATA FROM UPPER 1/2 BANK
CMP R0,R5 ;V177660 ;IS IT #110?
BEQ 4\$;V177662 ;YES - SKIP
PERR01 ;V177664 ;NO- TAKE ERROR TRAP
4\$: SOB R3,3\$;V177666 ;LOOP FOR 1/2 BANK
RETURN

CZMSP80 MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 313
 MTP032 WRITE RECOVERY TEST

10272 036246

MTP033: SUBTST <<MTP033 BRANCH GOBBLE TEST>>

```

:*****
:*SUBTEST      MTP033 BRANCH GOBBLE TEST
:*****

```

```

10273
10274 036246 000000
10275 036250 000000
10276 036252 000261
10277 036254 105511
10278 036256 100402
10279 036260 105212
10280 036262 000773
10281
10282
10283 036264 102401
10284 036266 104461
10285
10286 036270 000242
10287 036272 105212
10288 036274 103402
10289 036276 102001
10290 036300 100401
10291 036302 104461
10292
10293
10294
10295 036304 010701
10296 036306 162701 000036
10297 036312 010102
10298 036314 005202
10299
10300
10301 036316 010503
10302 036320 005725
10303 036322 010504
10304
10305
10306 036324 020537 002520
10307 036330 001001
10308 036332 000207
10309
10310
10311 036334 014344
10312 036336 001376
10313 036340 005011
10314 036342 000743
10315 000076
10316

```

```

      .DSABL  AMA
      0
BGTEST: 0      :MOVE TERMINATOR
BRGOBB: SEC    :TEST WORD (TWO BYTES)
      ADCB    (R1) :SET CARRY (TO BE ADDED TO 'BGTEST')
      BMI     1$  :INCREMENT LOW BYTE OF 'BGTEST'
      INCB    (R2) :BRANCH WHEN BIT7 IS SET
      BR      BRGOBB :INCREMENT HIGH BYTE OF 'BGTEST'
      :LOOP 128 TIMES

      :NOW CHECK FOR CORRECT CONDITION CODES
1$:   BVS     2$  :BR IF V-BIT SET (SHOULD BE)
      PERR35    :NO - REPORT ERROR AND ABORT TEST
      :COND CODES NOT EQUAL TO 1010
2$:   CLV
      INCB    (R2) :CLEAR V-BIT
      BCS     3$  :INCREMENT HIGH BYTE OF 'BGTEST' ONCE MORE
      BVC     3$  :BR IF C-BIT SET (SHOULD NOT BE)
      RMI     4$  :BR IF V-BIT CLEAR (SHOULD NOT BE)
      PERR35    :BR IF N-BIT SET (SHOULD BE)
      :NO - REPORT ERROR AND ABORT TEST
      :COND CODES NOT EQUAL TO 1010
4$:   :UPDATE TEST POINTERS
      MOV     PC,R1
5$:   SUB     #5$-BGTEST,R1
      MOV     R1,R2
      INC     R2

      :UPDATE MOVE REGISTERS
      MOV     R5,R3
      TST     (R5)+
      MOV     R5,R4
      :BUMP (SAFELY) BY 2
      :DONE?
      CMP     R5,#LINK1
      BNE     6$
      RETURN  :DONE?
      :NO - SKIP
      :YES - RETURN

6$:   :MOVE CODE 1 LOCATION
      MOV     -(R3),-(R4)
      BNE     6$
      CLR     (R1)
      BR      BRGOBB
      :CLEAR TEST WORD 'BGTEST'
      :RUN MOVED CODE AGAIN

GBLENGTH=-MTP033
      .ENABL  AMA

```


CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 314
MTP033 BRANCH GOBBLE TEST

10318 036344

MTP034: SUBTST <<MTP034 SOFT ERROR - BACKGROUND PATTERN TEST>>

: *SUBTEST MTP034 SOFT ERROR - BACKGROUND PATTERN TEST

10319 036344 010220
10320 036346 077102
10321 036350 000207
10322 036352 012401
10323 036354 020102
10324 036356 001402
10325 036360 104430
10326 036362 000240
10327 036364 077306
10328 036366 000207

1\$: MOV R2,(R0)+ :V177640
SOB R1,MTP034 :V177642
RETURN :V177644
2\$: MOV (R4)+,R1 :V177646
CMP R1,R2 :V177650
BEQ 3\$:V177652
PERRO2 :V177654
NOP :V177656
3\$: SOB R3,2\$:V177660
RETURN :V177662

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 315
 MTP034 SOFT ERROR - BACKGROUND PATTERN TEST

```

10330 036370          MTP035:SUBTST  <<MTP035      WORST CASE NOISE PARITY TEST>>
:*****
:*SUBTEST          MTP035  WORST CASE NOISE PARITY TEST
:*****
10331 036370  012737  000003  002074          MOV      #3,NOPAR          ;SET PARITY TRAPS TO RETURN TO 'PARTHERE''
10332
10333 036376          FOR R0 := #FIRST TO #LAST BY #4000
10334 036402  012737  000005  002146          MOV      #BIT2!BIT0,CSR  ;SET WRITE WRONG PARITY & PAR. TRAPS INTO CSR
10335 036410  104425          LOADCSR
10336 036412  012737  036446  002302          MOV      #1$,PARTHERE
10337 036420  011010          MOV      (R0),(R0)      ;WWP TEST LOCATION
10338 036422  005710          TST      (R0)
10339 036424  010037  002032          MOV      R0,ADDRESS
10340 036430  104050          ERROR   +50
10341 036432  004737  057532          CALL   PERBNK
10342 036436  032763  002000  002654          BIT      #BIT10,CONFIG+2(R3)
10343 036444  001002          BNE     2$
10344 036446  104426          1$:    READCSR
10345 036450  104512          ERRGEN
10346
10347 036452  104503          2$:    CLR1CSR
10348 036454  011010          MOV      (R0),(R0)      ;CLEAR WRONG PARITY IN MEMORY
10349 036456  012737  000001  002146          MOV      #BIT0,CSR
10350 036464  104425          LOADCSR
10351 036466  012737  036500  002302          MOV      #3$,PARTHERE
10352 036474  005710          TST      (R0)
10353 036476  000405          BR      4$
10354 036500  010037  002032          3$:    MOV      R0,ADDRESS
10355 036504  104050          ERROR   +50
10356 036506  004737  057532          CALL   PERBNK
10357 036512          4$:    END; OF FOR
10358
10359 036524  005037  002074          CLR      NOPAR          ;RESET PARITY TRAP ACTION
10360 03653C  000207          RETURN

```

10362 036532

MTP036: SUBTST <<MTP036 CORRECTION CODE TEST>>
:.....
: *SUBTEST MTP036 CORRECTION CODE TEST
:.....

10363
10364
10365
10366
10367 036532 104424
10368 036534 105037 002262
10369 036540 104513
10370 036542
10371 036542
10372 036546
10373 036552
10374 036556
10375 036566
10376 036572
10377 036574
10378 036600
10379 036600
10380
10381 036600 005237 002322
10382 036604
10383 036606
10384 036612 072227 000005
10385 036616 052702 000004
10386 036622
10387 036626 104425
10388 036630
10389 036632 005711
10390 036634
10391 036640
10392 036646
10393 036652
10394 036656
10395 036662 104052
10396 036664
10397 036664 005011
10398 036666
10399 036676 006305
10400 036700
10401 036702 000261
10402 036704 006105
10403 036706
10404 036706
10405 036716 005100
10406 036720
10407 036730 104503
10408 036732 104423
10409 036734 000207

:
: THIS TEST CHECKS TO SEE THAT EACH BIT OF A DATA WORD
: CAN BE CORRECTED INDIVIDUALLY FROM A ZERO TO A ONE AND
: VISA VERSA.
: CACHOFF :TURN OFF CACHE
: CLR PASFLG :CLEAR PASFLG
: CBREG :ENABLE CHECK/SYNDROME BIT REGISTER
: REPEAT :
: LET PASFLG := PASFLG + #1 :INCREMENT LOOP COUNTER
: LET R4 := #-1 :INDEX TO SINGLE BIT ERROR TABLE
: LET BITNO := #0 :CLEAR INNER LOOP COUNTER
: IFB PASFLG EQ #1 :SELECT DATA TO BE CORRECTED BY PASSNO
: LET R5 := #1 :DATA=0;BIT TO BE CORRECTED IS A ONE
: ELSE :
: LET R5 := #177776 :DATA=177776;BIT TO BE CORRECTED IS A ZERO
: END :
: REPEAT :
: INC BITNO :INCREMENT BIT POINTER
: LET R4 := R4 + #1 :POINT TO NEXT SET OF CHECK BITS
: LET R2 := PTABLE(R4) :GET NEXT SET OF CHECK BITS
: ASH #5,R2 :SHIFT TO LINE UP IN CSR
: BIS #BIT2,R2 :ENABLE DIAG MODE
: LET CSR := R2 :GET CHECK BITS TO BE WRITTEN
: LOADCSR :LOAD CSR WITH DATA
: LET (R1) := R0 :WRITE DATA TO TEST ADDRESS
: TST (R1) :CORRECT SBE
: IF (R1) NE R5 :WAS DATA CORRECTED???
: LET ADDRESS := #60000 :MOV ERROR INFORMATION IN
: LET CHECK := R2 :
: LET TSTDAT := R5 :
: LET TSTDAT+2 := (R1) :
: ERROR +52 :NO ERROR
: END :
: CLR (R1) :CLEAR LUT
: IFB PASFLG EQ #1 :SHIFT NEW DATA DEPENDING ON PASFLG
: ASL R5 :SHIFT BITNO TO THE LEFT
: ELSE :
: SEC :SET CARRY BIT AND.....
: ROL R5 :ROTATE LEFT
: END :
: UNTIL BITNO EQ #16. :UNTIL ALL BITS ARE DONE
: COM R0 :COMPLEMENT DATA AND REPEAT
: UNTILB PASFLG EQ #2 :UNTIL 2 PASSES ARE COMPLETE!
: CLR1CSR :CLEAR CSR
: CACHON :TURN CACHE
: RETURN :
:

MS11-P SINGLE BIT ERROR CHECK BIT TABLE

10410
10411
10412
10413 036736 002 007 037
036741 031 032 025
036744 026 020 057
PTABLE: .BYTE 2,7,37,31,32,25,26,20,57,51,52,45,46,40,75,70

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 316-1
MTPO36 CORRECTION CODE TEST

036747	051	052	045
036752	046	040	075
036755	070		

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 317
MTP036 CORRECTION CODE TEST

10415 036756

MTP037: SUBTST <<MTP037 CHECK ECC DISABLE TEST>>
:*****
: *SUBTEST MTP037 CHECK ECC DISABLE TEST
:*****

10416
10417
10418
10419
10420 036756 104424
10421 036760
10422 036764
10423 036770 104475
10424 036772
10425 037000 104475
10426 037002
10427 037004
10428 037010
10429 037014
10430 037022 104037
10431 037024
10432 037024 104423
10433 037026 000207

: THIS TEST CHECKS THAT ECC CAN BE DISABLED AND THAT
: NO CORRECTION TAKES PLACE WITH ECC DISABLED.
:
: CACHOFF ;TURN OFF CACHE
: LET GOOD := #0 ;GOOD DATA FOR ERROR PRINT OUT
: LET CHECK := #0 ;CLEAR CHECK BIT FIELD
: CB1CSR ;ENABLE SYNDROME/CHECK BIT REGISTER
: LET CHECK := #100 ;SBE CHECK BITS
: CB1CSR ;WRITE CHECK BITS TO CB REGISTER
: LET (R1) := #0 ;WRITE CHECK BITS TO MEMORY
: IF (R1) NE #0 ;WAS CORRECTION MADE????
: LET BAD := (R1) ;YES IT WAS.....ERROR
: LET ADDRESS := #60000 ;
: ERROR +37
:
: END
: CACHON ;TURN ON CACHE
: RETURN

10436 037030

MTP041: SUBTST <<MTP041 ADDRESS TO CSR ON DOUBLE BIT ERROR TEST>>
 :*****
 :*SUBTEST MTP041 ADDRESS TO CSR ON DOUBLE BIT ERROR TEST
 :*****

10437

10438

10439

10440

10441 037030

10442 037034 072427 000011

10443 037040 042704 170037

10444 037044

10445 037050

10446 037060

10447 037064 005037 002312

10448 037070 104475

10449 037072

10450 037072 105237 002262

10451 037076

10452 037102

10453 037106

10454 037114 104475

10455 037116

10456 037120 104503

10457 037122 005711

10458 037124 104426

10459 037126

10460 037132 042705 170037

10461 037136

10462 037140 060402

10463 037142 000240

10464 037144

10465 037150

10466 037154

10467 037160 104455

10468 037162

10469 037162

10470 037164 104475

10471 037166

10472 037176 104503

10473 037200 000207

10474

: THIS TEST CHECKS TO SEE IF THE CORRECT ADDRESS APPEARS
 : IN CSR BITS 5-11 ON A DOUBLE ERROR.

```

LET R4 := BANK           ;GET STARTING BANK NUMBER
ASH #9,R4                ;SHIFT INTO POSITION TO MATCH ADDRESS IN CSR
BIC #^C7740,R4          ;CLEAR OFF EXTRANEIOUS BITS
LET R0 := #-40           ;INIT CSR ADDRESS TO 0 - 1K (BIT 5 = 1K ADD.)
LET R1 := #FIRST - #4000 ;GET LOW ADDRESS IN BANK
LET PASFLG :B= #0       ;INIT PASFLG
CLR CHECK                ;CLEAR CHECK BIT FIELD TO BE LOADED
CB1CSR                   ;ENABLE CHECK/SYNDROME BIT REGISTER
REPEAT
  INCB PASFLG            ;INC LOOP COUNTER
  LET R0 := R0 + #40     ;INC CSR ADDRESS TO BE EXPECTED
  LET R1 := R1 + #4000
  LET CHECK := #1340    ;DOUBLE ERROR CHECK BITS
  CB1CSR                 ;WRITE DOUBLE ERROR CHECK BITS
  LET (R1) := #0        ;WRITE DATA AND D.E. CHK BITS AT A=0
  CLR1CSR                ;CLEAR CSR
  TST (R1)               ;READ ADDRESS TO GET DOUBLE ERROR
  READCSR                ;READ CSR FOR CORRECT ADDRESS
  LET R5 := CSR
  BIC #^C7740,R5
  LET R2 := R0
  ADD R4,R2
  NOP
  IF R2 NE R5
    LET BAD := R2
    LET GOOD := R5
    PERR31
  END
  LET (R1) := #0
  CB1CSR
UNTILB PASFLG EQ #16.
CLR1CSR
RETURN
    
```


CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 321
 MTP041 ADDRESS TO CSR ON DOUBLE BIT ERROR TEST

10477 037202

MTP042: SUBTST <<MTP042 EXTENDED ADDRESS TO CSR ON ERROR TEST>>

 :SUBTEST MTP042 EXTENDED ADDRESS TO CSR ON ERROR TEST

THIS TESTS THE EXTENDED UNIBUS ADDRESS IN THE
 CSR BY CAUSING A SINGLE ERROR, ENABLING BIT # 14, THEN CHECKING
 FOR THE PROPER ADDRESS IN THE CSR.

10478

10479

10480

10481

10482

10483 037202 104424

10484 037204

10485 037210 042704 177607

10486 037214 072427 000002

10487 037220 052704 040000

10488 037224 062737 000400 172352

10489 037232

10490 037236

10491 037242 042705 177770

10492 037246 072527 000011

10493 037252 052705 000020

10494 037256 104513

10495 037260

10496 037260 105237 002262

10497 037264

10498 037272 104425

10499 037274

10500 037276 104503

10501 037300 005711

10502 037302 104426

10503 037304 042737 020000 002146

10504 037312

10505 037320

10506 037326

10507 037332 104023

10508 037334

10509 037334

10510 037342 104425

10511 037344 104426

10512 037346 042737 020000 002146

10513 037354

10514 037362

10515 037370

10516 037374

10517 037402 104023

10518 037404

10519 037404

10520 037406 062705 000740

10521 037412 104513

10522 037416 104513

10523 037420

10524 037430 104503

10525 037432 104423

10526 037434 000207

10527

10528

CACHOFF

LET R4 := BANK

BIC #C170,R4

ASH #2,R4

BIS #BIT14,R4

ADD #400,KIPAR5

LET PASFLG :B= #0

LET R5 := BANK

BIC #C7,R5

ASH #9,R5

BIS #BIT4,R5

CBREG

REPEAT

INCB PASFLG

LET CSR := #104

LOADCSR

LET (R1) := #0

CLR1CSR

TST (R1)

READCSR

BIC #BIT13,CSR

IF CSR NE R5 THEN

LET BAD := CSR

LET GOOD := R5

ERROR +23

END

LET CSR := #40000

LOADCSR

READCSR

BIC #BIT13,CSR

IF CSR NE R4 THEN

LET BAD := CSR

LET GOOD := R4

SET HEADER

ERROR +23

END

LET (R1) := #0

LET R1 := #137776

ADD #740,R5

CBREG

UNTILB PASFLG EQ #2

CLR1CSR

CACHON

RETURN

:TURN OFF CACHE MEMORY

:GET BANK NUMBER TO FIGURE OUT EXTENDED ADDRESS

:CLEAR OFF LOWER BITS

:SHIFT TO LINE UP WITH CSR

:SET EXTENDED ADDRESS BIT

:SET UP PAR TO POINT TO TOP OF A BANK

:INIT LOOP COUNTER

:R5 GETS THE BANK NUMBER

:CLEAR ALL BUT THE LOWER BITS

:ROTATE INTO POSITION

:SET UP SBE INDICATOR ;;DATA TO BE EXPECTED

:ENABLE CHECK/SYNDROME BIT REGISTER

:INCR LOOP COUNTER

:WRITE CHECK BITS TO CSR WITH DIAG MODE

:LOAD CSR WITH DATA

:WRT ZEROS AT A=0 AND SINGLE ERROR BITS

:CLEAR CSR

:READ A=0;DATA BIT 0 SHOULD BE CORRECTED TO A 1

:READ CSR FOR DATA

:CLEAR POSSIBLE INHIBIT MODE IN DATA "CSR"

:HAS SINGLE ERROR BITS SET IN CSR?

:WRITE EUB BIT TO CSR

:READ FOR CORRECT EXTENDED UNIBUS ADDRESS

:CLEAR INHIBIT MODE POINTER IN DATA

:READ EUB ADDRESS

:CLEAR LUT

:SET UP NEW ADDRESS

:ADD TO GET NEW ADDRESS

:ENABLE CHECK/SYNDROME BIT REGISTER

:LOOP 2 TIMES

:CLEAR CSR

:TURN ON CACHE

10530 037436

MTP043: SUBTST <<MTP043 WRITE BYTE CLEARS SINGLE BIT ERROR TEST>>

 : *SUBTEST MTP043 WRITE BYTE CLEARS SINGLE BIT ERROR TEST

10531

: THIS TEST CHECKS TO SEE IF A SINGLE BIT ERROR WILL BE CORRECTED DURING
 : THE READ PORTION OF A WRITE BYTE AND THAT THE CORRECT CHECK BITS WILL
 : BE GENERATED ON A WRITE.

10532

10533

10534

10535

10536 037436 104424

CACHOFF ;TURN OFF CACHE

10537 037440 104513

CBREG ;ENABLE CHECK/SYNDROME BIT REGISTER

10538 037442 105037 002262

CLRB PASFLG ;CLEAR LOOP COUNTER

10539 037446

LET R2 := R1 + #1 ;R2 POINTS TO HIGH BYTE

10540 037452

LET R4 := #1 ;INITIAL DATA = 1

10541 037456

REPEAT

10542 037456 105237 002262

INCB PASFLG ;INCREMENT LOOP COUNTER

10543 037462

LET CSR := #604 ;WRITE CHECK BITS CORRESPONDING TO DATA OF 0

10544 037470 104425

LOADCSR ;WRITE CSR

10545 037472

LET (R1) := R4 ;WRITE DATA OF 1 CREATING A SINGLE BIT ERROR

10546 037474 104503

CLR1CSR ;WRITE CSR TO NORMAL MODE

10547 037476

LET (R2) :B= #377 ;WRITE BYTE OF WORD

10548 037502 104426

READCSR ;READ CSR

10549 037504 042737 177757 002146

BIC #*C20,CSR ;SEE IF SBE INDICATOR IS SET

10550 037512

IF CSR NE #20 ;IS SBE SET?????

10551 037522

LET GOOD := #20

10552 037530

LET BAD := CSR

10553 037536 104060

ERROR +60

10554 037540

END

10555 037540 104513

CBREG ;WRITE CSR TO DIAG MODE

10556 037542 005711

TST (R1) ;READ SA0 FOR CORRECT CHECK BITS

10557 037544 104426

READCSR ;READ CSR

10558 037546 042737 174037 002146

BIC #*C3740,CSR ;MASK OUT CHECK BIT FIELD

10559 037554

IF CSR NE #300 ;WERE CORRECT CHECK BITS GENERATED?????

10560 037564

SET HEADER

10561 037572

LET GOOD := #300

10562 037600

LET BAD := CSR

10563 037606 104061

ERROR +61

10564 037610

END

10565 037610 005302

DEC R2 ;POINT TO HIGH BYTE AND REPEAT

10566 037612

LET R4 := #400 ;BIT 0 OF HIGH BYTE

10567 037616

UNTILB PASFLG EQ #2 ;DO HIGH AND LOW BYTE

10568 037626 104423

CACHON ;TURN ON CACHE

10569 037630 000207

RETURN

CZMSPBC MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 323-1
 MTP044 SHIFTING CHECK BITS THROUGH THE CSR TEST

10625	040036		END		
10626	040036	040502	BIC R5,R2		:SHIFT CHECK BITS AND CREATE NEW DATA FOR CSR
10627	040040		IFB PASFLG EQ #1		:SELECT FUNCTION
10628					:DO A FIELD OF ZEROS--->ONES
10629	040050	006302	ASL R2		:SHIFT CHECK BITS
10630	040052		ELSE		:DO A FIELD OF ONES --->ZEROS
10631	040054	005105	COM R5		:TAKE OUT CHECK BIT FIELD
10632	040056	010546	MOV R5,-(SP)		:
10633	040060	040216	BIC R2,(SP)		:
10634	040062	040502	BIC R5,R2		:
10635	040064	052602	BIS (SP)+,R2		:
10636	040066	006302	ASL R2		:SHIFT CHECK BITS
10637	040070	010546	MOV R5,-(SP)		:PUT BACK CHECK BIT FIELD
10638	040072	040216	BIC R2,(SP)		:
10639	040074	040502	BIC R5,R2		:
10640	040076	052602	BIS (SP)+,R2		:
10641	040100	005105	COM R5		:COMPLEMENT DATA PATTERN
10642	040102		END		
10643	040102		LET R2 := R2 + #6		:ADD 6 SO THAT WRITE ON CSR WILL ENABLE DIAG MODE
10644	040106		UNTILB PASSNO EQ #6		:DO ALL CHECK BITS
10645	040116		LET R2 := #3706		:REPEAT WITH FIELD OF ONES
10646	040122		UNTILB PASFLG EQ #2		:
10647	040132	104503	CLR1CSR		:
10648	040134	005011	CLR (R1)		
10649	040136	062701 000100	ADD #100,R1		:ALL 256 TO ADDRESS
10650	040142		UNTIL R1 EQ #LAST+2		:ILC::REVB
10651	040150	104423	CACHON		:ILC::REVB
10652	040152	000207	RETURN		:

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 324
MTP044 SHIFTING CHECK BITS THROUGH THE CSR TEST

10654 040154

MTP045: SUBTST <<MTP045 SYNDROMES TO CSR ON DOUBLE BIT ERROR TEST>>

: *SUBTEST MTP045 SYNDROMES TO CSR ON DOUBLE BIT ERROR TEST

10655
10656
10657
10658
10659

:
: THIS TEST CHECKS TO SEE IF THE DOUBLE BIT ERROR INDICATOR IS SET
: ON A DOUBLE BIT ERROR AND THE CORRECT SYNDROMES ARE LATCHED INTO THE
: CSR. THIS TEST IS THEN REPEATED WITH MULTIPLE ERROR CHECK/SYNDROME BITS

10660 040154 104424
10661 040156 104513

: CACHOFF :TURN OFF CACHE
: CBREG :ENABLE CHECK/SYNDROME BIT REGISTER
: LET PASSNO := #0 :CLEAR LOOP COUNTER
: LET GOOD := #3744 :GOOD DATA
: LET CSR := #3144 :DBE CHECK BITS FOR CSR
: REPEAT

10666 040200 005237 002264
10667 040204 104425

: INC PASSNO
: LOADCSR :WRITE DBE CHECK BITS TO CSR
: LET (R1) := #0 :WRITE ZEROS AND DBL ERROR CHK BITS A=0
: CLR1CSR :CLEAR CSR OUT
: TST (R1) :READ A=0 TO GET DOUBLE BIT ERROR
: READCSR :WAS UNCORRECTABLE ERROR BIT SET???

10668 040206
10669 040210 104503
10670 040212 005711
10671 040214 104426

: IF #BIT15 OFF.IN CSR
: SET HEADER
: LET BAD := CSR
: ERROR +63 :BIT NOT SET

10672 040216
10673 040226
10674 040234
10675 040242 104063
10676 040244

: END
: CBREG :ENABLE SYNDROME BIT REGISTER
: READCSR :READ CSR FOR CORRECT SYNDROME BITS
: NOP :DEBUG AIDE
: BIC #^C3744,CSR :MASK SYNDROMES OUT
: IF CSR NE GOOD THEN :CHECK IF DOUBLE ERROR BIT IS SET
: LET BAD := CSR :BAD DATA
: SET HEADER
: ERROR +42

10677 040244 104513
10678 040246 104426
10679 040250 000240
10680 040252 042737 174033 002146

: END
: CLR (R1) :CLEAR LUT
: LET GOOD := #3604 :REPEAT WITH MULTIPLE ERROR SYNDROMES
: LET CSR := #3004 :MULTIPLE ERROR CHECK BITS

10681 040260
10682 040270
10683 040276
10684 040304 104042
10685 040306
10686 040306 005011

: UNTIL PASSNO EQ #2
: CLR1CSR
: CACHON
: RETURN

10687 040310
10688 040316
10689 040324
10690 040334 104503
10691 040336 104423
10692 040340 000207

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 325
 MTP045 SYNDROMES TO CSR ON DOUBLE BIT ERROR TEST

10694 040342

MTP046: SUBTST <<MTP046 CHECK SINGLE BIT ERRORS WITH ECC DISABLED>>

```
*****
:SUBTEST      MTP046 CHECK SINGLE BIT ERRORS WITH ECC DISABLED
*****
```

10695

10696

10697

10698

10699

10700 040342 005037 002264

10701 040346 104424

10702 040350

10703 040350

10704 040354 005000

10705 040356 105037 002262

10706 040362 104513

10707 040364

10708 040364

10709 040370

10710 040374

10711 040402

10712 040406

10713 040416

10714 040422

10715 040424

10716 040430

10717 040430

10718 040430 005237 002322

10719 040434

10720 040440

10721 040442 072227 000005

10722 040446 052702 000006

10723 040452

10724 040456 104425

10725 040462

10726 040464

10727 040466

10728 040476 104471

10729 040500

10730 040502 104507

10731 040504

10732 040504 005711

10733 040506 004737 040626

10734 040512 104426

10735 040514

10736 040524

10737 040532

10738 040540 104045

10739 040542

10740 040542 104503

10741 040544 005011

10742 040546

10743 040556 006305

10744 040560

10745 040562 000261

10746 040564 006105

10747 040566

```

:
: THIS TEST CHECKS TO SEE THAT FOR EACH BIT OF A DATA WORD THAT A SBE
: IS TREATED LIKE A UNCORRECTABLE ERROR WITH ECC DISABLED AND TRAPS
: ARE DETECTED.
:
CLR PASSNO      ;CLEAR OUTER LOOP COUNTER
CACHOFF        ;TURN OFF CACHE
REPEAT
LET PASSNO := PASSNO + #1
CLR R0        ;CLEAR DATA
CLRB PASFLG   ;CLEAR PASFLG
CBREG        ;ENABLE CHECK/SYNDROME BIT REGISTER
REPEAT
LET PASFLG :B= PASFLG + #1 ;INCREMENT LOOP COUNTER
LET R4 := #-1 ;INDEX TO SINGLE BIT ERROR TABLE
LET NOPAR := #1 ;ENABLE PARITY ACTION
LET BITNO := #0 ;CLEAR INNER LOOP COUNTER
IFB PASFLG EQ #1 ;SELECT DATA TO BE CORRECTED BY PASSNO
LET R5 := #1 ;DATA=0;BIT TO BE CORRECTED IS A ONE
ELSE ;DATA=177776;BIT TO BE CORRECTED IS A ZERO
LET R5 := #177776
END
REPEAT
LET PARCNT := #0 ;CLEAR PARITY COUNTER
INC BITNO ;INCREMENT BIT POINTER
LET R4 := R4 + #1 ;POINT TO NEXT SET OF CHECK BITS
LET R2 :B= PTABLE(R4) ;GET NEXT SET OF CHECK BITS
ASH #5,R2 ;SHIFT TO LINE UP IN CSR
BIS #BIT2!BIT1,R2 ;ENABLE DIAG MODE
LET CSR := R2 ;GET CHECK BITS TO BE WRITTEN
LOADCSR ;LOAD CSR WITH DATA
LET (R1) := R0 ;WRITE DATA TO TEST ADDRESS
IF PASSNO EQ #1 ;WRITE CSR
ECCIDIS ;FIRST PASS ;ECC DISABLE,NO PBL
ELSE ;SECOND PASS ;ECC DISABLE,PBL ENABLED
ENA1SBE
END
TST (R1) ;CORRECT SBE
CALL CHKTRP ;CHECK FOR CORRECT TRAP
READCSR ;READ THE CSR FOR UNCORRECTABLE ERROR
IF #BIT15 OFF.IN CSR ;IS UNCORRECTABLE ERROR BIT SET????
LET BAD := CSR
SET HEADER
ERROR +45
END
CLR1CSR
CLR (R1)
IFB PASFLG EQ #1 ;CLEAR LUT
ASL R5 ;SHIFT NEW DATA DEPENDING ON PASFLG
ELSE ;SHIFT BITNO TO THE LEFT
SEC ;SET CARRY BIT AND.....
ROL R5 ;ROTATE LEFT
END

```


CZMSPB0 MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 325-1
 MTPO46 CHECK SINGLE BIT ERRORS WITH ECC DISABLED

10748	040566		UNTIL BITNO EQ #16.	:UNTIL ALL BITS ARE DONE
10749	040576	005100	COM R0	:COMPLEMENT DATA AND REPEAT
10750	040600		UNTILB PASFLG EQ #2	:UNTIL 2 PASSES ARE COMPLETE!
10751	040610		UNTIL PASSNO EQ #2	:
10752	040620	104503	CLR1CSR	:
10753	040622	104423	CACHON	:TURN CACHE
10754	040624	000207	RETURN	:
10755				
10756				
10757	040626		CHKTRP: IF PASSNO EQ #1	:PASS 1 CHECK FOR NO TRAP
10758	040636		IF PARCNT EQ #1	:
10759	040646		SET HEADER	:
10760	040654	104057	ERROR +57	:
10761	040656		END	:
10762	040656		ELSE	:
10763	040660		IF PARCNT NE #1	:
10764	040670		SET HEADER	:
10765	040676	104062	ERROR +62	:
10766	040700		END	:
10767	040700		END	:
10768	040700	000207	RETURN	:

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 326
 MTP046 CHECK SINGLE BIT ERRORS WITH ECC DISABLED

10770 040702

MTP047: SUBTST <<MTP047 NO CSR UPDATE ON SBE WITH EXSISTING DBE>>

 :*SUBTEST MTP047 NO CSR UPDATE ON SBE WITH EXSISTING DBE

10771
 10772
 10773
 10774
 10775

:
 : THIS TEST CHECKS TO SEE THAT THE CSR CONTENTS WILL NOT CHANGE
 : WITH A SINGLE BIT ERROR WHEN A DOUBLE BIT ERROR ALREADY
 : EXISTS.
 :

10776 040702 104424

CACHOFF

:TURN OFF CACHE

10777 040704

LET R4 := BANK

:GET BANK NUMBER

10778 040710 072427 000011

ASH #9,R4

:SHIFT INTO PLACE

10779 040714 042704 170037

BIC #^C7740,R4

:MASK OUT UNWANTED BITS

10780 040720 052704 100000

BIS #BIT15,R4

:SET UP GOOD DATA

10781 040724 104513

CBREG

:ENABLE CHECK/SYNDROME BIT REGISTER

10782 040726

LET CSR := #3144

:CHECK BITS FOR DOUBLE BIT ERROR

10783 040734 104425

LOADCSR

10784 040736

LET (R1) := #0

:WRITE DBE CHECK BITS

10785 040740

LET CSR := #104

:WRITE SBE CHECK BITS

10786 040746 104425

LOADCSR

10787 040750

LET (R2) := #0

:WRITE SBE CHECK BITS AT ADDRESS + 4K

10788 040752 104503

CLR1CSR

:CLEAR CSR

10789 040754 005711

TST (R1)

:READ DBE LOCATION

10790 040756 104426

READCSR

:READ FOR CSR DBE INDICATOR

10791 040760 042737 020000 002146

BIC #BIT13,CSR

:CLEAR INHIBIT MODE POINTER

10792 040766

IF CSR NE R4

10793 040774

LET BAD := CSR

10794 041002

LET GOOD := R4

10795 041006

SET HEADER

10796 041014 104063

ERROR +63

10797 041016

END

10798 041016 052704 000020

BIS #20,R4

:SET BIT IN GOOD DATA

10799 041022 005712

TST (R2)

:READ SBE

10800 041024 104426

READCSR

:READ CSR FOR NO CHANGE

10801 041026 042737 020000 002146

BIC #BIT13,CSR

:CLEAR INHIBIT MODE POINTER

10802 041034

IF CSR NE R4

10803 041042

LET BAD := CSR

10804 041050

LET GOOD := R4

10805 041054

SET HEADER

10806 041062 104051

ERROR +51

10807 041064

END

10808 041064 104503

CLR1CSR

:CLEAR 1 CSR

10809 041066 005011

CLR (R1)

10810 041070 005012

CLR (R2)

10811 041072 104423

CACHON

:TURN ON CACHE

10812 041074 000207

RETURN

10814
10815
10816 041076

.SBTTL MISC SUBROUTINES

REGCOPY:SUBTST <<SUBR COPY R0 TO R4,R1 TO R3, & R2 TO R5>>
:*****
:*SUBTEST SUBR COPY R0 TO R4,R1 TO R3, & R2 TO R5
:*****

10817 041076 010004
10818 041100 010103
10819 041102 010205
10820 041104 000207
10821
10822 041106

MOV R0,R4
MOV R1,R3
MOV R2,R5
RETURN

FLIPWARN:SUBTST <<FLIP WARNING CONSTANTS IN WORST CASE NOISE TESTS>>
:*****
:*SUBTEST FLIP WARNING CONSTANTS IN WORST CASE NOISE TESTS
:*****

10823 041106
10824 041110 005237 002604
10825 041114 042737 177774 002604
10826 041122 022737 000001 002604
10827 041130 001414
10828 041132 022737 000002 002604
10829 041140 001413
10830 041142 022737 000003 002604
10831 041150 001414
10832 041152 005000
10833 041154 013704 002602
10834 041160 000414
10835 041162
10836 041166 000411
10837 041170 012700 000401
10838 041174 013704 002602
10839 041200 000404
10840 041202 012700 000401
10841 041206 012704 000401
10842 041212 010037 027526
10843 041216 010037 027542
10844 041222 010037 027566
10845 041226 010037 027602
10846 041232
10847 041234 000207

PUSH R0
INC FLIPLOC
BIC #^C3,FLIPLOC
CMP #1,FLIPLOC
BEQ 1\$
CMP #2,FLIPLOC
BEQ 2\$
CMP #3,FLIPLOC
BEQ 3\$
CLR R0
MOV ONES,R4
BR 4\$
1\$: CLEAR R0,R4
BR 4\$
2\$: MOV #401,R0
MOV ONES,R4
BR 4\$
3\$: MOV #401,R0
MOV #401,R4
4\$: MOV R0,WARN2
MOV R0,WARN3
MOV R0,WARN4
MOV R0,WARN5
POP R0
RETURN

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 328
FLIP WARNING CONSTANTS IN WORST CASE NOISE TESTS

10849 041236

BACKGND:SUBST <<SUBR WRITE BACKGROUND>>

:SUBTEST SUBR WRITE BACKGROUND

10850

:WRITES DATA FROM R2

10851 041236 104415

SAVREG

10852 041240 012700 060000

MOV #FIRST,R0

10853 041244 012701 040000

MOV #SIZE,R1

10854 041250 022737 000001 003754

CMP #1,PROTYP

10855 041256 001415

BEQ WARN6B

10856 041260 012737 000207 027410 WARN6A:

MOV #207,MTP000+4

:WARNING PUTTING 'RETURN' AFTER WRITE

10857 041266 012737 027404 002260

MOV #MTP000,SUPDOADD

10858 041274 004737 027212

CALL SUPD03

10859 041300 012737 000240 027410

MOV #240,MTP000+4

:RESTORE 'NOP' AFTER WRITE

10860 041306 104416

RESREG

10861 041310 000207

RETURN

10862 041312 000207 177644 WARN6B:

BMOV MTP000

10863 041320 012737 000207 027034 WARN6:

MOV #207,UIPAR2

:WARNING PUTTING 'RETURN' INSTRUCTION AFTER WRITE

10864 041326 004737 027034

CALL SUPD01

10865 041332 104416

RESREG

10866 041334 000207

RETURN

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 329
SUBR WRITE BACKGROUND

10868 041336

GETCSR: SUBTST <<SUBR GET CSR INFORMATION FROM CONFIGURATION TABLE>>
:*****
:SUBTEST SUBR GET CSR INFORMATION FROM CONFIGURATION TABLE
:*****

10869

10870

10871

10872

10873 041336 013702 002102

10874 041342 016203 002652

10875 041346 000303

10876 041350 006303

10877 041352 042703 177741

10878 041356 010337 002150

10879 041362 000207

:INPUTS : NONE
:
:OUTPUT : CSRNO = CSR NUMBER
:
MOV BANKINDEX,R2 :GET INDEX INTO CONFIG TABLE
MOV CONFIG(R2),R3 :MOV IT INTO R3
SWAB R3 :
ASL R3 :
BIC #^C36,R3 :CLEAR OFF SOME BITS
MOV R3,CSRNO :SAVE CSR NUMBER
RETURN

CZMSP80 MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 331
 SUBR GET CSR INFORMATION FROM CONFIGURATION TABLE

10882 041364

PCONFIG:SUBTST <<SUBR PRINT CONFIGURATION MAP>>

 :*SUBTEST SUBR PRINT CONFIGURATION MAP

```

10883 041364          PUSH    TKVEC,TKVEC+2,R0
10884 041376 010637 041664          MOV     SP,PCONFS          ;SAVE LAST GOOD SP
10885 041402 012737 041632 000060          MOV     #PCONF2,TKVEC
10886 041410 012737 000340 000062          MOV     #340,TKVEC+2
10887 041416 017700 141210          MOV     @STKB,R0          ;KILL ANY OLD INTERRUPT
10888 041422 042737 000200 177776          BIC     #BIT7,PSW         ;LOWER CPU PRIORITY TO 140
10889 041430 052777 000100 141172          BIS     #BIT6,@STKS      ;ENABLE KEYBOARD INTERRUPTS
10890
10891 041436          TYPE    MSG001
10892 041442          TYPE    MSG002
10893 041446          TYPE    MSG003
10894 041452 022737 000060 002554          CMP     #60,LASTBANK
10895 041460 002006          BGE     NOOJ
10896          ;IF FAT PAPER ON TERMINAL GOTO 1$
10897 041462          IF #SW4 SET.IN @SWR THEN JUMPTO PCONF1
10898 041476 012700 000074          NOOJ:  MOV     #60.,R0
10899 041502 010004          MOV     R0,R4
10900 041504          CLEAR  R1,R3
10901 041510          TYPE    MSG004
10902 041514 004737 041666          CALL   TCONFIG          ;GO TYPE CONFIGURATION (1ST HALF)
10903 041520 022737 000060 002554          CMP     #60,LASTBANK
10904 041526 002041          BGE     PCONF2
10905 041530          TYPE    $CRLF
10906 041534          TYPE    MSG017          ;PRINT SPACE(S)
10907 041540          TYPE    MSG011
10908 041544          TYPE    $CRLF
10909 041550          TYPE    MSG017          ;PRINT SPACE(S)
10910 041554          TYPE    MSG012
10911 041560 012701 000360          MOV     #60.*2+2,R1
10912 041564 010103          MOV     R1,R3
10913 041566 004737 041666          CALL   TCONFIG
10914 041572 000417          BR     PCONF2
10915
10916 041574 012700 000170          PCONF1: MOV    #120.,R0
10917 041600 010004          MOV    R0,R4
10918 041602          CLEAR R1,R3
10919 041606          TYPE  MSG014          ;SPACE
10920 041612          TYPE  MSG011
10921 041616          TYPE  MSG004
10922 041622          TYPE  MSG012
10923 041626 004737 041666          CALL  TCONFIG
10924
10925 041632 013706 041664          PCONF2: MOV    PCONFS,SP          ;RESTORE STACK
10926 041636 042777 000100 140764          BIC    #BIT6,@STKS
10927 041644 117700 140762          MOVB  @STKB,R0          ;READ CHAR TO KILL FLAG
10928 041650          POP   R0,TKVEC+2,TKVEC
10929 041662 000207          RETURN
10930
10931 041664 000000          PCONFS: 0          ;STACK SAVED HERE!
    
```


10934 041666

```

SUBTST <<SUBR TYPE CONFIGURATION>>
:*****
:*SUBTEST      SUBR      TYPE CONFIGURATION
:*****
:CALL:  MOV      #N,R0      ;N=NUMBER OF CHARACTERS
:        MOV      R0,R4      ;BACKUP
:        MOV      #K,R1      ;INDEX CONSTANT
:        MOV      R1,R3      ;BACKUP
:        CALL     TCONFIG     ;ACTUAL CALL
:        RETURN     ;ONLY RETURN
:*****

```

10935
10936
10937
10938
10939
10940
10941
10942
10943
10944
10945
10946
10947
10948
10949
10950
10951
10952
10953
10954
10955
10956
10957

041666 012737 000340 177776
041674
041700 032761 000001 002652
041706 001403
041710
041714 000402
041716
041722 062701 000004
041726 077014
041730 010400
041732 010301

```

:*****
:** ERROR **
:*****
TCONFIG:MOV      #340,PSW      ;DISABLE INTERUPTS
TYPE      MSG005
1$: BIT      #BIT0,CONFIG(R1) ;ERROR ON THIS BANK?
BEQ      2$      ;NO - SKIP
TYPE      MSG013      ;PRINT 'X'
BR      3$
2$: TYPE      MSG014      ;PRINT SPACE
3$: ADD      #4,R1      ;BUMP POINTER
SOB      R0,1$      ;LOOP UNTIL DONE
MOV      R4,R0
MOV      R3,R1

```

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 335
 SUBR TYPE CONFIGURATION

```

10960
10961
10962
10963 041734
10964
10965 041740 012737 000340 177776 TCFIG1:
10966 041746 032761 010000 002654
10967 041754 001014
10968 041756 032761 000002 002652
10969 041764 001004
10970 041766 112737 000040 074750
10971 041774 000424
10972 041776 112737 000055 074750 18$:
10973 042004 000420
10974 042006 016105 002652 1$:
10975 042012 042705 007777
10976 042016 000305
10977 042020 072527 177774
10978 042024 022705 000011
10979 042030 100002
10980 042032 062705 000007
10981 042036 062705 000060 2$:
10982 042042 110537 074750
10983 042046 16$:
10984 042052
10985 042062 062701 000004
10986 042066 077054
10987 042070 010400
10988 042072 010301
10989
10990
10991
10992
10993
10994 042074
10995 042100 033761 002104 002652 TCFIG2:
10996 042106 001432
10997 042110 016105 002654
10998 042114 000305
10999 042116 042705 177770
11000 042122 020527 000003
11001 042126 003022
11002 042130
11003 042136
11004 042144 112737 000120 074750
11005 042152
11006 042154 112737 000115 074750
11007 042162
11008 042162
11009 042164 112737 000114 074750
11010 042172
11011 042172 000403
11012 042174 112737 000040 074750 17$:
11013 042202 8$:
11014 042206
11015 042216 062701 000004
11016 042222 077052

:*****
:** INTERLEAVE **
:*****
TYPE MSG007
:THIS IS AN ENTRY POINT FROM ERROR REPORTS
:DISABLE INTERRUPTS
MOV #340,PSW
BIT #BIT12,CONFIG+2(R1)
BNE 1$
BIT #BIT1,CONFIG(R1) ;IS THERE ANY MEMORY HERE?
BNE 18$ ;BRANCH IF MEMORY PRESENT.
MOVB #' ,MSG015 ;MOVE A BLANK IN TO BE PRINTED
BR 16$ ;BRANCH TO TYPE ROUTINE
MOVB #'- ,MSG015
BR 16$
MOV CONFIG(R1),R5 1$:
BIC #^C170000,R5 ;GET CSR INTERLEAVE
SWAB R5
ASH #-4,R5
CMP #9,R5
BPL 2$
ADD #7,R5
ADD #60,R5 ;MAKE ASCII
MOVB R5,MSG015 ;PLUG INTO MEMORY
TYPE MSG015
IF NOTAB NE #0 THEN $RETURN
ADD #4,R1 ;BUMP POINTER
SOB R0,TCFIG1 ;LOOP UNTIL DONE
MOV R4,R0
MOV R3,R1

:*****
:** MEMORY TYPE **
:*****
.ENABL LSB
TYPE MSG009
BIT CPUBIT,CONFIG(R1)
BEQ 17$
MOV CONFIG+2(R1),R5
SWAB R5 ;GET MEMORY TYPE
BIC #^C7,R5 ;CLEAR NON INTERESTING BITS
CMP R5,#3 ;IS IT A LEGAL MEMORY TYPE
BGT 17$ ;IF IF SO BRANCH!!!!!!
IF #BIT0 SET.IN R5 ;IS IT AN ECC MEMORY????
IF #BIT1 SET.IN R5 ;IS IT A MS11-P OR A MS11-M???
MOVB #'P,MSG015 ;IT IS A MS11-P
ELSE
MOVB #'M,MSG015 ;IT IS A MS11-M
END
ELSE
MOVB #'L,MSG015 ;IT IS A MS11-L
END
BR 8$
MOVB #' ,MSG015
TYPE MSG015
IF NOTAB NE #0 THEN $RETURN
ADD #4,R1 ;BUMP POINTER
SOB R0,TCFIG2 ;LOOP UNTIL DONE

```

CZMSP80 MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 335-1
 SUBR TYPE CONFIGURATION

```

11017 042224 010400      MOV      R4,R0
11018 042226 010301      MOV      R3,R1
11019                      .DSABL   LSB
11020
11021                      :*****
11022                      **: CSR **
11023                      :*****
11024 042230              TYPE      MSG016
11025 042234 112737 000040 074750 TCFIG3: MOVB     #' ,MSG015
11026 042242 016105 002652      MOV     CONFIG(R1),R5
11027 042246 032705 000002      BIT     #BIT1,R5
11028 042252 001414      BEQ     16$
11029 042254 042705 170377      BIC     #^C7400,R5
11030 042260 000305      SWAB   R5
11031 042262 022705 000011      CMP     #9.,R5
11032 042266 100002      BPL     10$
11033 042270 062705 000007      ADD     #7,R5
11034 042274 062705 000060      ADD     #60,R5          ;MAKE ASCII
11035 042300 110537 074750      MOVB   R5,MSG015       ;PLUG INTO MEMORY
11036 042304              TYPE      MSG015
11037 042310              IF NOTAB NE #0 THEN $RETURN
11038 042320 062701 0000( 4      ADD     #4,R1          ;BUMP POINTER
11039 042324 077035
11040 042326 010400
11041 042330 010301
11042
11043                      :*****
11044                      **: PROTECTED **
11045                      :*****
11046 042332              TYPE      MSG010
11047 042336 105761 002652          11$:  TSTB   CONFIG(R1)       ;BANK PROTECTED?
11048 042342 100004          BPL     12$           ;NO - SKIP
11049 042344 112737 000120 074750      MOVB   #'P,MSG015
11050 042352 000407          BR     13$
11051 042354 032761 000100 002652      12$:  BIT     #BIT6,CONFIG(R1)  ;PROTECTED REGION OF ECC?
11052 042362 001406          BEQ     14$           ;NO - SKIP
11053 042364 112737 000111 074750      MOVB   #'I,MSG015
11054 042372          13$:  TYPE      MSG015
11055 042376 000402          BR     15$
11056 042400          14$:  TYPE      MSG014       ;PRINT SPACE
11057 042404 062701 000004      15$:  ADD     #4,R1       ;BUMP POINTER
11058 042410 077026          SOB     R0,11$       ;LOOP UNTIL DONE
11059 042412 010400      MOV     R4,R0
11060 042414 010301      MOV     R3,R1
11061 042416 000207      RETURN

```


CZMSPB0 MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 337
 TRAP PARITY ERROR HANDLER

11064
 11065
 11066
 11067
 11068
 11069
 11070
 11071
 11072
 11073
 11074
 11075
 11076
 11077
 11078
 11079
 11080
 11081
 11082
 11083
 11084
 11085
 11086
 11087
 11088
 11089
 11090
 11091
 11092
 11093

```

.SBTTL TRAP PARITY ERROR HANDLER
:*****
:VECTOR TO HERE FROM TRAPS TO 114
:IGNORE ERRORS BUT COUNT IF NOPAR FLAG = 1.
:*****
:
:      CODE      ACTION
:      ---
:      0          PRINT UNEXPECTED PARITY TRAP
:      1          COUNT ERROR
:      2          SET 'ABORT' / SETUP 'BADPC' / RETURN VIA PCBUMP
:      3          RETURN VIA 'PARTHERE'
:
:      PARITY:  CMP      #1,NOPAR          ;COUNTING PARITY ERRORS?
:              BNE      1$                ;NO - SKIP
:              INC      PARCNT            ;PARITY ERROR COUNTER + 1
:              RTI
:      1$:      CMP      #2,NOPAR          ;ACTION CODE = 2 ?
:              BNE      2$                ;NO - SKIP
:              SET      ABORTFLAG         ;YES
:              CALL     BADSTACK           ;FIND BAD SP,PC,PSW OFF STACK
:              ADD      PCBUMP,(SP)       ;UPDATE RETURN PC
:              BIC      #BIT2,2(SP)       ;SHOW FAILURE BY .NE.
:              RTI
:      2$:      CMP      #3,NOPAR          ;ACTION CODE = 3 ?
:              BNE      3$                ;NO - SKIP
:              MOV      PARTHERE,(SP)
:              RTI
:      3$:      CALL     BADSTACK           ;FIND BAD SP,PC,PSW OFF STACK
:              FATAL   32
    
```

042420 022737 000001 002074
 042426 001003
 042430 005237 002070
 042434 000002
 042436 022737 000002 002074
 042444 001013
 042454 004737 042626
 042460 063716 002324
 042464 042766 000004 000002
 042472 000002
 042474 022737 000003 002074
 042502 001003
 042504 013716 002302
 042510 000002
 042512 004737 042626
 042516

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 339
TRAP NON-EXISTANT MEMORY (HOLES) HANDLER

```

11096          .SBTTL TRAP NON-EXISTANT MEMORY (HOLES) HANDLER
11097          :*****
11098          :VECTOR TO HERE (SOMETIMES) FROM TRAPS TO 4
11099          :CODE IN NONEM DETERMINES ACTION AS FOLLOWS:
11100          : 1) IGNORE ERRORS BUT COUNT IF NONEM (NO NON-EXISTANT MEMORY) FLAG = 1.
11101          : 2) TO EXIT PATTERN 0 DURING SIZING IF NON-EXIST MEM ERROR
11102          :*****
11103
11104 042524 022737 000001 002076 NONEXIST: CMP #1,NONEM ;COUNTING NON-EXISTANT MEMORY ERRORS?
11105 042532 001011          BNE 2$ ;NO - SKIP
11106 042534 005237 002066          INC NEMCNT ;BUMP NON-EXISTANT MEMORY COUNTER
11107 042540 022737 000001 002066          CMP #1,NEMCNT ;FIRST ERROR?
11108 042546 001002          BNE 1$ ;NO - SKIP
11109 042550 010037 002032          MOV R0,ADDRESS ;ASSUME R0 CONTAINS THE ADDRESS ACCESSED
11110 042554 000002          1$: RTI
11111 042556 005237 002066          2$: INC NEMCNT ;BUMP NON-EXISTANT MEMORY COUNTER
11112 042562 012701 000001          MOV #1,R1 ;DUMMY UP R1 FOR A FORCED SOB EXIT
11113 042566 000002          RTI
11114
11115          :*****
11116          .SBTTL TRAP TIMEOUT (TRAP TO 4) HANDLER
11117 042570 004737 042626 TIMEOUT: CALL BADSTACK ;FIND BAD SP,PC,PSW OFF STACK
11118 042574          FATAL 6
11119          :*****
11120          .SBTTL TRAP MEMORY MANAGEMENT (TRAP TO 250) HANDLER
11121 042602 004737 042626 MMTRAP: CALL BADSTACK ;FIND BAD SP,PC,PSW OFF STACK
11122 042606          FATAL 7
11123          .SBTTL TRAP RESERVED INSTRUCTION HANDLER
11124 042614 004737 042626 PDP1105: CALL BADSTACK ;FIND BAD SP,PC,PSW OFF STACK
11125 042620          FATAL 5
11126
11132
11133 042626          BADSTACK: SUBTST <<FIND BAD SP, PC, & PSW FROM STACK>>
          :*****
          :*SUBTEST FIND BAD SP, PC, & PSW FROM STACK
          :*****
11134 042626 010637 002024          MOV SP,BADSP
11135 042632 062737 000002 002024          ADD #2,BADSP
11136 042640 016637 000002 002020          MOV 2(SP),BADPC
11137 042646 016637 000004 002030          MOV 4(SP),BADPSW
11138 042654 000207          RETURN

```

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 341
TRAP KERNEL TRAP HANDLER

```

11141          .SBTTL TRAP   KERNEL TRAP HANDLER
11142          :*****
11143          :KERNEL IS A TRAP THAT COMES HERE
11144          :*****
11145
11146 042656 042766 140000 000002 $KERNEL:      BIC      #140000,2(SP)
11147 042664 000002          RTI
11148          :*****
11149          .SBTTL TRAP   ENERGIZE TRAP HANDLER
11150 042666 052737 000001 177572 $ENERGIZE: BIS  #BITO,MMRO
11151 042674 000002          RTI
11152          :*****
11153          .SBTTL TRAP   DEENERGIZE TRAP HANDLER
11154 042676 042737 000001 177572 $DEENERGIZE: BIC #BITO,MMRO
11155 042704 000002          RTI
11156          :*****
11157          .SBTTL TRAP   CACHON TRAP HANDLER
11158 042706 005737 002542 $CACHN:  TST   CACHKN          ;IS THERE A CACHE
11159 042712 001406          BEQ   1$              ;NO - RETURN
11160 042714 013737 002542 177746          MOV   CACHKN,CONTRL          ;SETUP CACHE AS PER CONSTANT (USUALLY 1 = FULLY ON)
11161 042722 052737 000001 177746          BIS   #BITO,CONTRL          ;DISABLE TRAPS (BUT NOT ABORTS)
11162 042730 000002          1$:  RTI
11163          :*****
11164          .SBTTL TRAP   CACHOFF TRAP HANDLER
11165 042732 005737 002542 $CACHF:  TST   CACHKN          ;IS THERE A CACHE?
11166 042736 001403          BEQ   1$              ;NO - RETURN
11167          ;DISABLE TRAPS (NOT ABORTS), FORCE MISSES, FLUSH, BYPASS
11168 042740 053737 002546 177746          BIS   CACHKF,CONTRL
11169 042746 000002          1$:  RTI

```


TRAP LOAD CSR TRAP HANDLER

```

11172                                     .SBTTL TRAP LOAD CSR TRAP HANDLER
11173                                     ;LOAD CORRECT CSR WITH DATA IN CSR
11174                                     ;PROGRAM CSR'S ASSERT INHIBIT MODE POINTER WHEN LOADED
11175 042750                               $LOADC: PUSH    R0,R1                ;SAVE REGISTERS
11176 042754 013700 002150                MOV     CSRNO,R0                ;CREATE CSR ADDRESS
11177 042760                               IF INHECC IS TRUE THEN GOTO 3$ ;DON'T WANT INH. MODE POINTER ON
11178 042766 005737 002530                TST     PGMCSR                 ;PROGRAM IN INTERLEAVED SPACE?
11179 042772 100007                       BPL     1$                     ;BRANCH IF NOT
11180 042774 113701 002531                MOVB   PGMCSR+1,R1            ;CHECK SECOND CSR
11181 043000 042701 177740                BIC    #^C37,R1              ;CLEAR UNNECESSARY BITS
11182 043004 020137 002150                CMP    R1,CSRNO              ;IS THIS THE CURRENT CSR?
11183 043010 001404                       BEQ    2$                     ;BRANCH IF IT IS
11184 043012 123737 002530 002150 1$:    CMPS  PGMCSR,CSRNO            ;IS THIS THE CURRENT CSR?
11185 043020 001003                       BNE    3$                     ;BRANCH IF NOT
11186 043022 052737 020000 002146 2$:    BIS    #BIT13,CSR            ;SET THE INHIBIT MODE POINTER TO 1ST 16K
11187 043030 013760 002146 172100 3$:    MOV    CSR,CSRADD(R0)        ;LOAD THE CSR
11188 043036                               POP    R1,R0                  ;RESTORE REGISTERS
11189 043042 000002                       RTI
11190
11191                                     .SBTTL TRAP READ CSR TRAP HANDLER
11192                                     ;READ THE CORRECT CSR INTO LOCATIONS CSR
11193 043044                               $READC: PUSH    R0
11194 043046 013700 002150                MOV    CSRNO,R0
11195 043052 016037 172100 002146        MOV    CSRADD(R0),CSR        ;READ IT
11196 043060                               POP    R0
11197 043062 000002                       RTI

```

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 344
 TRAP TEST (R1) & READ CSR CAREFULLY

```

11199          .SBTTL TRAP TEST (R1) & READ CSR CAREFULLY
11200 043064          $TSTRD: PUSH R0,R2,R3
11201 043072 012700 172100      MOV #CSRADD,R0          :CREATE CSR ADDRESS
11202 043076 063700 002150      ADD CSRNO,R0
11203 043102 005002          CLR R2
11204 043104 005737 002530      TST PGMCSR
11205 043110 100007          BPL 1$
11206 043112 113703 002531      MOVB PGMCSR+1,R3
11207 043116 042703 000200      BIC #BIT7,R3
11208 043122 020337 002150      CMP R3,CSRNO
11209 043126 001404          BEQ 2$
11210 043130 123737 002530 002150 1$: CMPB PGMCSR,CSRNO
11211 043136 001002          BNE 3$
11212 043140 012702 020000      MOV #BIT13,R2
11213 043144 022737 000001 003754 3$: CMP #1,PROTYP          :IS THIS AN 11/44?
11214 043152 001403          BEQ 4$          :BRANCH IF IT IS
11215 043154 004737 043242      CALL TSTRD1
11216 043160 000405          BR 5$
11217 043162          4$: BMOV TSTRD1
11218 043170 004737 177640      CALL FASTCITY          :CALL TO THE USER INSTRUCTION PAR'S
11219          :IF SINGLE BIT ERROR ONLY - SET CARRY BIT
11220 043174          5$: POP R3,R2,R0
11221 043202          IF #BIT4 SET.IN CSR AND #BIT15 OFF.IN CSR
11222 043222 052766 000001 000002      BIS #BIT0,2(SP)
11223 043230          ELSE
11224 043232 042766 000001 000002      BIC #BIT0,2(SP)
11225 043240          END :OF IF #BIT4
11226 043240 000002          RTI
11227
11228 043242 010210          TSTRD1: MOV R2,(R0)          :V177640
11229 043244          TESTAREA          :V177642 :ENTER SUPERVISOR MODE
11230 043252 105711          TSTB (R1)          :V177646
11231 043254 042737 140000 177776      BIC #BIT15!BIT14,PSW          :V177650
11232 043262 011037 002146          MOV (R0),CSR          :V177656
11233 043266 000207          RETURN          :V177662

```

CZMSP80 MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 346
 TRAP ECC DISABLE ALL CSR'S TRAP HANDLER

11236						.SBTTL	TRAP	ECC DISABLE ALL CSR'S TRAP HANDLER
11237	043270	012737	000002	002146	\$ECCDIS:	MOV	#BIT1,CSR	
11238	043276	004737	044014			CALL	CSROUT	
11239	043302	000002				RTI		
11240						.SBTTL	TRAP	ECC DISABLE OF 1 SELECTED CSR TRAP HANDLER
11241	043304	012737	000002	002146	\$ECC1DIS:	MOV	#BIT1,CSR	
11242	043312	104425				LOADCSR		
11243	043314	000002				RTI		
11244						.SBTTL	TRAP	INITIALIZE ALL CSR'S TRAP HANDLER
11245	043316	012737	000001	002146	\$ECCINIT:	MOV	#BIT0,CSR	
11246	043324	004737	044014			CALL	CSROUT	
11247	043330	000002				RTI		
11248						.SBTTL	TRAP	INITIALIZE 1 SELECTED CSR TRAP HANDLER
11249	043332	012737	000001	002146	\$ECC1INIT:	MOV	#BIT0,CSR	
11250	043340	104425				LOADCSR		
11251	043342	000002				RTI		
11252						.SBTTL	TRAP	ENABLE SBE PARITY TRAPS ON ALL CSR'S
11253	043344	012737	000003	002146	\$ENASBE:	MOV	#BIT0!BIT1,CSR	
11254	043352	004737	044014			CALL	CSROUT	
11255	043356	000002				RTI		
11256						.SBTTL	TRAP	ENABLE SBE PARITY TRAPS ON 1 SELECTED CSR
11257	043360	012737	000003	002146	\$ENA1SBE:	MOV	#BIT0!BIT1,CSR	
11258	043366	104425				LOADCSR		
11259	043370	000002				RTI		
11260						.SBTTL	TRAP	WRITE CHECKBITS THRU ALL CSR'S TRAP HANDLER
11261	043372	013737	002312	002146	\$CBCSR:	MOV	CHECK,CSR	:BITS 11-5
11262	043400	052737	000006	002146		BIS	#BIT1!BIT2,CSR	:CHECK MODE
11263	043406	004737	044014			CALL	CSROUT	
11264	043412	000002				RTI		
11265						.SBTTL	TRAP	WRITE CHECKBITS THRU 1 SELECTED CSR TRAP HANDLER
11266	043414	013737	002312	002146	\$CB1CSR:	MOV	CHECK,CSR	:BITS 11-5
11267	043422	052737	000006	002146		BIS	#BIT1!BIT2,CSR	:CHECK MODE
11268	043430	104425				LOADCSR		
11269	043432	000002				RTI		

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 348
 TRAP WAS THERE A SBE ON ANY CSR TRAP HANDLER

11272						.SBTTL TRAP WAS THERE A SBE ON ANY CSR TRAP HANDLER
11273	043434					\$WASSBE: PUSH R1,R4
11274	043440	013701	002222			MOV TOTCSRS,R1 ;GET CSR'S BYTE
11275	043444	005004				CLR R4
11276	043446					BEGIN LWSBE
11277	043446					FOR CSRNO := #0 TO #36 BY #2
11278	043452	006301				ASL R1
11279	043454					ON.ERROR
11280	043456	104426				READCSR
11281	043460					IF #BIT4 SET.IN CSR
11282	043470					SET R4
11283	043474					LEAVE LWSBE
11284	043476					END ;OF IF #BIT4
11285	043476					END ;OF ON.ERROR
11286	043476					IF R1 EQ #0 THEN LEAVE LWSBE
11287	043502					END ;OF FOR CSRNO
11288	043520					END LWSBE
11289	043520	006004				ROR R4 ;SET C BIT FOR ERROR
11290	043522					POP R4,R1
11291	043526					ON.ERROR
11292	043530	052766	000001	000002		BIS #BIT0,2(SP)
11293	043536					ELSE
11294	043540	042766	000001	000002		BIC #BIT0,2(SP)
11295	043546					END ;OF ON.ERROR
11296	043546	000002				RTI
11297						.SBTTL TRAP WAS THERE A SBE IN 1 SELECTED CSR TRAP HANDLER
11298						:ON RETURN IF CARRY IS SET THERE WAS A SBE
11299	043550	104426				\$WAS1SBE: READCSR
11300	043552	042766	000001	000002		BIC #BIT0,2(SP) ;CLR C BIT ON STACK
11301	043560	032737	000020	002146		BIT #BIT4,CSR
11302	043566	001403				BEQ 1\$
11303	043570	052766	000001	000002		BIS #BIT0,2(SP) ;SET C BIT ON STACK
11304	043576	000002				1\$: RTI

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 350
 TRAP WAS THERE A DBE ON ANY CSR TRAP HANDLER

11307						.SBTTL TRAP WAS THERE A DBE ON ANY CSR TRAP HANDLER
11308	043600					\$WASDBE: PUSH R1,R4
11309	043604	013701	002222			MOV TOTCSRS,R1 ;GET CSR'S BYTE
11310	043610	005004				CLR R4
11311	043612					BEGIN LWDBE
11312	043612					FOR CSRNO := #0 TO #36 BY #2
11313	043616	006301				ASL R1
11314	043620					ON.ERROR
11315	043622	104426				READCSR
11316	043624					IF #BIT15 SET.IN CSR
11317	043634					SET R4
11318	043640					LEAVE LWDBE
11319	043642					END ;OF IF #BIT4
11320	043642					END ;OF ON.ERROR
11321	043642					IF R1 EQ #0 THEN LEAVE LWDBE
11322	043646					END ;OF FOR CSRNO
11323	043664					END LWDBE
11324	043664	006004				ROR R4 ;SET C BIT FOR ERROR
11325	043666					POP R4,R1
11326	043672					ON.ERROR
11327	043674	052766	000001	000002		BIS #BIT0,2(SP)
11328	043702					ELSE
11329	043704	042766	000001	000002		BIC #BIT0,2(SP)
11330	043712					END ;OF ON.ERROR
11331	043712	000002				RTI
11332						.SBTTL TRAP WAS THERE A DBE ON 1 SELECTED CSR TRAP HANDLER
11333						:ON RETURN IF CARRY IS SET THERE WAS A DBE
11334	043714	104426				\$WAS1DBE: READCSR
11335	043716	005737	002146			TST CSR ;DBE?
11336	043722	100004				BPL 3\$;NO - SKIP
11337	043724	052766	000001	000002		BIS #BIT0,2(SP) ;SET C BIT ON STACK
11338	043732	000002				RTI
11339	043734	042766	000001	000002	3\$:	BIC #BIT0,2(SP) ;CLR C BIT ON STACK
11340	043742	000002				RTI

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 352
 TRAP CLEAR ALL ECC CSR'S TRAP HANDLER

11343						.SBTTL TRAP CLEAR ALL ECC CSR'S TRAP HANDLER
11344	043744					\$CLRCSR: CLEAR CSR
11345	043750	004737	044014			CALL CSROUT
11346	043754	000002				RTI
11347						.SBTTL TRAP CLEAR 1 SELECTED CSR TRAP HANDLER
11348	043756					\$CLR1CSR: CLEAR CSR
11349	043762	104425				LOADCSR
11350	043764	000002				RTI
11351						.SBTTL TRAP ECC DISABLE, CHECK MODE, & WRITE CHECKBITS IN ALL CSR'S TRAP HANDLER
11352						:CHECKBITS ALREADY IN LOC "CSR"
11353	043766	052737	000006	002146	\$CHKDIS: BIS #BIT1:BIT2,CSR ;ECC DISABLE & DIAG CHECK MODE	
11354	043774	004737	044014			CALL CSROUT
11355	044000	000002				RTI
11356						.SBTTL TRAP ECC DISABLE, CHECK MODE, & WRITE CHECKBITS IN 1 SELECTED CSR
11357						:CHECKBITS ALREADY IN LOC "CSR"
11358	044002	052737	000006	002146	\$CHK1DIS: BIS #BIT1:BIT2,CSR ;ECC DISABLE & DIAG CHECK MODE	
11359	044010	104425				LOADCSR
11360	044012	000002				RTI

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 354
TRAP ECC DISABLE, CHECK MODE, & WRITE CHECKBITS IN 1 SELECTED

11363 044014

```
CSROUT: SUBTST <<SUBR WRITE IN ALL CSR'S>>
:*****
:*SUBTEST SUBR WRITE IN ALL CSR'S
:*****
```

11364 044014
11365 044016 013701 002222
11366 044022
11367 044022
11368 044026 006301
11369 044030
11370 044032 104425
11371 044034
11372 044034
11373 044040
11374 044056
11375 044056
11376 044060 000207
11377
11378 044062

```
PUSH R1
MOV TOTCSRS,R1 ;GET CSR'S BYTE
BEGIN LCSROUT
FOR CSRNO := #0 TO #36 BY #2
ASL R1
ON.ERROR
LOADCSR
END :OF ON.ERROR
IF R1 EQ #0 THEN LEAVE LCSROUT
END :OF FOR CSRNO
END LCSROUT
POP R1
RETURN
```

```
$INVALID: SUBTST <<TRAP INVALIDATE BACKGROUND PATTERN>>
:*****
:*SUBTEST TRAP INVALIDATE BACKGROUND PATTERN
:*****
```

11379 044062
11380 044066 013701 002100
11381 044072 006301
11382 044074 006301
11383 044076 042761 020000 002654
11384 044104
11385 044110 000002

```
PUSH R0,R1
MOV BANK,R1
ASL R1
ASL R1
BIC #BIT13,CONFIG+2(R1)
POP R1,R0
RTI
```

11387 044112

```

$ERRGEN:      SUBTST<<TRAP  GENERATE AND TEST ERROR ADDRESS>>
:*****
:*SUBTEST     TRAP  GENERATE AND TEST ERROR ADDRESS
:*****
11388 044112      PUSH  R0,R1,R2,R3
11389 044122      MOV   BANKINDEX,R3
11390 044126      TST  NOSUPER
11391 044132      BNE  6$
11392 044134      MOV  SIPAR3,R0          ;GENERATE WHAT ERROR ADDR SHOULD BE
11393 044140      BR   7$
11394 044142      MOV  UIPAR3,R0
11395 044146      ASH  #-5,R0
11396 044152      TST  EUFLAG
11397 044156      BNE  1$
11398 044160      BIC  #^C177,R0
11399 044164      SWAB R1          ;GET CURRENT ADDRESS BITS 11 AND 12
11400 044166      ASR  R1
11401 044170      ASR  R1
11402 044172      ASR  R1
11403 044174      BIC  #^C2,R1
11404 044200      ADD  R1,R0          ;ADD THEM TO THE ADJUSTED PAR VALUE
11405              ;GET ERROR ADDRESS FROM CSR UNDER TEST
11406 044202      MOV  CSR,R1
11407 044206      ASH  #-5,R1
11408 044212      BIC  #^C177,R1
11409 044216      TST  NO22BIT
11410 044222      BNE  2$          ;IS THIS AN 11/44 OR 11/24?
11411 044224      TST  EUFLAG
11412 044230      BEQ  2$          ;BRANCH IF NOT NECESSARY
11413 044232      PUSH R0          ;IS IT EUB?
11414 044234      MOV  CSRNO,R2        ;BRANCH IF NOT
11415 044240      BIS  #BIT14,CSRADD(R2) ;SAVE GENERATED ERROR ADDRESS
11416 044246      MOV  CSRADD(R2),R0 ;GET CSR NUMBER
11417 044252      BIC  #BIT14,CSRADD(R2) ;TURN ON EUB BIT CAREFULLY
11418 044260      BIC  #^C740,R0 ;GET CSR CONTENTS
11419 044264      ASL  R0          ;TURN OFF EUB BIT CAREFULLY
11420 044266      ASL  R0          ;CLEAR EVERYTHING BUT ERROR ADDR
11421 044270      ADD  R0,R1        ;SHIFT ADDR BITS 18-21 INTO POSITION
11422 044272      POP  R0          ;ADD TO CURRENT ERROR ADDRESS
11423 044274      CMP  R0,R1
11424 044276      BEQ  5$          ;COMPARE REAL AND GENERATED ERR. ADDR.
11425 044300      TST  INTFLAG
11426 044304      BEQ  3$          ;BRANCH IF THEY ARE THE SAME
11427 044306      ADD  #100,R0
11428 044312      TST  INT64K
11429 044316      BNE  4$          ;INTERLEAVED?
11430 044320      ADD  #100,R0
11431 044324      CMP  R0,R1
11432 044326      BEQ  5$          ;NO - WE HAVE AN ERROR
11433 044330      TST  SKPERR
11434 044334      BNE  5$          ;ARE WE SUPPOSED TO SKIP ERROR P.O.?
11435 044336      PERR36
11436 044340      MOV  R1,ERRADD
11437 044344      CLR  SKPERR
11438 044350      POP  R3,R2,R1,R0
11439 044360      RTI
11440

```

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 355-1
TRAP GENERATE AND TEST ERROR ADDRESS

11441 044362

SCBREG: SUBTST <<TRAP ENABLE CHECK/SYNDROME BIT REGISTER>>
:*****
:*SUBTEST TRAP ENABLE CHECK/SYNDROME BIT REGISTER
:*****

11442 044362 005037 002146
11443 044366 052737 000004 002146
11444 044374 104425
11445 044376 000002

CLR CSR
BIS #BIT2,CSR
LOADCSR
RTI
:
:ENABLE DIAGNOSTIC MODE
:LOAD CSR REGISTER
:

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 357
TRAP ENABLE CHECK/SYNDROME BIT REGISTER

11448 044400

CHKGEN: SUBTST<<SUBR GENERATE CHECK BITS>>

: *SUBTEST SUBR GENERATE CHECK BITS

11449

: CHECK BIT GENERATOR ROUTINE

11450

: CALLING SEQUENCE IS:

11451

: MOV #WORD1, SOURCE ; SOURCE = ADDRESS OF DATA

11452

: CALL CHKGEN

11453

11454

: CHECK BITS RETURNED IN BITS 11-5 OF LOCATION CHECK

11455

11456 044400

: PUSH R0, R1, R2, R3, R4, R5

11457 044414 012702 000077

: MOV #77, R2 ; DEFAULT CHECKBITS FOR DOUBLE WORD OF ZEROS

11458 044420 012703 044506

: MOV #CHKTAB, R3 ; ADDRESS OF CHECKBIT TABLE

11459 044424 013705 002310

: MOV SOURCE, R5 ; GET SOURCE ADDRESS

11460 044430 012501

: MOV (R5)+, R1 ; GET LSB'S

11461 044432 011500

: MOV (R5), R0 ; GET MSB'S

11462

11463 044434 006704

1\$: SXT R4 ; EXTEND SIGN OF DOUBLE WORD TO R4

11464 044436 142304

: BICB (R3)+, R4 ; ELIMINATE BITS THAT DON'T COUNT

11465 044440 074402

: XOR R4, R2 ; COMPLEMENT MASKED BITS IN CHECKBITS

11466 044442 073027 000001

: ASHC #1, R0 ; DOUBLE PRECISION LEFT SHIFT R0, R1

11467 044446 001372

: BNE 1\$; LOOP TILL ALL BITS ARE CHECKED

11468

11469 044450 042702 177600

: BIC #^C177, R2 ; KILL ALL JUNK BITS

11470 044454 000302

: SWAB R2 ; POSITION CHECKBITS IN BITS 11-5

11471 044456 006202

: ASR R2

11472 044460 006202

: ASR R2

11473 044462 006202

: ASR R2

11474 044464 010237 002312

: MOV R2, CHECK

11475 044470

: POP R5, R4, R3, R2, R1, R0

11476 044504 000207

: RETURN

Address	Hex	Dec	CHKTAB	Bit
11479	044506		:BYTE #3	
11480	044506	200	.BYTE ^C177	:BIT 31
11481	044507	301	.BYTE ^C076	:BIT 30
11482	044510	302	.BYTE ^C075	:BIT 29
11483	044511	203	.BYTE ^C174	:BIT 28
11484	044512	304	.BYTE ^C073	:BIT 27
11485	044513	205	.BYTE ^C172	:BIT 26
11486	044514	206	.BYTE ^C171	:BIT 25
11487	044515	307	.BYTE ^C070	:BIT 24
11488			:BYTE #2	
11489	044516	310	.BYTE ^C067	:BIT 23
11490	044517	211	.BYTE ^C166	:BIT 22
11491	044520	212	.BYTE ^C165	:BIT 21
11492	044521	313	.BYTE ^C064	:BIT 20
11493	044522	214	.BYTE ^C163	:BIT 19
11494	044523	315	.BYTE ^C062	:BIT 18
11495	044524	316	.BYTE ^C061	:BIT 17
11496	044525	217	.BYTE ^C160	:BIT 16
11497			:BYTE #1	
11498	044526	320	.BYTE ^C057	:BIT 15
11499	044527	221	.BYTE ^C156	:BIT 14
11500	044530	222	.BYTE ^C155	:BIT 13
11501	044531	323	.BYTE ^C054	:BIT 12
11502	044532	224	.BYTE ^C153	:BIT 11
11503	044533	325	.BYTE ^C052	:BIT 10
11504	044534	326	.BYTE ^C051	:BIT 9
11505	044535	227	.BYTE ^C150	:BIT 8
11506			:BYTE #0	
11507	044536	340	.BYTE ^C037	:BIT 7
11508	044537	241	.BYTE ^C136	:BIT 6
11509	044540	242	.BYTE ^C135	:BIT 5
11510	044541	343	.BYTE ^C034	:BIT 4
11511	044542	244	.BYTE ^C133	:BIT 3
11512	044543	345	.BYTE ^C032	:BIT 2
11513	044544	346	.BYTE ^C031	:BIT 1
11514	044545	247	.BYTE ^C130	:BIT 0

11517 044546

```

SUBTST<<SUBR MAPPER>>
*****
:SUBTEST SUBR MAPPER
*****
:THIS SUBROUTINE MAPS THE MEMORY BANK (16K WORDS = 1 BANK)
:IN R3 TO THE TEST PATTERN AREA (SUPERVISOR VIRTUAL (60000 - 157777) FOR
:THE 11/44 AND 11/45-55; USER VIRTUAL (60000 - 157777) FOR ALL OTHER
:PDP-11'S).
:CALL MOV BANKNO,R3 :SET UP BANK ARGUMENT
:CALL CALL MAPPER :ACTUAL CALL
:RETURN :ONLY RETURN

:SET SUPERVISOR/USER UP FOR 1 TO 1 MAP
MAPPER: PUSH RO,R1,R2,R4,R5
MOV #KIPAR0,R0 :FIRST AREA TO MAP TO
MOV #SIPAR0,R1 :FIRST ADDRESS REGISTER
MOV #SIPDR0,R4 :FIRST DESCRIPTOR REGISTER
TST NOSUPER :CAN WE USE SUPERVISOR MODE?
BEQ 4$ :YES, BRANCH
MOV #UIPAR0,R1 :FIRST ADDRESS REGISTER
MOV #UIPDR0,R4 :FIRST DESCRIPTOR REGISTER
4$: MOV #77406,R2 :CONSTANT FOR 4K PAGE, UP, R/W
MOV #8,R5 :COUNTER
1$: MOV (R0)+,(R1)+ :PUT IN SUPERVISOR ADDRESS
MOV R2,(R4)+ :PUT IN SUPERVISOR DESCRIPTOR
SOB R5,1$ :LOOP TILL DONE
MOV #177600,-(R1) :CORRECT LAST FIELD FOR PERIPHERALS PAGE

:SET UP SUPERVISOR/USER FOR TEST AREA
CMP #120.,R3 :MAP NOTHING (1 TO 1)?
BEQ 3$ :YES - SKIP
ASH #9.,R3 :BANK 1 STARTS AT 100,000 LESS 6 LSB'S
:FOR MEMORY MANAGEMENT = 1000
MOV #SIPAR3,R1 :SETUP FOR AUTO INCREMENTING
TST NOSUPER :DO WE HAVE SUPERVISOR MODE?
BEQ 5$ :YES - BRANCH
MOV #UIPAR3,R1 :SETUP FOR AUTO INCREMENTING
5$: MOV #4,R2 :COUNTER
2$: MOV R3,(R1)+ :PLUG IN PAR INFO
ADD #200,R3 :BUMP ADDRESS 4K
SOB R2,2$ :LOOP TILL DONE
TST SPLTCSR
BEQ 9$
SIB #10,R1
MOV R1,R2
ADD #4,R2
CMP #1,SPLTCSR
BEQ 10$
MOV R2,R0
MOV R1,R2
MOV R0,R1
10$: MOV (R1)+,(R2)+
MOV (R1),(R2)
MOV BANKINDEX,R0
TST INT64K
BEQ 11$

```

```

11518
11519
11520
11521
11522
11523
11524
11525
11526
11527
11528 044546
11529 044560 012700 172340
11530 044564 012701 172240
11531 044570 012704 172200
11532 044574 005737 002454
11533 044600 001404
11534 044602 012701 177640
11535 044606 012704 177600
11536 044612 012702 077406
11537 044616 012705 000010
11538 044622 012021
11539 044624 010224
11540 044626 077503
11541 044630 012741 177600
11542
11543
11544 044634 022703 000170
11545 044640 001516
11546 044642 072327 000011
11547
11548 044646 012701 172246
11549 044652 005737 002454
11550 044656 001402
11551 044660 012701 177646
11552 044664 012702 000004
11553 044670 010321
11554 044672 062703 000200
11555 044676 077204
11556 044700 005737 002236
11557 044704 001442
11558 044706 162701 000010
11559 044712 010102
11560 044714 062702 000004
11561 044720 022737 000001 002236
11562 044726 001403
11563 044730 010200
11564 044732 010102
11565 044734 010001
11566 044736 012122
11567 044740 011112
11568 044742 013700 002102
11569 044746 005737 002136
11570 044752 001403

```


CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 361-1

SUBR MAPPER

```

11571 044754 012700 004000      MOV    #4000,R0
11572 044760 000402      BR     12$
11573 044762 012700 010000      11$:  MOV    #10000,R0
11574 044766 005737 002454      12$:  TST    NOSUPER
11575 044772 001403      BEQ    13$
11576 044774 012701 177652      MOV    #UIPAR5,R1
11577 045000 000402      BR     14$
11578 045002 012701 172252      13$:  MOV    #SIPAR5,R1
11579 045006 060021      14$:  ADD    R0,(R1)+
11580 045010 060011      ADD    R0,(R1)
11581
11582      :IF WE ONLY HAVE AN 124K SYSTEM, WE DON'T WANT TO TEST THE
11583      :LAST 4K, WHERE THE UNIBUS DEVICE PAGE IS. INSTEAD, THE
11584      :PROGRAM WILL REMAP THE LAST 4K TO 8-12K. ALSO, IF THERE
11585      :IS A BANK 177 ON AN 11/44, THE PROGRAM WILL REMAP THE LAST
11586      :4K TO 8-12K FOR THE SAME REASON.
11586 045012 022737 000007 002554 9$:  CMP    #7,LASTBANK
11587 045020 001010      BNE    7$
11588 045022 005737 002452      TST    NO22BIT      :11/44 OR 24?
11589 045026 001423      BEQ    3$           :BRANCH IF SO
11590 045030 022737 000007 002100      CMP    #7,BANK      :BANK 7?
11591 045036 001017      BNE    3$           :NO - BRANCH
11592 045040 000404      BR     8$
11593 045042 022737 000177 002554 7$:  CMP    #177,LASTBANK
11594 045050 001012      BNE    3$
11595 045052 005737 002454      8$:  TST    NOSUPER
11596 045056 001404      BEQ    6$
11597 045060 013737 177652 177654      MOV    UIPAR5,UIPAR6
11598 045066 000403      BR     3$
11599 045070 013737 172252 172254 6$:  MOV    SIPAR5,SIPAR6
11600 045076      3$:  POP    R5,R4,R2,R1,R0
11601 045110 000207      RETURN
11602      .SBTTL  TRAP    MAP KERNEL (ALMOST 1 TO 1) TRAP HANDLER
11603 045112      SKMAP:  PUSH   R0,R1,R2,R3,R4
11604 045124 005000      CLR    R0           :1ST AREA TO MAP TO
11605 045126 012701 172340      MOV    #KIPAR0,R1  :FIRST ADDRESS
11606 045132 012702 077406      MOV    #77406,R2   :CONSTANT FOR 4K PAGE,UP,R/W
11607 045136 012703 172300      MOV    #KIPDR0,R3  :1ST PAGE DESCRIPTOR REGISTER
11608 045142 012704 000010      MOV    #8,R4       :COUNTER
11609 045146 010021      1$:  MOV    R0,(R1)+    :PUT IN KERNEL ADDRESS
11610 045150 010223      MOV    R2,(R3)+    :PUT IN KERNEL DISCRIPTOR
11611 045152 062700 000200      ADD    #200,R0     :ADD ADDRESS CONSTANT FOR 4K CHANGE
11612 045156 077405      SOB    R4,1$      :LOOP TILL DONE
11613 045160 012741 177600      MOV    #177600,-(R1) :THE PERIPHERALS PAGE TO KIPAR7
11614 045164 012741 177400      MOV    #177400,-(R1) :AND NEXT LOWER PAGE TO KIPAR6
11621 045170      POP    R4,R3,R2,R1,R0
11622 045202 000002      RTI

```

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 363
TRAP MAP KERNEL (ALMOST 1 TO 1) TRAP HANDLER

11625 045204

RELOCATE:SUBTST <<RELOCATE PROGRAM>>

:SUBTEST RELOCATE PROGRAM

11626 045204
11627 045220
11628 045234
11629 045246
11630 045246
11631 045246
11632 045254 004737 047032
11633 045260
11634 045302 013700 002100
11635 045306 010037 002430
11636 045312 013701 002564
11637 045316 004737 046422
11638 045322 004737 046754
11639 045326 013701 002102
11640 045332 052761 100000 002654
11641 045340 042761 020000 002654
11642 045346
11643 045350
11644 045350
11645 045364
11646 045374
11647 045400
11648 045400
11649 045404
11650 045404
11651 045404 013702 002554
11652 045410 006302
11653 045412 006302
11654 045414
11655 045420
11656 045430
11657 045440
11658 045450
11659 045460
11660
11661 045500
11662 045502
11663 045502
11664 045510
11665 045516 010137 002536
11666 045522
11667 045522
11668 045522
11669 045522
11670 045522
11671 045532
11672 045540
11673 045546 013701 002536
11674 045552 023727 002276 000030
11675 045560 001421
11676 045562 000420
11677 045564
11678 045564

IF #SW12 SET.IN @SWR THEN \$RETURN ERROR
IF APTFLAG IS TRUE OR ACTFLAG IS TRUE
IF \$PASS NE #0 THEN \$RETURN ERROR
END: OF IF APTFLAG
BEGIN LOADERBANK
FOR BANK := #1 TO LASTBANK
CALL EXBANK
IF ACFLAG IS TRUE AND PFLAG IS FALSE AND BMFLAG IS FALSE
MOV BANK,R0
MOV RO,LOADBANK
MOV LOADHOME,R1
CALL BANKMOV
CALL NEWLOAD ;MAP NEW LOADER BANK IN KERNEL
MOV BANKINDEX,R1
BIS #BIT15,CONFIG+2(R1) ;MARK LOADER
BIC #BIT13,CONFIG+2(R1) ;INVALIDATE BACKGROUND PATTERN
LEAVE LOADERBANK
END :OF IF ACFLAG
END :OF FOR BANK
IF #SW13 OFF.IN @SWR
TYPE MSG075 ;RELOCATION NOT POSSIBLE
END :OF IF #SW13
\$RETURN ERROR
END LOADERBANK
BEGIN FINDBANK
MOV LASTBANK,R2
ASL R2
ASL R2 ;R2 <- R2 * 4
FOR R1 := #2*2 TO R2 BY #4
IF #BIT7!BIT0 OFF.IN CONFIG(R1) ;IF NO ERRORS & NOT PROGRAM SPACE
IF #BIT15 OFF.IN CONFIG+2(R1) ;IF NOT LOADER BANK
IF CPUBIT SET.IN CONFIG(R1) ;IF ACCESSABLE
IF #BIT8 OFF.IN CONFIG+2(R1) THEN LEAVE FINDBANK ;IF PARITY
IF #BIT6 SET.IN CONFIG(R1) AND #BIT7 OFF.IN CONFIG(R1)
;IF 1ST PROTECTABLE ECC BANK
LEAVE FINDBANK
END :OF IF #BIT6
IF INHECC IS FALSE
SET INHECC
MOV R1,INHBANK
END: OF IF INHECC
END :OF IF CPUBIT
END :OF IF #BIT15
END :OF IF #BIT7
END :OF FOR
IF FULLREL IS FALSE
IF INHECC IS TRUE
MOV INHBANK,R1
CMP REALPAT,#30 ;IS THIS PATTERN 30?
BEQ RELENT1 ;YES - SKIP MESSAGE
BR RELENT1
END: OF IF INHECC
END: OF IF FULLREL

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 363-1
RELOCATE PROGRAM

11679	045564	005037	002534		CLR INHECC		:MAKE SURE FLAG IS TURNED OFF!
11680	045570				IF #SW13 OFF.IN @SWR		
11681	045600	023727	002276	000030	CMP REALPAT,#30		:IS THIS PATTERN 30?
11682	045606	001402			BEQ SKUB		:YES - SKIP MESSAGE
11683	045610				TYPE MSG075		:RELOCATION NOT POSSIBLE
11684	045614				END :OF IF #SW13		
11685	045614				SKUB: \$RETURN ERROR		
11686	045620				END FINDBANK		
11687	045620				CLEAR INHECC		:IF WE RELOCATED PROPERLY, THIS SHOULD BE OFF!
11688	045624	042761	020000	002654	RELENT1: BIC #BIT13,CONFIG+2(R1)		:INVALIDATE BACKGROUND PATTERN
11689	045632	005000			CLR R0		
11690	045634	071027	000004		DIV #4,R0		
11691	045640				RELOC1: LET NEWBANK := R0		
11692	045644	013737	002530	002532	MOV PGMCSR,PGMCSR+2		:SAVE CURRENT PGM. CSR
11693	045652	004737	046570		CALL USERMAP		:MAP NEWBANK TO USER PAR
11694	045656				USER		:ENTER USER MODE
11695	045664				BMOV 0,100000,SIZE		:MOVE PROGRAM
11696	045676	104417			KERNEL		:ENTER KERNEL MODE
11697	045700	022737	000001	003754	CMP #1,PROTYP		:IS THIS AN 11/44 ?
11698	045706	001021			BNE JMPRL1		:JUMP IF NOT
11699	045710	042737	000040	172516	BIC #BIT5,MMR3		:TURN OFF UNIBUS MAP
11700	045716	013700	002306		MOV NEWBANK,R0		
11701	045722	006200			ASR R0		
11702	045724				ON.ERROR		
11703	045726	012737	100000	170200	MOV #BIT15,MAPLO		
11704	045734				END :OF ON.ERROR		
11705	045734	010037	170202		MOV R0,MAPHO		
11706	045740	004737	046356		CALL LOWMAP		:SETUP LOWER 16K IN UNIBUS MAP
11707	045744	052737	000040	172516	BIS #BIT5,MMR3		:ENERGIZE UNIBUS MAP
11708	045752	042737	000001	177572	JMPRL1: BIC #BIT0,MMR0		:DEENERGIZE MEMORY MANAGEMENT
11709	045760	004737	046652		CALL NEWKERNEL		
11710	045764	013700	002306		MOV NEWBANK,R0		
11711	045770	006300			ASL R0		
11712	045772	006300			ASL R0		:R0 <- R0 * 4
11713	045774	016002	002652		MOV CONFIG(R0),R2		
11714	046000	000302			SWAB R2		
11715	046002	042702	177760		BIC #^C17,R2		
11716	046006	006302			ASL R2		
11717	046010	052737	000001	177572	BIS #BIT0,MMR0		:ENERGIZE MEMORY MANAGEMENT
11718	046016	010237	002530		MOV R2,PGMCSR		:PUT NEW PGM. CSR INTO PGMCSR
11719	046022	032760	010000	002654	BIT #BIT12,CONFIG+2(R0)		:IS THE NEW BANK INTERLEAVED?
11720	046030	001412			BEQ 1\$:BRANCH IF NOT INTERLEAVED
11721	046032	016002	002652		MOV CONFIG(R0),R2		
11722	046036	042702	007777		BIC #^C170000,R2		
11723	046042	072227	177775		ASH #-3,R2		
11724	046046	052702	100000		BIS #BIT15,R2		
11725	046052	050237	002530		BIS R2,PGMCSR		
11726	046056				1\$: SET RLFLAG		
11727	046064				\$RETURN NOERROR		

11730 046070

UNRELOCATE:SUBTST <<UNRELOCATE PROGRAM>>

```

:*****
:*SUBTEST      UNRELOCATE PROGRAM
:*****
    
```

11731

:RESTORE LOADERS

```

11732 046070
11733 046072 013701 002430
11734 046076 013700 002564
11735 046102 004737 046422
11736 046106 004737 046754
11737 046112
11738 046116 013737 002430 002100
11739 046124 004737 047032
11740 046130 013701 002102
11741 046134 042761 100000 002654
11742 046142 013737 002564 002100
11743 046150 004737 047032
11744 046154 013701 002102
11745 046160 042761 020000 002654
11746 046166
11747 046172
    
```

```

PUSH      R0
MOV       LOADBANK,R1
MOV       LOADHOME,R0
CALL     BANKMOV
CALL     NEWLOAD           ;MAP NEW LOADER BANK IN KERNEL SPACE
PUSH     BANK
MOV      LOADBANK,BANK
CALL     EXBANK
MOV     BANKINDEX,R1
BIC     #BIT15,CONFIG+2(R1) ;CLEAR LOADER FLAG
MOV     LOADHOME,BANK
CALL     EXBANK
MOV     BANKINDEX,R1
BIC     #BIT13,CONFIG+2(R1) ;INVALIDATE BACKGROUND PATTERN
POP     BANK
CLEAR   INHECC           ;MAKE SURE ECC TESTS ARE NOT INHIBITED!
    
```

11748

:RESTORE BANK 0

```

11749
11750 046176 042737 020000 002654
11751 046204
11752 046210 004737 046570
11753 046214
11754 046222
11755 046234 104417
11756 046236 042737 000001 177572
11757 046244 004737 046652
11758 046250 013737 002532 002530
11759 046256 052737 000001 177572
11760 046264 005037 002124
11761 046270 022737 000001 003754
11762 046276 001014
11763 046300 042737 000040 172516
11764 046306
11765 046316 004737 046356
11766 046322 052737 000040 172516
11767 046330 012700 002654
11768 046334 042710 020000
11769 046340 062700 000004
11770 046344 020027 003620
11771 046350 003771
11772 046352
11773 046354 000207
11774
11775 046356
    
```

```

BIC     #BIT13,CONFIG+2 ;INVALIDATE BACKGROUND PATTERN
LET     NEWBANK := #0
CALL    USERMAP         ;MAP NEWBANK TO USER PAR
USER    ;ENTER USER MODE
BMOV   0,100000,SIZE    ;MOVE PROGRAM
KERNEL ;ENTER KERNEL MODE
BIC     #BIT0,MMR0      ;DEENERGIZE MEMORY MANAGEMENT
CALL    NEWKERNEL
MOV     PGMCSR+2,PGMCSR ;RESTORE PREVIOUS PGM. CSR
BIS     #BIT0,MMR0      ;ENERGIZE MEMORY MANAGEMENT
CLR     RLFLAG
CMP     #1,PROTYP       ;IS THIS AN 11/44 ?
BNE     1$
BIC     #BIT5,MMR3      ;TURN OFF UNIBUS MAP
CLEAR   MAPLO,MAPHO
CALL    LOWMAP          ;SETUP LOWER 16K OF UNIBUS MAP
BIS     #BIT5,MMR3      ;ENERGIZE UNIBUS MAP
1$: MOV  #CONFIG+2,R0    ;MOVE 2ND WORD OF CONFIG TO R0
2$: BIC  #BIT13,(R0)    ;CLEAR BACKGROUND VALID BIT
    ADD  #4,R0          ;INCREMENT TO NEXT BANK
    CMP  R0,#3620      ;DONE?
    BLE  2$            ;NO - BRANCH
POP     R0
RETURN
    
```

LOWMAP: SUBTST <<SETUP LOWER 16K OF UNIBUS MAP>>

```

:*****
:*SUBTEST      SETUP LOWER 16K OF UNIBUS MAP
:*****
    
```

```

11776 046356
11777 046364 012700 170200
11778 046370 012701 170204
11779 046374 012702 000003
11780 046400 012011
    
```

```

PUSH     R0,R1,R2
MOV     #MAPLO,R0
MOV     #MAPL1,R1
MOV     #3,R2
1$: MOV  (R0)+,(R1)
    
```

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 365-1
SETUP LOWER 16K OF UNIBUS MAP

11781	046402	062721	020000	ADD	#BIT13,(R1)+
11782	046406	012021		MOV	(R0)+,(R1)+
11783	046410	077205		SOB	R2,1\$
11784	046412			POP	R2,R1,R0
11785	046420	000207		RETURN	

11788 046422

BANKMOV:SUBTST <<MOVE BANKS>>

:SUBTEST MOVE BANKS

11789

:MOVE 3/4 OF A BANK

11790

:CALLING SEQUENCE

11791

:R0 = DESTINATION BANK

11792

:R1 = SOURCE BANK

11793 046422 104415

SAVREG

11794 046424 004737 046570

CALL USERMAP

11795 046430 104416

RESREG

11796 046432 104415

SAVREG

11797 046434 072027 000011

ASH #9,R0

11798 046440 072127 000011

ASH #9,R1

11799 046444 012702 177650

MOV #UIPAR4,R2

11800 046450 012703 000200

MOV #200,R3

11801

11802 046454 010122

MOV R1,(R2)+

:MAP 1ST HALF BANK

11803 046456 060301

ADD R3,R1

:BUMP BY 4K

11804 046460 010122

MOV R1,(R2)+

11805 046462 060301

ADD R3,R1

11806

11807 046464 010022

MOV R0,(R2)+

11808 046466 060300

ADD R3,R0

11809 046470 010022

MOV R0,(R2)+

11810 046472 060300

ADD R3,R0

11811

11812 046474

USER

11813 046502

BMOV 100000,140000,SIZE/2

:MOV 1ST HALF BANK

11814 046514 104417

KERNEL

:ENTER KERNEL MODE

11815

11816 046516 012702 177650

MOV #UIPAR4,R2

11817

11818 046522 010122

MOV R1,(R2)+

:MAP 2ND HALF BANK

11819 046524 060301

ADD R3,R1

:BUMP BY 4K

11820 046526 010122

MOV R1,(R2)+

11821 046530 060301

ADD R3,R1

11822

11823 046532 010022

MOV R0,(R2)+

11824 046534 060300

ADD R3,R0

11825 046536 010022

MOV R0,(R2)+

11826 046540 060300

ADD R3,R0

11827

11828 046542

USER

11829 046550

BMOV 100000,140000,SIZE/4

:MOV 3ND FOURTH OF BANK

11830 046562 104417

KERNEL

:ENTER KERNEL MODE

11831

11832 046564 104416

RESREG

11833 046566 000207

RETURN

11836 046570

USERMAP:SUBTST <<SUBR MAP USER TO NEW BANK>>
:*****
:*SUBTEST SUBR MAP USER TO NEW BANK
:*****

11837 046570 012701 177640
11838 046574 012702 172340
11839 046600 012703 177600
11840 046604 012704 172300
11841 046610 012705 000004
11842 046614 012221
11843 046616 011423
11844 046620 077503
11845
11846 046622 013700 002306
11847 046626 072027 000011
11848
11849 046632 012705 000004
11850 046636 010021
11851 046640 062700 000200
11852 046644 011423
11853 046646 077505
11854 046650 000207
11855
11856 046652

MOV #UIPAR0,R1 ;COPY KERNEL PAR'S & PDR'S (0-3)
MOV #KIPAR0,R2
MOV #UIPDR0,R3
MOV #KIPDR0,R4
MOV #4,R5
1\$: MOV (R2)+,(R1)+
MOV (R4),(R3)+
SOB R5,1\$
MOV NEWBANK,R0
ASH #9.,R0 ;BANK 1 STARTS AT 100,000 LESS 6 LSB'S
;FOR MEMORY MANAGEMENT = 1000
MOV #4,R5
2\$: MOV R0,(R1)+ ;SETUP UIPAR(4-7)
ADD #200,R0 ;BUMP ADDRESS 4K
MOV (R4),(R3)+ ;SETUP UIPDR(4-7)
SOB R5,2\$
RETURN

NEWKERNEL:SUBTST <<SUBR SETUP KERNEL PAR'S FOR NEW BANK>>
:*****
:*SUBTEST SUBR SETUP KERNEL PAR'S FOR NEW BANK
:*****

11857 046652
11858 046660 012700 172340
11859 046664 013701 002306
11860 046670 072127 000011
11861
11862 046674 012705 000004
11863 046700 010120
11864 046702 062701 000200
11865 046706 077504
11866 046710
11867 046716 000207
11868
11869 046720

PUSH R0,R1,R5
MOV #KIPAR0,R0
MOV NEWBANK,R1
ASH #9.,R1 ;BANK 1 STARTS AT 100,000 LESS 6 LSB'S
;FOR MEMORY MANAGEMENT = 1000
MOV #4,R5
1\$: MOV R1,(R0)+ ;SETUP KIPAR(0-3)
ADD #200,R1
SOB R5,1\$
POP R5,R1,R0
RETURN

MAPKERNAL:SUBTST <<SUBR MAP KERNAL PARS 4 AND 5 TO A BANK>>
:*****
:*SUBTEST SUBR MAP KERNAL PARS 4 AND 5 TO A BANK
:*****

11870
11871 046720 013705 002100
11872 046724 072527 000011
11873 046730 013737 172350 002270
11874 046736 010537 172350
11875 046742 062705 000200
11876 046746 010537 172352
11877 046752 000207
11878
11879 046754

MOV BANK,R5 ;MOV BANK NUMBER TO R5
ASH #9.,R5 ;R5 ENTERS 100000 LESS SHIFT TO CREATE MAPPING
MOV KIPAR4,SAVPAR ;SAVE OLD PAR
MOV R5,KIPAR4 ;GET NEW PAR'S
ADD #200,R5
MOV R5,KIPAR5
RETURN

NEWLOAD:SUBTST <<SUBR SETUP KERNEL PAR'S FOR NEW LOADER BANK>>
:*****
:*SUBTEST SUBR SETUP KERNEL PAR'S FOR NEW LOADER BANK
:*****
:R0 CONTAINS THE DESTINATION BANK

11880

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 369-1
SUBR SETUP KERNEL PAR'S FOR NEW LOADER BANK

11881	046754				PUSH	R0,R1	
11882	046760	012701	172350		MOV	#KIPAR4,R1	
11883	046764	072027	000011		ASH	#9.,R0	:BANK 1 STARTS AT 100000 LESS 6 LSB'S (1000)
11884	046770	010021			MOV	R0,(R1)+	:SETUP KIPAR4
11885	046772	062700	000200		ADD	#200,R0	
11886	046776	010021			MOV	R0,(R1)+	:SETUP KIPAR5
11887	047000				POP	R1,R0	
11888	047004	000207			RETURN		
11889							
11890	047006						

```
UNMAP: SUBTST <<SUBR UNMAP KERNAL PAR'S 4 AND 5>>
:*****
:*SUBTEST SUBR UNMAP KERNAL PAR'S 4 AND 5
:*****
```

11891	047006	013737	002270	172350	MOV	SAVPAR,KIPAR4	:RESTORE KIPAR4
11892	047014	062737	000200	002270	ADD	#200,SAVPAR	:ADD 200 FOR NEXT PAR
11893	047022	013737	002270	172352	MOV	SAVPAR,KIPAR5	:RESTORE KIPAR5
11894	047030	000207			RETURN		:

11897 047032

EXBANK: SUBTST <<SUBR EXAMINE BANK>>

:SUBTEST SUBR EXAMINE BANK

:DOES THE FOLLOWING:
:(1) SETS UP 'BANKINDEX' AND R1 BASED ON VALUE OF 'BANK'.
:(2) SETS THE 'MKFLAG' IF THE BANK IS ECC.
:(3) SETS THE 'KPFLAG' IF THE BANK IS THE PROTECTED REGION OF ECC MEMORY.
:(4) SETS THE 'ACFLAG' IF THE BANK CAN BE ACCESSED BY THIS CPU.
:(5) SETS THE 'PFLAG' IF THE BANK IS IN PROGRAM SPACE.
:(6) SETS THE 'RRFLAG' IF RELOCATION IS REQUIRED TO TEST THIS BANK; HOWEVER,
IT COMPLEMENTS THIS FLAG IF THE RELOCATION FLAG 'RLFLAG' IS SET (THIS IS
NECESSARY FOR THE USE OF THE RECURSIVE 'MODE' SUBROUTINES). THE 'RRFLAG'
IS ALWAYS SET TO DISABLE TESTING IF FIELD SERVICE MODE 'SELECTED BANKS'
ARE BEING TESTED AND THIS BANK IS NOT SELECTED.
:(7) SETS THE 'BMFLAG' IF THE BANK IS A BAD MEMORY; HOWEVER, IT COMPLEMENTS
THIS FLAG IF THE 'WORST' FLAG IS NOT SET (THIS IS NECESSARY FOR THE USE
OF THE RECURSIVE 'MODE' SUBROUTINES).
:(8) SETS THE 'INTFLAG' IF THE BANK IS INTERLEAVED.
:(9) SETS THE 'INT64K' FLAG IF THE BANK IS INTERLEAVED ON 64K WORD BOUNDS.
:(10) SETS THE 'SKIPMK' FLAG IF THIS BANK IS INTERLEAVED, AND HAS ALREADY
BEEN TESTED.
:(11) SETS THE 'PMEMFLG' IF THE ECC MEMORY UNDER TEST IS A MS11-P

11898
11899
11900
11901
11902
11903
11904
11905
11906
11907
11908
11909
11910
11911
11912
11913
11914
11915
11916
11917
11918 047032
11919 047040
11920 047054
11921 047062
11922 047076
11923 047112 013701 002100
11924 047116 006301
11925 047120 006301
11926 047122 010137 002102
11927 047126 032761 000100 002652
11928 047134 001403
11929 047136
11930 047144 012700 000002 1\$:
11931 047150
11932 047164 005037 002114
11933 047170
11934 047170 005737 002114
11935 047174 001415
11936 047176 016102 002654
11937 047202 000302
11938 047204 042702 177770
11939 047210 020227 000003
11940 047214 003405
11941 047216
11942 047224 000137 047450
11943 047230 032761 000400 002654 2\$:
11944 047236 001412
11945 047240
11946 047246 032761 001000 002654
11947 047254 001403
11948 047256
11949 047264 032761 000200 002652 3\$:
11950 047272 001406

PUSH R0,R1,R2
CLEAR MKFLAG,KPFLAG,PMEMFLAG
SET ACFLAG
CLEAR PFLAG,RRFLAG,BMFLAG
CLEAR INTFLAG,INT64K,SKIPMK
MOV BANK,R1
ASL R1
ASL R1 ;R1 <- R1 * 4
MOV R1,BANKINDEX
BIT #BIT6,CONFIG(R1) ;PROTECTED REGION OF ECC MEMORY?
BEQ 1\$;NO - SKIP
SET KPFLAG
MOV #BIT1,R0
IF R0 SET.IN CPUBIT AND R0 OFF.IN CONFIG(R1)
CLR ACFLAG
END ;OF IF R0
TST ACFLAG ;ACTIVE MEMORY?
BEQ 2\$;BRANCH IF NOT
MOV CONFIG+2(R1),R2
SWAB R2
BIC #^C7,R2 ;ISOLATE MEM TYPE BITS
CMP R2,#3 ;IS THIS AN ILLEGAL MEM TYPE?
BLE 2\$;BRANCH IF NOT
SET BMFLAG ;SET BAD BANK FLAG
JMP ENEXBK ;JUMP OVER REST OF FLAG TESTS
BIT #BIT8,CONFIG+2(R1) ;IS THERE ECC THERE?
BEQ 3\$;NO - SKIP
SET MKFLAG ;YES - SET MKFLAG
BIT #BIT9,CONFIG+2(R1) ;IS IT A MS11-P????
BEQ 3\$;NO SKIP!!!!
SET PMEMFLG ;SET MS11-P FLAG
BIT #BIT7,CONFIG(R1) ;BANK = PROGRAM SPACE?
BEQ 5\$;NO - SKIP

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 371-1
 SUBR EXAMINE BANK

```

11951 047274          SET      PFLAG,RRFLAG
11952 047310 005737 002124      5$:  TST      RLFLAG          :IS PROGRAM RELOCATED?
11953 047314 001402          BEQ      6$              :NO - SKIP
11954 047316 005137 002122          COM      RRFLAG          :YES - COMPLEMENT RELOCATION REQUIRED FLAG
11955 047322 032761 000001 002652 6$:  BIT      #BIT0,CONFIG(R1) :ERRORS PRESENT IN THIS BANK?
11956 047330 001403          BEQ      8$              :NO - SKIP
11957 047332          SET      BMFLAG
11958 047340 005737 002566      8$:  TST      WORST          :IS THIS A WORST FIRST PASS?
11959 047344 001002          BNE     9$              :YES - SKIP
11960 047346 005137 002126          COM      BMFLAG          :NO - COMPLEMENT BAD MEMORY FLAG
11961 047352          9$:  IF SELONLY IS TRUE AND #BIT14 OFF.IN CONFIG+2(R1)
11962 047370          SET      RRFLAG
11963 047376          END ;OF IF SELONLY
11964 047376 032761 010000 002654  BIT      #BIT12,CONFIG+2(R1) :IS THIS BANK INTERLEAVED?
11965 047404 001421          BEQ      ENEXBK          :BRANCH IF IT IS NOT
11966 047406          SET      INTFLAG
11967 047414 032761 004000 002654  BIT      #BIT11,CONFIG+2(R1) :IS THIS BANK INTERLEAVED WITH 64K BOARDS?
11968 047422 001403          BEQ      10$           :BRANCH IF IT IS NOT
11969 047424          SET      INT64K
11970 047432 032761 000040 002652 10$: BIT      #BIT5,CONFIG(R1)  :SHOULD THIS BANK BE TESTED?
11971 047440 001403          BEQ      ENEXBK          :BRANCH IF IT SHOULD
11972 047442          SET      SKIPMK
11973 047450          ENEXBK: POP      R2,R1,R0 :RESTORE REGISTERS
11974 047456 000207          RETURN

```

11977 047460

```

BANKOK: SUBTST <<SUBR BANK OK?>>
:*****
:*SUBTEST SUBR BANK OK?
:*****
;TEST TO INSURE THAT THE TYPE OF MEMORY IN THE PRESENT BANK
;IS OF THE TYPE WE ARE TESTING "TMFLAG".
;RESULT IS RETURNED IN THE CONDITION CODES (OK = (=0)).
MOV TMFLAG,R0
COM R0
MOV MKFLAG,R1
XOR R0,R1
RETURN ;OK = (=OK)

```

11978
11979
11980
11981 047460 013700 002132
11982 047464 005100
11983 047466 013701 002116
11984 047472 074001
11985 047474 000207
11986
11987 047476
11988 047476

```

INCRPT:
INCPAT: SUBTST <<SUBR INCREMENT PATTERN TESTING >>
:*****
:*SUBTEST SUBR INCREMENT PATTERN TESTING
:*****
;INCREMENT THE PATTERN & SET UP THE CONDITION CODES
;RESULT - 2 BIT SET INDICATES OVERFLOW
INC PATTERN
CMP #30,PATTERN ;SET UP CONDITION CODES
RETURN ;NOT EQUAL TO ZERO IS GOOD (NO OVERFLOW)

```

11989
11990
11991 047476 005237 002110
11992 047502 022737 000030 002110
11993 047510 000207
11994
11995 047512
11996 047512

```

SETPAT:
HIPAT: SUBTST <<SUBR SET HIGHEST PATTERN TESTING TYPE>>
:*****
:*SUBTEST SUBR SET HIGHEST PATTERN TESTING TYPE
:*****
MOV #27,PATTERN ;SET HIGHEST PATTERN
RETURN

```

11997 047512 012737 000027 002110
11998 047520 000207
11999
12000 047522

```

INCBNK: SUBTST <<SUBR INCREMENT BANK & TEST>>
:*****
:*SUBTEST SUBR INCREMENT BANK & TEST
:*****
;RESULTS RETURNED IN CONDITION CODES
INC BANK
CMP LASTBANK,BANK ;TOO FAR?
RETURN

```

12001
12002 047522 005237 002100
12003 047526 023737 002554 002100
12004 047534 000207

12007 047536

BOOT: SUBTST <<BOOTSTRAP ROUTINE>>
:*****
:*SUBTEST BOOTSTRAP ROUTINE
:*****

12008
12009
12010
12011
12012
12013
12014 047536 104472
12015 047540
12016 047546
12017 047560 004737 024570
12018 047564 104421
12019 047566 005737 002452
12020 047572 001003
12021 047574 042737 000040 172516
12022 047602 005001
12023 047604 000005
12024 047606 012700 177406
12025 047612 010160 000004
12026 047616 012710 177400
12027 047622 012740 000005
12028 047626 105710
12029 047630 100376
12030 047632 062701 020000
12031 047636 005710
12032 047640 100761
12033 047642 005007

:INITIALIZE ALL CSR'S
:UNRELOCATE IF NECESSARY
:FLUSH OUT ANY DBE'S
:TURN OFF MEMORY MANAGEMENT
:TURN OFF THE UNIBUS MAP
:BOOT RKO OR RK1
ECCINIT ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
SET4 #BOOT1 ;TRAPS TO 4 GO TO BOOT1
IF RLFLAG IS TRUE THEN \$CALL UNRELOCATE
CALL MT0030 ;FLUSH OUT DBE'S
DEENERGIZE ;TURN OFF MEMORY MANAGEMENT
TST NO22BIT ;IS THIS AN 11/44 OR 11/24?
BNE BOOT1
BIC #BIT5,MMR3 ;TURN OFF THE UNIBUS MAP
BOOT1: CLR R1
1\$: RESET
MOV #177406,R0
MOV R1,4(R0)
MOV #177400,(R0)
MCV #5,-(R0)
2\$: TSTB (R0)
BPL 2\$
ADD #BIT13,R1
TST (R0)
BMI 1\$
CLR PC

CZMSP80 MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 377
BOOTSTRAP ROUTINE

12036 047644
12037 047644 004737 047676
12038 047650
12039 047664 000777
12040 047666
12041 047670 000000
12042 047672 000137 003656
12043 047676
12044
12045 047676

```
EXIT:  SUBTST  <<HALT PROGRAM>>
:*****
:*SUBTEST    HALT PROGRAM
:*****
      CALL    SHUTUP
EXIT2:  IF APTFLAG IS TRUE OR ACTFLAG IS TRUE
      BR      .
      ELSE
SEXHALT: HALT
      JMP     START
      END ;OF IF APTFLAG
```

12046
12047
12048
12049
12050
12051
12055 047676 104472
12056 047700
12057 047712
12058 047720 004737 024570
12059 047724
12060 047724 012700 000001
12061 047730 013701 002564
12062 047734 004737 046422
12063 047740 104421
12064 047742 005737 002452
12065 047746 001003
12066 047750 042737 000040 172516
12070 047756 000207
12071
12072 047760

```
SHUTUP: SUBTST  <<SHUTDOWN DIAGNOSTIC>>
:*****
:*SUBTEST    SHUTDOWN DIAGNOSTIC
:*****
      :INITIALIZE ALL CSR'S
      :UNRELOCATE
      :FLUSH OUT DBE'S
      :RESTORE LOADERS
      :TURN OFF MEMORY MANAGEMENT
      :UNMAP THE UNIBUS MAP
      ECCINIT      :TRAP ON DOUBLE BIT ERRORS (NORMAL)
      IF RLFLAG IS TRUE THEN $CALL UNRELOCATE
      IF QUICK IS FALSE
      CALL MT0030      :FLUSH OUT DBE'S
      END ;OF IF QUICK
      MOV #1,R0      :DESTINATION BANK
      MOV LOADHOME,R1 :SOURCE BANK
      CALL BANKMOV
      DEENERGIZE      :TURN OFF MEMORY MANAGEMENT
      TST NO22BIT      :DOES THIS PDP-11 HAVE 22-BIT ADDR?
      BNE 1$          :BRANCH IF NOT
      BIC #BITS,MMR3   :TURN OFF UNIBUS MAP
1$:  RETURN
```

12073 047760
12074 047774
12075 050002 012737 047760 060024
12076 050010 012737 000340 060026
12077 050016 012737 000000 127760
12078 050024 104417
12079 050026 000000

```
APTDOWN:SUBTST <<APT SHUTDOWN SEQUENCE>>
:*****
:*SUBTEST    APT SHUTDOWN SEQUENCE
:*****
      MAP #0      :MAP SUPERVISOR SPACE (TEST AREA) TO BANK #0
      TESTAREA    :ENTER TEST MODE
      MOV #APTDOWN,FIRST+24
      MOV #340,FIRST+26
      MOV #0,FIRST+APTDOWN
      KERNEL      :ENTER KERNEL MODE
APTHLT: HALT
```

CZMSP80 M511-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 379
APT SHUTDOWN SEQUENCE

12082 050030

```

SUBSTST <<BLOCK MOVE SUBROUTINE>>
:*****
:*SUBTEST      BLOCK MOVE SUBROUTINE
:*****
:BLOCK3 HAS 3 ARGUEMENTS
:BLOCK2 HAS 2 ARGUEMENTS
:BLOCK1 HAS 1 ARGUEMENTS
:
:ALL ARE CALLED BY THE BMOV MACRO
.ENABL  LSB
BLOCK1: PUSH  R0,R1,R2
        MOV   #FASTCITY,R2
        MOV   #16.,R1
        BR    3$
BLOCK2: PUSH  R0,R1,R2
        MOV   #16.,R1
        BR    2$
BLOCK3: PUSH  R0,R1,R2
        MOV   (R5)+,R1
2$:     MOV   (R5)+,R2
3$:     MOV   (R5)+,R0
1$:     MOV   (R0)+,(R2)+
        SOB  R1,1$
        POP  R2,R1,R0
        RTS  R5
        .DSABL  LSB

```

```

12083
12084
12085
12086
12087
12088
12089 050030
12090 050036 012702 177640
12091 050042 012701 000020
12092 050046 000413
12093
12094 050050
12095 050056 012701 000020
12096 050062 000404
12097
12098 050064
12099 050072 012501
12100 050074 012502
12101 050076 012500
12102
12103 050100 012022
12104 050102 077102
12105 050104
12106 050112 000205
12107

```

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 380
FIELD SERVICE MODE

12109
12110
12111 050114

.SBTTL FIELD SERVICE MODE

FIELDSERVICE:SUBTST <<SUBR FIELD SERVICE COMMAND MODE>>
:*****
:*SUBTEST SUBR FIELD SERVICE COMMAND MODE
:*****

12112 050114 104415
12113 050116
12114
12115 050122
12116 050136
12117 050142 104416
12118 050144 000207
12119 050146
12120 050146 005737 002542
12121 050152 001402
12122 050154
12123 050160
12124 050170 104424
12125 050172
12126 050200
12127 050204 104414
12128 050206
12129 050210 020027 000022
12130 050214 101403
12131 050216
12132 050222 000766
12133 050224
12134 050234 050312
12135 050236 050414
12136 050240 050524
12137 050242 050672
12138 050244 051146
12139 050246 051466
12140 050250 052404
12141 050252 052412
12142 050254 052704
12143 050256 053132
12144 050260 053424
12145 050262 053452
12146 050264 053474
12147 050266 053514
12148 050270 053536
12149 050272 053554
12150 050274 053640
12151 050276 053702
12152 050300 053716
12153 050302
12154 050310 000733

SAVREG
TYPE MSG020 ;FIELD SERVICE COMMAND MODE

IF RLFLAG IS TRUE OR NOFSMODE IS TRUE
TYPE MSG048 ;NOT AVAILABLE NOW - TRY LATER!
RESREG
RETURN
END ;OF IF RLFLAG
TST CACHKN
BEQ 1\$
PUSH CONTRL ;SAVE CACHE STATUS
1\$: PUSH CSRNO,KAMIKAZE ;SAVE CSR & KAMIKAZE STATUS
CACHOFF ;TURN CACHE OFF
SET KAMIKAZE
FS1: TYPE MSG026 ;COMMAND:
RDDEC ;READ A DECIMAL NUMBER
POP RO ;COMMAND --> RO
CMP RO,#18.
BLOS 1\$
TYPE MSG021
BR FS1

1\$: CASE RO
FSCMD0 ;EXIT FIELD SERVICE COMMANDS
FSCMD1 ;READ CSR
FSCMD2 ;LOAD CSR
FSCMD3 ;EXAMINE MEMORY
FSCMD4 ;MODIFY MEMORY
FSCMD5 ;SELECT BANK & PATTERN
FSCMD6 ;TYPE CONFIGURATION MAP
FSCMD7 ;SOB-A-LONG TEST
FSCMD8 ;ERROR SUMMARY
FSCMD9 ;REFRESH TEST
FCMD10 ;SET FILL COUNT
FCMD11 ;ENTER KAMIKAZE MODE
FCMD12 ;EXIT KAMIKAZE MODE
FCMD13 ;TURN CACHE OFF
FCMD14 ;TURN CACHE ON
FCMD15 ;TEST ONLY SELECTED BANKS
FCMD16 ;RESUME TESTING ALL BANKS
FCMD17 ;ENABLE TRACE
FCMD18 ;DISABLE TRACE
END ;OF CASE
BR FS1

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 382
SUBR FIELD SERVICE COMMAND MODE

12157 050312

```

FSCMD0: SUBTST <<COMMAND 0      EXIT>>
:*****
:*SUBTEST      COMMAND 0      EXIT
:*****
      TYPE      MSG103      ;LEAVING FIELD SERVICE MODE
      ADD      #2,SP
      IF SKIPKAMI IS TRUE
      ADD      #2,SP      ;THROW AWAY OLD KAMIKAZE FLAG
      CLR      SKIPKAMI
      ELSE
      POP      KAMIKAZE      ;RESTORE OLD KAMIKAZE FLAG
      END ;OF IF SKIPKAMI
      POP      CSRNO
      TST      CACHKN
      BEQ      RES0
      IF CACHKN EQ CACHKF      ;IF CACHE IS OFF
      ADD      #2,SP      ;THROW AWAY CACHE STATUS
      ELSE
      TST      CACHKN
      BEQ      RES0
      POP      CONTRL      ;RESTORE CACHE STATUS
      END ;OF IF CACHKN
RES0:  RESREG
      RETURN

```

12158 050312
12159 050316 062706 000002
12160 050322
12161 050330 062706 000002
12162 050334 005037 002006
12163 050340
12164 050342
12165 050346
12166 050346
12167 050352 005737 002542
12168 050356 001414
12169 050360
12170 050370 062706 000002
12171 050374
12172 050376 005737 002542
12173 050402 001402
12174 050404
12175 050410
12176 050410 104416
12177 050412 000207
12178
12179 050414

```

FSCMD1: SUBTST <<FS      COMMAND 1      READ CSR>>
:*****
:*SUBTEST      FS      COMMAND 1      READ CSR
:*****
      CALL      WHICHCSR
      MOV      SP,FSSTACK
      SET4     #RES1      ;TRAPS TO 4 GOTO RES1
      READCSR
      SET      NOERROR
      ERROR    +26      ;USE ERROR ROUTINE FOR PRINTOUT
      RES4
      RETURN      ;RESET TRAPS TO 4 TO DEFAULT
RES1:  TYPE      MSG025      ;THIS CSR DOES NOT EXIST
      MOV      FSSTACK,SP
      RES4
      RETURN      ;RESET TRAPS TO 4 TO DEFAULT

```

12180 050414 004737 053730
12181 050420 010637 002304
12182 050424
12183 050432 104426
12184 050434
12185 050442 104026
12186 050444
12187 050466 000207
12188 050470
12189 050474 013706 002304
12190 050500
12191 050522 000207

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 384
 FS COMMAND 1 READ CSR

12194 050524

```
FSCMD2: SUBTST <<FS COMMAND 2 LOAD CSR>>
:*****
:*SUBTEST FS COMMAND 2 LOAD CSR
:*****
```

```
12195 050524 004737 053730
12196 050530 010637 002304
12197 050534
12198 050542 104426
12199 050544
12200 050550
12201 050556 104026
12202 050560
12203 050602
12204 050606 104413
12205 050610
12206 050614 104425
12207 050616 104426
12208 050620
12209 050624
12210 050632 104026
12211 050634 000207
12212 050636
12213 050642 013706 002304
12214 050646
12215 050670 000207
```

```
CALL WHICHCSR
MOV SP,FSSTACK
SET4 #RES2 ;TRAPS TO 4 GOTO RES2
READCSR
TYPE MSG027
SET NOERROR
ERROR +26 ;USE ERROR ROUTINE FOR PRINTOUT
RES4 ;RESET TRAPS TO 4 TO DEFAULT
TYPE MSG023 ;FIRST CSR WORD
RDOCT ;READ AN OCTAL NUMBER
POP CSR ;PUT IN IN LOC 'CSR'
LOADCSR
READCSR
TYPE MSG028
SET NOERROR
ERROR +26 ;USE FOR PRINTOUT - NOT AN ERROR
RETURN
RES2: TYPE MSG025 ;THIS CSR DOES NOT EXIST
MOV FSSTACK,SP
RES4 ;RESET TRAPS TO 4 TO DEFAULT
RETURN
```

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 386
 FS COMMAND 2 LOAD CSR

12218 050672

```

FSCMD3: SUBTST <<FS COMMAND 3 EXAMINE MEMORY>>
:*****
:*SUBTEST FS COMMAND 3 EXAMINE MEMORY
:*****
12219 050672 PUSH BANK,NOPAR,PARTHERE,4
12220 050712 012737 000002 002074 MOV #2,NOPAR ;INDICATE PARITY ACTION
12221 050720 TYPE MSG029 ;EXAMINE MEMORY
12222 050724 1$: TYPE MSG031 ;PHYSICAL ADDRESS (0-17775776)??
12223 050730 104413 RDOCT ;READ OCTAL NUMBER ONTO STACK & $HIOCT
12224 050732 013737 065204 002100 MOV $HIOCT,BANK ;PUT MSB'S IN BANK
12225 050740 POP RO ;PUT LSB'S IN RO
12226 050742 000241 CLC
12227 050744 006100 ROL RO
12228 050746 006137 002100 ROL BANK
12229 050752 000241 CLC
12230 050754 006000 ROR RO
12231 050756 023737 002100 002554 CMP BANK,LASTBANK ;CHECK FOR BANK TOO HIGH
12232 050764 003357 BGT 1$ ;BRANCH IF TRUE
12233 050766 062700 060000 ADD #FIRST,RO
12234 050772 032700 000001 BIT #BIT0,RO ;CHECK FOR ODD ADDRESS
12235 050776 001352 BNE 1$ ;BRANCH IF ODD ADDRESS
12236 051000 020027 157776 CMP RO,#LAST ;CHECK FOR ADDRESS OVER 16K
12237 051004 101347 BHI 1$ ;BRANCH IF OVER 16K
12238 051006 012737 051060 002302 MOV #3$,PARTHERE ;INCASE OF ABORTS
12239 051014 SET4 #4$ ;TRAPS TO 4 GOTO 4$
12240 051022 MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
12241 051036 TESTAREA ;ENTER TEST MODE
12242 051044 011001 MOV (RO),R1
12243 051046 104417 KERNEL ;ENTER KERNEL MODE
12244 051050 TYPOCS R1
12245 051056 000410 BR EXCMD3
12246
12247 051060 3$: TYPE MSG032 ;PARITY ABORT
12248 051064 000405 BR EXCMD3
12249
12250 051066 062706 000004 4$: ADD #4,SP ;FIX STACK
12251 051072 TYPE MSG033 ;TIMEOUT TRAP
12252 051076 000400 BR EXCMD3
12253
12254 051100 104417 EXCMD3: KERNEL ;ENTER KERNEL MODE
12255 051102 POP 4,PARTHERE,NOPAR,BANK
12256 051122 RES4 ;RESET TRAPS TO 4 TO DEFAULT
12257 051144 000207 RETURN
    
```


CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 388
 FS COMMAND 3 EXAMINE MEMORY

```

12260 051146          FSCMD4: SUBTST <<FS  COMMAND 4  MODIFY MEMORY>>
:*****
:SUBTEST           FS  COMMAND 4  MODIFY MEMORY
:*****
12261 051146          PUSH   BANK,NOPAR,PARTHERE,4
12262 051166 012737 000003 002074  MOV   #3,NOPAR           ;INDICATE PARITY ACTION
12263 051174          TYPE   MSG036           ;MODIFY MEMORY
12264 051200          1$:   TYPE   MSG031           ;PHYSICAL ADDRESS (0-17775776)??
12265 051204 104413          RDOCT          ;READ OCTAL NUMBER ONTO STACK & $HIOCT
12266 051206 013737 065204 002100  MOV   $HIOCT,BANK      ;PUT MSB'S IN BANK
12267 051214          POP    RO              ;PUT LSB'S IN RO
12268 051216 000241          CLC
12269 051220 006100          ROL    RO
12270 051222 006137 002100          ROL    BANK
12271 051226 000241          CLC
12272 051230 006000          ROR    RO
12273 051232          IF BANK GT LASTBANK THEN GOTO 1$ ;CHECK FOR BANK TOO HIGH
12274 051242 062700 060000          ADD   #FIRST,RO
12275 051246          IF #BIT0 SET.IN RO THEN GOTO 1$ ;CHECK FOR ODD ADDRESS
12276 051254          IF RO HI #LAST THEN GOTO 1$     ;CHECK FOR ADDRESS OVER 16K
12277 051262 012737 051330 002302  MOV   #3$,PARTHERE    ;INCASE OF ABORTS
12278 051270          SET4  #4$             ;TRAPS TO 4 GOTO 4$
12279 051276          MAP   BANK           ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
12280 051312 104511          INVALIDATE
12281 051314          TESTAREA             ;ENTER TEST MODE
12282 051322 011001          MOV   (RO),R1
12283          ;GETTING HERE MEANS WE GOT LUCKY - NO TRAPS
12284 051324 104417          KERNEL           ;ENTER KERNEL MODE
12285 051326 000410          BR    5$
12286
12287 051330          3$:   TYPE   MSG032           ;PARITY ABORT
12288 051334 000431          BR    EXCMD4         ;EXIT
12289
12290 051336 062706 000004          4$:   ADD   #4,SP       ;FIX STACK
12291 051342          TYPE   MSG033           ;TIMEOUT TRAP
12292 051346 000424          BR    EXCMD4         ;EXIT
12293
12294 051350          5$:   TYPE   MSG037           ;OLD DATA WAS
12295 051354          TYPOCS R1             ;PRINT IT
12296 051362          TYPE   MSG039           ;INPUT NEW DATA
12297 051366 104413          RDOCT          ;READ ON OCTAL NUMBER ONTO THE STACK
12298 051370          POP    R1             ;GET NEW NUMBER
12299 051372          TESTAREA             ;ENTER TEST MODE
12300 051400 010110          MOV   R1,(RO)         ;PUT IT IN MEMORY
12301 051402 011001          MOV   (RO),R1        ;READ IT AGAIN
12302 051404 104417          KERNEL           ;ENTER KERNEL MODE
12303 051406          TYPE   MSG038           ;DATA IS NOW
12304 051412          TYPOCS R1             ;PRINT IT
12305
12306 051420 104417          EXCMD4: KERNEL       ;ENTER KERNEL MODE
12307 051422          POP    4,PARTHERE,NOPAR,BANK
12308 051442          RES4
12309 051464 000207          RETURN             ;RESET TRAPS TO 4 TO DEFAULT

```

12312 051466

FSCMD5: SUBTST <<FS COMMAND 5 SELECT BANK & PATTERN>>
 :*****
 :+SUBTEST FS COMMAND 5 SELECT BANK & PATTERN
 :*****

12313 051466
 12314 051516 010637 002304
 12315 051522
 12316 051526
 12317 051532 104413
 12318 051534
 12319 051540

PUSH BANK,PATTERN,TESTADD,PCBUMP,TKVEC,TKVEC+2
 MOV SP,FSSTACK ;SAVE LAST GOOD STACK POINTER
 TYPE MSG040 ;SELECT BANK & PATTERN TEST
 1\$: TYPE MSG030 ;BANK(0-177)?
 RDOCT ;READ AN OCTAL NUMBER ONTO THE STACK
 POP BANK ;PUT IT IN BANK
 IF BANK GT LASTBANK THEN GOTO 1\$;CHECK FOR BANK TOO HIGH

12320
 12321 051550 013701 002100
 12322 051554 006301
 12323 051556 006301
 12324 051560
 12325 051570
 12326 051574
 12327 051576

MOV BANK,R1
 ASL R1
 ASL R1
 IF CPUBIT OFF.IN CONFIG(R1)
 TYPE MSG041 ;BANK NOT ACCESSABLE
 GOTO 1\$
 END ;OF IF

12328
 12329 051576
 12330 051602 104413
 12331 051604
 12332 051610
 12333 051620
 12334 051626
 12335 051632 104413
 12336 051634
 12337 051636

2\$: TYPE MSG042 ;PATTERN(0-45)?
 RDOCT ;READ AN OCTAL NUMBER ONTO THE STACK
 POP PATTERN ;PUT IT IN PATTERN
 IF PATTERN GT #47 THEN GOTO 2\$;CHECK FOR PATTERN TO HIGH
 IF PATTERN EQ #0
 TYPE MSG043 ;PATTERN 0 DATA IS?
 RDOCT ;READ AN OCTAL NUMBER ONTO THE STACK
 POP R2 ;PUT IT IN R2
 END ;OF IF

12338
 12339
 12340 051636
 12341 051652 104511
 12342 051654 004737 047032
 12343 051660
 12344 051666
 12345 051672 000137 052306
 12346 051676

MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
 INVALIDATE
 CALL EXBANK ;SET NEW MARGINS
 IF RRFLAG IS TRUE
 TYPE MSG049 ;BANK REQUIRES RELOCATION
 JMP CMD5C
 END ;OF IF RRFLAG

12347 051676
 12348 051702 013737 002150 002152
 12349 051710 013702 002100
 12350 051714 072227 000002
 12351 051720 016203 002652
 12352 051724 072327 177770
 12353 051730 042703 177760
 12354 051734 006303
 12355 051736 010337 002150
 12356 051742 012737 052306 000060
 12357 051750 012737 000340 000062
 12358 051756 017700 130650
 12359 051762 042737 000200 177776
 12360 051770 052777 000100 130632

TYPE MSG046 ;TO ESCAPE TYPE ANY KEY!
 MOV CSRNO,SAVCSR ;SAVE OLD CSR NUMBER
 MOV BANK,R2
 ASH #2,R2 ;GENERATE INDEX INTO CONFIGURATION TABLE
 MOV CONFIG(R2),R3 ;R3 = LOW WORD OF CONFIGURATION TABLE FOR THIS BANK
 ASH #-10,R3 ;POSITION CSR CODE IN BITS 0-3
 BIC #^C17,R3 ;CLEAR ALL BUT THE CSR CODE
 ASL R3 ;ADJUST CSR NUMBER
 MOV R3,CSRNO
 MOV #CMD5C,TKVEC
 MOV #340,TKVEC+2
 MOV @STKB,R0 ;KILL ANY OLD INTERRUPT
 BIC #BIT7,PSW ;LOWER CPU PRIORITY TO 140
 BIS #BIT6,@STKS ;ENABLE KEYBOARD INTERRUPTS

12361
 12362
 12363 051776
 12364 052012 013701 002100
 12365 052016 006301

CMD5B: SET HEADER,MUT
 MOV BANK,R1
 ASL R1

CZMSP80 MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 390-1
FS COMMAND 5 SELECT BANK & PATTERN

12366	052020	006301			ASL	R1		
12367	052022	005037	002236		CLR	SPLTCSR		
12368	052026	005037	002262		CLR	PASFLG		
12369	052032	012737	060000	002410	MOV	#FIRST,TESTADD		
12370	052040	012737	060002	002412	MOV	#FIRST+2,TESTADD+2		
12371	052046				IF #BIT12	SET.IN CONFIG+2(R1)		
12372	052056	905237	002236		INC	SPLTCSR		
12373	052062				MAP	BANK		
12374	052076	012737	120000	002412	MOV	#120000,TESTADD+2		
12375	052104				END; OF IF	#BIT12		
12376	052104				IF #SWO	SET.IN @SWR		
12377	052114	104470			ECCDIS			:DISABLE ERROR CORRECTION
12378	052116				ELSE			
12379	052120				PUSH	CSRNO		
12380	052124	104502			CLRCSR			:CLEAR CSRS
12381	052126				POP	CSRNO		
12382	052132				END ;OF IF			
12383	052132	012737	000002	002074	MOV	#2,NOPAR		:PARITY ACTION
12384	052140	012737	000002	002324	MOV	#2,PCBUMP		:TRAPS ADD 2 TO PC
12385	052146	013700	002110		MOV	PATTERN,R0		
12386	052152	006300			ASL	R0		
12387	052154	004770	052166		CALL	@FSPAT(R0)		
12388	052160	005037	002074		CLR	NOPAR		
12389	052164	000712			BR	CMD5B		:LOOP TILL KEYBOARD INTERRUPT
12390								
12391	052166	020702			FSPAT: MT0000	:<1 SEC		DATA PATTERN TEST
12392	052170	020762			MT0001	:<1 SEC		ADDRESS TEST
12393	052172	021102			MT0002	:<1 SEC		COMPLEMENT ADDRESS TEST
12394	052174	021242			MT0003	: 1 SEC		3 XOR 9 WORST CASE NOISE TEST
12395	052176	021474			MT0004	: 1 SEC		ROTATING ZEROS TEST
12396	052200	021616			MT0005	: 1 SEC		ROTATING ONES TEST
12397	052202	021752			MT0006	:<1 SEC		INITIAL DATA TEST
12398	052204	022006			MT0007	:<1 SEC		ADDRESS BIT TEST
12399	052206	022050			MT0010	:<1 SEC		BYTE ADDRESSING TEST
12400	052210	022104			MT0011	:<2 SEC		CREATE SINGLE BIT ERROR TEST
12401	052212	022162			MT0012	:<1 SEC		WRITE BYTE CLEARS SBE TEST
12402	052214	022266			MT0013	: 1 SEC		CREATE DOUBLE BIT ERROR TEST
12403	052216	022352			MT0014	: 1 SEC		BASIC DOUBLE BIT ERROR TEST
12404	052220	022442			MT0015	: 1 SEC		WRITE INHIBIT OF BYTE WITH DBE
12405	052222	022520			MT0016	:<1 SEC		WRITE INHIBIT OF WORD WITH DBE
12406	052224	022576			MT0017	:<1 SEC		HOLDING 1'S & 0'S TEST
12407	052226	022620			MT0020	: 1 SEC		SYNDROMES TO CSR ON SBE TEST
12408	052230	022710			MT0021	: 1 SEC		MARCHING 0'S & 1'S TEST
12409	052232	023162			MT0022	:10 SEC		REFRESH & SHIFTING DIAGONAL TEST
12410	052234	023214			MT0023	:10 SEC		SHIFTING DIAGONAL TEST
12411	052236	023260			MT0024	:20 SEC		FAST GALLOPING PATTERN TEST
12412	052240	023524			MT0025	:<1 SEC		INTERRUPT ENABLE TEST
12413	052242	023602			MT0026	:<1 SEC		RANDOM DATA TEST
12414	052244	024104			MT0027	: 1 SEC		UNIQUE BANK TEST
12415	052246	024570			MT0030	: 1 SEC		FLUSH OUT DBE'S TEST
12416	052250	025072			MT0031	: 3 SEC		SOB-A-LONG TEST
12417	052252	025262			MT0032	:<1 SEC		WRITE RECOVERY TEST
12418	052254	025614			MT0033	:35 SEC		BRANCH GOBBLE TEST
12419	052256	026002			MT0034	: 1 SEC		SOFT ERROR TEST
12420	052260	026154			MT0035	:<1 SEC		WORST CASE NOISE PARITY TEST
12421	052262	026266			MT0036	: 1 SEC		CORRECTION CODE TEST
12422	052264	026340			MT0037	:<1 SEC		ECC DISABLE TEST

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 390-2
 FS COMMAND 5 SELECT BANK & PATTERN

12423	052266	026406			MT0040	: 1 SEC	NO WRITE ABORT WITH ECC DISABLED TEST
12424	052270	026410			MT0041	: 1 SEC	ADDRESS TO CSR ON DOUBLE BIT ERROR TEST
12425	052272	026452			MT0042	:<1 SEC	EXTENDED UNIBUS ADDRESS TEST
12426	052274	026516			MT0043	: 1 SEC	WRITE BYTE CLEARS SBE TEST
12427	052276	026556			MT0044	: 5 SEC	SHIFTING 1/0'S THROUGH THE CHECK BITS
12428	052300	026620			MT0045	: 1 SEC	SYNDROMES TO CSR ON DBE TEST
12429	052302	026660			MT0046	: 1 SEC	CHECK SINGLE BIT ERROR WITH ECC DISABLED TEST
12430	052304	026720			MT0047	:<1 SEC	NO CSR UPDATE WITH SBE ON DBE TEST
12431							
12432	052306	013706	002304		CMD5C: MOV	FSSTACK,SP	:RECOVER OLD STACK POINTER
12433	052312	042777	000100	130310	BIC	#BIT6,@\$TKS	
12434	052320				POP	TKVEC+2,TKVEC	
12435	052330	117700	130276		MOVB	@\$TKB,R0	:GET CHARACTER TO GET RID OF FLAG
12436	052334				POP	PCBUMP,TESTADD	
12437	052344				POP	PATTERN,BANK	
12438	052354				MAP	BANK	:REMAP OLD BANK
12439	052370	004737	047032		CALL	EXBANK	
12440	052374	013737	002152	002150	MOV	SAVCSR,CSRNO	:RESTORE CSRNO.
12441	052402	000207			RETURN		

;:IL

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 391
FS COMMAND 5 SELECT BANK & PATTERN

12443 052404

FSCMD6: SUBTST <<FS COMMAND 6 TYPE CONFIGURATION MAP>>
:.....
:*SUBTEST FS COMMAND 6 TYPE CONFIGURATION MAP
:.....

12444 052404 004737 041364
12445 052410 000207
12446

CALL PCONFIG
RETURN

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 393
 FS COMMAND 6 TYPE CONFIGURATION MAP

```

12449 052412      FSCMD7: SUBTST <<FS      COMMAND 7      SOB-A-LONG TEST>>
;*****
; *SUBTEST      FS      COMMAND 7      SOB-A-LONG TEST
;*****
12450 052412      PUSH      BANK,PATTERN,TKVEC,TKVEC+2,NOPAR
12451 052436      010637  002304      MOV      SP,FSSTACK      ;SAVE LAST GOOD STACK POINTER
12452 052442      TYPE      MSG055      ;SOB-A-LONG TEST
12453
12454 052446      IF #SW0 SET.IN @SWR
12455 052456      104470      ECCDIS      ;DISABLE ERROR CORRECTION
12456 052460      ELSE
12457 052462      104502      CLRCSR      ;CLEAR CSRS
12458 052464      END ;OF IF
12459 052464      TYPE      MSG056      ;BELL = EACH PASS COMPLETE
12460
12461 052470      TYPE      MSG046      ;TO ESCAPE TYPE ANY KEY!
12462 052474      012737  052620  000060      MOV      #CMD7C,TKVEC
12463 052502      012737  000340  000062      MOV      #340,TKVEC+2
12464 052510      017700  130116      MOV      @STKB,RO      ;KILL ANY OLD INTERRUPT
12465 052514      042737  000200  177776      BIC      #BIT7,PSW      ;LOWER CPU PRIORITY TO 140
12466 052522      052777  000100  130100      BIS      #BIT6,@STKS      ;ENABLE KEYBOARD INTERRUPTS
12467
12468
12469 052530      SET      HEADER,MUT
12470
12471 052544      CMD7B:  FOR BANK := #0 TO LASTBANK
12472 052550      004737  047032      CALL EXBANK
12473 052554      IF ACFLAG IS TRUE AND RRFLAG IS FALSE
12474 052570      104511      INVALIDATE
12475 052572      004737  025072      CALL MT0031
12476 052576      END ;OF IF ACFLAG
12477 052576      END ;OF FOR BANK
12478 052612      TYPE      $BELL      ;RING BELL
12479 052616      GOTO     CMD7B
12480
12481 052620      013706  002304      CMD7C:  MOV      FSSTACK,SP      ;RECOVER OLD STACK POINTER
12482 052624      042777  000100  127776      BIC      #BIT6,@STKS
12483 052632      117700  127774      MOV      @STKB,RO      ;READ CHAR TO KILL FLAG
12484 052636      POP      NOPAR,TKVEC+2,TKVEC,PATTERN,BANK
12485 052662      MAP      BANK      ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
12486 052676      004737  047032      CALL     EXBANK
12487 052702      000207      RETURN

```


12490 052704

FSCMD8: SUBTST <<FS COMMAND 8 ERROR SUMMARY>>
:*****
:*SUBTEST FS COMMAND 8 ERROR SUMMARY
:*****

12491 052704
12492 052716 013737 065712 002432
12493 052724 005337 002432
12494 052730
12495 052736 013737 172350 002266
12496 052744 012737 001000 172350
12497 052752
12498 052756
12499 052764
12500 052770
12501 052776 005037 002332
12502 053002
12503 053006 013703 002100
12504 053012 070327 000004
12505 053016
12506 053024
12507 053032
12508 053036
12509 053044
12510 053044
12511 053054 116300 002654
12512 053060 042700 177400
12513 053064
12514 053070
12515 053074
12516 053074
12517 053110
12518 053110 013737 002266 172350
12519 053116
12520 053130 000207

PUSH R0,R2,R3,BANK
MOV \$PASS,TEMP
DEC TEMP
TYPDEC TEMP
MOV KIPAR4,SAV4
MOV #1000,KIPAR4
TYPE MSG125
TYPDEC \$ERTTL
TYPE MSG079
IF \$ERTTL NE #0
CLR SUCCESS
FOR BANK := #0 TO LASTBANK
MOV BANK,R3
MUL #4,R3
IFB CONFIG+2(R3) NE #0
IF SUCCESS IS FALSE
TYPE MSG076
SET SUCCESS
END :OF IF SUCCESS
TYPOCS BANK,3
MOVB CONFIG+2(R3),R0
BIC #^C377,R0
TYPDEC R0
TYPE \$CRLF
END :OF IFB CONFIG(R3)
END :OF FOR BANK
END :OF IF \$ERTTL
MOV SAV4,KIPAR4
POP BANK,R3,R2,R0
RETURN

:::I.L.C.:REV B
:::I.L.C.:REV B
:PASSES COMPLETED
:ERROR(S) DETECTED
:BANK ERRORS
:::I.L.C.:REV B

12523 053132

```
FSCMD9: SUBTST <<FS COMMAND 9 REFRESH TEST>>
:*****
:SUBTEST FS COMMAND 9 REFRESH TEST
:*****
```

12524 053132

12525 053156 010637 002304

12526 053162

12527

12528 053166

12529 053176 104470

12530 053200

12531 053202 104502

12532 053204

12533 053204

12534

12535 053210

12536 053214 012737 053340 000060

12537 053222 012737 000340 000062

12538 053230 017700 127376

12539 053234 042737 000200 177776

12540 053242 052777 000100 127360

12541

12542 053250

12543

12544 053264

12545 053270 004737 047032

12546 053274

12547 053310 104511

12548 053312 004737 023162

12549 053316

12550 053316

12551 053332

12552 053336

12553

12554 053340 013706 002304

12555 053344 042777 000100 127256

12556 053352 117700 127254

12557 053356

12558 053402

12559 053416 004737 047032

12560 053422 000207

12561

```

PUSH BANK,PATTERN,TKVEC,TKVEC+2,NOPAR
MOV SP,FSSTACK ;SAVE LAST GOOD STACK POINTER
TYPE MSG073 ;REFRESH TEST

IF #SW0 SET.IN @SWR
ECCDIS ;DISABLE ERROR CORRECTION
ELSE
CLRCR ;CLEAR CSRS
END ;OF IF
TYPE MSG056 ;BELL = EACH PASS COMPLETE

TYPE MSG046 ;TO ESCAPE TYPE ANY KEY!
MOV #CMD9C,TKVEC
MOV #340,TKVEC+2
MOV @STKB,RO ;KILL ANY OLD INTERRUPT
BIC #BIT7,PSW ;LOWER CPU PRIORITY TO 140
BIS #BIT6,@STKS ;ENABLE KEYBOARD INTERRUPTS

SET HEADER,MUT

CMD9B: FOR BANK := #0 TO LASTBANK
CALL EXBANK
IF ACFLAG IS TRUE AND RRFLAG IS FALSE
INVALIDATE
CALL MT0022
END ;OF IF ACFLAG
END ;OF FOR BANK
TYPE $BELL ;RING BELL
GOTO CMD9B

CMD9C: MOV FSSTACK,SP ;RECOVER OLD STACK POINTER
BIC #BIT6,@STKS
MOVB @STKB,RO ;READ CHAR TO KILL FLAG
POP NOPAR,TKVEC+2,TKVEC,PATTERN,BANK
MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
CALL EXBANK
RETURN
```

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 399
FS COMMAND 9 REFRESH TEST

12564 053424

FCMD10: SUBTST <<FS COMMAND 10 SET FILL COUNT>>
:*****
:*SUBTEST FS COMMAND 10 SET FILL COUNT
:*****

12565 053424

12566 053426

12567 053432 104413

12568 053434

12569 053436 042700 177760

12570 053442 110037 002355

12571 053446

12572 053450 000207

12573

12574 053452

PUSH R0
TYPE MSG085 ;FILL COUNT(OCTAL)?
RDOCT
POP R0
BIC #^C17,R0
MOVB R0,\$FILLS
POP R0
RETURN

FCMD11: SUBTST <<FS COMMAND 11 ENTER KAMIKAZE MODE>>
:*****
:*SUBTEST FS COMMAND 11 ENTER KAMIKAZE MODE
:*****

12575 053452

12576 053456

12577 053472 000207

12578

12579 053474

TYPE MSG101 ;ENTERING KAMIKAZE MODE
SET KAMIKAZE,SKIPKAMI
RETURN

FCMD12: SUBTST <<FS COMMAND 12 EXIT KAMIKAZE MODE>>
:*****
:*SUBTEST FS COMMAND 12 EXIT KAMIKAZE MODE
:*****

12580 053474

12581 053500 005037 002004

12582 053504

12583 053512 000207

12584

12585 053514

TYPE MSG102 ;LEAVING KAMIKAZE MODE
CLR KAMIKAZE
SET SKIPKAMI
RETURN

FCMD13: SUBTST <<FS COMMAND 13 TURN CACHE OFF>>
:*****
:*SUBTEST FS COMMAND 13 TURN CACHE OFF
:*****

12586 053514

12587 053520 104424

12588 053522 013737 002542 002544

12589 053530 005037 002542

12590 053534 000207

12591

12592 053536

TYPE MSG106 ;CACHE IS OFF
CACHOFF ;TURN CACHE OFF
MOV CACHKN,CACHKN+2 ;SAVE OLD CACHE ON STATE
CLR CACHKN ;KEEP CACHE OFF
RETURN

FCMD14: SUBTST <<FS COMMAND 14 TURN CACHE ON>>
:*****
:*SUBTEST FS COMMAND 14 TURN CACHE ON
:*****

12593 053536

12594 053542 013737 002544 002542

12595 053550 104423

12596 053552 000207

12597

TYPE MSG107 ;CACHE IS ON (EXCEPT DURING ACTUAL PATTERNS)
MOV CACHKN+2,CACHKN ;RESTORE OLD CACHE ON STATE
CACHON ;TURN CACHE ON
RETURN

12610
12611 053554

```

FCMD15: SUBTST <<FS COMMAND 15 TEST ONLY SELECTED BANKS>>
:*****
:*SUBTEST FS COMMAND 15 TEST ONLY SELECTED BANKS
:*****
TYPE MSG105 ;ENTER BANKS IN OCTAL - USE NUMBER OUTSIDE RANGE TO TERMINAT
CALL CMD16A ;ERASE OLD SELECTIONS
BEGIN CMD16LOOP
REPEAT
TYPE MSG030 ;BANK(0-177)?
RDOCT ;READ AN OCTAL NUMBER ONTO THE STACK
POP R1 ;PUT IT IN R1
IF R1 GT #177 OR R1 LT #0
LEAVE CMD16LOOP
END ;OF IF R1
ASL R1
ASL R1 ;R1 <- R1 * 4
BIS #BIT14,CONFIG+2(R1)
END ;OF REPEAT
END CMD16LOOP
TYPE MSG110 ;ONLY SELECTED BANKS WILL BE TESTED
SET SELONLY
RETURN

```

12612 053554
12613 053560 004737 053650
12614 053564
12615 053564
12616 053564
12617 053570 104413
12618 053572
12619 053574
12620 053606
12621 053610
12622 053610 006301
12623 053612 006301
12624 053614 052761 040000 002654
12625 053622
12626 053624
12627 053624
12628 053630
12629 053636 000207
12630
12631 053640

```

FCMD16: SUBTST <<FS COMMAND 16 RESUME TESTING ALL BANKS>>
:*****
:*SUBTEST FS COMMAND 16 RESUME TESTING ALL BANKS
:*****
TYPE MSG111 ;ALL BANKS WILL BE TESTED
CLR SELONLY

```

12632 053640
12633 053644 005037 002000
12634
12635
12636 053650 013702 002554
12637 053654 006302
12638 053656 006302
12639 053660
12640 053662 042761 040000 002654
12641 053670
12642 053700 000207

```

:ENTRY POINT FROM CMD15
CMD16A: MOV LASTBANK,R2
ASL R2
ASL R2
FOR R1 := #0 TO R2 BY #4
BIC #BIT14,CONFIG+2(R1)
END ;OF FOR R1
RETURN

```

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 402
FS COMMAND 16 RESUME TESTING ALL BANKS

12645 053702

FCMD17: SUBTST <<FS COMMAND 17 ENABLE TRACE>>
:.....
:*SUBTEST FS COMMAND 17 ENABLE TRACE
:.....

12646 053702

12647 053706 012737 177777 006206
12648 053714 000207

TYPE MSG127
MOV #-1,TRACE
RETURN

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 404
FS COMMAND 17 ENABLE TRACE

12651 053716

FCMD18: SUBTST <<FS COMMAND 18 DISABLE TRACE>>
:.....
:*SUBTEST FS COMMAND 18 DISABLE TRACE
:.....

12652 053716
12653 053722 005037 006206
12654 053726 000207

TYPE MSG128
CLR TRACE
RETURN

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 406
 FS COMMAND 18 DISABLE TRACE

12657 053730

WHICHCSR:SUBTST <<SUBR DETERMINE CORRECT CSR>>

 :*SUBTEST SUBR DETERMINE CORRECT CSR

12658 053730 013700 002222
 12659 053734 022700 100000
 12660 053740 001003
 12661 053742 005037 002150
 12662 053746 000207
 12663
 12664 053750
 12665 053754 104412
 12666 053756
 12667 053760 011000
 12668 053762 020027 000106
 12669 053766 101370
 12670 053770 022700 000101
 12671 053774 103002
 12672 053776 162700 000007
 12673 054002 162700 000060
 12674 054006 006300
 12675 054010 010037 002150
 12676 054014 000207

MOV TOTCSRS,RO :GET CSR'S FLAG
 CMP #BIT15,RO :CSR 0?
 BNE 1\$:NO - SKIP
 CLR CSRNO :YES - SET IT UP
 RETURN
 1\$: TYPE MSG022 :WHICH CSR(0-F)
 RDLIN :GET CHARACTER
 POP RO :PUT IN RO
 MOV (RO),RO :PUT CHAR IN RO
 CMP RO,#106 :CHECK LIMIT
 BHI 1\$:IF BAD LOOP TILL HE TYPES IT RIGHT
 CMP #'A,RO
 BHIS 2\$
 SUB #7,RO
 2\$: SUB #60,RO
 ASL RO
 MOV RO,CSRNO
 RETURN

```

13225
13226 054016
13227 054030
13228 054036
13229 054044
13230 054052 104417
13231 054054
13232 054054
13233 054062
13234 054066
13235 054070
13236 054074
13237 054074 000137 057276
13238
13239 054100

13240 054100
13241 054112
13242 054114 005037 002146
13243 054120 104505
13244 054122
13245 054130 005711
13246 054132 104417
13247 054134 104426
13248 054136 013700 002146
13249 054142 104503
13250 054144 072027 177773
13251 054150 042700 177600
13252 054154
13253 054160 005037 002042
13254 054164
13255 054172 011137 002050
13256 054176 011437 002052
13257 054202 104417
13258 054204 110037 002054
13259 054210 105037 002055
13260 054214 004737 057532
13261 054220 104033
13262 054222
13263 054224
13264 054234 104506
13265 054236
13266 054240 104472
13267 054242
13268 054242 000002
    
```

```

.SBTTL ERROR DATA (SUPERVISOR) SETUP STUFF
$PER25: LET ADDRESS := R1 - #2
        IF ABORTFLAG IS FALSE
            TESTAREA                ;ENTER TEST MODE
            LET BAD := -2(R1)
            KERNEL                    ;ENTER KERNEL MODE
        END ;OF IF ABORTFLAG
        IF 177654 EQ #0
            LET GOOD := R2
        ELSE
            LET GOOD := R3
        END ;OF IF
        JMP PERRAW

PERRA3: SUBTST <<DATA WAS 3 WORDS>>
:*****
:*SUBTEST DATA WAS 3 WORDS
:*****
        IF BADPC EQ #0 THEN $CALL BADSTACK
        PUSH R0
        CLR CSR                        ;MAKE SURE CSR BIT HOLDER IS CLEAR
        CHKIDIS                        ;DISABLE ECC & WRITE CHECKBITS FROM 1 SELECTED CSR
        TESTAREA
        TST (R1)                        ;READ LOCATION TO READ CHECKBITS INTO CSR
        KERNEL
        READCSR                        ;GET CSR CONTENTS
        MOV CSR,R0                      ;SAVE CSR CONTENTS IN R0
        CLR1CSR                        ;RETURN CSR TO NORMAL MODE
        ASH #-5,R0                      ;MOVE CHECK BITS TO BOTTOM OF WORD
        BIC #^C177,R0                  ;CLEAR OFF EXTRANEIOUS GARBAGE
        LET ADDRESS := R1                ;SAVE VIRTUAL ADDRESS FOR PRINTOUT
        CLR GOOD                        ;FIRST TEST WORD WRITTEN SHOULD ALWAYS BE ZERO
        TESTAREA                        ;ENTER TEST MODE
        MOV (R1),BAD                    ;GET BAD DATA FROM MUT - FIRST WORD
        MOV (R4),BAD2                   ;AND SECOND WORD
        KERNEL                          ;ENTER KERNEL MODE
        MOV R0,BAD3                     ;MOVE BAD CHECKBITS FOR PRINTOUT
        CLRB BAD3+1                     ;CLEAR OFF THE OTHER UNUSED BITS
        CALL PERBNK                      ;MARK BANK AS BAD IN CONFIG TABLE
        ERROR +33
        POP R0                            ;RESTORE R0
        IF #SWO SET.IN @SWR
            ENASBE                        ;TRAP ON SINGLE BIT ERRORS
        ELSE
            ECCINIT                       ;TRAP ON UNCORRECTABLE ERRORS
        END; OF IF #SWO
        RTI
    
```

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 423
DATA WAS 3 WORDS

13271 054244
13272 054250
13273 054262
13274 054270
13275 054276
13276 054304 104417
13277 054306
13278 054306 000137 057276
13279
13280 054312

\$PER30: LET GOOD := R1
LET ADDRESS := (SP) - 16
IF ABORTFLAG IS FALSE
TESTAREA ;ENTER TEST MODE
LET BAD := @ADDRESS
KERNEL ;ENTER KERNEL MODE
END ;OF IF ABORTFLAG
JMP PERRAW

GETDATA:SUBTST <<GET DATA FROM ABORTED AREA IF POSSIBLE>>
:*****
:*SUBTEST GET DATA FROM ABORTED AREA IF POSSIBLE
:*****

13281 054312
13282 054324 010637 054410
13283 054330 012737 054370 000004
13284 054336 012737 054370 000114
13285 054344 013700 002032
13286 054350
13287 054356 011037 002050
13288 054362 104417
13289 054364 005037 002142
13290 054370 013706 054410
13291 054374
13292 054406 000207
13293 054410 000000

PUSH R0,4,114
MOV SP,GETDA1
MOV #1\$,4
MOV #1\$,114
MOV ADDRESS,R0
TESTAREA
MOV (R0),BAD
KERNEL
CLR ABORTFLAG
1\$: MOV GETDA1,SP ;RESTORE KNOWN GOOD STACK POINTER
POP 114,4,R0
RETURN
GETDA1: 0


```

13296
13297
13298
13299
13300 054412
13308
13309 054412 005737 002542
13310 054416 001403
13311 054420
13312 054424 104423
13313 054426 012737 055364 000024 5$:
13314 054434 012737 000340 000026
13315 054442
13316
13317 054462 012700 177700
13318 054466 012701 000021
13319 054472
13320 054474 077102
13321
13322 054476 005737 002454
13323 054502 001013
13324 054504 012700 172300
13325 054510 012701 000020
13326 054514
13327 054516 077102
13328 054520
13329
13330 054532 012700 172300
13331 054536 012701 177600
13332 054542 012702 172200
13333 054546 012703 000040
13334 054552 011021
13335 054554 012022
13336 054556 077303

```

```

.SBTTL POWER FAIL AUTO RESTART
.SBTTL ROUTINE POWER DOWN AND UP
:*****
:POWER DOWN ROUTINE
$PWRDN:
:SAVE CACHE STATUS
TST CACHKN
BEQ 5$
PUSH CONTRL
CACHON ;TURN CACHE ON
MOV #SILLUP,PWRVEC ;SET FOR FAST UP
MOV #340,PWRVEC+2 ;:PRIO:7
PUSH R0,R1,R2,R3,R4,R5,CSRNO
:SAVE USER PAR'S & PDR7
MOV #177700,R0
MOV #17.,R1
1$: PUSH -(R0)
SOB R1,1$
:SAVE SUPERVISOR PAR'S
TST NOSUPER
BNE PD1
MOV #172300,R0
MOV #16.,R1
2$: PUSH -(R0)
SOB R1,2$
IF RLFLAG IS TRUE THEN $CALL WOOPS
:COPY KERNEL MAP TO USER & SUPERVISOR
PD1: MOV #KIPDR0,R0
MOV #UIPDR0,R1
MOV #SIPDR0,R2
MOV #32.,R3
3$: MOV (R0),(R1)+
MOV (R0)+,(R2)+
SOB R3,3$

```


CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 428
 ROUTINE POWER DOWN AND UP

```

13385
13386
13387 054766
13391 054766 012737 055364 000024
13392
13393 054774 013706 055370
13394 055000 005037 055370
13395 055004 005237 055370
13396 055010 001375
13397
13398 055012
13399
13400 055016 022737 000001 003754
13401 055024 001006
13402 055026
13403 055036 004737 046356
13404
13405 055042 012700 172340
13406 055046 012702 172300
13407 055052 012701 000020
13408 055056
13409 055060 012722 077406
13410 055064 077104
13411
13412 055066 005737 002454
13413 055072 001002
13414 055074
13415 055100
13416
13417 055114 013701 002222
13418 055120 042701 177400
13419 055124
13420 055124
13421 055132 006201
13422 055134
13423 055136
13424 055142 104425
13425 055144
13426 055144
13427 055150
13428 055166
13429
13430 055166 012700 172300
13431 055172 012701 177600
13432 055176 012702 172200
13433 055202 012703 000040
13434 055206 011021
13435 055210 012022
13436 055212 077303

;*****
;POWER UP ROUTINE
$PWRUP:
MOV    #SILLUP,PWRVEC ;;SET FOR FAST DOWN
;RESTORE STACK POINTER
MOV    $$SAVR6,SP      ;;GET SP
CLR    $$SAVR6         ;;WAIT LOOP FOR THE TTY
1$:    INC    $$SAVR6   ;;WAIT FOR THE INC
        BNE    1$      ;;OF A WORD
;RESTORE POSSIBLE SOFTWARE SWITCH REGISTER
POP    @SWR
;RESTORE UNIBUS MAP
CMP    #1,PROTYP       ;IS THIS AN 11/44?
BNE    10$
POP    MAPLO,MAPHO
CALL   LOWMAP         ;SETUP LOWER 16K OF UNIBUS MAP
;RESTORE KERNEL PAR'S & PDR'S
10$:   MOV    #172340,R0
        MOV    #KIPDR0,R2
        MOV    #16.,R1
6$:    POP    (R0)+
        MOV    #77406,(R2)+
        SOB   R1,6$
;RESTORE MMR3,2,1,0
TST    NOSUPER
BNE    11$
POP    MMR3
11$:   POP    MMR2,MMR1,MMR0
;RESTORE ECC REGISTERS
MOV    TOTCSRS,R1     ;GET CSR'S
BIC    #177400,R1
BEGIN LCSRRESTORE
      FOR CSRNO := #36 DOWNT0 #0 BY #2
      ASR    R1
      ON.ERROR
      POP    CSR
      LOADCSR
      END ;OF ON.ERROR
      IF R1 EQ #0 THEN LEAVE LCSRRESTORE
      END ;OF FOR CSRNO
END LCSRRESTORE
;COPY KERNEL MAP TO USER & SUPERVISOR
MOV    #KIPDR0,R0
MOV    #UIPDR0,R1
MOV    #SIPDR0,R2
3$:    MOV    #32.,R3
        MOV    (R0),(R1)+
        MOV    (R0)+,(R2)+
        SOB   R3,3$

```


CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 430
ROUTINE POWER DOWN AND UP

13485 055372

WOOPS: SUBTST <<POWER FAIL WHILE RELOCATED>>

:SUBTEST POWER FAIL WHILE RELOCATED

13486	055372			
13487	055376	005037	002100	
13488	055402			
13489	055416			
13490	055424	013737	060024	055770
13491	055432	013737	060026	055772
13492	055440			
13493	055452	012737	055556	060024
13494	055460	012737	000340	060026
13495	055466			
13496	055500	012700	172340	
13497	055504	012701	135740	
13498	055510	012702	000010	
13499	055514	012021		
13500	055516	077202		
13501	055520	005737	002454	
13502	055524	001002		
13503	055526	013721	172516	
13504	055532	013721	177576	
13505	055536	013721	177574	
13506	055542	013721	177572	
13507	055546	104417		
13508	055550			
13509	055554	000207		

```

PUSH BANK
CLR BANK
MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
SUPERVISOR ;ENTER SUPERVISOR MODE
MOV FIRST+PWRVEC,WOOPSAV
MOV FIRST+PWRVEC+2,WOOPSAV+2
BMOV FIRST+WOOPUP,WOOPSAV+4,WOOPEND-WOOPUP/2+12.
MOV #WOOPUP,FIRST+PWRVEC
MOV #340,FIRST+PWRVEC+2
BMOV WOOPUP,FIRST+WOOPUP,WOOPEND-WOOPUP/2
MOV #KIPAR0,R0
MOV #FIRST+WOOPEND,R1
MOV #8.,R2
1$: MOV (R0)+,(R1)+
SOB R2,1$
TST NOSUPER
BNE 2$
MOV MMR3,(R1)+
2$: MOV MMR2,(R1)+
MOV MMR1,(R1)+
MOV MMRO,(R1)+
KERNEL ;ENTER KERNEL MODE
POP BANK
RETURN

```

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 432
POWER FAIL WHILE RELOCATED

13512 055556

WOOPUP: SUBTST <<POWER UP FROM BANK 0 TO RELOCATION>>

: *SUBTEST POWER UP FROM BANK 0 TO RELOCATION

13513 055556 012700 055740
13514 055562 012701 172340
13515 055566 012703 172300
13516 055572 012702 000010
13517 055576 012021
13518 055600 012723 077406
13519 055604 077204
13520 055606 005737 002454
13521 055612 001002
13522 055614 012037 172516
13523 055620 012037 177576
13524 055624 012037 177574
13525 055630 012037 177572
13526 055634 013706 055370
13527 055640
13528 055644 005037 002100
13529 055650
13530 055664
13531 055672 013737 055770 060024
13532 055700 013737 055772 060026
13533
13534
13535 055706 012700 055774
13536 055712 012701 000105
13537 055716 012702 135556
13538 055722 012022
13539 055724 077102
13540
13541 055726 104417
13542 055730
13543 055734 000137 054766
13544 055740 000014
13547 055770 000107

MOV #WOOPEND,R0
MOV #KIPAR0,R1
MOV #KIPDR0,R3
MOV #8.,R2
1\$: MOV (R0)+,(R1)+
MOV #77406,(R3)+
SOB R2,1\$
TST NOSUPER
BNE 3\$
3\$: MOV (R0)+,MMR3
MOV (R0)+,MMR2
MOV (R0)+,MMR1
MOV (R0)+,MMR0
MOV \$\$SAVR6,SP
PUSH BANK
CLR BANK
MAP BANK
SUPERVISOR ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
;ENTER SUPERVISOR MODE
MOV WOOPSAV,FIRST+PWRVEC
MOV WOOPSAV+2,FIRST+PWRVEC+2
;SIMULATE THE FOLLOWING BLOCK MOV BUT WITH NO STACK ACCESSES
;BMOV WOOPSAV+4,FIRST+WOOPUP,WOOPEND-WOOPUP/2+12.
MOV #WOOPSAV+4,R0
MOV #WOOPEND-WOOPUP/2+12.,R1
MOV #FIRST+WOOPUP,R2
2\$: MOV (R0)+,(R2)+
SOB R1,2\$
KERNEL ;ENTER KERNEL MODE
POP BANK
JMP \$PWRUP
WOOPEND: .REPT 12.
WOOPSAV: .REPT WOOPEND-WOOPUP/2+12.+2


```

13552
13553
13554
13555
13556
13557
13558
13559
13560
13561
13562
13563
13564
13565
13566
13567
13568
13569
13570
13571 056206 105737 002356
13572 056212 100407
13573 056214 010046
13574 056216 017600 000002
13575 056222 112046
13576 056224 001005
13577 056226 005726
13578 056230 012600
13579 056232 062716 000002
13580 056236 000002
13581 056240 122716 000011
13582 056244 001002
13583 056246 112716 000040
13584 056252 122716 000200
13585 056256 001006
13586 056260 005726
13587 056262
13588 056264 002646
13589 056266 105037 056520
13590 056272 000753
13591 056274 004737 056334
13592 056300 123726 002640
13593 056304 001346
13594 056306 013746 002354
13595
13596 056312 105366 000001
13597 056316 002770
13598 056320 004737 056334
13599 056324 105337 056520
13600 056330 000770
13601 056332 000000
13602 056334
13603 056336 116601 000004
13604 056342 005737 002542
13605 056346 001402
13606 056350
13607 056354
13608 056356 104424

```

.SBTTL IO SUBROUTINES

.SBTTL ROUTINE TYPE

```

*****
;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
;*NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
;*NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
;*NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.

```

```

;*
;*CALL:
;*1) USING A TRAP INSTRUCTION
;*      TYPE      MESADR      ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
;*OR
;*      TYPE
;*      MESADR
;*

```

```

$TYPE:  TSTB      $TPFLG      ;;IS THERE A TERMINAL?
        BMI       6$          ;;BR IF NO
1$:      MOV      RO,-(SP)     ;;SAVE RO
        MOV      @2(SP),RO    ;;GET ADDRESS OF ASCIZ STRING
4$:      MOVB     (RO)+,-(SP)  ;;PUSH CHARACTER TO BE TYPED ONTO STACK
        BNE      7$          ;;BR IF IT ISN'T THE TERMINATOR
        TST      (SP)+       ;;IF TERMINATOR POP IT OFF THE STACK
5$:      MOV      (SP)+,RO    ;;RESTORE RO
6$:      ADD      #2,(SP)     ;;ADJUST RETURN PC
        RTI
7$:      CMPB     #HT,(SP)    ;;BRANCH IF NOT <HT>
        BNE      11$
        MOVB     #' ,(SP)    ;;REPLACE TAB WITH SPACE
11$:     CMPB     #CRLF,(SP)  ;;BRANCH IF NOT <CRLF>
        BNE      8$
        TST      (SP)+       ;;POP <CR><LF> EQUIV
        TYPE     ;;TYPE A CR AND LF
        CLR      $CHARCNT    ;;CLEAR CHARACTER COUNT
        BR       4$          ;;GET NEXT CHARACTER
8$:      CALL     $TYPEPC     ;;GO TYPE THIS CHARACTER
9$:      CMPB     $FILLC,(SP)+ ;;IS IT TIME FOR FILLER CHARS.?
        BNE      4$          ;;IF NO GO GET NEXT CHAR.
        MOV      $NULL,-(SP)  ;;GET # OF FILLER CHARS. NEEDED
        AND      THE NULL CHAR.
10$:     DECB     1(SP)       ;;DOES A NULL NEED TO BE TYPED?
        BLT      9$          ;;BR IF NO--GO POP THE NULL OFF OF STACK
        CALL     $TYPEPC     ;;GO TYPE A NULL
        DECB     $CHARCNT    ;;DO NOT COUNT AS A COUNT
        BR       10$        ;;LOOP
XOCHAR: .WORD 0
$TYPEPC: PUSH     R1
        MOVB     4(SP),R1
        TST      CACHKN
        BEQ      2$
        PUSH     CONTRL
2$:      PUSH     RO
        CACHOFF      ;;TURN CACHE OFF

```

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 434-1
ROUTINE TYPE

```

13633 056360 105777 124250      3$:   TSTB  @STPS      ;;WAIT UNTIL PRINTER IS READY
13634 056364 100375                BPL   3$
13635 056366 005037 056332      CLR   XOCHAR
13636 056372 105777 124232      TSTB  @STKS      ;;CHECK FOR XOFF
13637 056376 100032                BPL   NC          ;;SKIP IF NO CHARACTER
13638 056400 117737 124226 056332      MOVB  @STKB,XOCHAR ;;SAVE THE CHARACTER
13639 056406 042737 177600 056332      BIC   #'C177,XOCHAR ;;STRIP OFF ASCII
13640 056414 023727 056332 000023      CMP   XOCHAR,#023 ;;WAS IT A CONTROL S?
13641 056422 001020                BNE   NC          ;;BRANCH IF NOT
13642 056424 105777 124200      CONTS3: TSTB @STKS    ;;WAIT FOR CHARACTER
13643 056430 100375                BPL   CONTS3
13644 056432 117737 124174 056332      MOVB  @STKB,XOCHAR ;;GET CHARACTER
13645 056440 042737 177600 056332      BIC   #'C177,XOCHAR ;;STRIP OFF ASCII
13646 056446                IF XOCHAR EQ #21  ;; IF IT IS A ^Q
13647 056456 000402                BR    NC
13648 056460                ELSE
13649 056462 000760                BR    CONTS3
13650 056464                END ;OF IF XOCHAR
13651 056464 110177 124146      NC:   MOVB  R1,@STPB  ;;LOAD CHAR TO BE TYPED INTO DATA REG.
13655 056470 122766 000015 000002      CMPB  #CR,2(SP)    ;;IS CHARACTER A CARRIAGE RETURN?
13656 056476 001003                BNE   1$          ;;BRANCH IF NO
13657 056500 105037 056520      CLRB  $CHARCNT    ;;YES--CLEAR CHARACTER COUNT
13658 056504 000406                BR    $TYPEX     ;;EXIT
13659 056506 122766 000012 000002      1$:   CMPB  #LF,2(SP)  ;;IS CHARACTER A LINE FEED?
13660 056514 001402                BEQ   $TYPEX     ;;BRANCH IF YES
13661 056516 105227                INCB  (PC)+      ;;COUNT THE CHARACTER
13662 056520 000000      $CHARCNT: .WORD  0  ;;CHARACTER COUNT STORAGE
13663 056522                $TYPEX: POP   R0
13664 056524 005737 002542      TST   CACHKN     ;IS THERE A CACHE?
13665 056530 001402                BEQ   2$        ;BRANCH IF NOT
13666 056532                POP   CONTRL   ;POP CACHE STATUS
13667 056536                2$:   POP   R1
13668 056540 000207                RETURN
13669 056542                SUPLIMIT:;!!!!!!!!!!!!!!!!THIS IS THE LIMIT ON SUPERVISOR MAPPED TO MUT SPACE

```

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 449
 ERROR DATA SETUP

```

14347          .SBTTL  ERROR DATA SETUP
14348
14349          USE THIS  IF THIS CONDITION DISCRIBES THE ERROR
14350
14351          PERR01     TRAP
14352                   BAD DATA IN R0 UNLESS ABORTED
14353                   THEN BAD DATA IS POINTED TO BY -(R4)
14354                   GOOD DATA IN R5
14355
14356          PERR02     TRAP
14357                   BAD DATA IN R1 UNLESS ABORTED
14358                   THEN BAD DATA IS POINTED TO BY -(R4)
14359                   GOOD DATA IN R2
14360
14361          PERR03     TRAP
14362                   BAD DATA IS POINTED TO BY -(R1)
14363                   GOOD DATA IN R4
14364
14365          PERR04     TRAP
14366                   BAD DATA IN R4 UNLESS ABORTED
14367                   THEN BAD DATA IS POINTED TO BY -2(R0)
14368                   GOOD DATA IN R2
14369
14370          PERR05     JSR    PC
14371                   BAD DATA IS POINTED TO BY -(R0)
14372                   GOOD DATA IN R2
14373                   RETURN AFTER SETTING UP GOOD,BAD,ADDRESS
14374
14375          PERR06     JSR    PC
14376                   BAD DATA IS POINTED TO BY -(R0)
14377                   GOOD DATA IS ZERO
14378                   RETURN AFTER SETTING UP GOOD,BAD,ADDRESS
14379
14380          PERR07     TRAP
14381                   BAD DATA IN R2 UNLESS ABORTED
14382                   THEN BAD DATA IS POINTED TO BY (R1)
14383                   GOOD DATA IN DATBUF
14384
14385          PERR10     TRAP
14386                   BAD DATA IN R2 UNLESS ABORTED
14387                   THEN BAD DATA IS POINTED TO BY 2(R1)
14388                   GOOD DATA IN DATBUF+2
14389
14390          PERR11     TRAP
14391                   BYTE TEST
14392                   BAD DATA IN RIGHT BYTE OF R0 UNLESS ABORTED
14393                   THEN BAD DATA IS POINTED TO BY (R1)
14394                   GOOD DATA IS A ZERO BYTE
14395
14396          PERR12     TRAP
14397                   BYTE TEST
14398                   BAD DATA IN RIGHT BYTE OF R0 UNLESS ABORTED
14399                   THEN BAD DATA IS POINTED TO BY (R1)
14400                   GOOD DATA IS A BYTE OF ONES
14401
14402          PERR13     TRAP
14403                   BAD DATA IN R0 UNLESS ABORTED

```


CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 449-1
 ERROR DATA SETUP

14404	:		THEN BAD DATA IS POINTED TO BY (R1)
14405	:		GOOD DATA IS ZERO
14406	:		
14407	:	PERR14	TRAP
14408	:		BAD DATA IN R0 UNLESS ABORTED
14409	:		THEN BAD DATA IS POINTED TO BY (R1)
14410	:		GOOD DATA IS ONES
14411	:		
14412	:	PERR15	TRAP
14413	:		BAD DATA IN R0 UNLESS ABORTED
14414	:		THEN BAD DATA IS POINTED TO BY (R1)
14415	:		GOOD DATA IN TSTDAT
14416	:		
14417	:	PERR16	TRAP
14418	:		BAD DATA IN R0 UNLESS ABORTED
14419	:		THEN BAD DATA IS POINTED TO BY (R1)
14420	:		GOOD DATA IN TSTDAT+2
14421	:		
14422	:	PERR17	TRAP
14423	:		BAD DATA IN R0 UNLESS ABORTED
14424	:		THEN BAD DATA IS POINTED TO BY (R1)
14425	:		GOOD DATA IN R2
14426	:		
14427	:	PERR20	TRAP
14428	:		BAD DATA IN R0 UNLESS ABORTED
14429	:		THEN BAD DATA IS POINTED TO BY (R1)
14430	:		GOOD DATA IN R3
14431	:		
14432	:	PERR21	TRAP
14433	:		7 BIT BYTE TEST
14434	:		BAD DATA IN RIGHT BYTE OF R0 UNLESS ABORTED
14435	:		THEN BAD DATA IS POINTED TO BY (R1)
14436	:		GOOD DATA IS A 7 BIT BYTE ON ONES
14437	:		
14438	:	PERR22	TRAP
14439	:		BAD DATA IN R2 UNLESS ABORTED
14440	:		THEN BAD DATA IS POINTED TO BY (R1)
14441	:		GOOD DATA IN R0
14442	:		
14443	:	PERR23	TRAP
14444	:		BAD DATA IN R0 UNLESS ABORTED
14445	:		THEN BAD DATA IS POINTED TO BY (R1)
14446	:		GOOD DATA IN R4
14447	:		
14448	:	PERR24	TRAP
14449	:		BAD DATA IN R0 UNLESS ABORTED
14450	:		THEN BAD DATA IS POINTED TO BY (R2)
14451	:		GOOD DATA IN R3
14452	:		
14453	:	PERR25	TRAP
14454	:		BAD DATA POINTED TO BY -(R1)
14455	:		GOOD DATA IN R2 UNLESS LOC V177654 IS SET
14456	:		THEN GOOD DATA IS IN R3
14457	:		
14458	:	PERR26	TRAP
14459	:		BAD DATA IS DOUBLE WORD POINTED TO BY R1 AND IN LOW 7 BITS OF R0
14460	:		GOOD DATA IS 000000,,100000,,100

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 449-2
 ERROR DATA SETUP

14461	:			
14462	:	PERR27	TRAP	
14463	:		BAD DATA IS DOUBLE WORD POINTED TO BY R1 AND IN LOW 7 BITS OF R0	
14464	:		GOOD DATA IS 000000,,000000,,077	
14465	:			
14466	:	PERR30	TRAP	
14467	:		BAD DATA IS POINTED TO BY -16(SP)	
14468	:		GOOD DATA IS IN R1	
14469	:			
14470	:	PERR31	TRAP	
14471	:		SPECIAL ECC FAILURE HANDLER	
14472	:			
14473	:	PERR32	TRAP	
14474	:		SPECIAL ECC FAILURE HANDLER	
14475	:			
14476	:	PERR33	TRAP	
14477	:		SPECIAL ECC FAILURE HANDLER	
14478	:			
14479	:	PERR34	TRAP	
14480	:		SPECIAL ECC FAILURE HANDLER	
14481	:			
14482	:	PERR35	TRAP	
14483	:		SPECIAL BRANCH GOBBLE FAILURE HANDLER	
14484	:			
14485	:		CALLING SEQUENCE FOR TRAP TYPES	
14486	:	BEQ	2\$:NO - ERROR,BRANCH FOR CARD
14487	:	PERRXX		:TRAP TO ERROR ROUTINE
14488	:	2\$: NEXT	INSTRUCTION	:CONTINUE TESTING

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 451
 ERROR DATA SETUP

14491	056542	010437	002032		\$PER01: MOV R4,ADDRESS	
14492	056546	162737	000002	002032	SUB #2,ADDRESS	
14493	056554	010037	002050		MOV R0,BAD	
14494	056560	010537	002042		MOV R5,GOOD	
14495	056564	000137	057276		JMP PERRAW	
14496						
14497	056570	010437	002032		\$PER02: MOV R4,ADDRESS	
14498	056574	162737	000002	002032	SUB #2,ADDRESS	
14499	056602	010137	002050		MOV R1,BAD	
14500	056606	010237	002042		MOV R2,GOOD	
14501	056612	000137	057276		JMP PERRAW	
14502						
14503	056616	010137	002032		\$PER03: MOV R1,ADDRESS	
14504	056622	162737	000002	002032	SUB #2,ADDRESS	
14505	056630	010437	002042		MOV R4,GOOD	
14506	056634	016137	177776	002050	MOV -2(R1),BAD	
14507	056642	000137	057276		JMP PERRAW	
14508						
14509	056646	010037	002032		\$PER04: MOV R0,ADDRESS	
14510	056652	162737	000002	002032	SUB #2,ADDRESS	
14511	056660	010437	002050		MOV R4,BAD	
14512	056664	010237	002042		MOV R2,GOOD	
14513	056670	000137	057276		JMP PERRAW	
14514						
14515	056674	010237	002042		PERR05: MOV R2,GOOD	
14516	056700	014037	002050		PERA05: MOV -(R0),BAD	
14517	056704	010037	002032		MOV R0,ADDRESS	
14518	056710	062700	000002		ADD #2,R0	:RESTORE R0
14519	056714	004737	042626		CALL BADSTACK	
14520	056720	000207			RETURN	
14521						
14522	056722	005037	002042		PERR06: CLR GOOD	
14523	056726	000764			BR PERA05	
14524						
14525	056730	010137	002032		\$PER07: MOV R1,ADDRESS	
14526	056734	010237	002050		MOV R2,BAD	
14527	056740	013737	002240	002042	MOV DATBUF,GOOD	
14528	056746	000137	057276		JMP PERRAW	
14529						
14530	056752				\$PER10: LET ADDRESS := R1 + #2	
14531	056764				LET BAD := R2	
14532	056770				LET GOOD := DATBUF+2	
14533	056776	000137	057276		JMP PERRAW	
14534						
14535	057002				\$PER11: LET ADDRESS := R1	
14536	057006				LET BAD := R0	
14537	057012				LET GOOD := #0	
14538	057016	000137	057350		JMP PERRAB	
14539						
14540	057022				\$PER12: LET ADDRESS := R1	
14541	057026				LET BAD := R0	
14542	057032				LET GOOD := #377	
14543	057040	000137	057350		JMP PERRAB	

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 453
 ERROR DATA SETUP

14546	057044		\$PER13: LET ADDRESS := R1
14547	057050		LET BAD := R0
14548	057054		LET GOOD := #0
14549	057060	000137 057276	JMP PERRAW
14550			
14551	057064		\$PER14: LET ADDRESS := R1
14552	057070		LET BAD := R0
14553	057074		LET GOOD := ONES
14554	057102	000137 057276	JMP PERRAW
14555			
14556	057106		\$PER15: LET ADDRESS := R1
14557	057112		LET BAD := R0
14558	057116		LET GOOD := TSTDAT
14559	057124	000137 057276	JMP PERRAW
14560			
14561	057130		\$PER16: LET ADDRESS := R1
14562	057134		LET BAD := R0
14563	057140		LET GOOD := TSTDAT+2
14564	057146	000453	BR PERRAW
14565			
14566	057150		\$PER17: LET ADDRESS := R1
14567	057154		LET BAD := R0
14568	057160		LET GOOD := R2
14569	057164	000444	BR PERRAW
14570			
14571	057166		\$PER20: LET ADDRESS := R1
14572	057172		LET BAD := R0
14573	057176		LET GOOD := R3
14574	057202	000435	BR PERRAW
14575			
14576	057204		\$PER21: LET ADDRESS := R1
14577	057210		LET BAD := R0
14578	057214		LET GOOD := #177
14579	057222	000477	BR PERRA7
14580			
14581	057224		\$PER22: LET ADDRESS := R1
14582	057230		LET BAD := R2
14583	057234		LET GOOD := R0
14584	057240	000416	BR PERRAW
14585			
14586	057242		\$PER23: LET ADDRESS := R1
14587	057246		LET BAD := R0
14588	057252		LET GOOD := R4
14589	057256	000407	BR PERRAW
14590			
14591	057260		\$PER24: LET ADDRESS := R2
14592	057264		LET BAD := R0
14593	057270		LET GOOD := R3
14594	057274	000400	BR PERRAW

14596 057276

PERRAW: SUBTST <<DATA WAS A WORD>>
:*****
:*SUBTEST DATA WAS A WORD
:*****

14597 057276 004737 057532

14598 057302

14599 057314

14600 057326 004737 057506

14601 057332

14602 057340 104011

14603 057342

14604 057344 104012

14605 057346

14606 057346 000002

14607

14608 057350

CALL PERBNK
IF ABORTFLAG IS TRUE THEN \$CALL GETDATA
IF BADPC EQ #0 THEN \$CALL BADSTACK
CALL PERXOR
IF ABORTFLAG IS FALSE
ERROR +11
ELSE
ERROR +12
END ;OF IF ABORTFLAG
RTI

PERRAB: SUBTST <<DATA WAS A BYTE>>
:*****
:*SUBTEST DATA WAS A BYTE
:*****

14609 057350 004737 057532

14610 057354

14611 057366

14612 057400 004737 057506

14613 057404

14614 057412 104014

14615 057414

14616 057416 104015

14617 057420

14618 057420 000002

CALL PERBNK
IF ABORTFLAG IS TRUE THEN \$CALL GETDATA
IF BADPC EQ #0 THEN \$CALL BADSTACK
CALL PERXOR
IF ABORTFLAG IS FALSE
ERROR +14
ELSE
ERROR +15
END ;OF IF ABORTFLAG
RTI

14621 057422

PERRA7: SUBTST <<DATA WAS A 7 BIT BYTE>>
:*****
:*SUBTEST DATA WAS A 7 BIT BYTE
:*****

14622 057422
14623 057434 004737 057506
14624 057440 004737 057532
14625 057444 104022
14626 057446 000002

IF BADPC EQ #0 THEN \$CALL BADSTACK
CALL PERXOR
CALL PERBNK
ERROR +22
RTI

14627
14628 057450
14629 057456
14630 057464 000137 054100

\$PER26: LET GOOD2 := #100000
LET GOOD3 := #100
JMP PERRA3

14631
14632 057470 005037 002044
14633 057474
14634 057502 000137 054100

\$PER27: CLR GOOD2
LET GOOD3 := #077
JMP PERRA3

14635
14636 057506

PERXOR: SUBTST <<DETERMINE XOR OF GOOD & BAD>>
:*****
:*SUBTEST DETERMINE XOR OF GOOD & BAD
:*****

14637 057506
14638 057510 013700 002042
14639 057514 013737 002050 002056
14640 057522 074037 002056
14641 057526
14642 057530 000207

PUSH R0
MOV GOOD,R0
MOV BAD,BAD XOR
XOR R0,BAD XOR
POP R0
RETURN

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 458
DETERMINE XOR OF GOOD & BAD

14645 057532

PERBNK: SUBTST<<LOG ERROR ON BAD BANK>>

:SUBTEST LOG ERROR ON BAD BANK

14646

:WHILE WE'RE HERE LET'S MARK THE BAD BANK IN THE CONFIGURATION TABLE

14647 057532

PUSH R0,R1

14648 057536 013701 002100

MOV BANK,R1

14649 057542 006301

ASL R1

14650 057544 006301

ASL R1

14651 057546 052761 000001 002652

BIS #BIT0,CONFIG(R1)

14652 057554 105261 002654

INCB CONFIG+2(R1)

14653 057560 001002

BNE 12\$

:BUMP BANK COUNTER

14654 057562 105361 002654

DECB CONFIG+2(R1)

:NO OVERFLOW - SKIP

14655 057566 126137 002654 002552 12\$:

CMPB CONFIG+2(R1),ERRMAX

:SET BACK TO 255.

14656 057574 101403

BLOS 11\$

:IS IT PAST MAX?

14657 057576

SET TOOMANY

:NO - SKIP

14658 057604

11\$:

POP R1,R0

:YES

14659 057610 000207

RETURN

14660

14661 057612 010037 002050

PERECC: MOV R0,BAD

14662 057616

IF ADDRESS EQ TESTADD

14663 057626 013737 002244 002042

MOV TSTDAT,GOOD

14664 057634

ELSE

14665 057636 013737 002246 002042

MOV TSTDAT+2,GOOD

14666 057644

END :OF IF (R1)

14667 057644 004737 057506

CALL PERXOR

14668 057650

SET HEADER

14669 057656 000207

RETURN

14670

14671 057660

\$PER31: IF REALPAT EQ #41

14672 057670 104023

ERROR +23

14673 057672

END

14674 057672 004737 057612

IF BADPC EQ #0 THEN \$CALL BADSTACK

14675 057704

CALL PERECC

14676 057710

IF REALPAT EQ #11

14677 057720 104037

ERROR +37

14678 057722

END :OF IF REALPAT

14679 057722

IF REALPAT EQ #15

14680 057732 104043

ERROR +43

14681 057734

END :OF IF REALPAT

14682 057734

IF REALPAT EQ #16

14683 057744 104044

ERROR +44

14684 057746

END :OF IF REALPAT

14685 057746

SET HEADER

14686 057754 000002

RTI

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 460
LOG ERROR ON BAD BANK

14689	057756				\$PER32: IF BADPC EQ #0 THEN \$CALL BADSTACK	
14690	057770	010137	002032		MOV R1,ADDRESS	
14691	057774	010037	002050		MOV R0,BAD	
14692	060000	010237	002042		MOV R2,GOOD	
14693	060004				SET HEADER	
14694	060012	104040			ERROR +40	
14695	060014				SET HEADER	
14696	060022	000002			RTI	
14697						
14698	060024				\$PER33: IF BADPC EQ #0 THEN \$CALL BADSTACK	
14699	060036	010137	002032		MOV R1,ADDRESS	
14700	060042	010037	002050		MOV R0,BAD	
14701	060046	105037	002051		CLRB BAD+1	
14702	060052	012737	000377	002042	MOV #377,GOOD	
14703	060060	004737	057506		CALL PERXOR	
14704	060064				SET HEADER	
14705	060072	104041			ERROR +41	
14706	060074				SET HEADER	
14707	060102	000002			RTI	
14708						
14709	060104				\$PER34: IF BADPC EQ #0 THEN \$CALL BADSTACK	
14710	060116				IF #BIT15!BIT4 OFF.IN CSR	
14711	060126	104016			ERROR +16	:NO SBE OR DBE
14712	060130				ELSE	
14713	060132	104001			ERROR +1	:EXPECTED SBE SO DBE MUST HAVE GOTTEN SET
14714	060134				END ;OF IF #BIT15!BIT4	
14715	060134	000002			RTI	
14716						
14717					:DURING BRANCH GOBBLE THE CONDITION CODES WERE WRONG	
14718	060136	004737	057532		\$PER35: CALL PERBNK	
14719	060142	004737	042626		CALL BADSTACK	
14720	060146	013737	002030	002050	MOV BADPSW,BAD	
14721	060154	012737	000012	002042	MOV #12,GOOD	
14722	060162	104047			ERROR +47	
14723	060164	062706	000004		ADD #4,SP	:FIX STACK FROM TRAP
14724	060170	000207			RETURN	:ABORTING TEST
14725						
14726	060172	010037	002042		\$PER36: MOV R0,GOOD	
14727	060176	010137	002050		MOV R1,BAD	
14728	060202				SET HEADER	
14729	060210	104023			ERROR +23	
14730	060212				SET HEADER	
14731	060220	000002			RTI	
14732						
14733	060222	104053			\$PER37: ERROR +53	:ILC::REV B
14734	060224	000002			RTI	:ILC::REV B
14735						
14736	060226	104054			\$PER40: ERROR +54	:ILC::REV B
14737	060230	000002			RTI	:ILC::REV B

```

14740
14741
14742
14743
14744
14745
14746
14747
14748
14749 060232 005237 065714
14750 060236
14751 060240 005037 065714
14752 060244 105237 065716
14753 060250
14754 060250 104410
14755 060252 005737 006206
14756 060256 001402
14757 060260 004737 064210
14758 060264
14759 060264 005737 061366
14760 060270 001410
14761 060272 013737 177766 061364
14762 060300 032737 000001 061364
14763 060306 001401
14764 060310 104177
14773 060312
14774 060330 005037 002416
14775 060334 000137 047644
14776 060340
14777 060340
14778 060346 000002
14779 060350
14780 060350
14781
14782 060360 000425
14783
14784 060362 013746 000004
14785 060366 012737 060406 000004
14786 060374 005737 177060
14787 060400 012637 000004
14788 060404 000430
14789 060406 062706 000004
14790 060412 022737 000001 003754
14791 060420 001002
14792 060422 005037 177766
14793 060426 012637 000004
14794 060432 000407
14795 060434
14796 060434 105737 002012
14797 060440 001412
14798 060442 032777 001000 122154
14799 060450 001404
14800 060452 013737 002612 002610
14801 060460 000410
14802 060462 105037 002012
14803 060466 011637 002610
14804 060472 011637 002612

```

```

.SBTTL ROUTINE SCOPE HANDLER
;*****
;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
;*AND LOAD THE DISPLAY DATA INTO THE DISPLAY REGISTER
;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
;*SW14=1 LOOP ON TEST
;*SW9=1 LOOP ON ERROR
;*CALL
;*
$SCOPE: SCOPE      ;;SCOPE=IOT
        INC      $DEVCT      ;TELL APT WE ARE ALIVE
        IF RESULT IS LT
        CLR      $DEVCT
        INCB     $UNIT
        END ;OF IF RESULT
        CKSWR      ;;TEST FOR CHANGE IN SOFT-SWR
        TST      TRACE
        BEQ      NOTRCE
        CALL     CONTT      ;TRACE
NOTRCE: TST      CPERRF      ;IS THERE A CPU ERROR REGISTER?
        BEQ      SKJ        ;BRANCH IF NOT
        MOV      @#177766,CPSAVE ;GET CONTENTS OF ERROR REGISTER
        BIT      #BIT0,CPSAVE  ;IS THE POWER FAIL MONITOR BIT SET?
        BEQ      SKJ        ;BRANCH IF NOT
        ERROR   +177        ;REPORT IF SO
SKJ:    IF STOPOK IS TRUE AND #SW8 SET.IN @SWR
        CLR      STOPOK
        JMP      EXIT
        END ;OF IF STOPOK
        IF NOSCOPE IS TRUE
        RTI
        END ;OF IF NOSCOPE
1$:     IF #SW14 SET.IN @SWR THEN GOTO $OVER
;*****START OF CODE FOR THE XOR TESTER*****
$XTSTR: BR      2$
        MOV      ERRVEC,-(SP)  ;:IF RUNNING ON THE 'XOR' TESTER CHANGE
        MOV      #1$,ERRVEC   ;:THIS INSTRUCTION TO A 'NOP' (NOP=240)
        TST      177060      ;:SAVE THE CONTENTS OF THE ERROR VECTOR
        MOV      (SP)+,ERRVEC ;:SET FOR TIMEOUT
        BR      $SVLAD       ;:RESTORE THE ERROR VECTOR
        ADD     #4,SP        ;:GO TO THE NEXT TEST
        CMP     #1,PROTOP    ;:FIX STACK FROM TRAP
        BNE     6$          ;:IS THIS AN 11/44?
        CLR     CPUERR      ;:BRANCH IF NOT
        MOV     (SP)+,ERRVEC ;:RESET CPU ERROR REGISTER
        BR     4$          ;:RESTORE THE ERROR VECTOR
2$:     ;*****END OF CODE FOR THE XOR TESTER*****
3$:     TSTB     $ERFLG      ;:LOOP ON THE PRESENT TEST
        BEQ     $SVLAD      ;:HAS AN ERROR OCCURRED?
        BIT     #SW9,@SWR   ;:BR IF NO
        BEQ     5$          ;:LOOP ON ERROR?
4$:     MOV     $LPERR,$LPADR ;:BR IF NO
        BR     $OVER        ;:SET LOOP ADDRESS TO LAST SCOPE
5$:     CLRB     $ERFLG     ;:ZERO THE ERROR FLAG
$SVLAD: MOV     (SP),$LPADR  ;:SAVE SCOPE LOOP ADDRESS
        MOV     (SP),$LPERR ;:SAVE ERROR LOOP ADDRESS

```

```

:R-C
:R-C
:R-C
:R-C
:R-C
:R-C
:R-C

```


CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 462-1
ROUTINE SCOPE HANDLER

14805	060476	005037	002360
14806	060502	004737	060514
14807	060506	013716	002610
14808	060512	000002	

\$OVER:	CLR	\$ESCAPE	::CLEAR THE ESCAPE FROM ERROR ADDRESS
	CALL	GETDIS	
	MOV	\$LPADR,(SP)	::FUDGE RETURN ADDRESS
	RTI		::FIXES PS

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 463
ROUTINE SCOPE HANDLER

14810 060514

GETDIS: SUBTST <<SUBR DISPLAY>>

: *SUBTEST SUBR DISPLAY

14811 060514 113737 002100 002011
14812 060522 113737 002276 002010
14813 060530
14814 060532 005737 002124
14815 060536 001403
14816 060540 052737 100000 002010
14817 060546
14821 060546 013777 002010 122052
14822 060554 013737 002010 000174
14823 060562
14824 060564 000207

1\$:
MOV BANK,\$BANK
MOV REALPAT,\$PATMAR
PUSH R0
TST RLFLAG ;ARE WE RELOCATED?
BEQ 1\$;NO - SKIP
BIS #BIT15,\$PATMAR ;YES - SET MSB
MOV \$PATMAR,@DISPLAY
MOV \$PATMAR,DISPREG ;SOFTWARE DISPLAY REGISTER
POP R0
RETURN

```

14827
14828
14829
14830
14831
14832
14833
14834
14835
14836
14837
14838
14839
14840
14841 060566 105037 061362
14842 060572
14843 060600 104410
14844 060602
14845 060602 105237 002012
14846 060606 001775
14847 060610 004737 060514
14848 060614 013737 002010 065710
14849 060622 032777 002000 121774
14850 060630 001404
14851 060632
14852 060636
14853 060642 005237 002616
14854 060646
14855 060650 012737 077777 002616
14856 060656
14857 060656
14858 060656 011637 002016
14859 060662 162737 000002 002016
14860 060670 010637 002022
14861 060674 016637 000002 002026
14862 060702 117737 121110 002013
14863
14864 060710 122737 000177 002013
14865 060716 001431
14866 060720 105737 061362
14867 060724 001024
14868 060726 005737 061366
14869 060732 001423
14870 060734 013737 177766 061364
14871 060742 032737 000001 061364
14872 060750 001414
14873 060752 042737 000001 177766
14874 060760 112737 002013 061362
14875 060766 112737 000177 002013
14876 060774 000402
14877 060776 105037 061362
14878 061002
14879 061002
14880 061010
14881 061016 013737 002020 002016
14882 061024 162737 000002 002016
14883 061032 013737 002024 002022

```

```

.SBTTL ROUTINE ERROR HANDLER
*****
*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
*AND GO TO $ERRTYP ON ERROR
*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
*SW15=1 HALT ON ERROR
*SW13=1 INHIBIT ERROR TYPEOUTS
*SW10=1 BELL ON ERROR
*SW9=1 LOOP ON ERROR
*CALL
*
ERROR N ;;ERROR=EMT AND N=ERROR ITEM NUMBER

.ENABL LSB
CLRB IBSAVE ;;R-C
IF NOERROR IS FALSE
CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
BACK:
1$: INCB $ERFLG ;;SET THE ERROR FLAG
BEQ 1$ ;;DON'T LET THE FLAG GO TO ZERO
CALL GETDIS ;;SETUP DISPLAY STUFF
MOV $PATMAR,$TESTN ;;FOR APT
BIT #SW10,@SWR ;;BELL ON ERROR?
BEQ 2$ ;;NO - SKIP
TYPE $BELL ;;RING BELL
TYPE MSG014 ;;CONTROL Z
2$: INC $ERTTL ;;COUNT THE NUMBER OF ERRORS
IF RESULT IS MI
MOV #77777,$ERTTL
END ;OF IF RESULT
END ;OF IF NOERROR
MOV (SP),ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
SUB #2,ERRPC
MOV SP,ERRSP
MOV 2(SP),ERRPSW
MOVB @ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE

CMPB #177,$ITEMB ;;IS THIS THE POWER FAIL CALL?
BEQ 1001$ ;;BRANCH IF SO
TSTB IBSAVE ;;2ND ERROR CALL?
BNE 1000$ ;;BRANCH IF SO
TST CPERRF ;;IS THERE A CPU ERROR REGISTER?
BEQ 1001$ ;;BRANCH IF NOT
MOV 177766,CPSAVE ;;SAVE CONTENTS
BIT #BIT0,CPSAVE ;;POWER MONITOR BIT SET?
BEQ 1001$ ;;BRANCH IF NOT
BIC #BIT0,177766 ;;CLEAR THE BIT
MOVB # $ITEMB,IBSAVE ;;MAKE IBSAVE NON-ZERO FOR DUAL CALL
MOVB #177,$ITEMB ;;SET $ITEMB TO POWER FAIL POINTER
BR 1001$
1000$: CLRB IBSAVE
1001$:
IF NOERROR IS FALSE
IF BADPC NE #0
MOV BADPC,ERRPC
SUB #2,ERRPC
MOV BADSP,ERRSP

```


CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 465-1
 ROUTINE ERROR HANDLER

14884	061040	013737	002030	002026	MOV BADPSW,ERRPSW	
14885	061046	005037	002020		CLR BADPC	
14886	061052				END ;IF	
14887	061052	013737	002016	065706	MOV ERRPC,\$FATAL	:FOR APT
14888	061060	004737	057532		CALL PERBNK	::LOG ERROR ON BANK
14889	061064				IF #SW13 SET.IN @SWR	
14890	061074	000420			BR 3\$	
14891	061076				END ;OF IF #SW13	
14892	061076				IF #SW5 SET.IN @SWR AND TOOMANY IS TRUE	
14893	061114				GOTO 3\$	
14894	061116				END ;OF IF #SW5	
14895	061116				END ;OF IF NOERROR	
14896	061116	004737	061370		CALL \$ERRTYP	::GO TO USER ERROR ROUTINE
14897	061122				IF MONFLG IS TRUE	::SHOULD WE RETURN TO XXDP MONITOR???
14898	061130	013706	002272		MOV SAVMON,SP	::GET MONITOR ADDRESS
14899	061134	000207			RTS PC	::GO TO MONITOR
14900	061136				END	::

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 466
 ROUTINE ERROR HANDLER

```

14902 061136          3$:   IF NOERROR IS FALSE
14903 061144 005777 121454      TST   @SWR          ;;HALT ON ERROR
14904 061150 100002          BPL   7$           ;;SKIP IF CONTINUE
14905 061152 000000          SHALT:  HALT          ;;HALT ON ERROR!
14906 061154 104410          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
14907 061156          7$:   IF NOSCOPE IS FALSE AND #SW9 SET.IN @SWR
14908 061174 013716 002612      MOV   $LPERR,(SP)   ;;FUDGE RETURN FOR LOOPING
14909 061200          END ;OF IF NOSCOPE
14910 061200 005737 002360      TST   $ESCAPE      ;;CHECK FOR AN ESCAPE ADDRESS
14911 061204 001402          BEQ   9$           ;;BR IF NONE
14912 061206 013716 002360      MOV   $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
14913 061212          9$:   IF DETFLAG IS FALSE
14914 061220 022737 000001 003754  CMP   #1,PROTYP    ;IS THIS AN 11/44?
14915 061226 001002          BNE 11$
14916 061230 005037 177766      CLR   CPUERR
14917 061234          11$:  IF ACTFLAG IS TRUE OR APTFLAG IS TRUE OR FATAL$ IS TRUE
14918 061256 012737 000001 065704  MOV   #1,$MSGTY   ;FOR APT
14919 061264 000137 047644      JMP   EXIT
14920 061270          END ;OF IF ACTFLAG
14921 061270          IF XXDPCHAIN IS TRUE AND SERTTL HI #20
14922 061306          TYPE   MSG066   ;ERROR COUNT EXCEEDED 20 - ABORTING FOR XXDP CHAIN
14923 061312 013700 000042      MOV   42,R0
14924 061316 005037 000042      CLR   42
14925 061322 000137 015214      JMP   $ZAP42
14926 061326          END ;OF IF XXDPCHAIN
14927 061326          END ;OF IF DETFLAG
14928 061326          ELSE
14929 061330          SET   HEADER
14930 061336          END ;OF IF NOERROR
14931 061336          10$:  CLEAR  TOOMANY,NOERROR
14932 061346 105737 061362      TSTB  IBSAVE      ;POWER FAIL ERROR CALL? ;R-C
14933 061352 001402          BEQ   213$       ;R-C
14934 061354 000137 060602      JMP   BACK        ;JUMP IF SO ;R-C
14935 061360 000002          213$:  RTI          ;;RETURN
14936 061362 000000          IBSAVE: .WORD 0 ;R-C
14937 061364 000000          CPSAVE: .WORD 0 ;R-C
14938 061366 000000          CPERRF: .WORD 0 ;R-C
14939          .DSABL  LSB

```

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 468
ROUTINE ERROR MESSAGE TYPEOUT

.SBTTL ROUTINE ERROR MESSAGE TYPEOUT

14942
14943
14944
14945
14946
14947
14948
14949 061370 104415
14950 061372
14951 061376 005000
14952 061400 153700 002013
14953 061404 001004
14954
14955 061406
14956 061414 000511
14957 061416 122700 000177
14958 061422 001003
14959 061424 012700 061700
14960 061430 000406
14961 061432 005300
14962 061434 006300
14963 061436 006300
14964 061440 006300
14965 061442 062700 066326
14966 061446 012037 061504
14967 061452 001417
14968 061454 005737 002426
14969 061460 001003
14970 061462 005737 007 :0
14971 061466 100011
14972 061470 005737 002062
14973 061474 001402
14974 061476
14975 061502
14976 061504 000000
14977 061506
14978 061512 012037 061536
14979 061516 001412
14980 061520 005737 002426
14981 061524 001003
14982 061526 005737 002600
14983 061532 100004
14984 061534
14985 061536 000000
14986 061540
14987 061544 012001
14988 061546 001427
14989 061550 012002

```
*****
;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
$ERRTYP:SAVREG
      TYPE      $CRLF      ::"CARRIAGE RETURN" & "LINE FEED"
      CLR      RO          ::PICKUP THE ITEM INDEX
      BISB     $ITEMB,RO
      BNE      1$         ::IF ITEM NUMBER IS ZERO, JUST
                          ::TYPE THE PC OF THE ERROR
      TYPOCT   ERRPC,<ERROR ADDRESS>
      BR       11$        ::GET OUT
1$:   CMPB     #177,RO     ::POWER MONITOR CALL?                :R-C
      BNE     100$        ::BRANCH IF NOT                    :R-C
      MOV     #PFECWS,RO  ::MOV ADDRESS OF PFE BIT ERROR TO RO    :R-C
      BR     110$        ::
100$: DEC     RO          ::ADJUST THE INDEX SO THAT IT WILL      :R-C
      ASL     RO          ::      WORK FOR THE ERROR TABLE
      ASL     RO
      ASL     RO
      ADD     #$ERRTB,RO  ::FORM TABLE POINTER
110$: MOV     (RO)+,3$    ::PICKUP "ERROR MESSAGE" POINTER      :R-C
      BEQ     4$          ::SKIP TYPEOUT IF NO POINTER
      TST     NOERROR     ::IS THIS REALLY AN ERROR?
      BNE     12$        ::YES - SKIP
      TST     HEADER      ::TYPE HEADER?
      BPL     4$          ::NO - SKIP
12$:  TST     FATALS      ::WAS IT A FATAL ERROR?
      BEQ     2$          ::NO - SKIP
      TYPE    MSG067     ::FATAL
2$:   TYPE    "ERROR MESSAGE"
3$:   .WORD   0           ::"ERROR MESSAGE" POINTER GOES HERE
      TYPE    $CRLF      ::"CARRIAGE RETURN" & "LINE FEED"
4$:   MOV     (RO)+,5$    ::PICKUP "DATA HEADER" POINTER
      BEQ     6$          ::SKIP TYPEOUT IF 0
      TST     NOERROR     ::IS THIS REALLY AN ERROR?
      BNE     13$        ::YES - SKIP
      TST     HEADER      ::TYPE HEADER?
      BPL     6$          ::NO - SKIP
13$:  TYPE    "DATA HEADER"
5$:   .WORD   0           ::"DATA HEADER" POINTER GOES HERE
      TYPE    $CRLF      ::"CARRIAGE RETURN" & "LINE FEED"
6$:   MOV     (RO)+,R1    ::PICKUP "DATA TABLE" POINTER
      BEQ     10$        ::BR IF NO DATA TO BE TYPED
      MOV     (RO)+,R2    ::PICKUP "DATA FORMAT" POINTER
```



```

15035 :*****
15036 :*** OCTAL ***
15037 :*****
15038 062010 TAG70$: TYPOCT @ (R1) ;:TYPE AN OCTAL NUMBER
15039 062016 000207 RETURN
15040 :*****
15041 :*** DECIMAL ***
15042 :*****
15043 :*****
15044 062020 TAG71$: TYPDEC @ (R1) ;:TYPE A DECIMAL NUMBER
15045 062026 000207 RETURN
15046 :*****
15047 :*** INTERLEAVE ***
15048 :*****
15049 :*****
15050 062030 TAG72$: PUSH R1,R5
15051 062034 013701 002100 MOV BANK,R1
15052 062040 070127 000004 MUL #4,R1
15053 062044 SET NOTAB ;INDICATE NO TABLE TO BE PRINTED - NOW
15054 062052 TYPE MSG014
15055 062056 004737 041740 CALL TCFIG1
15056 062062 005037 002370 CLR NOTAB
15057 062066 POP R5,R1
15058 062072 TYPE MSG014 ;1 SPACE
15059 062076 000207 RETURN
15060 :*****
15061 :*** CSR ***
15062 :*****
15063 :*****
15064 062100 TAG73$: PUSH R1,R5
15065 062104 013701 002100 MOV BANK,R1
15066 062110 070127 000004 MUL #4,R1
15067 062114 SET NOTAB
15068 062122 004737 042234 CALL TCFIG3
15069 062126 005037 002370 CLR NOTAB
15070 062132 POP R5,R1
15071 062136 000207 RETURN
15072 :*****
15073 :*** PATTERN ***
15074 :*****
15075 :*****
15076 062140 TAG74$: TYPOCS REALPAT,<TYPE (0-77)>,2,2
15077 062150 000207 RETURN
15078 :*****
15079 :*** BANK ***
15080 :*****
15081 :*****
15082 062152 TAG75$: TYPOCS BANK,<TYPE (0-167)>,3
15083 062162 000207 RETURN

```

```

15085
15086
15087
15088 062164
15089 062170 013701 002100
15090 062174 070127 000004
15091 062200
15092 062206
15093 062212 004737 042100
15094 062216 005037 002370
15095 062222
15096 062226 000207
15097
15098
15099
15100
15101 062230
15102 062234 000207
15103
15104
15105
15106
15107 062236 013737 002032 002036
15108 062244 162737 060000 002036
15109 062252 013737 002100 002040
15110 062260 006237 002040
15111 062264 103003
15112 062266 052737 100000 002036
15113 062274 012746 002036
15114 062300 004737 065564
15115 062304 062706 000002
15116 062310
15117 062314 000207
15118
15119
15120
15121
15122 062316
15123 062322
15124 062332
15125 062336 000207

:*****
:*** MTYPE ***
:*****
TAG76$: PUSH R1,R5
        MOV BANK,R1
        MUL #4,R1
        SET NOTAB
        TYPE MSG019
        CALL TCFIG2
        CLR NOTAB
        POP R5,R1
        RETURN

:*****
:*** UNKNOWN DATA ***
:*****
TAG77$: TYPE MSG061
        RETURN

:*****
:*** PHYSICAL ADDRESS ***
:*****
TAG78$: MOV ADDRESS,PHYADD
        SUB #FIRST,PHYADD
        MOV BANK,PHYADD+2
        ASR PHYADD+2
        BCC 1$
        BIS #BIT15,PHYADD
1$: MOV #PHYADD,-(SP) ;POINTER TO DOUBLE WORD ON STACK
    CALL $DB20 ;CALL DOUBLE PRECISION CONVERSION ROUTINE
    ADD #2,SP ;FIX STACK
    TYPE $OCT8
    RETURN

:*****
:*** OCTAL BYTE ***
:*****
TAG79$: TYPE MSG018 ;2 SPACES
        TYPOCS @ (R1),<TYPE BYTE>,3,2
        TYPE MSG014 ;SPACE
        RETURN
    
```


15169 062340

DETAIL: SUBTST <<SUBR DETAILED ERROR REPORT>>

: *SUBTEST SUBR DETAILED ERROR REPORT

15170	062340	005237	002216		INC	DETFLAG
15171	062344	022737	000003	002216	CMP	#3,DETFLAG
15172	062352	101473			BLOS	4\$
15173	062354	022737	000002	002216	CMP	#2,DETFLAG
15174	062362	001435			BEQ	2\$
15175	062364				PUSH	HEADER,MUT
15176	062374				SET	HEADER
15177	062402	005037	002106		CLR	MUT
15178	062406	010037	002176		MOV	R0,DETRO
15179	062412	012700	002200		MOV	#DETR1,R0
15180	062416	010120			MOV	R1,(R0)+
15181	062420	010220			MOV	R2,(R0)+
15182	062422	010320			MOV	R3,(R0)+
15183	062424	010420			MOV	R4,(R0)+
15184	062426	010520			MOV	R5,(R0)+
15185	062430	013720	002022		MOV	ERRSP,(R0)+
15186	062434	013720	002026		MOV	ERRPSW,(R0)+
15187	062440	013700	002176		MOV	DETRO,R0
15188	062444				SET	NOERROR
15189	062452	104013			ERROR	+13
15190	062454	000423			BR	1\$
15191	062456			2\$:	PUSH	HEADER,MUT
15192	062466				SET	HEADER
15193	062474	005037	002106		CLR	MUT
15194	062500				SET	NOERROR
15195	062506	104031			ERROR	+31
15196	062510	022737	000001	003754	CMP	#1,PROTYP
15197	062516	001002			BNE	1\$
15198	062520	005037	177766		CLR	CPUERR
15199	062524			1\$:	POP	MUT,HEADER
15200					:WARNING RECURSIVE	
15201	062534	004737	062340		CALL	DETAIL
15202	062540	000207			RETURN	

:IS THIS AN 11/44?

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 478-1
SUBR DETAILED ERROR REPORT

15262 063054
15263 063060
15264 063064
15265 063070
15266 063074
15267 063106
15268 063110
15269 063114
15270 063114
15271 063120 005037 002216
15272 063124
15273 063126 000207

TYPE \$CRLF
TYPOCT RO
TYPE MSG018 :2 SPACES
TYPOCT (RO)
END ;OF FOR RO
ELSE
TYPE MSG091 :IS EMPTY
END ;OF IF RO
TYPE \$CRLF
CLR DETFLAG
POP RO
RETURN


```

15311 .SBTTL ROUTINE BINARY TO OCTAL (ASCII) AND TYPE
15312
15313 *****
15314 *THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
15315 *OCTAL (ASCII) NUMBER AND TYPE IT.
15316 *$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
15317 *CALL:
15318 *      MOV      NUM,-(SP)      ::NUMBER TO BE TYPED
15319 *      TYPOS    ::CALL FOR TYPEOUT
15320 *      .BYTE    N              ::N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
15321 *      .BYTE    M              ::M=1 OR 0
15322 *                                  ::1=TYPE LEADING ZEROS
15323 *                                  ::0=SUPPRESS LEADING ZEROS
15324
15325 *$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
15326 *$TYPOS OR $TYPOC
15327 *CALL:
15328 *      MOV      NUM,-(SP)      ::NUMBER TO BE TYPED
15329 *      TYPON    ::CALL FOR TYPEOUT
15330
15331 *$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
15332 *CALL:
15333 *      MOV      NUM,-(SP)      ::NUMBER TO BE TYPED
15334 *      TYPOC    ::CALL FOR TYPEOUT
15335
15336 063130 017646 000000 063353 $TYPOS: MOV      @ (SP),-(SP)      ::PICKUP THE MODE
15337 063134 116637 000001          MOVVB   1(SP), $OFILL    ::LOAD ZERO FILL SWITCH
15338 063142 112637 063355          MOVVB   (SP)+, $OMODE+1  ::NUMBER OF DIGITS TO TYPE
15339 063146 062716 000002          ADD     #2, (SP)      ::ADJUST RETURN ADDRESS
15340 063152 000406          BR      $TYPON
15341 063154 112737 000001 063353 $TYPOC: MOVVB   #1, $OFILL    ::SET THE ZERO FILL SWITCH
15342 063162 112737 000006 063355 MOVVB   #6, $OMODE+1  ::SET FOR SIX(6) DIGITS
15343 063170 112737 000005 063352 $TYPON: MOVVB   #5, $OCNT    ::SET THE ITERATION COUNT
15344 063176 010346          MOV     R3,-(SP)     ::SAVE R3
15345 063200 010446          MOV     R4,-(SP)     ::SAVE R4
15346 063202 010546          MOV     R5,-(SP)     ::SAVE R5
15347 063204 113704 063355          MOVVB   $OMODE+1,R4  ::GET THE NUMBER OF DIGITS TO TYPE
15348 063210 005404          NEG     R4
15349 063212 062704 000006          ADD     #6,R4        ::SUBTRACT IT FOR MAX. ALLOWED
15350 063216 110437 063354          MOVVB   R4, $OMODE  ::SAVE IT FOR USE
15351 063222 113704 063353          MOVVB   $OFILL,R4   ::GET THE ZERO FILL SWITCH
15352 063226 016605 000012          MOV     12(SP),R5   ::PICKUP THE INPUT NUMBER
15353 063232 005003          CLR     R3          ::CLEAR THE OUTPUT WORD
15354 063234 006105          1$:    ROL     R5    ::ROTATE MSB INTO 'C'
15355 063236 000404          BR      3$         ::GO DO MSB
15356 063240 006105          2$:    ROL     R5    ::FORM THIS DIGIT
15357 063242 006105          ROL     R5
15358 063244 006105          ROL     R5
15359 063246 010503          MOV     R5,R3
15360 063250 006103          3$:    ROL     R3    ::GET LSB OF THIS DIGIT
15361 063252 105337 063354          DECB   $OMODE      ::TYPE THIS DIGIT?
15362 063256 100016          BPL    6$          ::BR IF NO
15363 063260 042703 177770          BIC    #177770,R3  ::GET RID OF JUNK
15364 063264 001002          BNE    4$          ::TEST FOR 0
15365 063266 005704          TST   R4          ::SUPPRESS THIS 0?
15366 063270 001403          BEQ   5$          ::BR IF YES
15367 063272 005204          4$:    INC    R4    ::DON'T SUPPRESS ANYMORE 0'S
    
```

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 481-1
 ROUTINE BINARY TO OCTAL (ASCII) AND TYPE

15368	063274	052703	000060		BIS	#'0,R3	::MAKE THIS DIGIT ASCII
15369	063300	052703	000040	5\$:	BIS	#',R3	::MAKE ASCII IF NOT ALREADY
15370	063304	110337	063350		MOVB	R3,8\$::SAVE FOR TYPING
15371	063310				TYPE	8\$::GO TYPE THIS DIGIT
15372	063314	105337	063352	6\$:	DECB	\$OCNT	::COUNT BY 1
15373	063320	003347			BGT	2\$::BR IF MORE TO DO
15374	063322	002402			BLT	7\$::BR IF DONE
15375	063324	005204			INC	R4	::INSURE LAST DIGIT ISN'T A BLANK
15376	063326	000744			BR	2\$::GO DO THE LAST DIGIT
15377	063330	012605		7\$:	MOV	(SP)+,R5	::RESTORE R5
15378	063332	012604			MOV	(SP)+,R4	::RESTORE R4
15379	063334	012603			MOV	(SP)+,R3	::RESTORE R3
15380	063336	016666	000002 000004		MOV	2(SP),4(SP)	::SET THE STACK FOR RETURNING
15381	063344	012616			MOV	(SP)+,(SP)	
15382	063346	000002			RTI		::RETURN
15383	063350	000		8\$:	.BYTE	0	::STORAGE FOR ASCII DIGIT
15384	063351	000			.BYTE	0	::TERMINATOR FOR TYPE ROUTINE
15385	063352	000			\$OCNT:	.BYTE 0	::OCTAL DIGIT COUNTER
15386	063353	000			\$OFILL:	.BYTE 0	::ZERO FILL SWITCH
15387	063354	000000			\$OMODE:	.WORD 0	::NUMBER OF DIGITS TO TYPE

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 482
 ROUTINE CONVERT BINARY TO DECIMAL AND TYPE

```

15389          .SBTTL  ROUTINE CONVERT BINARY TO DECIMAL AND TYPE
15390          :*****
15391          :*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
15392          :*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
15393          :*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
15394          :*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
15395          :*REPLACED WITH SPACES.
15396          :*CALL:
15397          :*      MOV      NUM,-(SP)          ;;PUT THE BINARY NUMBER ON THE STACK
15398          :*      TYPDS   R0,R1,R2,R3,R5   ;;GO TO THE ROUTINE
15399 063356      $TYPDS: PUSH      R0,R1,R2,R3,R5
15400 063370      MOV      #20200,-(SP)     ;;SET BLANK SWITCH AND SIGN
15401 063374      MOV      20(SP),R5        ;;GET THE INPUT NUMBER
15402 063400      BPL      1$                ;;BR IF INPUT IS POS.
15403 063402      NEG      R5                ;;MAKE THE BINARY NUMBER POS.
15404 063404      MOVVB   #'-,1(SP)        ;;MAKE THE ASCII NUMBER NEG.
15405 063412      CLR      R0                ;;ZERO THE CONSTANTS INDEX
15406 063414      MOV      #$DBLK,R3        ;;SETUP THE OUTPUT POINTER
15407 063420      MOVVB   #' ,(R3)+        ;;SET THE FIRST CHARACTER TO A BLANK
15408 063424      CLR      R2                ;;CLEAR THE BCD NUMBER
15409 063426      MOV      $DTBL(R0),R1    ;;GET THE CONSTANT
15410 063432      SUB      R1,R5            ;;FORM THIS BCD DIGIT
15411 063434      BLT     4$                ;;BR IF DONE
15412 063436      INC     R2                ;;INCREASE THE BCD DIGIT BY 1
15413 063440      BR      3$                ;;
15414 063442      4$:   ADD      R1,R5      ;;ADD BACK THE CONSTANT
15415 063444      TST     R2                ;;CHECK IF BCD DIGIT=0
15416 063446      BNE     5$                ;;FALL THROUGH IF 0
15417 063450      TSTB   (SP)              ;;STILL DOING LEADING 0'S?
15418 063452      BMI     7$                ;;BR IF YES
15419 063454      5$:   ASLB   (SP)        ;;MSD?
15420 063456      BCC     6$                ;;BR IF NO
15421 063460      MOVVB   1(SP),-1(R3)     ;;YES--SET THE SIGN
15422 063466      6$:   BIS     #'0,R2      ;;MAKE THE BCD DIGIT ASCII
15423 063472      7$:   BIS     #' ,R2      ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
15424 063476      MOVVB   R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
15425 063500      TST     (R0)+            ;;JUST INCREMENTING
15426 063502      CMP     R0,#10           ;;CHECK THE TABLE INDEX
15427 063506      BLT     2$                ;;GO DO THE NEXT DIGIT
15428 063510      BGT     8$                ;;GO TO EXIT
15429 063512      MOV     R5,R2            ;;GET THE LSD
15430 063514      BR      6$                ;;GO CHANGE TO ASCII
15431 063516      8$:   TSTB   (SP)+        ;;WAS THE LSD THE FIRST NON-ZERO?
15432 063520      BPL     9$                ;;BR IF NO
15433 063522      MOVVB   -1(SP),-2(R3)    ;;YES--SET THE SIGN FOR TYPING
15434 063530      9$:   CLRB   (R3)         ;;SET THE TERMINATOR
15435 063532      POP     R5,R3,R2,R1,R0
15436 063544      TYPE   $DBLK             ;;NOW TYPE THE NUMBER
15437 063550      MOV     2(SP),4(SP)      ;;ADJUST THE STACK
15438 063556      MOV     (SP)+,(SP)
15439 063560      RI     R1                ;;RETURN TO USER
15440 063562      $DTBL: 10000.
15441 063564      1000.
15442 063566      100.
15443 063570      10.
15444 063572      000000 000000 000000 $DBLK: .WORD 0,0,0,0
          063600      000000
    
```



```

15446
15447
15448
15449
15450
15451
15452
15453 063602
15459 063602 005737 056332
15460 063606 001406
15461 063610 013746 056332
15462 063614 005037 056332
15463 063620 000137 063642
15464 063624 105777 117000
15465 063630 100130
15466 063632 117746 116774
15467 063636 042716 177600
15468 063642 022716 000006
15469 063646 001002
15470 063650 004737 050114
15471 063654 022716 000024
15472 063660 001002
15473 063662 004737 064210
15474 063666 022716 000003
15475 063672 001454
15476 063674 022716 000023
15477 063700 001002
15478 063702 004737 064264
15479 063706 022716 000013
15480 063712 001005
15481 063714
15482 063720 013706 002144
15483 063724 000207
15484 063726 022737 000176 002624
15485 063734 001067
15486 063736 022716 000007
15487 063742 001064
15488 063744 005737 002060
15489 063750 001061
15490 063752
15491 063756
15492 063762
15493 063770
15494 063774 005046
15495 063776 005046
15496 064000 105777 116624
15497 064004 100375
15498 064006 117746 116620
15499 064012 042716 177600
15500 064016 021627 000003
15501 064022 001006
15502 064024
15503 064030 062706 000006
15504 064034 000137 047536
15505 064040 021627 000025
15506 064044 001005
15507 064046

```

```

.SBTTL ROUTINE TTY INPUT
:*****
:*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
:*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
:*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
:*WHEN OPERATING IN TTY FLAG MODE.
.ENABLE LSB
$CKSWR:
TST X0CHAR ;;SOMETHING THERE?
BEQ NOCH ;; GO ON IF NOT
MOV X0CHAR,-(SP) ;; USE IT
CLR X0CHAR
JMP CONTS1
NOCH: TSTB @STKS ;;CHAR THERE?
BPL 12$ ;;IF NO, DON'T WAIT AROUND
MOVB @STKB,-(SP) ;;SAVE THE CHAR
BIC #'C177,(SP) ;;STRIP-OFF THE ASCII
CONTS1: CMP #6,(SP) ;;IS IT CONTROL F?
BNE 1$ ;;NO SKIP
CALL FIELDSERVICE
1$: CMP #24,(SP) ;;IS IT CONTROL T?
BNE 16$ ;;NO - SKIP
CALL CONTT ;;YES - CALL CONTROL T ROUTINE
16$: CMP #3,(SP) ;;IS IT CONTROL C?
BEQ 5$ ;;YES EXIT *****NOTE***** STACK IS SCREWED UP!
2$: CMP #23,(SP) ;;IS IT CONTROL S?
BNE 17$ ;;NO - SKIP
CALL CONTS ;;YES - CALL CONTROL S ROUTINE
17$: CMP #13,(SP) ;;IS IT CONTROL K?
BNE 6$ ;;NO - SKIP
TYPE $CNTLK ;;TYPE A ^K
MOV CTLKVEC,SP ;;RESET KSP TO AFTER PATTERN EXEC ROUTINE
RETURN ;;RETURN TO PATTERN EXEC ROUTINE
6$: CMP #SWREG,SWR ;;IS THE SOFT-SWR SELECTED?
BNE CKEND ;;BRANCH IF NO
CMP #7,(SP) ;;IS IT A CONTROL G?
BNE CKEND ;;NO, RETURN TO USER
TST $AUTO ;;ARE WE RUNNING IN AUTO-MODE?
BNE CKEND ;;BRANCH IF YES
TYPE $CNTLG ;;ECHO THE CONTROL-G (^G)
$GTSWR: TYPE $MSWR ;;TYPE CURRENT CONTENTS
TYPOCT @SWR ;;OF THE SWR
TYPE $MNEW ;;PROMPT FOR NEW SWR
3$: CLR -(SP) ;;CLEAR COUNTER
CLR -(SP) ;;THE NEW SWR
4$: TSTB @STKS ;;CHAR THERE?
BPL 4$ ;;IF NOT TRY AGAIN
MOVB @STKB,-(SP) ;;PICK UP CHAR
BIC #'C177,(SP) ;;MAKE IT 7-BIT ASCII
5$: CMP (SP),#3 ;;IS IT A CONTROL-C?
BNE 7$ ;;BRANCH IF NOT
TYPE $CNTLC ;;YES, ECHO CONTROL-C (^C)
ADD #6,SP ;;CLEAN UP STACK
JMP BOOT ;;CONTROL-C RESTART
7$: CMP (SP),#25 ;;IS IT A CONTROL-U?
BNE 9$ ;;BRANCH IF NOT
TYPE $CNTLU ;;YES, ECHO CONTROL-U (^U)

```

CZMSPBG MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 483-1
 ROUTINE TTY INPUT

15508	064052	062706	000006		8\$:	ADD	#6,SP	::	IGNORE PREVIOUS INPUT
15509	064056	000746				BR	3\$::	LET'S TRY IT AGAIN
15510	064060	021627	000015		9\$:	CMP	(SP),#15	::	IS IT A <CR>?
15511	064064	001016				BNE	13\$::	BRANCH IF NO
15512	064066	005766	000004			TST	4(SP)	::	YES, IS IT THE FIRST CHAR?
15513	064072	001403				BEQ	10\$::	BRANCH IF YES
15514	064074	016677	000002	116522		MOV	2(SP),@SWR	::	SAVE NEW SWR
15515	064102	062706	000006		10\$:	ADD	#6,SP	::	CLEAR UP STACK
15516	064106					TYPE	\$CRLF	::	ECHO <CR> AND <LF>
15517	064112	000002			12\$:	RTI		::	RETURN
15518	064114	062706	000002		CKEND:	ADD	#2,SP	::	FIX STACK
15519	064120	000002				RTI		::	RETURN
15520	064122	004737	056334		13\$:	CALL	\$TYPEC	::	ECHO CHAR
15521	064126	021627	000060			CMP	(SP),#60	::	CHAR < 0?
15522	064132	002420				BLT	15\$::	BRANCH IF YES
15523	064134	021627	000067			CMP	(SP),#67	::	CHAR > 7?
15524	064140	003015				BGT	15\$::	BRANCH IF YES
15525	064142	042726	000060			BIC	#60,(SP)+	::	STRIP-OFF ASCII
15526	064146	005766	000002			TST	2(SP)	::	IS THIS THE FIRST CHAR
15527	064152	001403				BEQ	14\$::	BRANCH IF YES
15528	064154	006316				ASL	(SP)	::	NO, SHIFT PRESENT
15529	064156	006316				ASL	(SP)	::	CHAR OVER TO MAKE
15530	064160	006316				ASL	(SP)	::	ROOM FOR NEW ONE.
15531	064162	005266	000002		14\$:	INC	2(SP)	::	KEEP COUNT OF CHAR
15532	064166	056616	177776			BIS	-2(SP),(SP)	::	SET IN NEW CHAR
15533	064172	000702				BR	4\$::	GET THE NEXT ONE
15534	064174				15\$:	TYPE	\$QUES	::	TYPE ?<CR><LF>
15535	064200	000724				BR	8\$::	SIMULATE CONTROL-U
15536	064202	136	113	015	\$CNTLK:	.ASCIZ	/'K/<15><12>	::	CONTROL K ASCII STRING
15536	064205	012	000						
15537						.EVEN			
15538						.DSABL	LSB		

15541 064210

CONTT: SUBTST <<CONTROL T>>
:*****
:*SUBTEST CONTROL T
:*****

15542 064210
15543 064212
15553 064216
15554 064224
15555 064230
15556 064230
15557 064234
15558 064244
15559 064250
15563 064260
15564 064262 000207
15565
15566 064264

PUSH R0
TYPE \$CRLF
IF RLFLAG IS TRUE
TYPE MSG092 :RELOCATED
END ;OF IF RLFLAG
TYPE MSG093 :BANK=
TYPOCS BANK,,3 :TYPE 3 DIGITS
TYPE MSG095 :PAT=
TYPOCS REALPAT,,2 :TYPE 2 DIGITS
POP R0
RETURN

CONTS: SUBTST <<CONTROL S & CONTROL Q>>
:*****
:*SUBTEST CONTROL S & CONTROL Q
:*****

15567 064264
15568 064266 105777 116336
15569 064272 100375
15570 064274 117716 116332
15571 064300 042716 177600
15572 064304
15573 064312 000137 063642
15574 064316
15575 064320 000762
15576 064322

CONTS2: POP R0 :GET RID OF RETURN ADDRESS FROM STACK
TSTB @STKS :WAIT FOR CHARACTER
BPL CONTS2
MOVB @STKB,(SP) :REPLACE OVER OLD CHARACTER ON STACK
BIC #'C177,(SP) :STRIP ALL BUT ASCII
IF (SP) EQ #21 :IF IT IS A CONTROL Q
JMP CONTS1
ELSE
BR CONTS2
END ;OF IF (SP)

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 486
CONTROL S & CONTROL Q

```

15578
15579
15580
15581
15582
15583
15584
15585
15586 064322 011646
15587 064324 016666 000004 000002
15588 064332 105777 116272
15589 064336 100375
15590 064340 117766 116266 000004
15591 064346 042766 177600 000004
15592 064354 026627 000004 000023
15593 064362 001013
15594 064364 105777 116240
15595 064370 100375
15596 064372 117746 116234
15597 064376 042716 177600
15598 064402 022627 000021
15599 064406 001366
15600 064410 000750
15601 064412 026627 000004 000021
15602 064420 001744
15603 064422 026627 000004 000140
15604 064430 002407
15605 064432 026627 000004 000175
15606 064440 003003
15607 064442 042766 000040 000004
15608 064450 000002
15609
15610
15611
15612
15613
15614
15615 064452 010346
15616 064454 005046
15617 064456 012703 064750
15618 064462 022703 064774
15619 064466 101477
15620 064470 104411
15621 064472 112613
15622 064474 122713 000003
15623 064500 001016
15624 064502
15625 064506 005726
15626 064510 012603
15627 064512 032777 000400 116104
15628 064520 001404
15629 064522 005037 002416
15630 064526 000137 047644
15631 064532 000137 047536
15632 064536 122713 000177
15633 064542 001022
15634 064544 005716

*****
*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
*CALL:
*   RDCHR          ::INPUT A SINGLE CHARACTER FROM THE TTY
*   RETURN HERE   ::CHARACTER IS ON THE STACK
*                 ::WITH PARITY BIT STRIPPED OFF
*
$RDCHR: MOV      (SP),-(SP)      ::PUSH DOWN THE PC
        MOV      4(SP),2(SP)    ::SAVE THE PS
1$:    TSTB     @STKS           ::WAIT FOR
        BPL      1$            ::A CHARACTER
        MOVB     @STKB,4(SP)    ::READ THE TTY
        BIC      #'C<177>,4(SP) ::GET RID OF JUNK IF ANY
        CMP      4(SP),#23     ::IS IT A CONTROL-S?
        BNE      3$            ::BRANCH IF NO
2$:    TSTB     @STKS           ::WAIT FOR A CHARACTER
        BPL      2$            ::LOOP UNTIL ITS THERE
        MOVB     @STKB,-(SP)    ::GET CHARACTER
        BIC      #'C177,(SP)   ::MAKE IT 7-BIT ASCII
        CMP      (SP)+,#21     ::IS IT A CONTROL-Q?
        BNE      2$            ::IF NOT DISCARD IT
        BR       1$            ::YES, RESUME
3$:    CMP      4(SP),#21      ::IS IT A RANDOM CONTROL-Q?      :R-C
        BEQ      1$            ::BRANCH BACK IF SO                :R-C
        CMP      4(SP),#140    ::IS IT UPPER CASE?
        BLT      4$            ::BRANCH IF YES
        CMP      4(SP),#175    ::IS IT A SPECIAL CHAR?
        BGT      4$            ::BRANCH IF YES
        BIC      #40,4(SP)     ::MAKE IT UPPER CASE
4$:    RTI                    ::GO BACK TO USER
*****
*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
*CALL:
*   RDLIN          ::INPUT A STRING FROM THE TTY
*   RETURN HERE   ::ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
*                 ::TERMINATOR WILL BE A BYTE OF ALL 0'S
*
$RDLIN: MOV      R3,-(SP)      ::SAVE R3
        CLR      -(SP)        ::CLEAR THE RUBOUT KEY
1$:    MOV      #$TTYIN,R3    ::GET ADDRESS
2$:    CMP      #$TTYIN+20.,R3 ::BUFFER FULL?
        BLOS    8$            ::BR IF YES
        RDCHR   ::GO READ ONE CHARACTER FROM THE TTY
        MOVB    (SP)+,(R3)    ::GET CHARACTER
        CMPB    #3,(R3)      ::IS IT A CONTROL-C?
        BNE     3$            ::BRANCH IF NO
        TYPE    $CNTLC       ::TYPE A CONTROL-C (^C)
        TST     (SP)+        ::CLEAN RUBOUT KEY OFF OF THE STACK
        MOV     (SP)+,R3     ::RESTORE R3
        BIT     #BIT8,@SWR   ::IS THERE A HALT FLAG SET IN THE SWR?
        BEQ    11$          ::BRANCH IF NOT TO BOOT ROUTINE
        CLR    STOPOK       ::GET READY TO HALT PROGRAM
        JMP    EXIT         ::GO HALT PROGRAM
11$:   JMP    BOOT          ::GOTO CONTROL-C RESTART
3$:    CMPB    #177,(R3)    ::IS IT A RUBOUT
        BNE    5$            ::BR IF NO
        TST    (SP)         ::IS THIS THE FIRST RUBOUT?

```

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 486-1
CONTROL S & CONTROL Q

15635	064546	001007				BNE	4\$::BR IF NO
15636	064550	112737	000134	064746		MOVB	#'\,10\$::TYPE A BACK SLASH
15637	064556					TYPE	10\$	
15638	064562	012716	177777			MOV	#-1,(SP)	::SET THE RUBOUT KEY
15639	064566	005303			4\$:	DEC	R3	::BACKUP BY ONE
15640	064570	020327	064750			CMP	R3,#STTYIN	::STACK EMPTY?
15641	064574	103434				BLO	8\$::BR IF YES
15642	064576	111337	064746			MOVB	(R3),10\$::SETUP TO TYPEOUT THE DELETED CHAR.
15643	064602					TYPE	10\$::GO TYPE
15644	064606	000725				BR	2\$::GO READ ANOTHER CHAR.
15645	064610	005716			5\$:	TST	(SP)	::RUBOUT KEY SET?
15646	064612	001406				BEQ	6\$::BR IF NO
15647	064614	112737	000134	064746		MOVB	#'\,10\$::TYPE A BACK SLASH
15648	064622					TYPE	10\$	
15649	064626	005016				CLR	(SP)	::CLEAR THE RUBOUT KEY
15650	064630	122713	000025		6\$:	CMPB	#25,(R3)	::IS CHARACTER A CTRL U?
15651	064634	001003				BNE	7\$::BR IF NO
15652	064636					TYPE	\$CNTLU	::TYPE A CONTROL 'U'
15653	064642	000705				BR	1\$::GO START OVER
15654	064644	122713	000022		7\$:	CMPB	#22,(R3)	::IS CHARACTER A '^R'?
15655	064650	001011				BNE	9\$::BRANCH IF NO
15656	064652	105013				CLRB	(R3)	::CLEAR THE CHARACTER
15657	064654					TYPE	\$CRLF	::TYPE A 'CR' & 'LF'
15658	064660					TYPE	\$TTYIN	::TYPE THE INPUT STRING
15659	064664	000676				BR	2\$::GO PICKUP ANOTHER CHACTER
15660	064666				8\$:	TYPE	\$QUES	::TYPE A '?'
15661	064672	000671				BR	1\$::CLEAR THE BUFFER AND LOOP
15662	064674	111337	064746		9\$:	MOVB	(R3),10\$::ECHO THE CHARACTER
15663	064700					TYPE	10\$	
15664	064704	122723	000015			CMPB	#15,(R3)+	::CHECK FOR RETURN
15665	064710	001264				BNE	2\$::LOOP IF NOT RETURN
15666	064712	105063	177777			CLRB	-1(R3)	::CLEAR RETURN (THE 15)
15667	064716					TYPE	\$LF	::TYPE A LINE FEED
15668	064722	005726				TST	(SP)+	::CLEAN RUBOUT KEY FROM THE STACK
15669	064724	012603				MOV	(SP)+,R3	::RESTORE R3
15670	064726	011646				MOV	(SP),-(SP)	::ADJUST THE STACK AND PUT ADDRESS OF THE
15671	064730	016666	000004	000002		MOV	4(SP),2(SP)	::FIRST ASCII CHARACTER ON IT
15672	064736	012766	064750	000004		MOV	#STTYIN,4(SP)	
15673	064744	000002				RTI		::RETURN
15674	064746	000			10\$:	.BYTE	0	::STORAGE FOR ASCII CHAR. TO TYPE
15675	064747	000				.BYTE	0	::TERMINATOR
15676	064750	000024				\$TTYIN:	.REPT 20.	::RESERVE SIZE BYTES FOR TTY INPUT
15679	064774	136	103	015		\$CNTLC:	.ASCIZ /^C/<15><12>	::CONTROL 'C'
	064777	012	000					
15680	065001	136	125	015		\$CNTLU:	.ASCIZ /^U/<15><12>	::CONTROL 'U'
	065004	012	000					
15681	065006	136	107	015		\$CNTLG:	.ASCIZ /^G/<15><12>	::CONTROL 'G'
	065011	012	000					
15682	065013	015	012	123		\$MSWR:	.ASCIZ <15><12>/SWR = /	
	065016	127	122	040				
	065021	075	040	000				
15683	065024	040	040	116		\$MNEW:	.ASCIZ / NEW = /	
	065027	105	127	040				
	065032	075	040	000				
15684						.EVEN		

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 487
 ROUTINE READ AN OCTAL NUMBER FROM THE TTY

```

15686                                     .SBTTL  ROUTINE READ AN OCTAL NUMBER FROM THE TTY
15687                                     :*****
15688                                     :*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
15689                                     :*CHANGE IT TO BINARY.
15690                                     :*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
15691                                     :*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED
15692                                     :*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
15693                                     :*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
15694                                     :*CALL:
15695                                     :*      RDOCT          ::READ AN OCTAL NUMBER
15696                                     :*      RETURN HERE  ::LOW ORDER BITS ARE ON TOP OF THE STACK
15697                                     :*                                     ::HIGH ORDER BITS ARE IN $HIOCT
15698 065036 011646 $RDOCT: MOV      (SP),-(SP)      ::PROVIDE SPACE FOR THE
15699 065040 016666 000004 000002      MOV      4(SP),2(SP)      ::INPUT NUMBER
15700 065046                                     PUSH     RO,R1,R2
15701 065054 104412 1$:  RDLIN                                     ::READ AN ASCII LINE
15702 065056 012600      MOV      (SP)+,RO      ::GET ADDRESS OF 1ST CHARACTER
15703 065060 010037 065164      MOV      RO,$$          ::AND SAVE IT
15704 065064 005001      CLR      R1            ::CLEAR DATA WORD
15705 065066 005002      CLR      R2
15706 065070 112046 2$:  MOV8     (RO)+,-(SP)     ::PICKUP THIS CHARACTER
15707 065072 001420      BEQ      3$           ::IF ZERO GET OUT
15708 065074 122716 000060      CMPB    #'0,(SP)      ::MAKE SURE THIS CHARACTER
15709 065100 003026      BGT      4$           ::IS AN OCTAL DIGIT
15710 065102 122716 000067      CMPB    #'7,(SP)
15711 065106 002423      BLT      4$
15712 065110 006301      ASL     R1            ::*2
15713 065112 006102      ROL     R2
15714 065114 006301      ASL     R1            ::*4
15715 065116 006102      ROL     R2
15716 065120 006301      ASL     R1            ::*8
15717 065122 006102      ROL     R2
15718 065124 042716 177770      BIC     #'C7,(SP)     ::STRIP THE ASCII JUNK
15719 065130 062601      ADD     (SP)+,R1      ::ADD IN THIS DIGIT
15720 065132 000756      BR      2$           ::LOOP
15721 065134 005726 3$:  TST     (SP)+         ::CLEAN TERMINATOR FROM STACK
15722 065136 010166 000012      MOV     R1,12(SP)     ::SAVE THE RESULT
15723 065142 010237 065204      MOV     R2,$HIOCT
15724 065146      POP     R2,R1,RO
15725 065154 000002      RTI                                     ::RETURN
15726 065156 005726 4$:  TST     (SP)+         ::CLEAN PARTIAL FROM STACK
15727 065160 105010      CLRB    (RO)         ::SET A TERMINATOR
15728 065162      TYPE                                     ::TYPE UP THRU THE BAD CHAR.
15729 065164 000000 5$:  .WORD   0
15730 065166      TYPE   MSG062        :INPUT MUST BE A
15731 065172      TYPE   MSG063        :N OCTAL
15732 065176      TYPE   MSG064        :NUMBER
15733 065202 000724      BR      1$           ::TRY AGAIN
15734 065204 000000 $HIOCT: .WORD   0      ::HIGH ORDER BITS GO HERE
15735                                     .SBTTL  ROUTINE READ A DECIMAL NUMBER FROM THE TTY
15736                                     :*****
15737                                     :*THIS ROUTINE WILL READ A DECIMAL (ASCII) NUMBER FROM THE TTY AND
15738                                     :*CHANGE IT TO BINARY. IF TOO MANY CHARACTERS OR ANY ILLEGAL CHARACTERS
15739                                     :*ARE READ A "?" FOLLOWED BY A CARRIAGE RETURN-LINE FEED WILL BE TYPED.
15740                                     :*THE COMPLETE NUMBER MUST BE RETYPED. THE INPUT IS TERMINATED BY THE
15741                                     :*USER TYPING A CARRIAGE RETURN. THE RANGE OF THE INPUT NUMBER IS
15742

```


CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 487-1
 ROUTINE READ A DECIMAL NUMBER FROM THE TTY

```

15743 ;*POSITIVE 32767 TO NEGATIVE 32768.
15744 ;*CALL:
15745 ;* RDDEC ;:READ A DECIMAL NUMBER
15746 ;* RETURN HERE ;:NUMBER IS ON TOP OF THE STACK
15747 ;
15748
15749 065206 011646 SRDDEC: MOV (SP),-(SP) ;:PROVIDE SPACE FOR
15750 065210 016666 000004 000002 MOV 4(SP),2(SP) ;:THE INPUT NUMBER
15751 065216 PUSH R0,R1,R2
15752 065224 104412 1$: RDLIN ;:READ AN ASCII LINE
15753 065226 012600 MOV (SP)+,R0 ;:ADDRESS OF 1ST CHAR.
15754 065230 010037 065354 MOV R0,6$ ;:SAVE INCASE OF BAD INPUT
15755 065234 005046 CLR -(SP) ;:CLEAR DATA WORD
15756 065236 005002 CLR R2 ;:SIGN SET POSITIVE
15757 065240 122710 000055 CMPB #'-(R0) ;:SEE IF A MINUS SIGN WAS TYPED
15758 065244 001001 BNE 2$ ;:BR IF NO MINUS SIGN
15759 065246 112002 MOVB (R0)+,R2 ;:SAVE FOR LATER USE
15760 065250 112001 2$: MOVB (R0)+,R1 ;:PICKUP THIS CHARACTER
15761 065252 001424 BEQ 3$ ;:GET OUT IF ZERO
15762 065254 122701 000060 CMPB #'0,R1 ;:MAKE SURE THIS CHARACTER
15763 065260 003032 BGT 5$ ;:IS A DIGIT BETWEEN 0 & 9
15764 065262 122701 000071 CMPB #'9,R1
15765 065266 002427 BLT 5$
15766 065270 032716 170000 BIT #'C7777,(SP) ;:DON'T LET NUMBER GET TO BIG
15767 065274 001024 BNE 5$ ;:BR IF NUMBER WOULD OVERFLOW
15768 065276 006316 ASL (SP) ;:*2
15769 065300 011646 MOV (SP),-(SP) ;:SAVE FOR LATER
15770 065302 006316 ASL (SP) ;:*4
15771 065304 006316 ASL (SP) ;:*8
15772 065306 062616 ADD (SP)+,(SP) ;:*10
15773 065310 102416 BVS 5$ ;:OVERFLOW ISN'T ALLOWED
15774 065312 162701 000060 SUB #'0,R1 ;:STRIP AWAY THE ASCII JUNK
15775 065316 060116 ADD R1,(SP) ;:ADD IN THIS DIGIT
15776 065320 102412 BVS 5$ ;:OVERFLOW ISN'T ALLOWED
15777 065322 000752 BR 2$ ;:LOOP
15778 065324 005702 3$: TST R2 ;:CHECK IF NUMBER IS NEG
15779 065326 001401 BEQ 4$ ;:BR IF NO
15780 065330 005416 NEG (SP) ;:YES--NEGATE THE NUMBER
15781 065332 012666 000012 4$: MOV (SP)+,12(SP) ;:SAVE THE RESULT
15782 065336 POP R2,R1,R0
15783 065344 000002 RTI ;:RETURN
15784
15785 065346 005726 5$: TST (SP)+ ;:CLEAN PARTIAL NUMBER FROM STACK
15786 065350 105010 CLRB (R0) ;:SET A TERMINATOR
15787 065352 TYPE ;:TYPE THE INPUT UP TO BAD CHAR.
15788 065354 000000 6$: .WORD 0 ;:POINTER GOES HERE
15789 065356 TYPE MSG062 ;:INPUT MUSST BE A
15790 065362 TYPE MSG065 ;:DECIMAL
15791 065366 TYPE MSG064 ;:NUMBER
15792 065372 000714 BR 1$ ;:TRY AGAIN

```

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 488
 ROUTINE SAVE AND RESTORE R0-R5

15794
 15795
 15796
 15797
 15798
 15799
 15800
 15801
 15802
 15803
 15804
 15805
 15806
 15807
 15808
 15809
 15810
 15811 065374
 15812 065374
 15813 065410 016646 000022
 15814 065414 016646 000022
 15815 065420 016646 000022
 15816 065424 016646 000022
 15817 065430 000002
 15818
 15819
 15820
 15821
 15822 065432
 15823 065432 012666 000022
 15824 065436 012666 000022
 15825 065442 012666 000022
 15826 065446 012666 000022
 15827 065452
 15828 065466 000002

.SBTTL ROUTINE SAVE AND RESTORE R0-R5

```

*****
;*SAVE R0-R5
;*CALL:
;*   SAVREG
;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
;*
;*TOP---(+16)
;* +2---(+18)
;* +4---R5
;* +6---R4
;* +8---R3
;*+10---R2
;*+12---R1
;*+14---R0

$SAVREG:
      PUSH   R0,R1,R2,R3,R4,R5
      MOV    22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
      MOV    22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
      MOV    22(SP),-(SP)    ;;SAVE PS OF CALL
      MOV    22(SP),-(SP)    ;;SAVE PC OF CALL
      RTI

;*RESTORE R0-R5
;*CALL:
;*   RESREG
$RESREG:
      MOV    (SP)+,22(SP)    ;;RESTORE PC OF CALL
      MOV    (SP)+,22(SP)    ;;RESTORE PS OF CALL
      MOV    (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
      MOV    (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
      POP    R5,R4,R3,R2,R1,R0
      RTI

```

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 489
 ROUTINE RANDOM NUMBER GENERATOR

15830
 15831
 15832
 15833
 15834
 15835
 15836
 15837
 15838
 15839
 15840
 15841 065470
 15842 065476 013700 002572
 15843 065502 013701 002570
 15844 065506 012702 000007
 15845 065512 006300
 15846 065514 006101
 15847 065516 077203
 15848 065520 063700 002572
 15849 065524 005501
 15850 065526 063701 002570
 15851 065532 062700 001057
 15852 065536 005501
 15853 065540 062701 047401
 15854 065544 010037 002572
 15855 065550 010137 002570
 15856 065554
 15857 065562 000207

.SBTTL ROUTINE RANDOM NUMBER GENERATOR

```

*****
: THIS ROUTINE IS A DOUBLE PRECISION PSEUDO RANDOM NUMBER GENERATOR
: WITH A RANGE OF 0 TO 2**(+33)-1.
: CALL:
: *   CALL   $RAND           :: CALL THE ROUTINE
: *   RETURN          :: RETURN HERE THE RANDOM
: *                               :: NUMBER WILL BE IN
: *                               :: $HINUM,$LONUM
:
$RAND:  PUSH   R0,R1,R2
        MOV   SEEDLO,R0      : SET R0 WITH LOW
        MOV   SEEDHI,R1      : SET R1 WITH HIGH
        MOV   #7,R2          : SET SHIFT COUNT
1$:     ASL   R0              :: SHIFT R0 LEFT AND
        ROL   R1              :: ROTATE CARRY INTO R1 AND
        SOB   R2,1$
        ADD   SEEDLO,R0      : ADD NUMBER TO MAKE X 129
        ADC   R1              :: PROPOGATE CARRY
        ADD   SEEDHI,R1      : ADD NUMBER TO MAKE X 129
        ADD   #1057,R0       :: ADD LOW CONSTANT
        ADC   R1              :: PROPOGATE CARRY
        ADD   #47401,R1      : ADD HIGH CONSTANT
        MOV   R0,SEEDLO      : SAVE R0
        MOV   R1,SEEDHI      : SAVE R1
        POP   R2,R1,R0
        RETURN

```


CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 491
 ROUTINE DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT

```

15860                                     .SBTTL  ROUTINE DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT
15861                                     :*****
15862                                     :*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
15863                                     :*UNSIGNED OCTAL ASCII NUMBER.
15864                                     :*CALL
15865                                     :*      MOV      #PNTR,-(SP)      ;; POINTER TO LOW WORD OF BINARY NUMBER
15866                                     :*      CALL    $DB20           ;; CALL THE ROUTINE
15867                                     :*      RETURN                    ;; THE ADDRESS OF THE FIRST ASCII CHAR. IS ON THE STACK
15868
15869
15870 065564 104415      $DB20: SAVREG      ;; SAVE ALL REGISTERS
15871 065566 016601      MOV      2(SP),R1      ;; PICKUP THE POINTER TO LOW WORD
15872 065572 012705      MOV      #SOCTVL+13.,R5      ;; POINTER TO DATA TABLE
15873 065576 012704      MOV      #12.,R4           ;; DO ELEVEN CHARACTERS
15874 065602 012703      MOV      #*C7,R3           ;; MASK
15875 065606 012100      MOV      (R1)+,R0          ;; LOWER WORD
15876 065610 012101      MOV      (R1)+,R1          ;; HIGH WORD
15877 065612 005002      CLR      R2              ;; TERMINATOR
15878 065614 110245      1$:  MOVB    R2,-(R5)      ;; PUT CHARACTER IN DATA TABLE
15879 065616 010002      MOV      R0,R2           ;; GET THIS DIGIT
15880 065620 005304      DEC      R4              ;; COUNT THIS CHARACTER
15881 065622 003007      BGT     3$              ;; BR IF NOT THE LAST DIGIT
15882 065624 001405      BEQ     2$              ;; BR IF IT IS THE LAST DIGIT
15883 065626 005205      INC     R5              ;; ALL DIGITS DONE-ADJUST POINTER FOR FIRST
15884 065630 010566      MOV     R5,2(SP)        ;; ASCII CHAR. & PUT IT ON THE STACK
15885 065634 104416      RESREG                    ;; RESTORE ALL REGISTERS
15886 065636 000207      RETURN                    ;; RETURN TO USER
15887 065640 006203      2$:  ASR     R3              ;; POSITION THE MASK FOR THE LAST DIGIT
15888 065642 006001      3$:  ROR     R1              ;; POSITION THE BINARY NUMBER FOR
15889 065644 006000      ROR     R0              ;;
15890 065646 006001      RCR     R1              ;; THE NEXT OCTAL DIGIT
15891 065650 006000      ROR     R0
15892 065652 006001      ROR     R1
15893 065654 006000      ROR     R0
15894 065656 040302      BIC     R3,R2           ;; MASK OUT ALL JUNK
15895 065660 062702      ADD     #'0,R2          ;; MAKE THIS CHAR. ASCII
15896 065664 000753      BR      1$              ;; GO PUT IT IN THE DATA TABLE
15897 065666 000016      $OCTVL: .REPT 14.      ;; RESERVE DATA TABLE
15900 065672      $OCT8=$OCTVL+4        ;; POINTER TO 11 DIGIT NUMBER

```

TABLES

15902			.SBTTL	TABLES	
15903					
15904			.SBTTL	APT MAILBOX-ETABLE	
15905	065704		\$MAIL:		
15906	065704	000000	\$MSGTY:	.WORD 0	::MESSAGE TYPE CODE
15907	065706	000000	\$FATAL:	.WORD 0	::FATAL ERROR NUMBER (ERROR PC)
15908	065710	000000	\$TESTN:	.WORD 0	::TEST PATTERN NUMBER
15909	065712	000000	\$PASS:	.WORD 0	::PASS COUNT
15910	065714	000000	\$DEVCT:	.WORD 0	::DEVICE COUNT
15911	065716	000000	\$UNIT:	.WORD 0	::I/O UNIT NUMBER
15912	065720	000000	\$MSGAD:	.WORD 0	::MESSAGE ADDRESS
15913	065722	000000	\$MSGLG:	.WORD 0	::MESSAGE LENGTH
15914	065724		\$ETABLE:		::APT ENVIRONMENT TABLE
15915	065724	000	\$ENV:	.BYTE 0	::ENVIRONMENT BYTE ;SET TO A 1 FOR APT AUTO MODE
15916			;NOTE:	IF BIT #7 IS SET IN \$ENVM THE TABLE BELOW (BEGINNING AT \$MAMS1 AND	
15917			:	ENDING AT \$MADR4) MUST BE FILLED IN TO INDICATE THE PROPER AMOUNT OF	
15918			:	EACH TYPE OF MEMORY.	
15919	065725	000	\$ENVM:	.BYTE 0	::ENVIRONMENT MODE
15920					::BIT7(200)=USE APT SIZE INFO ;BIT5(40)=NO CONSOLE
15921	065726	000101	\$SWREG:	.WORD 101	::APT SWITCH REGISTER
15922	065730	000000	\$USWR:	.WORD 0	::USED TO LIMIT THE NUMBER OF PASSES
15923	065732	000000	\$CPUOP:	.WORD 0	::CPU TYPE,OPTIONS
15924			:		BITS 15-11=CPU TYPE
15925			:		11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
15926			:		11/70=06,PDQ=07,Q=10
15927			:		BIT 10=REAL TIME CLOCK
15928			:		BIT 9=FLOATING POINT PROCESSOR
15929			:		BIT 8=MEMORY MANAGEMENT
15930	065734	001	\$MAMS1:	.BYTE 1	::HIGH ADDRESS,M.S. BYTE ;DEFAULT - 54K
15931	065735	004	\$MTYP1:	.BYTE 4	::MEM. TYPE,BLK#1
15932			:		MEM.TYPE BYTE -- (HIGH BYTE)
15933			:		900 NSEC CORE=001
15934			:		300 NSEC BIPOLAR=002
15935			:		PARITY MOS=003
15936			:		ERROR CORRECTING MOS=004
15937	065736	177776	\$MADR1:	.WORD 177776	::HIGH ADDRESS,BLK#1
15938			:		MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
15939	065740	000	\$MAMS2:	.BYTE 0	::HIGH ADDRESS,M.S. BYTE
15940	065741	000	\$MTYP2:	.BYTE 0	::MEM. TYPE,BLK#2
15941	065742	000000	\$MADR2:	.WORD 0	::MEM.LAST ADDRESS,BLK#2
15942	065744	000	\$MAMS3:	.BYTE 0	::HIGH ADDRESS,M.S.BYTE
15943	065745	000	\$MTYP3:	.BYTE 0	::MEM. TYPE,BLK#3
15944	065746	000000	\$MADR3:	.WORD 0	::MEM.LAST ADDRESS,BLK#3
15945	065750	000	\$MAMS4:	.BYTE 0	::HIGH ADDRESS,M.S.BYTE
15946	065751	000	\$MTYP4:	.BYTE 0	::MEM. TYPE,BLK#4
15947	065752	000000	\$MADR4:	.WORD 0	::MEM.LAST ADDRESS,BLK#4
15948	065754	000000	\$VECT1:	.WORD 0	::INTERRUPT VECTOR#1,BUS PRIORITY#1
15949	065756	000000	\$VECT2:	.WORD 0	::INTERRUPT VECTOR#2BUS PRIORITY#2
15950	065760	000000	\$BASE:	.WORD 0	::BASE ADDRESS OF EQUIPMENT UNDER TEST
15951	065762	000000	\$DEVN:	.WORD 0	::DEVICE MAP
15952					
15953	065764	000000	\$CDW1:	.WORD 0	
15954	065766	000000	\$CDW2:	.WORD 0	

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 493
APT MAILBOX-ETABLE

```

15956 ;THE FOLLOWING LOCATIONS SPECIFY WHICH PATTERNS
15957 ;ARE TO BE RUN FOR PARTICULAR MEMORIES
15958 ;
15959 ;REFERENCE THE TABLE LISTED BELOW TO RELATE BITS TO PATTERNS.
15960 ;BIT0 SET WILL RUN THE FIRST ENTRY IN THE TABLE, BIT0 SET
15961 ;IN THE SECOND WORD WILL RUN THE 17TH ENTRY IN THE TABLE ...
15962 ;
15963 ;NOTE** NULL TESTS DO NOT TAKE ANY TIME
15964 ;
15965 065770 177777 ;FIELD SERVICE VALUE
15966 065772 177777 ;ECC CSR TESTS 177777 TABLE = MKCSRT:
15967 065774 177777 ;ECC CSR TESTS 177777 TABLE = MKCSRT:
15968 065776 177777 ;ECC PATTERNS 103777 TABLE = MKPAT:
15969 066000 177777 ;ECC PATTERNS 177777 TABLE = MKPAT:
15970 066002 177777 ;PARITY PATTERNS 003777 TABLE = MJPAT:
15974 066004 ;PARITY PATTERNS 177774 TABLE = MJPAT:
15975 ;
15976 ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
15977 ;INTERFACE SPEC.
15978 066004
15979 066004 000000 $APTHD:
15980 066006 065704 $SHIBTS: .WORD 0 ;:TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
15981 066010 000043 $SMBADR: .WORD $MAIL ;:ADDRESS OF APT MAILBOX (BITS 0-15)
15982 066012 001274 $STSTM: .WORD 35. ;:RUN TIME OF LONGEST TEST
15983 066014 000000 $SPASTM: .WORD 700. ;:RUN TIME IN SECS. OF 1ST PASS ON 128K (QUICK VERIFY)
15984 066016 000040 $SUNITM: .WORD 0. ;:EXTRA RUN TIME OF A PASS FOR EACH ADDITIONAL 128K (QV)
;:LENGTH MAILBOX-ETABLE(WORDS)

```


CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 494
 ROUTINE TRAP DECODER

.SBTTL ROUTINE TRAP DECODER

15986
 15987
 15988
 15989
 15990
 15991
 15992
 15993

```

:*****
:*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
:*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
:*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
:*GO TO THAT ROUTINE.

```

15994 066020 010046
 15995 066022 016600 000002
 15996 066026 005740
 15997 066030 111000
 15998 066032 006300
 15999 066034 016000 066062
 16000 066040 000200

```

$TRAP:  MOV    R0,-(SP)      ;;SAVE R0
        MOV    2(SP),R0    ;;GET TRAP ADDRESS
        TST   -(R0)       ;;BACKUP BY 2
        MOVB  (R0),R0     ;;GET RIGHT BYTE OF TRAP
        ASL   R0          ;;POSITION FOR INDEXING
        MOV   $TRPAD(R0),R0 ;;INDEX TO TABLE
        RTS   R0         ;;GO TO ROUTINE

```

16001
 16002
 16003
 16004

```

;;THIS IS USE TO HANDLE THE "GETPRI" MACRO

```

16005 066042 011646
 16006 066044 016666 000004 000002
 16007 066052 000002
 16008
 16009 066054
 16010 066060 000000

```

$TRAP2: MOV   (SP),-(SP)   ;;MOVE THE PC DOWN
        MOV   4(SP),2(SP)  ;;MOVE THE PSW DOWN
        RTI                          ;;RESTORE THE PSW

$NOTRAP:TYPE  MSG006      :UNDEFINED TRAP INSTRUCTION
$HALT2: HALT

```

TRAP TABLE

16013
16014
16015
16016
16017
16018
16019
16020 066062 066042
16021 066064 056206
16022 066066 063154
16023 066070 063130
16024 066072 066054
16025 066074 063356
16026 066076 066054
16027
16028 066100 063756
16029 066102 063602
16030
16031 066104 064322
16032 066106 064452
16033 066110 065036
16034 066112 065206
16035
16036 066114 065374
16037 066116 065432
16038
16039 066120 042656
16040 066122 042666
16041 066124 042676
16042
16043 066126 045112
16044
16045 066130 042706
16046 066132 042732
16047
16048 066134 042750
16049 066136 043044
16050
16051 066140 056542
16052 066142 056570
16053 066144 056616
16054 066146 056646
16055 066150 056730
16056 066152 056752
16057 066154 057002
16058 066156 057022
16059 066160 057044
16060 066162 057064
16061 066164 057106
16062 066166 057130
16063 066170 057150
16064 066172 057166
16065 066174 057204
16066 066176 057224
16067 066200 057242
16068 066202 057260
16069 066204 054016

.SBTTL TRAP TABLE

.*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
.*BY THE "TRAP" INSTRUCTION.

```

:      ROUTINE
:      -----
$TRPAD: .WORD  $TRAP2
$TYPE   :CALL=TYPEIT   TRAP+1(104401) TTY TYPEOUT ROUTINE
$TYPOC  :CALL=TYPOC    TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
$TYPOS  :CALL=TYPOS    TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
$NOTRAP:$TYPON  :CALL=TYPON   TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
$TYPDS  :CALL=TYPDS   TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
$NOTRAP:$TYPBN  :CALL=TYPBN   TRAP+6(104406) TYPE BINARY (ASCII) NUMBER

$GTSWR  :CALL=GTSWR   TRAP+7(104407) GET SOFT-SWR SETTING
$CKSWR  :CALL=CKSWR   TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR

$RDCHR  :CALL=RDCHR   TRAP+11(104411) TTY /PEIN CHARACTER ROUTINE
$RDLIN  :CALL=RDLIN   TRAP+12(104412) TTY TYPEIN STRING ROUTINE
$RDOCT  :CALL=RDOCT   TRAP+13(104413) READ AN OCTAL NUMBER FROM TTY
$RDDEC  :CALL=RDDEC   TRAP+14(104414) READ A DECIMAL NUMBER FROM TTY

$SAVREG :CALL=SAVREG   TRAP+15(104415) SAVE R0-R5 ROUTINE
$RESREG :CALL=RESREG   TRAP+16(104406) RESTORE R0-R5 ROUTINE

$KERNEL :CALL=KERNEL   TRAP+17(104417) ENTER KERNEL MODE
$ENERGIZ:CALL=ENERGIZ  TRAP+20(104420) TURN ON MEMORY MANAGEMENT & TRAPS
$DEENERGI:CALL=DEENERGI TRAP+21(104421) TURN OFF MEMORY MANAGEMENT & TRAPS

$KMAP   :CALL=KMAP     TRAP+22(104422) MAP KERNEL 1 TO 1

$CACHN  :CALL=CACHON   TRAP+23(104423) TURN CACHE ON
$CACHF  :CALL=CACHOFF  TRAP+24(104424) TURN CACHE OFF

$LOADC  :CALL=LOADCSR  TRAP+25(104425) LOAD CORRECT CSR
$READC  :CALL=READCSR  TRAP+26(104426) READ CORRECT CSR

$PER01  :CALL=PERR01   TRAP+27(104427) PROGRAM DETECTED ERROR
$PER02  :CALL=PERR02   TRAP+30(104430) PROGRAM DETECTED ERROR
$PER03  :CALL=PERR03   TRAP+31(104431) PROGRAM DETECTED ERROR
$PER04  :CALL=PERR04   TRAP+32(104432) PROGRAM DETECTED ERROR
$PER07  :CALL=PERR07   TRAP+33(104433) PROGRAM DETECTED ERROR
$PER10  :CALL=PERR10   TRAP+34(104434) PROGRAM DETECTED ERROR
$PER11  :CALL=PERR11   TRAP+35(104435) PROGRAM DETECTED ERROR
$PER12  :CALL=PERR12   TRAP+36(104436) PROGRAM DETECTED ERROR
$PER13  :CALL=PERR13   TRAP+37(104437) PROGRAM DETECTED ERROR
$PER14  :CALL=PERR14   TRAP+40(104440) PROGRAM DETECTED ERROR
$PER15  :CALL=PERR15   TRAP+41(104441) PROGRAM DETECTED ERROR
$PER16  :CALL=PERR16   TRAP+42(104442) PROGRAM DETECTED ERROR
$PER17  :CALL=PERR17   TRAP+43(104443) PROGRAM DETECTED ERROR
$PER20  :CALL=PERR20   TRAP+44(104444) PROGRAM DETECTED ERROR
$PER21  :CALL=PERR21   TRAP+45(104445) PROGRAM DETECTED ERROR
$PER22  :CALL=PERR22   TRAP+46(104446) PROGRAM DETECTED ERROR
$PER23  :CALL=PERR23   TRAP+47(104447) PROGRAM DETECTED ERROR
$PER24  :CALL=PERR24   TRAP+50(104450) PROGRAM DETECTED ERROR
$PER25  :CALL=PERR25   TRAP+51(104451) PROGRAM DETECTED ERROR

```


TRAP TABLE

16070	066206	057450	\$PER26	:CALL=PERR26	TRAP+52(104452)	PROGRAM DETECTED ERROR	
16071	066210	057470	\$PER27	:CALL=PERR27	TRAP+53(104453)	PROGRAM DETECTED ERROR	
16072	066212	054244	\$PER30	:CALL=PERR30	TRAP+54(104454)	PROGRAM DETECTED ERROR	
16073	066214	057660	\$PER31	:CALL=PERR31	TRAP+55(104455)	PROGRAM DETECTED ERROR	
16074	066216	057756	\$PER32	:CALL=PERR32	TRAP+56(104456)	PROGRAM DETECTED ERROR	
16075	066220	060024	\$PER33	:CALL=PERR33	TRAP+57(104457)	PROGRAM DETECTED ERROR	
16076	066222	060104	\$PER34	:CALL=PERR34	TRAP+60(104460)	PROGRAM DETECTED ERROR	
16077	066224	060136	\$PER35	:CALL=PERR35	TRAP+61(104461)	PROGRAM DETECTED ERROR	
16078	066226	060172	\$PER36	:CALL=PERR36	TRAP+62(104462)	PROGRAM DETECTED ERROR	
16079	066230	060222	\$PER37	:CALL=PERR37	TRAP+63(104463)	PROGRAM DETECTED ERROR	::ILC:REV B
16080	066232	060226	\$PER40	:CALL=PERR40	TRAP+64(104464)	PROGRAM DETECTED ERROR	::ILC:REV B
16081	066234	066054	\$NOTRAP	:CALL=PERR41	TRAP+65(104465)	PROGRAM DETECTED ERROR	
16082	066236	066054	\$NOTRAP	:CALL=PERR42	TRAP+66(104466)	PROGRAM DETECTED ERROR	
16083	066240	066054	\$NOTRAP	:CALL=PERR43	TRAP+67(104467)	PROGRAM DETECTED ERROR	
16084							
16085	066242	043270	\$ECCDIS	:CALL=ECCDIS	TRAP+70(104470)	DISABLE ECC ON ALL CSR'S	
16086	066244	043304	\$ECC1DIS	:CALL=ECC1DIS	TRAP+71(104471)	DISABLE ECC ON 1 SELECTED CSR	
16087	066246	043316	\$ECCINIT	:CALL=ECCINIT	TRAP+72(104472)	INITIALIZE ALL MK11 CSR'S	
16088	066250	043332	\$ECC1INIT	:CALL=ECC1INIT	TRAP+73(104473)	INITIALIZE 1 SELECTED MK11 CSR	
16089	066252	043372	\$CBCSR	:CALL=CBCSR	TRAP+74(104474)	WRITE GENERATED CHECKBITS IN ALL CSR'S	
16090	066254	043414	\$CB1CSR	:CALL=CB1CSR	TRAP+75(104475)	WRITE GENERATED CHECKBITS IN 1 SELECTED CSR	
16091	066256	043434	\$WASSBE	:CALL=WASSBE	TRAP+76(104476)	WAS THERE A SBE ON ANY CSR?	
16092	066260	043550	\$WAS1SBE	:CALL=WAS1SBE	TRAP+77(104477)	WAS THERE A SBE ON 1 SELECTED CSR?	
16093	066262	043600	\$WASDBE	:CALL=WASDBE	TRAP+100(104500)	WAS THERE A DBE ON ANY CSR?	
16094	066264	043714	\$WAS1DBE	:CALL=WAS1DBE	TRAP+101(104501)	WAS THERE A DBE ON 1 SELECTED CSR?	
16095	066266	043744	\$CLRCSR	:CALL=CLRCSR	TRAP+102(104502)	CLEAR ALL CSR'S	
16096	066270	043756	\$CLR1CSR	:CALL=CLR1CSR	TRAP+103(104503)	CLEAR 1 SELECTED CSR	
16097	066272	043766	\$CHKDIS	:CALL=CHKDIS	TRAP+104(104504)	DISABLE ECC & WRITE CKBITS FROM ALL CSR'S	
16098	066274	044002	\$CHK1DIS	:CALL=CHK1DIS	TRAP+105(104505)	DISABLE ECC & WRITE CKBITS FROM 1 CSR	
16099	066276	043344	\$ENASBE	:CALL=ENASBE	TRAP+106(104506)	ENABLE TRAPS ON SBE'S FROM ALL CSR'S	
16100	066300	043360	\$ENA1SBE	:CALL=ENA1SBE	TRAP+107(104507)	ENABLE TRAPS ON SBE'S FROM 1 SELECTED CSR	
16101	066302	043064	\$TSTRD	:CALL=TSTREAD	TRAP+110(104510)	TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)	
16102	066304	044062	\$INVALID	:CALL=INVALID	TRAP+111(104511)	INVALIDATE BACKGROUND PATTERN ON BANK	
16103	066306	044112	\$ERRGEN	:CALL=ERRGEN	TRAP+112(104512)	TEST ERROR ADDRESS	
16104	066310	044362	\$CBREG	:CALL=CBREG	TRAP+112(104513)	ENABLE CHECK/SYNDROME BIT REGISTER	
16105	066312	066054	\$NOTRAP				
16106	066314	066054	\$NOTRAP				
16107	066316	066054	\$NOTRAP				
16108	066320	066054	\$NOTRAP				
16109	066322	066054	\$NOTRAP				
16110	066324	066054	\$NOTRAP				

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 498
TRAP TABLE

16113 177776 ST = 177776 ;STATUS REGISTER

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 500
 TABLE ERROR POINTER

16116
 16117
 16118
 16119
 16120
 16121
 16122
 16123
 16124
 16125
 16126
 16127
 16128
 16129
 16130 066326
 16131 066326 071014
 16132 066330 073151
 16133 066332 067352
 16134 066334 067733
 16135
 16136 066336 070001
 16137 066340 072460
 16138 066342 067176
 16139 066344 067611
 16140
 16141 066346 070037
 16142 066350 072540
 16143 066352 067214
 16144 066354 067726
 16145
 16146 066356 070071
 16147 066360 072540
 16148 066362 067224
 16149 066364 067726
 16150
 16151 066366 070137
 16152 066370 072574
 16153 066372 067234
 16154 066374 067611
 16155
 16156 066376 070214
 16157 066400 072574
 16158 066402 067234
 16159 066404 067611
 16160
 16161 066406 070241
 16162 066410 072574
 16163 066412 067234
 16164 066414 067611
 16165
 16166 066416 072225
 16167 066420 073673
 16168 066422 067530
 16169 066424 067611

.SBTTL TABLE ERROR POINTER

;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
 ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
 ;*LOCATION \$ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
 ;*NOTE1: IF \$ITEMB IS 0 THE ONLY PERTINENT DATA IS (ERRPC).
 ;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:

;* EM ;;POINTS TO THE ERROR MESSAGE
 ;* DH ;;POINTS TO THE DATA HEADER
 ;* DT ;;POINTS TO THE DATA
 ;* DF ;;POINTS TO THE DATA FORMAT

\$ERRTB: ;ERROR 1
 EM24
 DH13
 DT13
 DF11
 ;ERROR 2
 EM2
 DH1
 DT1
 DF2
 ;ERROR 3
 EM3
 DH3
 DT3
 DF9
 ;ERROR 4
 EM4
 DH3
 DT4
 DF9
 ;ERROR 5
 EM5
 DH5
 DT5
 DF2
 ;ERROR 6
 EM6
 DH5
 DT5
 DF2
 ;ERROR 7
 EM7
 DH5
 DT5
 DF2
 ;ERROR 10
 EM53
 DH25
 DT25
 DF2

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 502
 TABLE ERROR POINTER

16172			:ERROR	11
16173	066426	070301	EM11	
16174	066430	072720	DH7	
16175	066432	067266	DT7	
16176	066434	067635	DF3	
16177			:ERROR	12
16178	066436	070301	EM11	
16179	066440	072720	DH7	
16180	066442	067266	DT7	
16181	066444	067650	DF4	
16182			:ERROR	13
16183	066446	070323	EM12	
16184	066450	073030	DH10	
16185	066452	067316	DT10	
16186	066454	067611	DF2	
16187			:ERROR	14
16188	066456	070301	EM11	
16189	066460	072720	DH7	
16190	066462	067266	DT7	
16191	066464	067663	DF5	
16192			:ERROR	15
16193	066466	070301	EM11	
16194	066470	072720	DH7	
16195	066472	067266	DT7	
16196	066474	067676	DF6	
16197			:ERROR	16
16198	066476	070347	EM13	
16199	066500	073151	DH13	
16200	066502	067352	DT13	
16201	066504	067733	DF11	
16202			:ERROR	17
16203	066506	070401	EM14	
16204	066510	073151	DH13	
16205	066512	067352	DT13	
16206	066514	067733	DF11	
16207			:ERROR	20
16208	066516	070445	EM15	
16209	066520	073151	DH13	
16210	066522	067352	DT13	
16211	066524	067733	DF11	
16212			:ERROR	21
16213	066526	072254	EM55	
16214	066530	073731	DH26	
16215	066532	067542	DT26	
16216	066534	067611	DF2	
16217			:ERROR	22
16218	066536	070513	EM17	
16219	066540	072720	DH7	
16220	066542	067266	DT7	
16221	066544	067663	DF5	
16222			:ERROR	23
16223	066546	072074	EM50	
16224	066550	073545	DH23	
16225	066552	067466	DT23	
16226	066554	067744	DF13	

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 504
 TABLE ERROR POINTER

16229			:ERROR	24	
16230	066556	070553	EM19		
16231	066560	073151	DH13		
16232	066562	067352	DT13		
16233	066564	067733	DF11		
16234			:ERROR	25	
16235	066566	070625	EM20		
16236	066570	073151	DH13		
16237	066572	067352	DT13		
16238	066574	067733	DF11		
16239			:ERROR	26	
16240	066576	000000	0		:NO MESSAGE
16241	066600	073144	DH12		
16242	066602	067346	DT12		
16243	066604	067611	DF2		
16244			:ERROR	27	
16245	066606	070704	EM21		
16246	066610	073126	DH11		
16247	066612	067340	DT11		
16248	066614	067611	DF2		
16249			:ERROR	30	
16250	066616	070740	EM22		
16251	066620	073151	DH13		
16252	066622	067352	DT13		
16253	066624	067733	DF11		
16254			:ERROR	31	
16255	066626	000000	0		:NO MESSAGE
16256	066630	073246	DH14		
16257	066632	067374	DT14		
16258	066634	067611	DF2		
16259			:ERROR	32	
16260	066636	070765	EM23		
16261	066640	072574	DH5		
16262	066642	067234	DT5		
16263	066644	067611	DF2		
16264			:ERROR	33	
16265	066646	071073	EM25		
16266	066650	073325	DH15		
16267	066652	067412	DT16		
16268	066654	067711	DF7		
16269			:ERROR	34	
16270	066656	071120	EM26		
16271	066660	073444	DH16		
16272	066662	067442	DT17		
16273	066664	067635	DF3		

CZMSP80 MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 506
 TABLE ERROR POINTER

16276			:ERROR	35
16277	066666	072200	EM52	
16278	066670	073673	DH25	
16279	066672	067530	DT25	
16280	066674	067611	DF2	
16281			:ERROR	36
16282	066676	071171	EM27	
16283	066700	073444	DH16	
16284	066702	067442	DT17	
16285	066704	067724	DF8	
16286			:ERROR	37
16287	066706	071666	EM35	
16288	066710	072720	DH7	
16289	066712	067266	DT7	
16290	066714	067635	DF3	
16291			:ERROR	40
16292	066716	071261	EM29	
16293	066720	072720	DH7	
16294	066722	067266	DT7	
16295	066724	067635	DF3	
16296			:ERROR	41
16297	066726	071343	EM30	
16298	066730	072720	DH7	
16299	066732	067266	DT7	
16300	066734	067663	DF5	
16301			:ERROR	42
16302	066736	072375	EM60	
16303	066740	073466	DH20	
16304	066742	067466	DT23	
16305	066744	067744	DF13	
16306			:ERROR	43
16307	066746	071453	EM32	
16308	066750	072720	DH7	
16309	066752	067266	DT7	
16310	066754	067635	DF3	
16311			:ERROR	44
16312	066756	071560	EM33	
16313	066760	072720	DH7	
16314	066762	067266	DT7	
16315	066764	067635	DF3	
16316			:ERROR	45
16317	066766	072130	EM51	
16318	066770	073624	DH24	
16319	066772	067510	DT24	
16320	066774	067754	DF14	
16321			:ERROR	46
16322	066776	071753	EM36	
16323	067000	072653	DH6	
16324	067002	067252	DT6	
16325	067004	067611	DF2	

TABLE ERROR POINTER

16328			:ERROR	47
16329	067006	072022	EM40	
16330	067010	072515	DH2	
16331	067012	067450	DT20	
16332	067014	067611	DF2	
16333			:ERROR	50
16334	067016	072275	EM56	
16335	067020	073747	DH27	
16336	067022	067550	DT27	
16337	067024	067610	DF1	
16338			:ERROR	51
16339	067026	072437	EM61	
16340	067030	073624	DH24	
16341	067032	067510	DT24	
16342	067034	067754	DF14	
16343			:ERROR	52
16344	067036	071261	EM29	
16345	067040	073151	DH13	
16346	067042	067352	DT13	
16347	067044	067733	DF11	
16348			:ERROR	53
16349	067046	071073	EM25	
16350	067050	074023	DH30	
16351	067052	067570	DT30	
16352	067054	067772	DF16	
16353			:ERROR	54
16354	067056	072327	EM57	
16355	067060	074023	DH30	
16356	067062	067570	DT30	
16357	067064	067772	DF16	
16358			:ERROR	55
16359	067066	070625	EM20	
16360	067070	073624	DH24	
16361	067072	067510	DT24	
16362	067074	067754	DF14	
16363			:ERROR	56
16364	067076	070625	EM20	
16365	067100	074023	DH30	
16366	067102	067570	DT30	
16367	067104	067772	DF16	
16368			:ERROR	57
16369	067106	070553	EM19	
16370	067110	073624	DH24	
16371	067112	067510	DT24	
16372	067114	067754	DF14	
16373			:ERROR	60
16374	067116	070347	EM13	
16375	067120	073466	DH20	
16376	067122	067466	DT23	
16377	067124	067744	DF13	
16378			:ERROR	61
16379	067126	071343	EM30	
16380	067130	073466	DH20	
16381	067132	067466	DT23	
16382	067134	067744	DF13	
16383			:ERROR	62
16384	067136	070625	EM20	

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 508-1
TABLE ERROR POINTER

16385	067140	073624	DH24	
16386	067142	067510	DT24	
16387	067144	067754	DF14	
16388			:ERROR	63
16389	067146	070740	EM22	
16390	067150	073624	DH24	
16391	067152	067510	DT24	
16392	067154	067754	DF14	
16393			:ERROR	64
16394	067156	100344	EM62	
16395	067160	072574	DH5	
16396	067162	067234	DT5	
16397	067164	067611	DF2	
16398			:ERROR	65
16399	067166	070740	EM22	
16400	067170	073466	DH20	
16401	067172	067466	DT23	
16402	067174	067744	DF13	

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 509
 ERROR DATA TAGS (DT)

Address	DT	Field 1	Field 2	Field 3	Field 4	Field 5	Field 6	Field 7	Field 8
16404							.SBTTL	ERROR DATA TAGS (DT)	
16405	067176	002016	002032	002042	DT1:	.WORD	ERRPC, ADDRESS, GOOD, BAD, 0		
	067204	002050	000000						
16406	067210	002016	000000		DT2:	.WORD	ERRPC, 0		
16407	067214	002016	002034	002070	DT3:	.WORD	ERRPC, PADDRESS, PARCNT, 0		
	067222	000000							
16408	067224	002016	002032	002066	DT4:	.WORD	ERRPC, ADDRESS, NEMCNT, 0		
	067232	000000							
16409	067234	002016	177572	177574	DT5:	.WORD	ERRPC, MMRO, MMR1, MMR2, MMR3, CPUERR, 0		
	067242	177576	172516	177766					
	067250	000000							
16410	067252	002016	002420	002376	DT6:	.WORD	ERRPC, APTPAR, LSIZE, APTECC, MSIZE, 0		
	067260	002422	002400	000000					
16411	067266	002016	002174	002032	DT7:	.WORD	ERRPC, DUMMY, ADDRESS, DUMMY, GOOD, BAD, BADXOR		
	067274	002174	002042	002050					
	067302	002056							
16412	067304	002174	002174	002174		.WORD	DUMMY, DUMMY, DUMMY, DUMMY, 0		
	067312	002174	000000						
16413	067316	002176	002200	002202	DT10:	.WORD	DETRO, DETR1, DETR2, DETR3, DETR4, DETR5, DETSP, DETPSW, 0		
	067324	002204	002206	002210					
	067332	002212	002214	000000					
16414	067340	002016	002146	000000	DT11:	.WORD	ERRPC, CSR, 0		
16415	067346	002146	000000		DT12:	.WORD	CSR, 0		
16416	067352	002016	002174	002032	DT13:	.WORD	ERRPC, DUMMY, ADDRESS, DUMMY, TSTDAT, TSTDAT+2, CHECK, CSR, 0		
	067360	002174	002244	002246					
	067366	002312	002146	000000					
16417	067374	177746	177572	177574	DT14:	.WORD	CONTRL, MMRO, MMR1, MMR2, MMR3, CPUERR, 0		
	067402	177576	172516	177766					
	067410	000000							
16418	067412	002016	002174	002174	DT16:	.WORD	ERRPC, DUMMY, DUMMY, GOOD, GOOD2, GOOD3		
	067420	002042	002044	002046					
16419	067426	002050	002052	002054		.WORD	BAD, BAD2, BAD3, DUMMY, DUMMY, 0		
	067434	002174	002174	000000					
16420	067442	002016	002174	000000	DT17:	.WORD	ERRPC, DUMMY, 0		
16421	067450	002016	002042	002050	DT20:	.WORD	ERRPC, GOOD, BAD, 0		
	067456	000000							
16422	067460	002016	002174	000000	DT22:	.WORD	ERRPC, DUMMY, 0		
16423	067466	002016	002174	002042	DT23:	.WORD	ERRPC, DUMMY, GOOD, BAD, DUMMY, DUMMY, DUMMY, DUMMY, 0		
	067474	002050	002174	002174					
	067502	002174	002174	000000					
16424	067510	002016	002174	002146	DT24:	.WORD	ERRPC, DUMMY, CSR, DUMMY, DUMMY, DUMMY, DUMMY, 0		
	067516	002174	002174	002174					
	067524	002174	000000						
16425	067530	002016	002042	002146	DT25:	.WORD	ERRPC, GOOD, CSR, CSRNO, 0		
	067536	002150	000000						
16426	067542	002016	002050	000000	DT26:	.WORD	ERRPC, BAD, 0		
16427	067550	002016	002174	002032	DT27:	.WORD	ERRPC, DUMMY, ADDRESS, DUMMY, DUMMY, DUMMY, DUMMY, 0		
	067556	002174	002174	002174					
	067564	002174	000000						
16428	067570	002016	002174	002174	DT30:	.WORD	ERRPC, DUMMY, DUMMY, GOOD, BAD, CSR, DUMMY, 0		
	067576	002042	002050	002146					
	067604	002174	000000						

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 517
MESSAGES

16521					.SBTTL	MESSAGES	
16522	074102	200	040	040	MSG001:	.ASCIZ <CRLF>/	MEMORY CONFIGURATION MAP/
16523	074164	200	040	040	MSG002:	.ASCIZ <CRLF>/	16K WORD BANKS/
16524	074241	200	040	040	MSG003:	.ASCII <CRLF>/	1 2 3/
16525	074303	040	040	040		.ASCIZ /	4 5 6 7 /
16526	074346	200	040	040	MSG004:	.ASCII <CRLF>/	012345670123456701234567/
16527	074407	060	061	062		.ASCIZ /012345670123456701234567012345670123/	
16528	074454	200	105	122	MSG005:	.ASCIZ <CRLF>/ERRORS /	
16529	074466	200	125	116	MSG006:	.ASCIZ <CRLF>/UNDEFINED TRAP INSTRUCTION/<32>	
16530	074523	200	111	116	MSG007:	.ASCIZ <CRLF>/INTRLV /	:INTERLEAVED CSR #
16531	074535	200	115	105	MSG009:	.ASCIZ <CRLF>/MEMTYPE /	:MEMORY TYPE
16532	074547	200	120	122	MSG010:	.ASCIZ <CRLF>/PROTECT /	:MEMORY PROTECTED
16533	074561	040	040	040	MSG011:	.ASCIZ /	0 1 2 3 4 5 6/
16534	074647	064	065	066	MSG012:	.ASCIZ /4567012345670123456701234567012345670123456701234567/	
16535	074744	130	000		MSG013:	.ASCIZ /X/	
16536	074746	040	000		MSG014:	.ASCIZ / /	:SPACE
16537	074750	000	000		MSG015:	.BYTE 0,0	:FOR SINGLE ASCII CHARACTERS & TERMINATOR
16538	074752	200	103	123	MSG016:	.ASCIZ <CRLF>/CSR /	
16539	074764	040	040	040	MSG017:	.ASCIZ / / /	:8 SPACES
16540	074775	040	040	000	MSG018:	.ASCIZ / / /	:2 SPACES
16541	075000	040	040	040	MSG019:	.ASCIZ / / /	:3 SPACES
16542	075004	200	106	123	MSG020:	.ASCIZ <CRLF>/FS COMMAND MODE/	
16543	075025	200	103	117	MSG021:	.ASCII <CRLF>/COMMANDS AVAILABLE:/	
16544	075051	200	060	040		.ASCII <CRLF>/0 = EXIT/	
16545	075062	200	061	040		.ASCII <CRLF>/1 = READ CSR/	
16546	075077	200	062	040		.ASCII <CRLF>/2 = LOAD CSR/	
16547	075114	200	063	040		.ASCII <CRLF>/3 = EXAMINE MEMORY/	
16548	075137	200	064	040		.ASCII <CRLF>/4 = MODIFY MEMORY/	
16549	075161	200	065	040		.ASCII <CRLF>/5 = SELECT BANK & TEST/	
16550	075210	200	066	040		.ASCII <CRLF>/6 = TYPE CONFIG MAP/	
16551	075234	200	067	040		.ASCII <CRLF>/7 = SOB-A-LONG TEST/	
16552	075260	200	070	040		.ASCII <CRLF>/8 = ERROR SUMMARY/	
16553	075302	200	071	075		.ASCII <CRLF>/9= REFRESH TEST/	
16554	075323	200	061	060		.ASCII <CRLF>/10= SET FILL COUNT/	
16555	075346	200	061	061		.ASCII <CRLF>/11= ENTER KAMIKAZE MODE/	
16556	075376	200	061	062		.ASCII <CRLF>/12= EXIT KAMIKAZE MODE/	
16557	075425	200	061	063		.ASCII <CRLF>/13= TURN CACHE OFF/	
16558	075450	200	061	064		.ASCII <CRLF>/14= TURN CACHE ON/	
16559	075472	200	061	065		.ASCII <CRLF>/15= TEST SELECTED BANKS/	
16560	075522	200	061	066		.ASCII <CRLF>/16= TEST ALL BANKS/	
16561	075545	200	061	067		.ASCII <CRLF>/17= ENABLE TRACE/	
16562	075566	200	061	070		.ASCII <CRLF>/18= DISABLE TRACE/	
16563	075610	015	012	000		.BYTE 15,12,0	
16564	075613	200	127	110	MSG022:	.ASCIZ <CRLF>/WHICH CSR(0-F)? /	
16565	075635	200	103	123	MSG023:	.ASCIZ <CRLF>/CSR WORD? /	
16566	075651	200	103	123	MSG025:	.ASCIZ <CRLF>/CSR DOES NOT EXIST/	
16567	075675	200	103	117	MSG026:	.ASCIZ <CRLF>/COMMAND:/	
16568	075707	200	117	114	MSG027:	.ASCIZ <CRLF>/OLD CSR WAC/	
16569	075724	200	103	123	MSG028:	.ASCIZ <CRLF>/CSR IS NOW/	
16570	075740	200	105	130	MSG029:	.ASCIZ <CRLF>/EXAMINE MEMORY/	
16571	075760	200	102	101	MSG030:	.ASCIZ <CRLF>/BANK(0-16)? /	
16572	075777	200	120	110	MSG031:	.ASCIZ <CRLF>/PHYSICAL ADDRESS(0-17757776)? /	
16573	076037	200	120	101	MSG032:	.ASCIZ <CRLF>/PARITY ABORT/<32>	
16574	076056	200	124	111	MSG033:	.ASCIZ <CRLF>/TIMEOUT TRAP/<32>	
16575	076075	200	102	131	MSG034:	.ASCIZ <CRLF>/BYPASSING ECC TESTS ON BANK /	
16576	076133	040	104	125	MSG034:	.ASCIZ / DUE TO SBE LOCATIONS/	
16577	076161	121	126	000	MSG035:	.ASCIZ /QV/	

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 517-1
 MESSAGES

16578	076164	200	115	117	MSG036:	.ASCIZ	<CRLF>/MODIFY MEMORY/
16579	076203	200	117	114	MSG037:	.ASCIZ	<CRLF>/OLD DATA WAS /
16580	076222	200	104	101	MSG038:	.ASCIZ	<CRLF>/DATA IS NOW /
16581	076240	200	111	116	MSG039:	.ASCIZ	<CRLF>/INPUT NEW DATA? /
16582	076262	200	123	105	MSG040:	.ASCIZ	<CRLF>/SELECT BANK & TEST/
16583	076306	200	102	101	MSG041:	.ASCIZ	<CRLF>/BANK NOT ACCESSABLE/
16584	076333	200	124	105	MSG042:	.ASCIZ	<CRLF>/TEST(0-47)? /
16585	076351	200	124	105	MSG043:	.ASCIZ	<CRLF>/TEST 0 DATA IS? /
16586	076373	200	124	117	MSG046:	.ASCIZ	<CRLF>/TO ESCAPE TYPE ANY KEY/<CRLF><12><12>
16587	076426	200	124	105	MSG047:	.ASCIZ	<CRLF>/TEST COMPLETE/
16588	076445	040	116	117	MSG048:	.ASCIZ	/ NOT AVAILABLE NOW - TRY LATER! /
16589	076505	200	102	101	MSG049:	.ASCIZ	<CRLF>/BANK REQUIRES RELOCATION/ .EVEN
16591	076540	120	117	127	MSG051:	.ASCIZ	/POWER RECOVERY/
16592	076557	200	123	117	MSG055:	.ASCIZ	<CRLF>/SOB-A-LONG TEST/
16593	076600	200	102	105	MSG056:	.ASCIZ	<CRLF>/BELL = EACH PASS COMPLETE/
16594	076633	200	040	040	MSG058:	.ASCIZ	<CRLF>/ CSR CSR .../
16595	076655	077	077	077	MSG061:	.ASCIZ	/?????/
16596	076664	111	116	120	MSG062:	.ASCIZ	/INPUT MUST BE A/
16597	076704	116	040	117	MSG063:	.ASCIZ	/N OCTAL /
16598	076715	116	125	115	MSG064:	.ASCIZ	/NUMBER/<CRLF>
16599	076725	040	104	105	MSG065:	.ASCIZ	/ DECIMAL /
16600	076737	200	105	122	MSG066:	.ASCIZ	<CRLF>/ERRORS > 20 - ABORTING FOR XXDP CHAIN/
16601	077006	106	101	124	MSG067:	.ASCIZ	/FATAL /
16602	077015	113	040	127	MSG070:	.ASCIZ	/K WORDS OF MEMORY TOTAL/<CRLF>
16603	077046	200	122	105	MSG073:	.ASCIZ	<CRLF>/REFRESH TEST/
16604	077064	200	122	105	MSG075:	.ASCIZ	<CRLF>/RELOCATION NOT POSSIBLE/<32>
16605	077116	200	040	040	MSG076:	.ASCIZ	<CRLF>/ BANK ERRORS/<CRLF>
16606	077137	200	105	116	MSG077:	.ASCIZ	<CRLF>/END PASS #/
16607	077153	040	105	122	MSG079:	.ASCIZ	/ ERROR(S) DETECTED/<CRLF>
16608	077177	200	106	111	MSG085:	.ASCIZ	<CRLF>/FILL COUNT(OCTAL)? /
16609	077224	200	113	105	MSG088:	.ASCIZ	<CRLF>/KERNEL STACK/
16610	077242	200	123	125	MSG089:	.ASCIZ	<CRLF>/SUPERVISOR STACK/
16611	077264	200	125	123	MSG090:	.ASCIZ	<CRLF>/USER STACK/
16612	077300	040	111	123	MSG091:	.ASCIZ	/ IS EMPTY/
16613	077312	122	105	114	MSG092:	.ASCIZ	/RELOCATED /
16614	077326	102	101	116	MSG093:	.ASCIZ	/BANK=/
16615	077334	040	040	124	MSG095:	.ASCIZ	/ TEST=/
16616	077344	200	105	116	MSG101:	.ASCIZ	<CRLF>/ENTERING KAMIKAZE MODE/
16617	077374	200	114	105	MSG102:	.ASCIZ	<CRLF>/LEAVING KAMIKAZE MODE/
16618	077423	200	114	105	MSG103:	.ASCIZ	<CRLF>/LEAVING FS MODE/<CRLF>
16619	077445	032	000		MSG104:	.BYTE	32,0 ;CONTROL Z
16620	077447	200	105	116	MSG105:	.ASCIZ	<CRLF>/ENTER BANKS - USE NUMBER 200 TO TERMINATE/
16621	077522	200	103	101	MSG106:	.ASCIZ	<CRLF>/CACHE IS OFF/
16622	077540	200	103	101	MSG107:	.ASCIZ	<CRLF>/CACHE IS ON (EXCEPT DURING ACTUAL PATTERNS)/
16623	077615	200	117	116	MSG110:	.ASCIZ	<CRLF>/ONLY SELECTED BANKS WILL BE TESTED/
16624	077661	200	101	114	MSG111:	.ASCIZ	<CRLF>/L BANKS WILL BE TESTED/
16625	077713	113	040	117	MSG112:	.ASCIZ	/K OF MS11-L/<CRLF>
16626	077730	113	040	117	MSG113:	.ASCIZ	/K OF MS11-M/<CRLF>
16627	077745	113	040	117	MSG114:	.ASCIZ	/K OF MS11-P/<CRLF>
16628	077762	200	040	040	MSG117:	.ASCIZ	<CRLF>" 11/44"
16629	077774	200	040	040	MSG119:	.ASCIZ	<CRLF>/ NO/
16630	100003	040	103	101	MSG120:	.ASCIZ	/ CACHE AVAILABLE/
16631	100024	040	103	101	MSG121:	.ASCIZ	/ CACHE BYPASSED/
16632	100044	200	103	123	MSG122:	.ASCII	<CRLF>/CSR NUMBER /
16633	100060	000			MSG122:	.BYTE	0
16634	100061	040	103	117		.ASCIZ	/ CONTROLS TOO MANY BANKS/

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 517-2
 MESSAGES

16635	100112	040	120	101	MSG125: .ASCIZ / PASSES COMPLETED/
16636	100134	200	120	122	MSG126: .ASCIZ <CRLF>/PROGRAM CSR COULD NOT BE DETERMINED/
16637	100201	200	124	122	MSG127: .ASCIZ <CRLF>/TRACE ENABLED/
16638	100220	200	124	122	MSG128: .ASCIZ <CRLF>/TRACE DISABLED/
16639	100240	200	200	040	MSG008: .ASCIZ <CRLF><CRLF>/ CSR MAP/<CRLF>
16640	100272	200	040	103	MSG000: .ASCIZ <CRLF>" CZMSPB MS11-L/M/P MEMORY DIAGNOSTIC"
16641	100341	200	200	000	MSG129: .ASCIZ <CRLF><CRLF>
16642	100344	120	122	117	EM62: .ASCIZ /PROCESSOR NOT SUPPORTED BY THIS DIAGNOSTIC/
16643					.EVEN
16649	100420				\$\$END
16650	100420				END:
16651	100420	001236			.PRINT 60000-SUPLIMIT ;SUPERVISOR ADDRESSES LEFT
16655		000200			.END START3

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 517-3

SYMBOL TABLE

ABORTF	002142	B0	004616	B62	043616	CPUBIT	002104	DH13	073151
ACFLAG	002114	B1	005552	B63	044022	CPUERR=	177766	DH14	073246
ACTFLA	002346	B10	013464	B64	044026	CR	= 000015	DH15	073325
ADDRES	002032	B100	054626	B65	045246	CRLF	= 000200	DH16	073444
ANA2	010004	B101	055124	B66	045254	CSR	002146	DH19	073461
APTDOW	047760	B102	055132	B67	045404	CSRADD=	172100	DH2	072515
APTECC	002422	B103	062574	B7	013422	CSRCAS	020176	DH20	073466
APTFLA	002350	B104	062600	B70	045420	CSRFBA	002230	DH23	073545
APTHAN	015320	B105	062676	B71	052550	CSRFIR	002224	DH24	073624
APHTLT	050026	B106	062770	B72	053006	CSRHOL	002524	DH25	073673
APTPAR	002420	B107	063054	B73	053270	CSRINC	002326	DH26	073731
APTSIZ	002442	B11	014216	B74	053564	CSRINF	002460	DH27	073747
BACK	060602	B12	014304	B75	053564	CSRINT	002234	DH3	072540
BACKGN	041236	B13	014346	B76	053662	CSRLAS	002226	DH30	074023
BAD	002050	B14	014572	B77	054622	CSRLBA	002232	DH5	072574
BADPC	002020	B15	015364	CACHKF	002546	CSRLOO	002330	DH6	072653
BADPSW	002030	B16	017002	CACHKN	002542	CSRMAP	005776	DH7	072720
BADSP	002024	B17	017252	CACHOF=	104424	CSRNO	002150	DIAGFL	002002
BADSTA	042626	B2	006012	CACHON=	104423	CSROUT	044014	DISPLA	002626
BADXOR	002056	B20	017260	CACHVE=	000114	CSRSTU=	000021	DISPRE=	000174
BAD2	002052	B21	017276	CBCSR =	104474	CSRIS	002320	DISPTB	014772
BAD3	002054	B22	017340	CBITS	002314	CTEST	006660	DOBACK	015354
BAFPAF	015454	B23	017412	CBREG =	104513	CTLKVE	002144	DONE	006644
BAFPAR	015562	B24	024172	CB1CSR=	104475	DATARG=	177754	DSWR	= 177570
BAKPAT	002620	B25	024176	CHECK	002312	DATBUF	002240	DT1	067176
BANK	002100	B26	024376	CHKDIS=	104504	DBEMSK	002254	DT10	067316
BANKIN	002102	B27	024404	CHKGEN	044400	DDISP =	177570	DT11	067340
BANKMO	046422	B3	006062	CHKTAB	044506	DEENER=	104421	DT12	067346
BANKOK	047460	B30	024702	CHKTRP	040626	DETAIL	062340	DT13	067352
BAWPAF	015670	B31	032414	CHK1DI=	104505	DETFLA	002216	DT14	067374
BAWPAR	016020	B32	034132	CKEND	064114	DETPSW	002214	DT16	067412
BGTEST	036250	B33	034170	CKSWR =	104410	DETRO	002176	DT17	067442
BITNO	002322	B34	034612	CLRCR=	104502	DETR1	002200	DT2	067210
BIT0 =	000001	B35	034650	CLREX	007754	DETR2	002202	DT20	067450
BIT1 =	000002	B36	034664	CLRMEM	007644	DETR3	002204	DT22	067460
BIT10 =	002000	B37	035004	CLR1CS=	104503	DETR4	002206	DT23	067466
BIT11 =	004000	B4	006370	CMD16A	053650	DETR5	002210	DT24	067510
BIT12 =	010000	B40	035164	CMD16L=	000074	DETSP	002212	DT25	067530
BIT13 =	020000	B41	035206	CMD5B	052012	DET1	063030	DT26	067542
BIT14 =	040000	B42	036402	CMD5C	052306	DF1	067610	DT27	067550
BIT15 =	100000	B43	036542	CMD7B	052544	DF11	067733	DT3	067214
BIT2 =	000004	B44	036600	CMD7C	052620	DF13	067744	DT30	067570
BIT3 =	000010	B45	037072	CMD9B	053264	DF14	067754	DT4	067224
BIT4 =	000020	B46	037260	CMD9C	053340	DF15	067763	DT5	067234
BIT5 =	000040	B47	037456	CONFG	002446	DF16	067772	DT6	067252
BIT6 =	000100	B5	006424	CONFIE	003656	DF2	067611	DT7	067266
BIT7 =	000200	B50	037632	CONF IG	002652	DF3	067635	DUMMY	002174
BIT8 =	000400	B51	037652	CONTF L	002220	DF4	067650	DUMPCS=	000103
BIT9 =	001000	B52	037662	CONTRL=	177746	DF5	067663	ECCDIS=	104470
BLOCK1	050030	B53	040200	CONTS	064264	DF6	067676	ECCINI=	104472
BLOCK2	050050	B54	040350	CONTS1	063642	DF7	067711	ECCTYP	005716
BLOCK3	050064	B55	040364	CONTS2	064266	DF8	067724	ECC1DI=	104471
BMFLAG	002126	B56	040430	CONTS3	056424	DF9	067726	ECC1IN=	104473
BOOT	047536	B57	043446	CONTT	064210	DH1	072460	EMTVEC=	000030
BOOT1	047602	B6	012576	COUNT	002366	DH10	073030	EM11	070301
BRGOBB	036252	B60	043452	CPERRF	061366	DH11	073126	EM12	070323
BSIZE	002372	B61	043612	CPSAVE	061364	DH12	073144	EM13	070347

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 517-4

SYMBOL TABLE

EM14	070401	E10	013550	E64	044056	HIPAT	047512	LOADER=	000065
EM15	070445	E100	054662	E65	045404	HOLDLO=	000016	LOADHO	002564
EM17	070513	E101	055166	E66	045364	HT	= 000011	LOOP	014744
EM19	070553	E102	055166	E67	045620	I	002450	LOWMAP	046356
EM2	070001	E103	062642	E7	013564	IBSAVE	061362	LSIZE	002376
EM20	070625	E104	067642	E70	045532	IIII	= 177777	LWDBE	= 000061
EM21	070704	E105	062726	E71	052612	ILLCSR	014126	LWSBE	= 000057
EM22	070740	E106	063022	E72	053110	IMPTES	013066	LO	004626
EM23	070765	E107	063106	E73	053332	INCBNK	047522	L1	004670
EM24	071014	E11	014276	E74	053624	INCPAT	047476	L10	005166
EM25	071073	E12	014344	E75	053624	INCRPT	047476	L100	013056
EM26	071120	E13	014374	E76	053700	INHBAN	002536	L101	013060
EM27	071171	E14	014712	E77	054662	INHECC	002534	L102	013326
EM29	071261	E15	015452	FASTCI=	177640	INTFLA	002134	L103	013326
EM3	070037	E16	017022	FATAL\$	002062	INT64K	002136	L104	013226
EM30	071343	E17	017606	FCMD10	053424	INVALI=	104511	L105	013234
EM32	071453	E2	006056	FCMD11	053452	IOTVEC=	000020	L106	013326
EM33	071560	E20	017566	FCMD12	053474	JMPRL1	045752	L107	013300
EM35	071666	E21	017516	FCMD13	053514	KAMIKA	002004	L11	005174
EM36	071753	E22	017516	FCMD14	053536	KAMITE	027000	L110	013306
EM4	070071	E23	017466	FCMD15	053554	KDIAG	= 000010	L111	013332
EM40	072022	E24	024354	FCMD16	053640	KDPAR0=	172360	L112	013550
EM5	070137	E25	024340	FCMD17	053702	KDPAR6=	172374	L113	013550
EM50	072074	E26	024562	FCMD18	053716	KDPAR7=	172376	L114	013540
EM51	072130	E27	024546	FIELDS	050114	KERNEL=	104417	L115	013536
EM52	072200	E3	006200	FINDBA=	000067	KERSTK=	002000	L116	013542
EM53	072225	E30	024760	FINT	007176	KFLAG	002526	L117	014266
EM55	072254	E31	032500	FIRST	= 060000	KIPAR0=	172340	L12	005244
EM56	072275	E32	034406	FLIPLO	002604	KIPAR4=	172350	L120	014262
EM57	072327	E33	034374	FLIPWA	041106	KIPAR5=	172352	L121	014254
EM6	070214	E34	035156	FLUSH	015064	KIPAR6=	172354	L122	014260
EM60	072375	E35	035142	FSCMD0	050312	KIPDR0=	172300	L123	014266
EM61	072437	E36	034756	FSCMD1	050414	KMAP	= 104422	L124	014362
EM62	100344	E37	035126	FSCMD2	050524	KPFLAG	002112	L125	014432
EM7	070241	E4	006566	FSCMD3	050672	KSIZE	002374	L126	014544
ENASBE=	104506	E40	035202	FSCMD4	051146	KSTACK	002562	L127	014744
ENA1SB=	104507	E41	035226	FSCMD5	051466	LAST	= 157776	L13	005240
END	100420	E42	036524	FSCMD6	052404	LASTBA	002554	L130	014702
ENERGI=	104420	E43	036730	FSCMD7	052412	LASTBL	002556	L131	014620
ENEXBK	047450	E44	036716	FSCMD8	052704	LASTER	002014	L132	014622
ERRADD	002456	E45	037176	FSCMD9	053132	LBLS0	= 000610	L133	014662
ERRGEN=	104512	E46	037430	FSINFL	002440	LBLS1	= 000107	L134	014676
ERRMAX	002552	E47	037626	FSPAT	052166	LBLS2	= 000602	L135	014742
ERROR	= 104000	E5	006556	FSSTAC	002304	LBLS3	= 000575	L136	014744
ERRPC	002016	E50	040150	FS1	050200	LBLS4	= 000447	L137	015060
ERRPSW	002026	E51	040132	FS7FLA	002444	LBLS5	= 000451	L14	005244
ERRSP	002022	E52	040116	FULLRE	002540	LBLS6	= 000023	L140	015170
ERRVEC=	000004	E53	040334	GBLENG=	000076	LCSROU=	000063	L141	015200
EUFLAG	002130	E54	040620	GETCSR	041336	LCSRRE=	000101	L142	015350
EVEN	002362	E55	040610	GETDAT	054312	LCSRSA=	000077	L143	015350
EXBANK	047032	E56	040576	GETDA1	054410	LEGALC=	000010	L144	015436
EXCMD3	051100	E57	043520	GETDIS	060514	LF	= 000012	L145	015550
EXCMD4	051420	E6	013014	GOOD	002042	LINK1	002520	L146	015656
EXIT	047644	E60	043520	GOOD2	002044	LINK2	002522	L147	016006
EXIT2	047650	E61	043664	GOOD3	002046	LKS	= 177546	L15	005504
EO	004644	E62	043664	GTSWR	= 104407	LOADBA	002430	L150	016136
E1	005670	E63	044056	HEADER	002600	LOADCS=	104425	L151	016266

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 517-5

SYMBOL TABLE

L152	016440	L235	023172	L322	034756	L42	006420	L501	052120
L153	016570	L236	023224	L323	034706	L420	042162	L502	052132
L154	016742	L237	023270	L324	034720	L421	042172	L503	052462
L155	017022	L24	005764	L325	034736	L422	042216	L504	052464
L156	017060	L240	023540	L326	034744	L423	042320	L505	052576
L157	017070	L241	023550	L327	034770	L424	043232	L506	053110
L16	005504	L242	023550	L33	006144	L425	043240	L507	053074
L160	017516	L243	023560	L330	035126	L426	043476	L51	006546
L161	017500	L244	024324	L331	035026	L427	043476	L510	053044
L162	017550	L245	024272	L332	035040	L43	006424	L511	053202
L163	020020	L246	024324	L333	035072	L430	043540	L512	053204
L164	020130	L247	024370	L334	035056	L431	043546	L513	053316
L165	020266	L25	005774	L335	035070	L432	043642	L514	053606
L166	020314	L250	024532	L336	035114	L433	043642	L515	053610
L167	020320	L251	024500	L337	035102	L434	043704	L516	054054
L17	005616	L252	024532	L34	006154	L435	043712	L517	054070
L170	020322	L253	024744	L340	035114	L436	044034	L520	054074
L171	020366	L254	024744	L341	035226	L437	045220	L521	054112
L172	020422	L255	025052	L342	036574	L44	006454	L522	054240
L173	020426	L256	025024	L343	036600	L440	045234	L523	054242
L174	020430	L257	025102	L344	036664	L441	045246	L524	054306
L175	020576	L260	025272	L345	036702	L442	045246	L525	054532
L176	021252	L261	025624	L346	036706	L443	045350	L526	054640
L177	022120	L262	026114	L351	037024	L444	045400	L527	055144
L2	004776	L263	026152	L352	037162	L445	045522	L530	056462
L20	005644	L264	026252	L354	037334	L446	045522	L531	056464
L200	022130	L265	026276	L355	037404	L447	045522	L532	057314
L201	022130	L266	026350	L357	037540	L45	006526	L533	057326
L202	022140	L267	026420	L36	006252	L450	045502	L534	057344
L203	022176	L27	006146	L360	037610	L451	045522	L535	057346
L204	022206	L270	026462	L362	037762	L452	045564	L536	057366
L205	022206	L271	026526	L363	040036	L453	045564	L537	057400
L206	022216	L272	026566	L364	040054	L454	045614	L54	006602
L207	022246	L273	026630	L365	040102	L455	045734	L540	057416
L21	005662	L274	026670	L37	006256	L456	047170	L541	057420
L210	022252	L275	026730	L371	040244	L457	047376	L542	057434
L211	022302	L276	027022	L372	040306	L46	006516	L543	057636
L212	022312	L277	027030	L374	040424	L460	047560	L544	057644
L213	022312	L3	005020	L375	040430	L461	047664	L545	057672
L214	022322	L30	006126	L376	040502	L462	047670	L546	057704
L215	022366	L300	027034	L377	040504	L463	047676	L547	057722
L216	022376	L301	030652	L4	005036	L464	047712	L55	006644
L217	022376	L302	030650	L40	006322	L465	047724	L550	057734
L220	022406	L303	030652	L400	040542	L466	050136	L551	057746
L221	022456	L304	032350	L401	040562	L467	050146	L552	057770
L222	022466	L305	032406	L402	040566	L47	006520	L553	060036
L223	022466	L306	032466	L406	040660	L470	050302	L554	060116
L224	022476	L31	006120	L407	040656	L471	050342	L555	060132
L225	022530	L310	034164	L41	006364	L472	050346	L556	060134
L226	022544	L311	034170	L410	040700	L473	050376	L557	060250
L227	022554	L312	034264	L411	040700	L474	050410	L56	011564
L23	005766	L313	034342	L412	041016	L475	051576	L560	060340
L230	022554	L314	034360	L413	041064	L476	051636	L561	060350
L231	022634	L315	034364	L414	041476	L477	051676	L562	060656
L232	022644	L32	006126	L415	042062	L5	005114	L563	060656
L233	022644	L320	034634	L416	042164	L50	006542	L564	061116
L234	022662	L321	034644	L417	042154	L500	052104	L565	061052

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 517-6

SYMBOL TABLE

L566	061076	MMR0 =	177572	MSG049	076505	MTPA21	034434	MT0004	021474
L567	061116	MMR1 =	177574	MSG051	076540	MTPA24	035260	MT0005	021616
L57	011570	MMR2 =	177576	MSG055	076557	MTPA25	035670	MT0006	021752
L570	061136	MMR3 =	172516	MSG056	076600	MTPA26	036020	MT0007	022006
L571	061330	MMTRAP	042602	MSG058	076633	MTPB03	027554	MT0010	022050
L572	061200	MMVEC =	000250	MSG061	076655	MTPB04	027706	MT0011	022104
L573	061326	MONFLG	002274	MSG062	076664	MTPB21	034464	MT0012	022162
L574	061256	MSEEDH	002574	MSG063	076704	MTPB24	035320	MT0013	022266
L575	061270	MSEEDL	002576	MSG064	076715	MTPB25	035712	MT0014	022352
L576	061326	MSGA12	100060	MSG065	076725	MTPB26	036034	MT0015	022442
L577	061336	MSGA34	076075	MSG066	076737	MTPC03	027614	MT0016	022520
L6	005174	MSGB34	076133	MSG067	077006	MTPC21	034520	MT0017	022576
L60	011572	MSG000	100272	MSG070	077015	MTPC24	035334	MT0020	022620
L600	061672	MSG001	074102	MSG073	077046	MTPC25	035752	MT0021	022710
L601	062620	MSG002	074164	MSG075	077064	MTPC26	036070	MT0022	023162
L602	063024	MSG003	074241	MSG076	077116	MTPD03	027632	MT0023	023214
L603	063030	MSG004	074346	MSG077	077137	MTPD21	034554	MT0024	023260
L604	063110	MSG005	074454	MSG079	077153	MTPD25	035616	MT0025	023524
L605	063114	MSG006	074466	MSG085	077177	MTPD26	036110	MT0026	023602
L606	064230	MSG007	074523	MSG088	077224	MTPE25	035640	MT0027	024104
L607	064320	MSG008	100240	MSG089	077242	MTP000	027404	MT0030	024570
L61	012404	MSG009	074535	MSG090	077264	MTP001	027430	MT0031	025072
L610	064322	MSG010	074547	MSG091	077300	MTP002	027462	MT0032	025262
L62	012420	MSG011	074561	MSG092	077312	MTP005	027726	MT0033	025614
L63	012544	MSG012	074647	MSG093	077326	MTP006	027762	MT0034	026002
L64	012522	MSG013	074744	MSG095	077334	MTP007	030162	MT0035	026154
L65	012534	MSG014	074746	MSG101	077344	MTP010	030262	MT0036	026266
L66	012550	MSG015	074750	MSG102	077374	MTP011	030370	MT0037	026340
L67	013000	MSG016	074752	MSG103	077423	MTP012	031166	MT0040	026406
L7	005174	MSG017	074764	MSG104	077445	MTP013	031554	MT0041	026410
L70	013000	MSG018	074775	MSG105	077447	MTP014	032270	MT0042	026452
L71	013000	MSG019	075000	MSG106	077522	MTP015	032514	MT0043	026516
L72	012656	MSG020	075004	MSG107	077540	MTP016	033260	MT0044	026556
L73	012662	MSG021	075025	MSG110	077615	MTP017	034042	MT0045	026620
L74	013000	MSG022	075613	MSG111	077661	MTP020	034120	MT0046	026660
L75	012732	MSG023	075635	MSG112	077713	MTP022	034604	MT0047	026720
L76	013000	MSG025	075651	MSG113	077730	MTP025	035352	MT0999	026764
L77	013052	MSG026	075675	MSG114	077745	MTP030	036126	MT1	017030
MAINT =	177750	MSG027	075707	MSG117	077762	MTP031	036136	MT2	017034
MAPHO =	170202	MSG028	075724	MSG119	077774	MTP032	036214	MUT	002106
MAPKER	046720	MSG029	075740	MSG120	100003	MTP033	036246	NC	056464
MAPLO =	170200	MSG030	075760	MSG121	100024	MTP034	036344	NEMCNT	002066
MAPL1 =	170204	MSG031	075777	MSG122	100044	MTP035	036370	NEWBAN	002306
MAPPER	044546	MSG032	076037	MSG125	100112	MTP036	036532	NEWKER	046652
MASK	002316	MSG033	076056	MSG126	100134	MTP037	036756	NEWLOA	046754
MBERR	014004	MSG035	076161	MSG127	100201	MTP041	037030	NOCH	063624
MEMDON	015012	MSG036	076164	MSG128	100220	MTP042	037202	NOERRO	002426
MFPT =	000007	MSG037	076203	MSG129	100341	MTP043	037436	NOFSMO	002424
MJPAT	020622	MSG038	076222	MSIZE	002400	MTP044	037632	NONEM	002076
MJTEST	020516	MSG039	076240	MTA030	024602	MTP045	040154	NONEXI	042524
MKCNT	017670	MSG040	076262	MTEST	016754	MTP046	040342	NOOJ	041476
MKCONT	017050	MSG041	076306	MTLA11	030416	MTP047	040702	NOPAR	002074
MKCSRT	020206	MSG042	076333	MTLB11	030430	MTST3	012424	NORES	003666
MKFLAG	002116	MSG043	076351	MTLC11	030442	MT0000	020702	NOSCOPI	002436
MKLOOP	017232	MSG046	076373	MTLD11	030536	MT0001	020762	NOSUPE	002454
MKPAT	020436	MSG047	076426	MTPA03	027514	MT0002	021102	NOTAB	002370
MKTEST	020276	MSG048	076445	MTPA04	027652	MT0003	021242	NOTRCE	060264

CZMSP30 MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 517-7
 SYMBOL TABLE

NO22B1	002452	PERR30=	104454	SAVREG=	104415	SUPDR1	002160	TRAPVE=	000034
NULLFL	002342	PERR31=	104455	SAV4	002266	SUPDR2	002162	TSTBAN	012312
NXTCSR	005644	PERR32=	104456	SBEMSK	002250	SUPDR3	002164	TSTDAT	002244
OLDCAC	002300	PERR33=	104457	SBENT	020150	SUPDR4	002166	TSTRD1	043242
OLDCSR	002154	PERR34=	104460	SBESYN	034414	SUPDR5	002170	TSTREA=	104510
ONES	002602	PERR35=	104461	SBETES	017672	SUPDR6	002172	TST1	005524
PADDRE	002034	PERR36=	104462	SCOPE =	000004	SUPLIM	056542	TST2	011406
PAFBAF	016150	PERR37=	104463	SDPAR0=	172260	SUPSTK=	000740	TST3	011572
PAFBAW	016300	PERR40=	104464	SDPAR5=	172272	SWPAT	002622	TST4	012560
PARBAF	016452	PERR41=	104465	SDPAR6=	172274	SWR	002624	TST5	014744
PARBAW	016602	PERR42=	104466	SDPAR7=	172276	SWREG =	000176	TST6	015016
PARCNT	002070	PERR43=	104467	SEEDHI	002570	SW0 =	000001	TYPDS =	104405
PARITY	042420	PERXOR	057506	SEEDLO	002572	SW1 =	000002	TYPEIT=	104401
PARTHE	002302	PFECDF	062004	SELONL	002000	SW10 =	002000	TYPOC =	104402
PARVEC=	000114	PFECDH	061744	SETPAT	047512	SW11 =	004000	TYPOS =	104403
PASFLG	002262	PFECDT	061774	SHADL1	012454	SW12 =	010000	TYP50 =	000000
PASSNO	002264	PFECM	061710	SHUTUP	047676	SW13 =	020000	TYP51 =	000002
PATERR	002072	PFECWS	061700	SIPARO=	172240	SW14 =	040000	TYP52 =	000000
PATPLU	004610	PFLAG	002120	SIPAR3=	172246	SW15 =	100000	TYP53 =	000000
PATTER	002110	PGMCSR	002530	SIPAR5=	172252	SW2 =	000004	TYP54 =	000000
PCBUMP	002324	PHEBE	014006	SIPAR6=	172254	SW3 =	000010	TYP55 =	000000
PCONFI	041364	PHYADD	002036	SIPDR0=	172200	SW4 =	000020	TYP56 =	000002
PCONFS	041664	PMEMFL	002140	SIZE =	040000	SW5 =	000040	T12A	033260
PCONF1	041574	PROTYP	003754	SKIPKA	002006	SW6 =	000100	T12B	033302
PCONF2	041632	PSIZE	002402	SKIPMK	002340	SW7 =	000200	UDPAR0=	177660
PDP110	042614	PSW =	177776	SKJ	060312	SW8 =	000400	UDPAR7=	177676
PD1	054532	PTABLE	036736	SKPERR	002064	SW9 =	001000	UIPAR0=	177640
PERA05	056700	PWRVEC=	000024	SKUB	045614	SYSSIZ	003756	UIPAR1=	177642
PERBNK	057532	QUICK	002434	SKUJ	014010	TAG2\$	012044	UIPAR2=	177644
PERECC	057612	QVFLAG	002344	SOBK	002560	TAG3\$	012100	UIPAR3=	177646
PERRAB	057350	RANODD	036050	SOBLEN=	000056	TAG4\$	027130	UIPAR4=	177650
PERRAW	057276	RDCHR =	104411	SOFTPA	002606	TAG70\$	062010	UIPAR5=	177652
PERRA3	054100	RDDEC =	104414	SOURCE	002310	TAG71\$	062020	UIPAR6=	177654
PERRA7	057422	RDLIN =	104412	SPLTCS	002236	TAG72\$	062030	UIPDRO=	177600
PERR01=	104427	RDOCT =	104413	SSP =	%000006	TAG73\$	062100	UNITOP	002414
PERR02=	104430	READCS=	104426	ST =	177776	TAG74\$	062140	UNMAP	047006
PERR03=	104431	READON	002406	STACK =	002000	TAG75\$	062152	UNRELO	046070
PERR04=	104432	REALPA	002276	START	003656	TAG76\$	062164	UPPFLG	002263
PERR05	056674	REFRES	035160	START1	000300	TAG77\$	062230	USERMA	046570
PERR06	056722	REFSUB	035230	START2	000310	TAG78\$	062236	USESTK=	000700
PERR07=	104433	REGCOP	041076	START3	000200	TAG79\$	062316	USP =	%000006
PERR10=	104434	RELENT	045624	STAR27	024164	TAG9\$	011672	WARN1	011760
PERR11=	104435	RELOCA	045204	STOPOK	002416	TBG4\$	027306	WARN2	027526
PERR12=	104436	RELOC1	045640	STRIPE	002364	TCFIG1	041740	WARN3	027542
PERR13=	104437	RESREG=	104416	SUBAAA	004646	TCFIG2	042100	WARN4	027566
PERR14=	104440	RESTAR	002614	SUBAAB	004776	TCFIG3	042234	WARN5	027602
PERR15=	104441	RESVEC=	000010	SUBAAI	012450	TCONFI	041666	WARN6	041320
PERR16=	104442	RESO	050410	SUBAAP	014170	TEMP	002432	WARN6A	041260
PERR17=	104443	RES1	050470	SUBAAR	013374	TESTAD	002410	WARN6B	041312
PERR20=	104444	RES2	050636	SUBAAS	011402	TESTMO	002550	WARN7	024142
PERR21=	104445	RLFLAG	002124	SUCCESS	002332	TIME	002336	WASDBE=	104500
PERR22=	104446	RRFLAG	002122	SUPDOA	002260	TIMEOU	042570	WASSBE=	104476
PERR23=	104447	RTNVAL=	%000000	SUPDO1	027034	TKVEC =	000060	WAS1DB=	104501
PERR24=	104450	RWCSR	006210	SUPDO2	027050	TMFLAG	002132	WAS1SB=	104477
PERR25=	104451	SAVCSR	002152	SUPDO3	027212	TOOMAN	002404	WHICHC	053730
PERR26=	104452	SAVMON	002272	SUPDO4	027226	TOTCSR	002222	WOOPEN	055740
PERR27=	104453	SAVPAR	002270	SUPDRO	002156	TRACE	006206	WOOPS	055372

CZMSPBO MS11-L/M/P MEMORY DIAG. MACRO M1113 06-JUL-82 10:35 PAGE 517-8
 SYMBOL TABLE

WOOPSA	055770	\$DIDDO=	000000	\$L	=	000110	\$PER10	056752	\$SWR	=	163000
WOOPUP	055556	\$DDAGA	015350	\$LF		002647	\$PER11	057002	\$SWREG		065726
WORST	002566	\$DOAGN	015244	\$LL	=	000106	\$PER12	057022	\$T	=	000611
XOCHAR	056332	\$DOWN	054762	\$LOADC		042750	\$PER13	057044	\$TESTN		065710
XXDPCH	002352	\$DTBL	063562	\$LPADR		002610	\$PER14	057064	\$TKB		002632
ZEROS	002334	\$ECCDI	043270	\$LPERR		002612	\$PER15	057106	\$TKS		002630
\$APTHD	066004	\$ECCIN	043316	\$LS	=	000000	\$PER16	057130	\$TN	=	000007
\$AUTO	002060	\$ECC1D	043304	\$MADR1		065736	\$PER17	057150	\$TPB		002636
\$BANK	002011	\$ECC1I	043332	\$MADR2		065742	\$PER20	057166	\$TPFLG		002356
\$BASE	065760	\$ENASB	043344	\$MADR3		065746	\$PER21	057204	\$TPS		002634
\$BELL	002641	\$ENA1S	043360	\$MADR4		065752	\$PER22	057224	\$TRAP		066020
\$CACHF	042732	\$ENDAD	015234	\$MAIL		065704	\$PER23	057242	\$TRAP2		066042
\$CACHN	042706	\$ENERG	042666	\$MAMS1		065734	\$PER24	057260	\$TRPAD		066062
\$CBCSR	043372	\$ENV	065724	\$MAMS2		065740	\$PER25	054016	\$TSTM		066010
\$CBREG	044362	\$ENVN	065725	\$MAMS3		065744	\$PER26	057450	\$TSTRD		043064
\$CB1CS	043414	\$EOP	015070	\$MAMS4		065750	\$PER27	057470	\$TTYIN		064750
\$CDW1	065764	\$ERFLG	002012	\$MBADR		066006	\$PER30	054244	\$TYPDS		063356
\$CDW2	065766	\$ERRGE	044112	\$MNEW		065024	\$PER31	057660	\$TYPE		056206
\$CHARC	056520	\$ERROR	060566	\$MSGAD		065720	\$PER32	057756	\$TYPEC		056334
\$CHKDI	043766	\$ERRTB	066326	\$MSGLG		065722	\$PER33	060024	\$TYPEX		056522
\$CHK1D	044002	\$ERRTY	061370	\$MSGTY		065704	\$PER34	060104	\$TYPOC		063154
\$CKSWR	063602	\$ERTTL	002616	\$MSWR		065013	\$PER35	060136	\$TYPON		063170
\$CLRCS	043744	\$ESCAP	002360	\$MTYP1		065735	\$PER36	060172	\$TYPOS		063130
\$CLR1C	043756	\$ETABL	065724	\$MTYP2		065741	\$PER37	060222	\$T1	=	000000
\$CMTAG	002000	\$ETEND	066004	\$MTYP3		065745	\$PER40	060226	\$T2	=	000610
\$CMTGE	002542	\$EXHAL	047670	\$MTYP4		065751	\$PWRDN	054412	\$UNIT		065716
\$CNTLC	064774	\$ES	=	\$NOTRA		066054	\$PWRUP	054766	\$UNITM		066014
\$CNTLG	065006	\$FATAL	065706	\$NULL		002354	\$QUES	002645	\$USWR		065730
\$CNTLK	064202	\$FILLC	002640	\$NWTST=		000001	\$R	=	\$VECT1		065754
\$CNTLU	065001	\$FILLS	002355	\$SOCNT		063352	\$RAND	065470	\$VECT2		065756
\$CPUOP	065732	\$FS	=	\$SOCTLV		065666	\$RDCHR	064322	\$WASDB		043600
\$CRLF	002646	\$GTSWR	063756	\$SOCT8 =		065672	\$RDDEC	065206	\$WASS3		043434
\$DBLK	063572	\$HALT	061152	\$SOMODE		063354	\$RDLIN	064452	\$WAS1D		043714
\$DB20	065564	\$HALT2	066060	\$OVER		060502	\$RDOCT	065036	\$WAS1S		043550
\$DDW0	065770	\$HIBTS	066004	\$OS	=	000000	\$READC	043044	\$XTSTR		060360
\$DDW1	065772	\$HIOCT	065204	\$PASS		065712	\$RESRE	065432	\$Y\$	=	000000
\$DDW2	065774	\$ILLUP	055364	\$PASTM		066012	\$SAVRE	065374	\$ZAP42		015214
\$DDW3	065776	\$INVAL	044062	\$PATMA		002010	\$SAVR6	055370	\$Z\$	=	000000
\$DDW4	066000	\$ITEMB	002013	\$PER01		056542	\$SCOPE	060232	\$S\$	=	000000
\$DDW5	066002	\$IS	=	\$PER02		056570	\$STN	=	\$T	=	000574
\$DEENE	042676	\$KERNE	042656	\$PER03		056616	\$SVLAD	060466	\$TT	=	000602
\$DEVCT	065714	\$KMAP	045112	\$PER04		056646	\$SV\$	=	\$OFILL		063353
\$DEVM	065762	\$KS	=	\$PER07		056730					

. ABS. 100420 000
 000000 001
 ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 26136 WORDS (103 PAGES)
 DYNAMIC MEMORY: 21558 WORDS (82 PAGES)
 ELAPSED TIME: 00:26:58
 CZMSPB.BIN,CZMSPB/CR/-SP=CZMSPB/ML,CZMSPB.P11

CZMSPB SYMBOL	CREATED BY	MACRO	GN	6-JUL-82	AT	10:46	PAGE	1											
SYMBOL	CROSS REFERENCE	VALUE	REFERENCES																
ABORTF	002142	#139-5610	*337-11083	421-13227	423-13273	*423-13289	454-14598	454-14601	454-14610	454-14613									
ACFLAG	002114	#139-5599	170-6874	172-6914	182-7056	191-7354	193-7370	195-7400	197-7430	199-7467									
		201-7508	203-7546	205-7591	207-7629	211-7723	238-8456	238-8485	240-8528	363-11633									
		*371-11920	*371-11932	371-11934	393-12473	397-12546													
ACTFLA	002346	#139-5676	*153-5992	167-6735	173-6950	217-7919	217-7934	226-8189	226-8198	226-8213									
		227-8224	229-8238	229-8248	231-8263	234-8383	253-8825	363-11627	377-12038	466-14917									
ADDRES	002032	#139-5575	*160-6207	*167-6729	*273-9224	*273-9230	*273-9251	*273-9264	*275-9340	*275-9354									
		*277-9428	*282-9487	*283-9571	*283-9577	*285-9669	*285-9676	*305-10130	*305-10142	*315-10339									
		*315-10354	*316-10391	*317-10429	*323-10606	*323-10622	*339-11109	*421-13226	*421-13226	*421-13252									
		*423-13272	*423-13272	423-13275	423-13285	*451-14491	*451-14492	*451-14497	*451-14498	*451-14503									
		*451-14504	*451-14509	*451-14510	*451-14517	*451-14525	*451-14530	*451-14530	*451-14535	*451-14540									
		*453-14546	*453-14551	*453-14556	*453-14561	*453-14566	*453-14571	*453-14576	*453-14581	*453-14586									
		*453-14591	458-14662	*460-14690	*460-14699	473-15107	509-16405	509-16408	509-16411	509-16416									
		509-16427																	
ANA2	010004	#166-6460																	
APTDOW	047760	153-5986	#377-12072	377-12075	*377-12077														
APTECC	002422	#139-5700	*186-7243	186-7248	186-7248	509-16410													
APTFLA	002350	#139-5677	*153-5985	186-7219	189-7336	226-8189	226-8198	226-8213	227-8224	229-8238									
		229-8248	231-8263	234-8383	253-8825	363-11627	377-12038	466-14917											
APTHAN	015320	#189-7338	189-7344																
APTHLT	050026	#377-12079																	
APTPAR	002420	#139-5699	*186-7240	186-7248	509-16410														
APTSIZ	002442	#139-5708	*153-5982	186-7219															
BACK	060602	#465-14844	466-14934																
BACKGN	041236	222-8126	222-8144	224-8172	232-8279	234-8346	246-8721	246-8726	#328-10849										
BAD	002050	#139-5581	*157-6113	*160-6255	*282-9501	*282-9515	*291-9795	*291-9807	*317-10428	*319-10465									
		*321-10505	*321-10514	*322-10552	*322-10562	*323-10608	*323-10621	*324-10674	*324-10682	*325-10736									
		*326-10793	*326-10803	*421-13229	*421-13255	*423-13275	*423-13287	*451-14493	*451-14499	*451-14506									
		*451-14511	*451-14516	*451-14526	*451-14531	*451-14536	*451-14541	*453-14547	*453-14552	*453-14557									
		*453-14562	*453-14567	*453-14572	*453-14577	*453-14582	*453-14587	*453-14592	456-14639	*458-14661									
		*460-14691	*460-14700	*460-14701	*460-14720	*460-14727	509-16405	509-16411	509-16419	509-16421									
		509-16423	509-16426	509-16428															
BADPC	002020	#139-5570	*339-11136	421-13240	454-14599	454-14611	456-14622	458-14674	460-14689	460-14698									
		460-14709	465-14880	465-14881	*465-14885														
BADPSW	002030	#139-5574	*339-11137	460-14720	465-14884														
BADSP	002024	#139-5572	*339-11134	*339-11135	465-14883														
BADSTA	042626	337-11084	337-11092	339-11117	339-11121	339-11124	#339-11133	421-13240	451-14519	454-14599									
		454-14611	456-14622	458-14674	460-14689	460-14698	460-14709	460-14719											
BADXOR	002056	#139-5584	*456-14639	*456-14640	509-16411														
BAD2	002052	#139-5582	*421-13256	509-16419															
BAD3	002054	#139-5583	*421-13258	*421-13259	509-16419														
BAFPAF	015454	187-7259	#193-7366	193-7391															
BAFPAR	015562	187-7260	#195-7396	195-7421															
BAKPAT	002620	#141-5753	*147-5882	232-8278	234-8345														
BANK	002100	#139-5593	*169-6750	*169-6762	169-6763	169-6765	169-6772	*169-6791	*170-6871	*170-6877									
		*170-6880	*170-6884	170-6886	*172-6910	*173-6947	173-6947	*173-6949	174-6959	*182-7052									
		*182-7082	182-7082	*191-7352	*191-7360	191-7360	*193-7367	*195-7397	*197-7427	*199-7464									
		*201-7503	*203-7541	*205-7586	*207-7624	211-7692	211-7719	211-7756	*211-7772	213-7828									
		213-7853	232-8289	232-8303	234-8356	236-8406	236-8419	*238-8454	238-8458	*238-8476									
		238-8476	*238-8483	238-8486	*238-8505	238-8505	*240-8525	*240-8534	240-8534	*240-8544									
		242-8562	244-8591	246-8655	255-8833	257-8866	319-10441	321-10484	321-10490	326-10777									
		354-11380	361-11590	*363-11631	363-11634	*363-11644	363-11644	365-11737	*365-11738	*365-11742									

CZMSPB CREATED BY MACRO ON 6-JUL-82 AT 10:46

PAGE 2
CREF

SYMBOL CROSS REFERENCE
SYMBOL VALUE

REFERENCES

		*365-11746	369-11871	371-11923	*373-12002	373-12003	386-12219	*386-12224	*386-12228	386-12231
		386-12240	*386-12255	388-12261	*388-12266	*388-12270	388-12273	388-12279	*388-12307	390-12313
		*390-12318	390-12319	390-12321	390-12340	390-12349	390-12364	390-12373	*390-12437	390-12438
		393-12450	*393-12471	*393-12477	393-12477	*393-12484	393-12485	395-12491	*395-12502	395-12503
		395-12510	*395-12516	395-12516	*395-12519	397-12524	*397-12544	*397-12550	397-12550	*397-12557
		397-12558	430-13486	*430-13487	430-13488	*430-13508	432-13527	*432-13528	432-13529	*432-13542
		458-14648	463-14811	472-15051	472-15065	472-15082	473-15089	473-15109	485-15557	
BANKIN	002102	#139-5594	*169-6768	169-6824	170-6873	172-6913	182-7054	211-7698	226-8203	246-8685
		246-8711	329-10873	355-11389	361-11568	363-11639	365-11740	365-11744	*371-11926	
BANKMO	046422	189-7335	363-11637	365-11735	#367-11788	377-12062				
BANKOK	047460	201-7506	203-7544	205-7589	207-7627	#373-11977				
BAWPAF	015670	187-7261	#197-7426	197-7450	197-7458					
BAWPAR	016020	187-7262	#199-7463	199-7487	199-7495					
BGTEST	036250	#313-10275	313-10296							
BITNO	002322	#139-5666	*291-9774	*291-9781	291-9817	*316-10373	*316-10381	316-10404	*325-10711	*325-10719
		325-10748								
BIT0	= 000001	#111-4749	158-6137	159-6170	159-6177	160-6208	160-6246	160-6272	162-6309	162-6322
		166-6532	166-6552	166-6590	166-6611	166-6621	166-6655	166-6663	169-6831	169-6850
		315-10334	315-10349	333-10949	335-11002	341-11150	341-11154	341-11161	344-11222	344-11224
		346-11245	346-11249	346-11253	346-11257	348-11292	348-11294	348-11300	348-11303	350-11327
		350-11329	350-11337	350-11339	363-11655	363-11708	363-11717	365-11756	365-11759	371-11955
		386-12234	388-12275	458-14651	462-14762	465-14871	465-14873			
BIT1	= 000002	#111-4748	151-5955	158-6144	158-6147	159-6171	159-6177	162-6309	162-6322	162-6330
		163-6374	163-6404	166-6532	166-6552	166-6559	166-6665	182-7095	182-7118	182-7130
		325-10723	335-10968	335-11003	335-11027	346-11237	346-11241	346-11253	346-11257	346-11262
		346-11267	352-11353	352-11358	371-11930					
BIT10	= 002000	#111-4739	315-10342							
BIT11	= 004000	#111-4738	158-6139	166-6592	182-7071	279-9465	283-9612	287-9714	371-11967	
BIT12	= 010000	#111-4737	155-6049	155-6050	155-6056	166-6588	182-7071	182-7101	182-7132	226-8204
		335-10966	363-11719	371-11964	390-12371	478-15240	478-15241	478-15256		
BIT13	= 020000	#111-4736	155-6049	155-6056	157-6106	157-6107	174-6961	174-6977	189-7297	246-8691
		246-8705	273-9293	279-9462	282-9498	283-9609	287-9711	321-10503	321-10512	323-10604
		326-10791	326-10801	343-11186	344-11212	354-11383	363-11641	363-11688	365-11745	365-11750
		365-11768	365-11781	375-12030	478-15240	478-15256				
BIT14	= 040000	#111-4735	160-6247	166-6640	166-6642	255-8855	297-9976	297-9977	321-10487	344-11231
		355-11415	355-11417	363-11694	365-11753	367-11812	367-11828	371-11961	400-12624	400-12640
		426-13339	426-13345	429-13442	429-13446	430-13489	432-13530			
BIT15	= 100000	#111-4734	157-6126	163-6416	166-6605	166-6631	213-7822	213-7840	213-7843	213-7847
		273-9235	324-10672	325-10735	326-10780	344-11221	344-11231	350-11316	363-11640	363-11656
		363-11694	363-11703	363-11724	365-11741	365-11753	367-11812	367-11828	406-12659	426-13339
		429-13446	460-14710	463-14816	473-15112					
BIT2	= 000004	#111-4747	154-6029	154-6030	154-6031	160-6247	162-6311	166-6534	166-6578	291-9785
		291-9803	315-10334	316-10385	325-10723	337-11086	346-11262	346-11267	352-11353	352-11358
		355-11443								
BIT3	= 000010	#111-4746	154-6029	154-6030	154-6033	157-6104	160-6222	160-6232	160-6262	160-6280
BIT4	= 000020	#111-4745	137-5536	145-5839	145-5840	157-6104	166-6605	166-6631	189-7331	213-7822
		213-7847	273-9235	321-10493	344-11221	348-11281	348-11301	460-14710		
BIT5	= 000040	#111-4744	153-5978	166-6525	166-6596	166-6626	189-7331	363-11699	363-11707	365-11763
		365-11766	371-11970	375-12021	377-12066					
BIT6	= 000100	#111-4743	172-6930	331-10889	331-10926	335-11051	363-11659	371-11927	390-12360	390-12433
		393-12466	393-12482	397-12540	397-12555					
BIT7	= 000200	#111-4742	153-5981	154-6000	154-6001	170-6882	273-9284	279-9451	283-9598	287-9700

CZMSPB SYMBOL	CREATED BY	MACRO	ON	DATE	TIME	PAGE	CREF			
SYMBOL	CROSS REFERENCE VALUE	REFERENCES								
BIT8	= 000400	331-10888	344-11207	363-11655	363-11659	371-11949	390-12359	393-12465	397-12539	
BIT9	= 001000	#111-4741	184-7168	363-11658	371-11943	486-15627				
BLOCK1	050030	#111-4740	174-6987	174-7002	184-7169	371-11946				
		169-6759	220-8033	220-8055	220-8081	222-8105	222-8108	222-8134	222-8154	232-8307
		232-8310	232-8314	232-8317	234-8366	236-8423	236-8431	238-8449	240-8518	244-8620
		246-8689	328-10862	344-11217	#379-12089					
BLOCK2	050050	#379-12094								
BLOCK3	050064	222-8109	222-8110	222-8135	222-8155	234-8367	234-8368	236-8424	236-8426	242-8564
		246-8658	363-11695	365-11754	367-11813	367-11829	#379-12098	430-13492	430-13495	
BMFLAG	002126	#139-5604	197-7432	199-7469	203-7548	207-7631	363-11633	*371-11921	*371-11941	*371-11957
		*371-11960								
BOOT	047536	#375-12007	483-15504	486-15631						
BOOT1	047602	375-12015	375-12020	#375-12022						
BRGOBB	036252	#313-10276	313-10280	313-10314						
BSIZE	002372	#139-5689	*184-7182	*184-7183	*184-7184	*184-7185	184-7186			
CACHKF	002546	#141-5730	*145-5832	341-11168	382-12169					
CACHKN	002542	#141-5729	*145-5834	154-6037	*154-6037	*154-6038	341-11158	341-11160	341-11165	380-12120
		382-12167	382-12169	382-12172	399-12588	*399-12588	*399-12589	399-12594	*399-12594	425-13309
		429-13466	434-13604	434-13664						
CACHOF	= 104424	#110-4625	154-6036	162-6290	166-6521	167-6719	169-6781	169-6798	169-6815	169-6841
		172-6908	211-7735	213-7806	255-8857	257-8894	282-9484	291-9767	296-9938	316-10367
		317-10420	321-10483	322-10536	323-10579	324-10660	325-10701	326-10776	380-12124	399-12587
		434-13608								
CACHON	= 104423	#110-4624	147-5901	163-6424	166-6697	167-6703	167-6722	169-6789	169-6802	169-6823
		169-6846	173-6955	211-7747	213-7858	213-7866	255-8859	257-8896	282-9523	291-9821
		296-9916	316-10408	317-10432	321-10525	322-10568	323-10651	324-10691	325-10753	326-10811
		399-12595	425-13312							
CACHVE	= 000114	#111-4764	111-4765							
CBCSR	= 104474	#110-4668								
CBITS	002314	#139-5663	*166-6561	*166-6564	166-6566					
CBREG	= 104513	#110-4683	174-6988	174-7003	282-9488	291-9770	291-9798	316-10369	321-10494	321-10522
		322-10537	322-10555	324-10661	324-10677	325-10706	326-10781			
CB1CSR	= 104475	#110-4669	166-6542	273-9215	275-9325	277-9420	283-9560	285-9659	305-10119	317-10423
		317-10425	319-10448	319-10454	319-10470	323-10582				
CHECK	002312	#139-5662	*166-6492	*273-9290	*273-9291	*279-9456	*279-9458	*279-9460	*283-9603	*283-9605
		*283-9607	*287-9705	*287-9707	*287-9709	*316-10392	*317-10422	*317-10424	*319-10447	*319-10453
		346-11261	346-11266	*357-11474	509-16416					
CHKDIS	= 104504	#110-4676								
CHKGEN	044400	273-9203	275-9316	277-9409	283-9547	285-9647	303-10059	#357-11448		
CHKTAB	044506	357-11458	#359-11479							
CHKTRP	040626	325-10733	#325-10757							
CHK1DI	= 104505	#110-4677	174-6966	174-6978	174-6993	174-7008	421-13243			
CKEND	064114	483-15485	483-15487	483-15489	#483-15518					
CKSWR	= 104410	#110-4608	462-14754	465-14843	466-14906					
CLRCSR	= 104502	#110-4674	162-6339	163-6417	#217-7922	238-8444	240-8537	390-12380	#393-12457	#397-12531
CLREX	007754	164-6437	164-6451	#164-6455						
CLRMEM	007644	162-6344	163-6425	#164-6437						
CLR1CS	= 104503	#110-4675	166-6541	166-6550	166-6687	174-6970	174-6983	174-6998	174-7012	174-7026
		213-7814	213-7838	213-7862	273-9181	273-9242	273-9255	273-9301	275-9308	275-9328
		275-9383	277-9389	277-9403	277-9422	279-9473	282-9521	283-9541	283-9562	283-9621
		285-9663	287-9723	291-9789	291-9820	303-10094	315-10347	316-10407	319-10456	319-10472
		321-10500	321-10524	322-10546	323-10647	324-10669	324-10690	325-10740	325-10752	326-10788

CZMSPB CREATED BY MACRO ON 6-JUL-82 AT 10:46 PAGE 4
SYMBOL CROSS REFERENCE SYMBOL VALUE REFERENCES CREF

SYMBOL	VALUE	REFERENCES	CREF
CMD16A	053650	326-10808 421-13249	
CMD5B	052012	400-12613 #400-12636	
CMD5C	052306	#390-12364 390-12389	
CMD7B	052544	390-12345 390-12356 #390-12432	
CMD7C	052620	#393-12471 393-12479	
CMD9B	053264	393-12462 #393-12481	
CMD9C	053340	#397-12544 397-12552	
CONFGE	002446	397-12536 #397-12554	
CONFIE	003656	#139-5710 *166-6613 *166-6657 *169-6832 *169-6851 *182-7078 *182-7157	
CONFIG	002652	#143-5796 151-5951 *154-6000 *154-6001 166-6525 *166-6581 *166-6585 *166-6588 *166-6592	
		*166-6596 166-6597 *166-6597 166-6598 *166-6598 *166-6611 *166-6612 166-6614 *166-6614	
		166-6615 *166-6615 166-6624 *166-6624 166-6625 *166-6625 *166-6626 *166-6655 *166-6656	
		*167-6731 169-6773 *169-6831 *169-6833 *169-6850 *169-6854 *170-6882 172-6923 *172-6930	
		173-6935 174-6987 174-7002 *174-7029 *174-7030 182-7057 182-7065 *182-7071 *182-7072	
		182-7094 182-7101 182-7118 182-7130 182-7132 184-7167 184-7168 184-7169 184-7189	
		189-7296 211-7699 211-7710 226-8204 246-8691 *246-8705 246-8712 315-10342 329-10874	
		333-10949 335-10966 335-10968 335-10974 335-10995 335-10997 335-11026 335-11047 335-11051	
		*354-11383 *363-11640 *363-11641 363-11655 363-11656 363-11657 363-11658 363-11659 363-11659	
		*363-11688 363-11713 363-11719 363-11721 *365-11741 *365-11745 *365-11750 365-11767 371-11927	
		371-11931 371-11936 371-11943 371-11946 371-11949 371-11955 371-11961 371-11964 371-11967	
		371-11970 390-12324 390-12351 390-12371 395-12505 395-12511 *400-12624 *400-12640 *458-14651	
		*458-14652 *458-14654 458-14655	
CONFL	002220	#139-5635 *187-7254 *193-7381 *195-7411 *197-7443 *199-7480 *201-7514 *203-7554 *205-7597	
		*207-7637 209-7673 *211-7770	
CONTRL	= 177746	#111-4770 145-5828 154-6016 *154-6029 *154-6030 154-6031 154-6033 *341-11160 *341-11161	
		*341-11168 380-12122 *382-12174 425-13311 *429-13468 434-13606 *434-13666 509-16417	
CONTS	064264	483-15478 #485-15566	
CONTS1	063642	483-15463 #483-15468 485-15573	
CONTS2	064266	#485-15568 485-15569 485-15575	
CONTS3	056424	#434-13642 434-13643 434-13649	
CONTT	064210	462-14757 478-15206 483-15473 #485-15541	
COUNT	002366	#139-5687 *296-9917 296-9920 *296-9920 *296-9921 296-9922 *296-9929 *296-9936 297-9941	
		*297-9941 *297-9942 297-9943 *297-9962	
CPERRF	061366	*145-5868 *145-5870 462-14759 465-14868 #466-14938	
CPSAVE	061364	*462-14761 462-14762 *465-14870 465-14871 #466-14937 470-15030	
CPUBIT	002104	#139-5595 *151-5955 167-6731 169-6833 169-6854 184-7167 184-7189 335-10995 363-11657	
		371-11931 390-12324	
CPUERR	= 177766	#111-4776 *145-5871 *151-5967 *164-6457 *167-6732 *169-6792 *382-12186 *382-12190 *384-12202	
		*384-12214 *386-12256 *388-12308 *462-14792 *466-14916 *476-15198 509-16409 509-16417	
CR	= 000015	#111-4704 434-13655	
CRLF	= 000200	#111-4705 434-13584 513-16476 517-16522 517-16523 517-16524 517-16526 517-16528 517-16529	
		517-16530 517-16531 517-16532 517-16538 517-16542 517-16543 517-16544 517-16545 517-16546	
		517-16547 517-16548 517-16549 517-16550 517-16551 517-16552 517-16553 517-16554 517-16555	
		517-16556 517-16557 517-16558 517-16559 517-16560 517-16561 517-16562 517-16564 517-16565	
		517-16566 517-16567 517-16568 517-16569 517-16570 517-16571 517-16572 517-16573 517-16574	
		517-16575 517-16578 517-16579 517-16580 517-16581 517-16582 517-16583 517-16584 517-16585	
		517-16586 517-16586 517-16587 517-16589 517-16592 517-16593 517-16594 517-16598 517-16600	
		517-16602 517-16603 517-16604 517-16605 517-16605 517-16606 517-16607 517-16608 517-16609	
		517-16610 517-16611 517-16616 517-16617 517-16618 517-16618 517-16620 517-16621 517-16622	
		517-16623 517-16624 517-16625 517-16626 517-16627 517-16628 517-16629 517-16632 517-16636	
		517-16637 517-16638 517-16639 517-16639 517-16639 517-16640 517-16641 517-16641	

CZMSPB CREATED BY MACRO ON 6-JUL-82 AT 10:46

PAGE 5
CREF

SYMBOL CROSS REFERENCE
SYMBOL VALUE
CSR 002146

REFERENCES

#139-5612	*160-6220	*160-6230	*160-6256	*160-6278	166-6547	166-6631	*174-6961	*174-6977
*174-6989	*174-6991	*174-7004	*174-7006	213-7822	213-7847	273-9235	273-9235	*282-9489
*282-9498	282-9499	282-9501	*291-9786	*291-9792	291-9793	291-9795	*291-9800	291-9804
291-9807	*315-10334	*315-10349	*316-10386	319-10459	*321-10497	*321-10503	321-10504	321-10505
*321-10509	*321-10512	321-10513	321-10514	*322-10543	*322-10549	322-10550	322-10552	*322-10558
322-10559	322-10562	*323-10592	*323-10600	323-10603	*323-10616	323-10619	323-10621	*324-10664
324-10672	324-10674	*324-10680	324-10681	324-10682	*324-10688	*325-10724	325-10735	325-10736
*326-10782	*326-10785	*326-10791	326-10792	326-10793	*326-10801	326-10802	326-10803	*343-11186
343-11187	*343-11195	344-11221	344-11221	*344-11232	*346-11237	*346-11241	*346-11245	*346-11249
*346-11253	*346-11257	*346-11261	*346-11262	*346-11266	*346-11267	348-11281	348-11301	350-11316
350-11335	*352-11344	*352-11348	*352-11353	*352-11358	355-11406	*355-11442	*355-11443	*384-12205
*421-13242	421-13248	426-13356	*428-13423	460-14710	478-15209	478-15218	*478-15224	509-16414
509-16415	509-16416	509-16424	509-16425	509-16428				

CSRADD = 172100

#113-4981	157-6096	*162-6311	*162-6314	*162-6316	*162-6319	162-6321	*163-6364	163-6379
*163-6396	163-6409	*166-6534	*166-6537	*166-6635	166-6637	*166-6640	166-6641	*166-6642
*343-11187	343-11195	344-11201	*355-11415	355-11416	*355-11417			

CSRCAS 020176

211-7744	#215-7872							
----------	-----------	--	--	--	--	--	--	--

CSRFBA 002230

#139-5640	*211-7693	211-7718	*211-7763					
-----------	-----------	----------	-----------	--	--	--	--	--

CSRFIR 002224

#139-5638	*211-7718	211-7724	211-7726	*211-7760	211-7760			
-----------	-----------	----------	----------	-----------	----------	--	--	--

CSRHOL 002524

#139-5719	*211-7703	*211-7713	211-7715					
-----------	-----------	-----------	----------	--	--	--	--	--

CSRINC 002326

#139-5668								
-----------	--	--	--	--	--	--	--	--

CSRINF 002460

#139-5715	*157-6103	*157-6104	157-6112	157-6113	*158-6137	*158-6144	*158-6147	159-6169
159-6170	159-6171	159-6177	160-6208	*160-6222	*160-6232	160-6246	*160-6262	160-6272
*160-6280	162-6309	162-6322	162-6330	163-6374	163-6404	166-6532	166-6552	166-6559
166-6582	166-6663	166-6665	182-7049	*182-7060	182-7088	182-7090		

CSRINT 002234

#139-5642	*211-7695	*211-7709	211-7762					
-----------	-----------	-----------	----------	--	--	--	--	--

CSRLAS 002226

#139-5639	*211-7724	*211-7725	211-7751					
-----------	-----------	-----------	----------	--	--	--	--	--

CSRLBA 002232

#139-5641	*211-7694	*211-7707	211-7760					
-----------	-----------	-----------	----------	--	--	--	--	--

CSRLOO 002330

#139-5669	*211-7697	*211-7708	211-7766					
-----------	-----------	-----------	----------	--	--	--	--	--

CSRMAP 005776

157-6128	#159-6151							
----------	-----------	--	--	--	--	--	--	--

CSRNO 002150

#139-5613	*160-6206	*162-6297	*162-6303	*163-6358	*163-6371	*163-6390	*163-6401	*166-6513
*166-6520	166-6634	*172-6926	172-6927	172-6928	*173-6938	*211-7715	*211-7716	*246-8716
246-8717	*329-10878	343-11176	343-11182	343-11184	343-11194	344-11202	344-11208	344-11210
*348-11277	*348-11287	348-11287	*350-11312	*350-11322	350-11322	*354-11367	*354-11373	354-11373
355-11414	380-12123	*382-12166	390-12348	*390-12355	390-12379	*390-12381	*390-12440	*406-12661
*406-12675	425-13315	*426-13352	*426-13359	426-13359	*428-13420	*428-13427	428-13427	*429-13462
478-15209	*478-15214	*478-15222	478-15222	*478-15224	509-16425			
346-11238	346-11246	346-11254	346-11263	352-11345	352-11354	#354-11363		

CSROUT 044014

#139-5665	*160-6213	*160-6214	160-6224	160-6228	160-6229			
-----------	-----------	-----------	----------	----------	----------	--	--	--

CSRIS 002320

157-6129	#162-6290							
----------	-----------	--	--	--	--	--	--	--

CTEST 006660

#139-5611	*211-7742	*211-7743	*217-7931	*217-7932	*219-7983	*219-7984	483-15482	
-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	--

CTLKVE 002144

#111-4773	154-6020							
-----------	----------	--	--	--	--	--	--	--

DATARG = 177754

#139-5644	*267-9048	*267-9049	267-9050	267-9051	267-9053	267-9058	267-9062	*267-9064
*267-9064	*267-9067	*267-9068	267-9069	267-9070	267-9072	267-9077	267-9081	*267-9083
*267-9083	*273-9188	*273-9189	273-9194	273-9195	273-9276	*273-9278	*273-9278	*275-9309
*275-9310	275-9313	275-9314	275-9373	*275-9375	*275-9375	*277-9391	*277-9392	277-9397
277-9398	*283-9529	*283-9530	283-9535	283-9536	*285-9630	*285-9631	285-9636	285-9637
451-14527	451-14532							

DATBUF 002240

#139-5644	*267-9048	*267-9049	267-9050	267-9051	267-9053	267-9058	267-9062	*267-9064
*267-9064	*267-9067	*267-9068	267-9069	267-9070	267-9072	267-9077	267-9081	*267-9083
*267-9083	*273-9188	*273-9189	273-9194	273-9195	273-9276	*273-9278	*273-9278	*275-9309
*275-9310	275-9313	275-9314	275-9373	*275-9375	*275-9375	*277-9391	*277-9392	277-9397
277-9398	*283-9529	*283-9530	283-9535	283-9536	*285-9630	*285-9631	285-9636	285-9637
451-14527	451-14532							

DBEMSK 002254

#139-5647	*277-9395	*277-9396	277-9404	277-9406	277-9414	277-9416	279-9436	*279-9438
*279-9438	*279-9455	279-9459	*279-9461	279-9462	*279-9467	*279-9468	*283-9533	*283-9534
283-9542	283-9544	283-9552	283-9554	283-9584	*283-9586	*283-9586	*283-9602	283-9606
*283-9608	283-9609	*283-9614	*283-9615	*285-9634	*285-9635	285-9642	285-9644	285-9652

CZMSPB		CREATED BY	MACRO	ON 6-JUL-82 AT 10:46		PAGE 6					
SYMBOL	CROSS REFERENCE	VALUE	REFERENCES								
			285-9654	287-9686	*287-9688	*287-9688	*287-9704	287-9708	*287-9710	287-9711	*287-9716
			*287-9717								
DDISP	=	177570	#111-4698	141-5758	151-5962						
DEENER	=	104421	#110-4621	155-6044	375-12018	377-12063					
DETAIL		062340	470-15021	#476-15169	476-15201						
DETFLA		002216	#139-5634	466-14913	470-15020	*476-15170	476-15171	476-15173	*478-15271		
DETSPW		002214	#139-5633	509-16413							
DETRO		002176	#139-5626	*476-15178	476-15187	509-16413					
DETR1		002200	#139-5627	476-15179	509-16413						
DETR2		002202	#139-5628	509-16413							
DETR3		002204	#139-5629	509-16413							
DETR4		002206	#139-5630	509-16413							
DETR5		002210	#139-5631	509-16413							
DETSP		002212	#139-5632	509-16413							
DET1		063030	478-15239	#478-15256							
DF1		067610	508-16337	#511-16432							
DF11		067733	500-16134	502-16201	502-16206	502-16211	504-16233	504-16238	504-16253	508-16347	#511-16441
DF13		067744	502-16226	506-16305	508-16377	508-16382	508-16402	#511-16442			
DF14		067754	506-16320	508-16342	508-16362	508-16372	508-16387	508-16392	#511-16443		
DF15		067763	#511-16444								
DF16		067772	508-16352	508-16357	508-16367	#511-16445					
DF2		067611	500-16139	500-16154	500-16159	500-16164	500-16169	502-16186	502-16216	504-16243	504-16248
			504-16258	504-16263	506-16280	506-16325	508-16332	508-16397	#511-16433		
DF3		067635	502-16176	504-16273	506-16290	506-16295	506-16310	506-16315	#511-16434		
DF4		067650	502-16181	#511-16435							
DF5		067663	502-16191	502-16221	506-16300	#511-16436					
DF6		067676	502-16196	#511-16437							
DF7		067711	504-16268	#511-16438							
DF8		067724	506-16285	#511-16439							
DF9		067726	500-16144	500-16149	#511-16440						
DH1		072460	500-16137	#515-16495							
DH10		073030	502-16184	#515-16502							
DH11		073126	504-16246	#515-16503							
DH12		073144	504-16241	#515-16504							
DH13		073151	500-16132	502-16199	502-16204	502-16209	504-16231	504-16236	504-16251	508-16345	#515-16505
DH14		073246	504-16256	#515-16507							
DH15		073325	504-16266	#515-16508							
DH16		073444	504-16271	506-16283	#515-16510						
DH19		073461	#515-16511								
DH2		072515	508-16330	#515-16496							
DH20		073466	506-16303	508-16375	508-16380	508-16400	#515-16512				
DH23		073545	502-16224	#515-16513							
DH24		073624	506-16318	508-16340	508-16360	508-16370	508-16385	508-16390	#515-16514		
DH25		073673	500-16167	506-16278	#515-16515						
DH26		073731	502-16214	#515-16516							
DH27		073747	508-16335	#515-16517							
DH3		072540	500-16142	500-16147	#515-16497						
DH30		074023	508-16350	508-16355	508-16365	#515-16518					
DH5		072574	500-16152	500-16157	500-16162	504-16261	508-16395	#515-16498			
DH6		072653	506-16323	#515-16499							
DH7		072720	502-16174	502-16179	502-16189	502-16194	502-16219	506-16288	506-16293	506-16298	506-16308
			506-16313	#515-16500							

CZMSPB		CREATED BY MACRO ON 6-JUL-82 AT 10:46		PAGE 7							
SYMBOL	CROSS REFERENCE	VALUE	REFERENCES	CREF							
DIAGFL		002002	#139-5560 *233-8335 *233-8337 296-9934								
DISPLA		002626	#141-5758 *151-5962 *151-5969 429-13460 463-14821								
DISPRE	=	000174	#113-4977 151-5969 *429-13461 *463-14822								
DISPTB		014772	187-7257 #187-7259								
DOBACK		015354	187-7268 187-7279 #191-7350								
DONE		006644	160-6272 #160-6282								
DSWR	=	177570	#111-4697 141-5757 151-5961								
DT1		067176	500-16138 #509-16405								
DT10		067316	502-16185 #509-16413								
DT11		067340	504-16247 #509-16414								
DT12		067346	504-16242 #509-16415								
DT13		067352	500-16133 502-16200 502-16205 502-16210 504-16232 504-16237 504-16252 508-16346 #509-16416								
DT14		067374	*145-5835 *145-5847 *145-5853 504-16257 #509-16417								
DT16		067412	504-16267 #509-16418								
DT17		067442	504-16272 506-16284 #509-16420								
DT2		067210	#509-16406								
DT20		067450	508-16331 #509-16421								
DT22		067460	#509-16422								
DT23		067466	502-16225 506-16304 508-16376 508-16381 508-16401 #509-16423								
DT24		067510	506-16319 508-16341 508-16361 508-16371 508-16386 508-16391 #509-16424								
DT25		067530	500-16168 506-16279 #509-16425								
DT26		067542	502-16215 #509-16426								
DT27		067550	508-16336 #509-16427								
DT3		067214	500-16143 #509-16407								
DT30		067570	508-16351 508-16356 508-16366 #509-16428								
DT4		067224	500-16148 #509-16408								
DT5		067234	*145-5846 *145-5852 500-16153 500-16158 500-16163 504-16262 508-16396 #509-16409								
DT6		067252	506-16324 #509-16410								
DT7		067266	502-16175 502-16180 502-16190 502-16195 502-16220 506-16289 506-16294 506-16299 506-16309								
DUMMY		002174	506-16314 #509-16411								
			#139-5624 509-16411 509-16411 509-16412 509-16412 509-16412 509-16412 509-16412 509-16416 509-16416								
			509-16418 509-16418 509-16419 509-16419 509-16420 509-16422 509-16423 509-16423 509-16423 509-16423								
			509-16423 509-16423 509-16424 509-16424 509-16424 509-16424 509-16424 509-16424 509-16427 509-16427								
			509-16427 509-16427 509-16427 509-16428 509-16428 509-16428 509-16428								
ECCDIS	=	104470	#110-4664 217-7920 240-8523 390-12377 393-12455 397-12529								
ECCINI	=	104472	#110-4666 167-6704 #167-6738 #173-6953 211-7769 #217-7937 240-8540 240-8547 240-8551								
			375-12014 377-12055 #421-13266								
ECCTYP		005716	157-6108 #158-6136								
ECC1DI	=	104471	#110-4665 174-7022 213-7807 213-7830 273-9213 273-9220 273-9297 275-9323 275-9378								
			277-9418 279-9470 283-9557 283-9617 285-9657 287-9719 303-10099 305-10107 305-10117								
			325-10728								
ECC1IN	=	104473	#110-4667 213-7857 213-7865 282-9504 282-9507 303-10064 305-10110								
EMTVEC	=	000030	#111-4759 *147-5887 *147-5888								
EM11		070301	502-16173 502-16178 502-16188 502-16193 #513-16460								
EM12		070323	502-16183 #513-16461								
EM13		070347	502-16198 508-16374 #513-16462								
EM14		070401	502-16203 #513-16463								
EM15		070445	502-16208 #513-16464								
EM17		070513	502-16218 #513-16465								
EM19		070553	504-16230 508-16369 #513-16466								
EM2		070001	500-16136 #513-16454								
EM20		070625	504-16235 508-16359 508-16364 508-16384 #513-16467								

CZMSPB SYMBOL	CREATED BY CROSS REFERENCE VALUE	MACRO ON 6-JUL-82 AT 10:46 REFERENCES	PAGE 8 CREF
EM21	070704	504-16245 #513-16468	
EM22	070740	504-16250 508-16389 508-16399 #513-16469	
EM23	070765	504-16260 #513-16470	
EM24	071014	500-16131 #513-16471	
EM25	071073	504-16265 508-16349 #513-16472	
EM26	071120	504-16270 #513-16473	
EM27	071171	506-16282 #513-16474	
EM29	071261	506-16292 508-16344 #513-16475	
EM3	070037	500-16141 #513-16455	
EM30	071343	506-16297 508-16379 #513-16476	
EM32	071453	506-16307 #513-16478	
EM33	071560	506-16312 #513-16479	
EM35	071666	506-16287 #513-16480	
EM36	071753	506-16322 #513-16481	
EM4	070071	500-16146 #513-16456	
EM40	072022	508-16329 #513-16482	
EM5	070137	500-16151 #513-16457	
EM50	072074	502-16223 #513-16483	
EM51	072130	506-16317 #513-16484	
EM52	072200	506-16277 #513-16485	
EM53	072225	500-16166 #513-16486	
EM55	072254	502-16213 #513-16487	
EM56	072275	508-16334 #513-16488	
EM57	072327	508-16354 #513-16489	
EM6	070214	500-16156 #513-16458	
EM60	072375	506-16302 #513-16490	
EM61	072437	508-16339 #513-16491	
EM62	100344	508-16394 #517-16642	
EM7	070241	500-16161 #513-16459	
ENASBE =	104506	#110-4678 167-6736 173-6951 217-7935 421-13264	
ENA1SB =	104507	#110-4679 303-10068 303-10101 305-10112 #325-10730	
END	100420	#517-16650	
ENERGI =	104420	#110-4620 155-6071 189-7332	
ENEXBK	047450	371-11942 371-11965 371-11971 #371-11973	
ERRADD	002456	#139-5714 213-7825 213-7850 *355-11436	
ERRGEN =	104512	#110-4682 213-7824 213-7849 273-9241 273-9267 279-9433 315-10345	
ERRMAX	002552	#141-5732 458-14655	
ERROR =	104000	#111-4692 154-6005 154-6024 154-6027 157-6114 160-6221 160-6231 160-6258 #160-6260	
		160-6279 167-6726 167-6730 174-7031 186-7249 273-9236 275-9341 277-9429 282-9495	
		282-9502 282-9516 291-9796 291-9808 303-10073 305-10131 305-10143 315-10340 315-10355	
		316-10395 317-10430 321-10507 321-10517 322-10553 322-10563 324-10675 324-10684 325-10738	
		325-10760 325-10765 326-10796 326-10806 337-11093 339-11118 339-11122 339-11125 382-12185	
		384-12201 384-12210 421-13261 454-14602 #454-14604 454-14614 #454-14616 456-14625 458-14672	
		458-14677 458-14680 458-14683 460-14694 460-14705 460-14711 #460-14713 460-14722 460-14729	
		460-14733 460-14736 462-14764 476-15189 476-15195	
ERRPC	002016	#139-5569 *465-14858 *465-14859 465-14862 *465-14881 *465-14882 465-14887 468-14955 470-15030	
		509-16405 509-16406 509-16407 509-16408 509-16409 509-16410 509-16411 509-16414 509-16416	
		509-16418 509-16420 509-16421 509-16422 509-16423 509-16424 509-16425 509-16426 509-16427	
		509-16428	
ERRPSW	002026	#139-5573 *465-14861 *465-14884 476-15186	
ERRSP	002022	#139-5571 *465-14860 *465-14883 476-15185	
ERRVEC =	000004	#111-4752 *147-5897 *147-5898 462-14784 *462-14785 *462-14787 *462-14793	

CZMSPB		CREATED BY	MACRO ON 6-JUL-82 AT 10:46		PAGE 9						
SYMBOL	CROSS REFERENCE	VALUE	REFERENCES								
EUFLAG		002130	#139-5605	222-8087	355-11396	355-11411					
EVEN		002362	#139-5685	*296-9905	296-9906	*297-9968	297-9968				
EXBANK		047032	170-6872	172-6912	182-7053	191-7353	193-7369	195-7399	197-7429	199-7466	
			203-7543	205-7588	207-7626	238-8455	238-8484	240-8526	240-8545	363-11632	
			365-11743	*371-11897	390-12342	390-12439	393-12472	393-12486	397-12545	397-12559	
EXCMD3		051100	386-12245	386-12248	386-12252	#386-12254					
EXCMD4		051420	388-12288	388-12292	#388-12306						
EXIT		047644	#377-12036	462-14775	466-14919	486-15630					
EXIT2		047650	#377-12038								
E1		005670	#157-6122								
E2		006056	#159-6166								
E3		006200	#159-6187								
E31		032500	#282-9519								
E32		034406	#291-9819								
E33		034374	#291-9817								
E4		006566	#160-6271								
E43		036730	#316-10406								
E44		036716	#316-10404								
E45		037176	#319-10471								
E46		037430	#321-10523								
E47		037626	#322-10567								
E5		006556	#160-6270								
E50		040150	#323-10650								
E51		040132	#323-10646								
E52		040116	#323-10644								
E53		040334	#324-10689								
E54		040620	#325-10751								
E55		040610	#325-10750								
E56		040576	#325-10748								
FASTCI	=	177640	#111-4806	169-6787	169-6821	255-8858	344-11218	379-12090			
FATALS		002062	#139-5586	*167-6726	*167-6730	*303-10073	*337-11093	*339-11118	*339-11122	*339-11125	
			468-14972							466-14917	
FCMD10		053424	380-12144	#399-12564							
FCMD11		053452	380-12145	#399-12574							
FCMD12		053474	380-12146	#399-12579							
FCMD13		053514	380-12147	#399-12585							
FCMD14		053536	380-12148	#399-12592							
FCMD15		053554	380-12149	#400-12611							
FCMD16		053640	380-12150	#400-12631							
FCMD17		053702	380-12151	#402-12645							
FCMD18		053716	380-12152	#404-12651							
FIELDS		050114	#380-12111	483-15470							
FINT		007176	162-6343	#163-6350							
FIRST	=	060000	#113-4984	169-6776	169-6809	169-6839	172-6911	211-7693	219-7976	219-7977	
			220-8038	220-8062	222-8091	222-8127	222-8145	224-8173	232-8284	234-8350	
			238-8459	238-8488	240-8530	242-8564	242-8569	242-8570	244-8596	244-8626	
			246-8658	246-8661	246-8662	246-8663	246-8664	246-8681	249-8762	251-8789	
			289-9734	296-9918	296-9937	297-9973	297-9975	297-9976	315-10333	319-10445	
			*377-12075	*377-12076	*377-12077	386-12233	388-12274	390-12369	390-12370	430-13490	
			430-13492	*430-13493	*430-13494	430-13495	430-13497	*432-13531	*432-13532	432-13537	
FLIPLO		002604	#141-5745	*147-5876	222-8117	*327-10824	*327-10825	327-10826	327-10828	327-10830	
FLIPWA		041106	222-8090	#327-10822							

CZMSPB CREATED BY MACRO ON 6-JUL-82 AT 10:46

PAGE 10
CREF

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES
FLUSH		015064	#187-7284
FSCMD0		050312	#380-12134 #382-12157
FSCMD1		050414	380-12135 #382-12179
FSCMD2		050524	380-12136 #384-12194
FSCMD3		050672	380-12137 #386-12218
FSCMD4		051146	380-12138 #388-12260
FSCMD5		051466	380-12139 #390-12312
FSCMD6		052404	380-12140 #391-12443
FSCMD7		052412	380-12141 #393-12449
FSCMD8		052704	380-12142 #395-12490
FSCMD9		053132	380-12143 #397-12523
FSINFL		002440	#139-5707 *189-7295
FSPAT		052166	390-12387 #390-12391
FSSTAC		002304	#139-5659 *382-12181 382-12189 *384-12196 384-12213 *390-12314 390-12432 *393-12451 393-12481
			*397-12525 397-12554
FS1		050200	#380-12126 380-12132 380-12154
FS7FLA		002444	#139-5709 238-8478
FULLRE		002540	#139-5724 *240-8513 *240-8541 *240-8552 363-11671
GBLENG	=	000076	246-8658 246-8661 #313-10315
GETCSR		041336	227-8231 231-8270 247-8733 248-8748 249-8760 250-8771 251-8787 #329-10868
GETDAT		054312	#423-13280 454-14598 454-14610
GETDA1		054410	*423-13282 423-13290 #423-13293
GETDIS		060514	255-8834 257-8867 462-14806 #463-14810 465-14847
GOOD		002042	#139-5578 *160-6219 *160-6229 *160-6277 *282-9491 282-9499 *282-9514 *291-9794 *291-9806
			*317-10421 *319-10466 *321-10506 *321-10515 *322-10551 *322-10561 *323-10607 *323-10620 *324-10663
			324-10681 *324-10687 *326-10794 *326-10804 *421-13233 *421-13235 *421-13253 *423-13271 *451-14494
			*451-14500 *451-14505 *451-14512 *451-14515 *451-14522 *451-14527 *451-14532 *451-14537 *451-14542
			*453-14548 *453-14553 *453-14558 *453-14563 *453-14568 *453-14573 *453-14578 *453-14583 *453-14588
			*453-14593 456-14638 *458-14663 *458-14665 *460-14692 *460-14702 *460-14721 *460-14726 509-16405
			509-16411 509-16418 509-16421 509-16423 509-16425 509-16428
GOOD2		002044	#139-5579 *456-14628 *456-14632 509-16418
GOOD3		002046	#139-5580 *456-14629 *456-14633 509-16418
GTSWR	=	104407	#110-4607 155-6064
HEADER		002600	#141-5743 *147-5877 *187-7274 *187-7276 *191-7355 *191-7358 *209-7667 *209-7682 *211-7738
			*273-9250 *273-9253 *273-9263 *273-9266 *275-9339 *275-9342 *275-9353 *275-9356 *277-9427
			*277-9430 *282-9500 *282-9513 *291-9805 *305-10132 *305-10144 *321-10516 *322-10560 *323-10609
			*323-10623 *324-10673 *324-10683 *325-10737 *325-10759 *325-10764 *326-10795 *326-10805 *390-12363
			*393-12469 *397-12542 *458-14668 *458-14685 *460-14693 *460-14695 *460-14704 *460-14706 *460-14728
			*460-14730 *466-14929 468-14970 468-14982 *470-15018 476-15175 *476-15176 476-15191 *476-15192
			*476-15199
HIPAT		047512	205-7584 207-7622 #373-11996
HT	=	000011	#111-4702 434-13581
I		002450	#139-5711 *149-5934 *149-5940 149-5940 *166-6528 *166-6530 166-6578 166-6586 166-6590
			166-6617 166-6621 166-6667 166-6672 166-6674 166-6678 *184-7180 *184-7186 184-7197
			*184-7197 184-7211 *211-7737 211-7740 *211-7746 211-7746 *238-8453 238-8463 238-8469
			*238-8477 238-8477 *238-8482 238-8492 238-8498 *238-8506 238-8506
IBSAVE		061362	*465-14841 465-14866 *465-14874 *465-14877 466-14932 #466-14936
IIII	=	177777	157-6122 157-6122 157-6122 #157-6122 159-6166 159-6166 159-6166 #159-6166 159-6187
			159-6187 159-6187 #159-6187 160-6270 160-6270 160-6270 #160-6270 160-6271 160-6271
			160-6271 #160-6271 282-9519 282-9519 282-9519 #282-9519 291-9817 291-9817 291-9817
			#291-9817 291-9819 291-9819 291-9819 #291-9819 316-10404 316-10404 316-10404 #316-10404
			316-10406 316-10406 316-10406 #316-10406 319-10471 319-10471 319-10471 #319-10471 321-10523

CZMSPB CREATED BY MACRO ON 6-JUL-82 AT 10:46

PAGE 11
CREF

SYMBOL	CROSS REFERENCE VALUE	REFERENCES
ILLCSR	014126	321-10523 321-10523 #321-10523 322-10567 322-10567 322-10567 #322-10567 323-10644 323-10644
IMPTES	013066	323-10644 #323-10644 323-10646 323-10646 323-10646 #323-10646 323-10650 323-10650
INCBNK	047522	#323-10650 324-10689 324-10689 324-10689 #324-10689 325-10748 325-10748 325-10748 #325-10748
INCPAT	047476	325-10750 325-10750 325-10750 #325-10750 325-10751 325-10751 325-10751 #325-10751
INCRPT	047476	182-7092 182-7109 #182-7149
INHBAN	002536	172-6932 173-6940 #174-6958
INHECC	002534	193-7382 195-7412 197-7444 199-7481 201-7515 203-7555 205-7598 207-7638 #373-12000
INTFLA	002134	193-7378 197-7440 #373-11988
INT64K	002136	201-7518 203-7558 #373-11987
INVALI	= 104511	#139-5723 *363-11665 363-11673
IOTVEC	= 000020	#139-5722 211-7691 343-11177 363-11663 *363-11664 363-11672 *363-11679 *363-11687 *365-11747
JMPRL1	045752	#139-5607 170-6874 172-6917 173-6934 182-7064 211-7704 355-11425 *371-11922 *371-11966
KAMIKA	002004	#139-5608 170-6875 355-11428 361-11569 *371-11922 *371-11969
KAMITE	027000	#110-4681 211-7720 217-7929 219-7981 238-8457 388-12280 390-12341 393-12474 397-12547
KDIAG	= 000010	#111-4757 *147-5885 *147-5886
KDPAR0	= 172360	363-11698 #363-11708
KDPAR6	= 172374	#139-5561 253-8825 380-12123 *380-12125 *382-12164 *399-12576 *399-12581
KDPAR7	= 172376	233-8323 233-8331 234-8341 242-8557 244-8586 246-8650 #253-8824
KERNEL	= 104417	#296-9904 296-9920 297-9941 297-9967
		#111-4897 222-8109 222-8111 222-8135 222-8136 222-8155 222-8156 234-8368 234-8371
		236-8424 236-8427 236-8432
		#111-4903 *222-8112 *234-8372
		#111-4904 *222-8137 *222-8157 *236-8425
		#110-4618 169-6788 169-6801 169-6822 169-6845 174-7035 213-7856 213-7864 242-8565
		244-8621 246-8659 255-8860 257-8897 363-11696 365-11755 367-11814 367-11830 377-12078
		386-12243 386-12254 388-12284 388-12302 388-12306 421-13230 421-13246 421-13257 423-13276
		423-13288 426-13341 426-13347 429-13444 429-13448 430-13507 432-13541
KERSTK	= 002000	#111-4689
KFLAG	002526	#139-5720 273-9233 275-9333
KIPAR0	= 172340	#111-4887 361-11529 361-11605 369-11838 369-11858 430-13496 432-13514
KIPAR4	= 172350	#111-4891 *162-6292 *163-6353 *163-6368 *163-6418 *163-6422 *163-6426 *164-6440 *164-6449
		164-6450 *166-6508 *166-6516 166-6602 166-6650 *166-6689 166-6690 166-6692 369-11873
		*369-11874 369-11882 *369-11891 395-12495 *395-12496 *395-12518
KIPAR5	= 172352	#111-4892 *166-6509 *166-6517 166-6653 *166-6670 *166-6676 *166-6682 *166-6690 *321-10488
		*369-11876 *369-11893
KIPAR6	= 172354	#111-4893
KIPDRO	= 172300	#111-4867 361-11607 369-11840 425-13330 428-13406 428-13430 432-13515
KMAP	= 104422	#110-4622 155-6069
KPFLAG	002112	#139-5598 *371-11919 *371-11929
KSIZE	002374	#139-5690
KSTACK	002562	#141-5736 144-5805 166-6518 189-7328 478-15229
LAST	= 157776	#113-4985 211-7694 220-8060 232-8282 234-8351 242-8571 244-8597 244-8625 246-8665
		289-9736 296-9919 297-9940 297-9977 315-10357 323-10650 386-12236 388-12276
LASTBA	002554	#141-5733 *144-5810 *145-5851 166-6501 166-6503 169-6763 170-6886 173-6947 182-7082
		*182-7124 184-7163 191-7360 234-8362 238-8476 238-8483 240-8534 331-10894 331-10903
		361-11586 361-11593 363-11644 363-11651 373-12003 386-12231 388-12273 390-12319 393-12477
		395-12516 397-12550 400-12636
LASTBL	002556	#141-5734 *166-6499 *166-6506 166-6692
LASTER	002014	#139-5568 *189-7301
LBSO	= 000405	157-6122 159-6166 159-6187 160-6271 282-9519 291-9819 316-10406 319-10471 321-10523
		322-10567 323-10650 324-10689 325-10751
LBS1	= 000404	160-6270 291-9817 316-10404 323-10646 325-10750

CZMSPB CREATED BY MACRO ON 6-JUL-82 AT 10:46

PAGE 12
CREF

SYMBOL	VALUE	REFERENCES
LBLS2	= 000403	323-10644 325-10748
LF	= 000012	#111-4703 434-13659
LINK1	002520	#139-5717 *169-6756 *169-6760 169-6775 169-6814 *238-8447 *238-8451 238-8467 238-8496 *240-8519 *240-8521 240-8531 *242-8571 *242-8579 *244-8595 *244-8608 *246-8665 *246-8673 309-10216 311-10263 313-10306
LINK2	002522	#139-5718 *169-6757 *169-6761 169-6800 *244-8598 *244-8609 244-8627 244-8634
LKS	= 177546	#111-4699
LOADBA	002430	#139-5703 *363-11635 365-11733 365-11738
LOADCS	= 104425	#110-4627 282-9490 291-9787 315-10335 315-10350 316-10387 321-10498 321-10510 322-10544 323-10593 323-10601 324-10667 325-10725 326-10783 326-10786 346-11242 346-11250 346-11258 346-11268 352-11349 352-11359 354-11370 355-11444 384-12206 428-13424
LOADHO	002564	#141-5737 189-7333 363-11636 365-11734 365-11742 377-12061
LOOP	014744	#187-7253 189-7347
LOWMAP	046356	363-11706 365-11765 #365-11775 428-13403
LSIZE	002376	#139-5691 *184-7162 *184-7175 184-7199 184-7201 186-7248 509-16410
LSTSS	= *****	157-6122 157-6122 157-6125 157-6125 159-6166 159-6166 159-6187 159-6187 160-6270 160-6270 160-6271 160-6271 282-9519 282-9519 291-9817 291-9817 291-9819 291-9819 316-10404 316-10404 316-10406 316-10406 319-10471 319-10471 321-10523 321-10523 322-10567 322-10567 323-10644 323-10644 323-10646 323-10646 323-10650 323-10650 324-10689 324-10689 325-10748 325-10748 325-10750 325-10750 325-10751 325-10751
MAINT	= 177750	#111-4771 145-5830 154-6018
MAPHO	= 170202	#113-4912 *363-11705 *365-11764 426-13374 *428-13402
MAPKER	046720	227-8229 231-8268 247-8737 248-8747 250-8770 251-8779 251-8795 251-8803 251-8811 #369-11869
MAPLO	= 170200	#113-4911 *363-11703 *365-11764 365-11777 426-13374 *428-13402
MAPL1	= 170204	#113-4913 365-11778
MAPPER	044546	155-6070 169-6772 173-6948 174-6959 211-7719 242-8562 244-8591 246-8655 255-8833 257-8866 #361-11528 377-12073 386-12240 388-12279 390-12340 390-12373 390-12438 393-12485 397-12558 430-13488 432-13529
MASK	002316	#139-5664
MBERR	014004	*182-7087 *182-7103 182-7107 *182-7111 *182-7116 #182-7126
MEMDON	015012	187-7258 #187-7268
MFPT	= 000007	#111-4706 145-5856
MJPAT	020622	149-5925 219-7980 219-7980 219-7985 #219-7993
MJTEST	020516	209-7680 #219-7973
MKCNT	017670	*211-7717 *211-7762 211-7762 #211-7774
MKCONT	017050	209-7675 #211-7686
MKCSRT	020206	149-5911 #215-7878
MKFLAG	002116	#139-5600 172-6915 182-7061 209-7670 240-8527 *371-11919 *371-11945 373-11983
MKLOOP	017232	#211-7715 211-7768
MKPAT	020436	149-5918 217-7928 217-7928 217-7933 #217-7945
MKTEST	020276	191-7356 209-7678 #217-7918
MMRO	= 177572	#111-4780 *341-11150 *341-11154 *363-11708 *363-11717 *365-11756 *365-11759 426-13362 *428-13415 430-13506 *432-13525 509-16409 509-16417
MMR1	= 177574	#111-4781 426-13362 *428-13415 430-13505 *432-13524 509-16409 509-16417
MMR2	= 177576	#111-4782 426-13362 *428-13415 430-13504 *432-13523 509-16409 509-16417
MMR3	= 172516	#111-4783 145-5837 *145-5838 *145-5839 145-5840 *189-7331 *363-11699 *363-11707 *365-11763 *365-11766 *375-12021 *377-12066 426-13365 *428-13414 430-13503 *432-13522 509-16409 509-16417
MMTRAP	042602	147-5899 #339-11121
MMVEC	= 000250	#111-4767 *147-5899 *147-5900
MONFLG	002274	#139-5655 *144-5803 *154-6004 *154-6023 *154-6026 465-14897
MPT	= *****	187-7280 311-10227 339-11127 361-11615 377-12052 377-12067 399-12598 406-12677 425-13301

CZMSPB CREATED BY MACRO ON 6-JUL-82 AT 10:46

PAGE 13
CREF

CZMSPB SYMBOL	CROSS REFERENCE VALUE	REFERENCES
		428-13388 430-13475 434-13609 434-13652 435-13671 462-14765 463-14818 470-15006 473-15126
		479-15275 483-15454 485-15544 485-15560 493-15971
MSEEDH	002574	#141-5741 *147-5878 147-5880
MSEEDL	002576	#141-5742 *147-5879 147-5881
MSG12	J0060	*182-7155 #517-16633
MSG34	076075	211-7755 #517-16575
MSG34	076133	211-7757 #517-16576
MSG000	100272	154-6007 #517-16640
MSG001	074102	331-10891 #517-16522
MSG002	074164	331-10892 #517-16523
MSG003	074241	331-10893 #517-16524
MSG004	074346	331-10901 331-10921 #517-16526
MSG005	074454	333-10948 #517-16528
MSG006	074466	494-16009 #517-16529
MSG007	074523	335-10963 #517-16530
MSG008	100240	159-6153 #517-16639
MSG009	074535	159-6167 335-10994 #517-16531
MSG010	074547	335-11046 #517-16532
MSG011	074561	331-10907 331-10920 #517-16533
MSG012	074647	331-10910 331-10922 #517-16534
MSG013	074744	333-10951 #517-16535
MSG014	074746	159-6164 159-6184 331-10919 333-10953 335-11056 465-14852 472-15054 472-15058 473-15124
MSG015	074750	#517-16536 *159-6162 159-6163 *159-6172 *159-6174 *159-6178 *159-6181 159-6183 *335-10970 *335-10972
		*335-10982 335-10983 *335-11004 *335-11006 *335-11009 *335-11012 335-11013 *335-11025 *335-11035
		335-11036 *335-11049 *335-11053 335-11054 #517-16537
MSG016	074752	159-6154 335-11024 #517-16538
MSG017	074764	331-10906 331-10909 #517-16539
MSG018	074775	470-15013 473-15122 478-15219 478-15234 478-15249 478-15264 #517-16540
MSG019	075000	473-15092 #517-16541
MSG020	075004	380-12113 #517-16542
MSG021	075025	380-12131 #517-16543
MSG022	075613	406-12664 #517-16564
MSG023	075635	384-12203 #517-16565
MSG025	075651	382-12188 384-12212 #517-16566
MSG026	075675	380-12126 #517-16567
MSG027	075707	384-12199 #517-16568
MSG028	075724	384-12208 #517-16569
MSG029	075740	386-12221 #517-16570
MSG030	075760	390-12316 400-12616 #517-16571
MSG031	075777	386-12222 388-12264 #517-16572
MSG032	076037	386-12247 388-12287 #517-16573
MSG033	076056	386-12251 388-12291 #517-16574
MSG035	076161	189-7306 #517-16577
MSG036	076164	388-12263 #517-16578
MSG037	076203	388-12294 #517-16579
MSG038	076222	388-12303 #517-16580
MSG039	076240	388-12296 #517-16581
MSG040	076262	390-12315 #517-16582
MSG041	076306	390-12325 #517-16583
MSG042	076333	390-12329 #517-16584
MSG043	076351	390-12334 #517-16585

CZMSPB	CREATED BY	MACRO	ON	DATE	TIME	PAGE	REF	
SYMBOL	CROSS REFERENCE	VALUE	REFERENCES					
MSG046	076373	390-12347	393-12461	397-12535	#517-16586			
MSG047	076426	#517-16587						
MSG048	076445	380-12116	#517-16588					
MSG049	076505	390-12344	#517-16589					
MSG051	076540	429-13464	#517-16591					
MSG055	076557	393-12452	#517-16592					
MSG056	076600	393-12459	397-12533	#517-16593				
MSG058	076633	478-15210	#517-16594					
MSG061	076655	473-15101	#517-16595					
MSG062	076664	487-15730	487-15789	#517-16596				
MSG063	076704	487-15731	#517-16597					
MSG064	076715	487-15732	487-15791	#517-16598				
MSG065	076725	487-15790	#517-16599					
MSG066	076737	466-14922	#517-16600					
MSG067	077006	468-14974	#517-16601					
MSG070	077015	184-7212	#517-16602					
MSG073	077046	397-12526	#517-16603					
MSG075	077064	363-11646	363-11683	#517-16604				
MSG076	077116	395-12507	#517-16605					
MSG077	077137	189-7304	#517-16606					
MSG079	077153	395-12499	#517-16607					
MSG085	077177	399-12566	#517-16608					
MSG088	077224	478-15228	#517-16609					
MSG089	077242	478-15244	#517-16610					
MSG090	077264	478-15259	#517-16611					
MSG091	077300	478-15253	478-15268	#517-16612				
MSG092	077312	485-15554	#517-16613					
MSG093	077326	485-15556	#517-16614					
MSG095	077334	485-15558	#517-16615					
MSG101	077344	399-12575	#517-16616					
MSG102	077374	399-12580	#517-16617					
MSG103	077423	382-12158	#517-16618					
MSG104	077445	470-15023	#517-16619					
MSG105	077447	400-12612	#517-16620					
MSG106	077522	399-12586	#517-16621					
MSG107	077540	399-12593	#517-16622					
MSG110	077615	400-12627	#517-16623					
MSG111	077661	400-12632	#517-16624					
MSG112	077713	184-7202	#517-16625					
MSG113	077730	184-7206	#517-16626					
MSG114	077745	184-7210	#517-16627					
MSG117	077762	154-6021	#517-16628					
MSG119	077774	154-6040	#517-16629					
MSG120	100003	154-6041	#517-16630					
MSG121	100024	154-6035	#517-16631					
MSG122	100044	182-7156	#517-16632					
MSG125	100112	395-12497	#517-16635					
MSG126	100134	163-6427	#517-16636					
MSG127	100201	402-12646	#517-16637					
MSG128	100220	404-12652	#517-16638					
MSG129	100341	159-6188	#517-16641					
MSIZE	002400	#139-5692	*184-7162	*184-7172	184-7203	184-7205	186-7248	509-16410

CZMSPB		CREATED BY MACRO ON 6-JUL-82 AT 10:46		PAGE 15							
SYMBOL	CROSS REFERENCE	REFERENCES	VALUE	CREF							
MTA030	024602	#240-8514	240-8546								
MTEST	016754	193-7376	195-7406	197-7438	199-7475	201-7512	203-7552	205-7595	207-7635	#209-7666	
MTLA11	030416	273-9183	#273-9188	273-9283							
MTLB11	030430	#273-9191	273-9279								
MTLC11	030442	#273-9194	273-9273								
MTLD11	030536	#273-9211	273-9295								
MTPA03	027514	222-8099	222-8105	#261-8950							
MTPA04	027652	222-8131	222-8134	222-8152	#265-9009	265-9029					
MTPA21	034434	232-8293	232-8307	#292-9829							
MTPA24	035260	234-8366	#299-9993	301-10052							
MTPA25	035670	303-10063	303-10098	305-10106	#305-10117						
MTPA26	036020	236-8410	236-8411	236-8423	#307-10149	307-10192					
MTPB03	027554	222-8102	222-8108	#261-8973	263-9006						
MTPB04	027706	222-8135	*222-8150	*222-8152	222-8155	265-9013	#265-9022	265-9035			
MTPB21	034464	232-8295	232-8310	#292-9846							
MTPB24	035320	234-8360	234-8367	299-10034	#301-10037						
MTPB25	035712	303-10100	305-10108	#305-10124							
MTPB26	036034	236-8414	236-8415	236-8431	#307-10156						
MTPC03	027614	222-8109	261-8975	#263-8991	263-9005						
MTPC21	034520	232-8298	232-8314	#292-9863							
MTPC24	035334	234-8368	301-10043	#301-10045							
MTPC25	035752	303-10102	305-10111	305-10113	#305-10136						
MTPC26	036070	236-8424	307-10150	307-10159	#307-10174						
MTPD03	027632	222-8110	263-8997	#263-8999							
MTPD21	034554	232-8300	232-8317	#294-9881							
MTPD25	035616	303-10075	303-10077	303-10080	303-10082	#303-10098					
MTPD26	036110	*236-8410	*236-8414	236-8426	#307-10187						
MTPE25	035640	303-10086	303-10088	303-10091	303-10093	#305-10106					
MTP000	027404	220-8030	220-8033	#259-8913	259-8915	*328-10856	328-10857	*328-10859	328-10862		
MTP001	027430	220-8052	220-8055	#259-8924							
MTP002	027462	220-8078	220-8081	#259-8936							
MTP005	027726	222-8149	222-8150	222-8154	#265-9031						
MTP006	027762	224-8166	#267-9045								
MTP007	030162	224-8176	#269-9088								
MTP010	030262	224-8183	#271-9128								
MTP011	030370	226-8194	#273-9164								
MTP012	031166	226-8209	#275-9305								
MTP013	031554	226-8218	#277-9387								
MTP014	032270	227-8232	#282-9478								
MTP015	032514	229-8243	#283-9526								
MTP016	033260	229-8252	#285-9625								
MTP017	034042	229-8257	#289-9727								
MTP020	034120	231-8271	#291-9761								
MTP022	034604	233-8326	233-8334	#296-9899							
MTP025	035352	234-8388	#303-10055								
MTP030	036126	240-8518	240-8522	#309-10195							
MTP031	036136	242-8564	#309-10201	309-10223							
MTP032	036214	244-8620	244-8639	#311-10252							
MTP033	036246	246-8658	#313-10272	313-10315							
MTP034	036344	238-8449	238-8466	238-8472	238-8495	238-8501	246-8689	246-8695	246-8701	#314-10318	
		314-10320									
MTP035	036370	246-8722	#315-10330								

CZMSPB SYMBOL	CREATED BY CROSS REFERENCE VALUE	MACRO ON 6-JUL-82 AT 10:46	REFERENCES	PAGE 17 CREF							
MT1	017030		219-8015	219-8016	219-8017	219-8018	#253-8819				
MT2	017034		209-7671	#209-7680							
MUT	002106		209-7679	#209-7681							
			#139-5596	*187-7274	*187-7277	*191-7355	*191-7357	*209-7668	*209-7681	*390-12363	*393-12469
			*397-12542	470-15016	476-15175	*476-15177	476-15191	*476-15193	*476-15199		
NC	056464		434-13637	434-13641	434-13647	#434-13651					
NEMCNT	002066		#139-5588	*167-6714	167-6727	*169-6771	169-6795	169-6807	169-6829	169-6853	*339-11106
			339-11107	*339-11111	509-16408						
NEWBAN	002306		#139-5660	240-8544	*363-11691	363-11700	363-11710	*365-11751	369-11846	369-11859	
NEWKER	046652		363-11709	365-11757	#369-11856						
NEWLOA	046754		363-11638	365-11736	#369-11879						
NOCH	063624		483-15460	#483-15464							
NOERRO	002426		#139-5702	*382-12184	*384-12200	*384-12209	465-14842	465-14879	466-14902	*466-14931	468-14968
			468-14980	470-15020	*476-15188	*476-15194					
NOFSMO	002424		#139-5701	*238-8452	*238-8479	*238-8507	*240-8524	*240-8541	*240-8552	380-12115	
NONEM	002076		#139-5592	*167-6715	*169-6752	*169-683C	*169-6847	339-11104			
NONEXI	042524		167-6716	169-6753	#339-11104						
NOOJ	041476		331-10895	#331-10898							
NOPAR	002074		#139-5591	*144-5815	*144-5821	*157-6097	*164-6439	*167-6713	*169-6751	*169-6793	*211-7736
			*217-7924	*217-7939	*219-7974	*219-7986	*226-8219	*240-8515	*242-8561	*244-8590	*246-8654
			*282-9486	*303-10060	*303-10095	*315-10331	*315-10359	*325-10710	337-11077	337-11081	337-11088
			386-12219	*386-12220	*386-12255	388-12261	*388-12262	*388-12307	*390-12383	*390-12388	393-12450
			*393-12484	397-12524	*397-12557						
NORES	003666		144-5801	#144-5803							
NOSCOF	002436		#139-5706	*232-8276	*232-8319	*234-8343	*234-8380	*240-8524	*240-8541	*240-8552	*242-8559
			*242-8581	*244-8588	*244-8643	*246-8652	*246-8676	462-14777	466-14907		
NOSUPF	002454		#139-5713	*145-5845	*145-5864	155-6045	220-8040	220-8064	242-8572	244-8599	246-8666
			257-8889	355-11390	361-11532	361-11549	361-11574	361-11595	425-13322	426-13343	426-13363
			428-13412	429-13439	430-13501	432-13520	478-15238				
NOTAB	002370		#139-5688	335-10984	335-11014	335-11037	*472-15053	*472-15056	*472-15067	*472-15069	*473-15091
			*473-15094								
NOTRCE	060264		462-14756	#462-14758							
NO22BI	002452		#139-5712	*145-5848	*145-5850	154-6003	166-6497	189-7329	355-11409	361-11588	375-12019
			377-12064								
NULLFL	002342		#139-5674	*253-8821							
NXTCSR	005644		157-6098	#157-6116							
OLDCAC	002300		#139-5657								
OLDCSR	002154		#139-5615	*172-6909	172-6927	*172-6928					
ONES	002602		#141-5744	169-6813	174-6979	174-6981	213-7833	213-7835	222-8125	296-9908	296-9910
			327-10833	327-10838	453-14553						
PADDRE	002034		#139-5576	509-16407							
PAFBAF	016150		187-7263	#201-7500	201-7525	201-7533					
PAFBAW	016300		187-7264	#203-7538	203-7565	203-7570	203-7578				
PARBAF	016452		187-7265	#205-7583	205-7608	205-7616					
PARBAW	016602		187-7266	#207-7621	207-7648	207-7653	207-7661				
PARCNT	002070		#139-5589	*167-6712	167-6724	*169-6770	169-6805	169-6827	169-6849	*282-9485	282-9494
			*282-9509	282-9512	*282-9522	*325-10718	325-10758	325-10763	*337-11079	509-16407	
PARITY	042420		147-5893	#337-11077							
PARTHE	002302		#139-5658	*303-10062	*305-10124	*305-10136	*305-10139	*315-10336	*315-10351	337-11090	386-12219
			*386-12238	*386-12255	388-12261	*388-12277	*388-12307				
PARVEC	= 000114		#111-4765	*147-5893	*147-5894						
PASFLG	002262		#139-5649	*160-6234	*160-6237	160-6239	160-6246	160-6257	160-6264	160-6271	*209-7669

CZMSPB SYMBOL	CREATED BY VALUE	CROSS REFERENCE	MACRO REFERENCES	ON 6-JUL-82 AT 10:46	PAGE 19 CREF								
PERR27	= 104453		#110-4650										
PERR30	= 104454		#110-4651	309-10208									
PERR31	= 104455		#110-4652	273-9225	273-9231	283-9572	283-9578	285-9670	285-9677	319-10467			
PERR32	= 104456		#110-4653	273-9246	273-9259								
PERR33	= 104457		#110-4654	275-9347									
PERR34	= 104460		#110-4655	273-9252	273-9265	275-9355							
PERR35	= 104461		#110-4656	313-10284	313-10291								
PERR36	= 104462		#110-4657	355-11435									
PERR37	= 104463		#110-4658	323-10610									
PERR40	= 104464		#110-4659	323-10624									
PERR41	= 104465		#110-4660										
PERR42	= 104466		#110-4661										
PERR43	= 104467		#110-4662										
PERXOR	057506		454-14600	454-14612	456-14623	#456-14636	458-14667	460-14703					
PFECDF	062004		470-15026	#470-15031									
PFECDH	061744		470-15026	#470-15028									
PFECDT	061774		470-15026	#470-15030									
PFECFM	061710		470-15026	#470-15027									
PFECWS	061700		468-14959	#470-15026									
PFLAG	002120		#139-5601	172-6929	363-11633	*371-11921	*371-11951						
PGMCSR	002530		#139-5721	*162-6291	*162-6337	162-6342	*163-6385	*163-6415	*163-6416	*163-6428	246-8717		
			343-11178	343-11180	343-11184	344-11204	344-11206	344-11210	363-11692	*363-11692	*363-11718		
			*363-11725	365-11758	*365-11758								
PHEBE	014006		#182-7127	*182-7129	*182-7134	182-7137	182-7139	182-7141	182-7143	*182-7145	*182-7147		
			273-9182	273-9239									
PHYADD	002036		#139-5577	*473-15107	*473-15108	*473-15109	*473-15110	*473-15112	473-15113				
PMEMFL	002140		#139-5609	226-8192	226-8201	226-8216	227-8227	229-8241	229-8247	231-8267	234-8386		
			247-8731	248-8743	249-8758	250-8767	251-8777	251-8785	251-8793	251-8801	251-8809		
			*371-11919	*371-11948									
PROTYP	003754		#145-5826	*145-5861	145-5862	145-5871	151-5967	164-6457	167-6732	169-6754	169-6783		
			169-6792	169-6817	220-8028	220-8050	220-8076	222-8096	222-8129	222-8147	232-8285		
			232-8287	234-8352	234-8354	236-8402	236-8404	238-8445	238-8464	238-8470	238-8493		
			238-8499	240-8516	244-8617	244-8637	246-8687	246-8693	246-8699	328-10854	344-11213		
			363-11697	365-11761	382-12186	382-12190	384-12202	384-12214	386-12256	388-12308	426-13372		
			428-13400	462-14790	466-14914	476-15196							
PSIZE	002402		#139-5693	*184-7162	*184-7170	184-7207	184-7209	186-7248					
PSW	= 177776		#111-4694	*155-6049	*155-6050	*155-6056	*169-6782	*169-6799	*169-6816	*169-6842	*174-6962		
			*213-7805	*242-8563	*244-8611	*246-8657	*255-8855	*257-8888	*331-10888	*333-10947	*335-10965		
			*344-11229	*344-11231	*363-11694	*365-11753	*367-11812	*367-11828	*377-12074	*386-12241	*388-12281		
			*388-12299	*390-12359	*393-12465	*397-12539	*421-13228	*421-13244	*421-13254	*423-13274	*423-13286		
			*426-13339	*426-13345	*429-13442	*429-13446	*430-13489	*432-13530	*478-15240	*478-15241	*478-15256		
PTABLE	036736		291-9783	316-10383	#316-10413	325-10721							
PWRVEC	= 000024		#111-4758	*147-5891	*147-5892	*153-5986	*425-13313	*425-13314	*426-13380	*428-13391	*429-13463		
			430-13490	430-13491	*430-13493	*430-13494	*432-13531	*432-13532					
QUICK	002434		#139-5705	*153-5985	377-12057								
QVFLAG	002344		#139-5675	*153-5985	*153-5990	189-7305	*189-7307	246-8724	273-9275	275-9372	279-9441		
			283-9589	287-9691									
RANODD	036050		*236-8413	*236-8430	#307-10164	*307-10168							
RDCHR	= 104411		#110-4610	486-15620									
RDDEC	= 104414		#110-4613	380-12127									
RDLIN	= 104412		#110-4611	406-12665	487-15701	487-15752							
RDOCT	= 104413		#110-4612	384-12204	386-12223	388-12265	388-12297	390-12317	390-12330	390-12335	399-12567		

CZMSPB SYMBOL	CREATED BY CROSS REFERENCE VALUE	MACRO ON 6-JUL-82 AT 10:46	PAGE 20 CREF							
READCS	= 104426	REFERENCES 400-12617								
		#110-4628	166-6545	273-9234	282-9497	291-9791	291-9799	303-10072	305-10129	305-10141
		315-10344	319-10458	321-10502	321-10511	322-10548	322-10557	323-10602	323-10615	324-10671
		324-10678	325-10734	326-10790	326-10800	348-11280	348-11299	350-11315	350-11334	382-12183
		384-12198	384-12207	421-13247	426-13355	478-15217				
READON	002406	#139-5695								
REALPA	002276	#139-5656	*220-8024	*220-8037	*220-8059	*222-8088	*222-8123	*222-8141	*224-8163	*224-8170
		*224-8180	*226-8193	*226-8202	*226-8217	*227-8228	*229-8242	*229-8251	*229-8256	*231-8266
		*232-8277	*233-8325	*233-8333	*234-8344	*234-8387	*236-8394	*238-8443	*240-8514	*242-8560
		*244-8589	*246-8653	*246-8680	*246-8710	*247-8732	*248-8744	*249-8759	*250-8768	*251-8778
		*251-8786	*251-8794	*251-8802	*251-8810	*253-8820	363-11674	363-11681	458-14671	458-14676
		458-14679	458-14682	463-14812	472-15076	485-15559				
REFRES	035160	296-9934	#297-9971							
REFSUB	035230	297-9974	297-9978	#297-9982						
REGCOP	041076	220-8027	220-8049	#327-10816						
RELENT	045624	363-11675	363-11676	#363-11688						
RELOCA	045204	193-7388	195-7418	197-7455	199-7492	201-7530	203-7575	205-7613	207-7658	240-8538
		#363-11625								
RELOC1	045640	#363-11691								
RESREG	= 104416	#110-4616	222-8101	222-8115	234-8376	236-8416	236-8433	328-10860	328-10865	367-11795
		367-11832	380-12117	382-12176	470-15019	491-15885				
		*137-5548	*137-5550	#141-5749	151-5948					
RESTAR	002614	#111-4753	*147-5895	*147-5896						
RESVEC	= 000010	382-12168	382-12173	#382-12176						
RESO	050410	382-12182	#382-12188							
RES1	050470	384-12197	#384-12212							
RES2	050636	#139-5603	193-7385	195-7415	197-7452	199-7489	201-7527	203-7572	205-7610	207-7655
RLFLAG	002124	*363-11726	*365-11760	371-11952	375-12016	377-12056	380-12115	425-13328	463-14814	485-15553
		#139-5602	191-7354	193-7372	195-7402	197-7434	199-7471	201-7510	203-7550	205-7593
		207-7633	211-7723	238-8456	238-8485	240-8528	*371-11921	*371-11951	*371-11954	*371-11962
		390-12343	393-12473	397-12546						
		157-6111	#160-6202							
RWCSR	006210	#139-5614	*390-12348	390-12440						
SAVCSR	002152	#139-5654	*144-5804	465-14898						
SAVMON	002272	#139-5653	*369-11873	369-11891	*369-11892	369-11893				
SAVPAR	002270	#110-4615	222-8098	222-8106	234-8359	234-8365	236-8409	236-8422	328-10851	367-11793
SAVREG	= 104415	367-11796	380-12112	468-14949	491-15870					
SAV4	002266	#139-5652	*395-12495	395-12518						
SBEMSK	002250	#139-5646	*273-9191	*273-9192	273-9207	273-9209	273-9270	*273-9272	*273-9272	*275-9311
		*275-9312	275-9317	275-9319	275-9330	275-9367	*275-9369	*275-9369	*277-9393	*277-9394
		277-9404	277-9406	277-9410	277-9412	279-9442	*279-9444	*279-9444	*279-9454	279-9457
		*279-9464	279-9465	279-9467	*283-9531	*283-9532	283-9542	283-9544	283-9548	283-9550
		283-9590	*283-9592	*283-9592	*283-9601	283-9604	*283-9611	283-9612	283-9614	*285-9632
		*285-9633	285-9642	285-9644	285-9648	285-9650	287-9692	*287-9694	*287-9694	*287-9703
		287-9706	*287-9713	287-9714	287-9716					
SBENT	020150	213-7810	213-7812	213-7817	213-7819	213-7828	213-7834	213-7836	213-7842	213-7844
		213-7853	#213-7862							
SBESYN	034414	291-9801	#291-9826							
SBETES	017672	211-7733	#213-7777							
SCOPE	= 000004	#111-4693	157-6074	167-6705	169-6742	172-6890	187-7253	187-7270	255-8861	257-8898
SDPARO	= 172260	#111-4857	222-8110	222-8112	234-8367	234-8369	234-8370	236-8426		
SDPARS	= 172272	#111-4862	*222-8114	*234-8371						

CZMSPB CREATED BY MACRO ON 6-JUL-82 AT 10:46

PAGE 21
CREF

SYMBOL	VALUE	REFERENCES
SDPAR6	= 172274	#111-4863 *236-8428
SDPAR7	= 172276	#111-4864 *222-8113
SEEDHI	002570	#141-5739 *147-5880 236-8397 *236-8436 489-15843 489-15850 *489-15855
SEEDLO	002572	#141-5740 *147-5881 236-8396 *236-8435 489-15842 489-15848 *489-15854
SELONL	002000	#139-5559 187-7273 211-7690 371-11961 *400-12628 *400-12633
SETPAT	047512	195-7404 199-7473 #373-11995
SHADL1	012454	#170-6872 170-6887
SHUTUP	047676	189-7316 377-12037 #377-12045
SIPARO	= 172240	#111-4847 361-11530
SIPAR3	= 172246	#111-4850 273-9184 *273-9185 355-11392 361-11548
SIPAR5	= 172252	#111-4852 220-8042 220-8066 242-8574 244-8601 246-8668 273-9185 *273-9186 361-11578
		361-11599
SIPAR6	= 172254	#111-4853 220-8042 220-8066 242-8574 244-8601 246-8668 *361-11599
SIPDRO	= 172200	#111-4827 361-11531 425-13332 428-13432
SIZE	= 040000	#113-4986 167-6718 169-6778 169-6811 220-8026 220-8039 220-8061 222-8128 222-8146
		236-8401 238-8461 238-8490 240-8529 244-8594 244-8631 246-8682 328-10853 363-11695
		365-11754 367-11813 367-11829
SKIPKA	002006	#139-5562 382-12160 *382-12162 *399-12576 *399-12582
SKIPMK	002340	#139-5673 172-6916 209-7674 *371-11922 *371-11972
SKJ	060312	462-14760 462-14763 #462-14773
SKPERR	002064	#139-5587 *213-7823 *213-7848 355-11433 *355-11437
SKUB	045614	363-11682 #363-11685
SKUJ	014010	182-7125 #182-7128
SOBK	002560	#141-5735 242-8566
SOBLEN	= 000056	242-8564 242-8569 #309-10223
SOF TPA	002606	#141-5746 246-8683
SOURCE	002310	#139-5661 *273-9202 *275-9315 *277-9408 *283-9546 *285-9646 *303-10058 357-11459
SPLTCS	002236	#139-5643 *172-6919 *173-6941 *211-7696 *211-7706 211-7728 *211-7761 *211-7764 *211-7771
		361-11556 361-11561 *390-12367 *390-12372
SSP	=%000006	#111-4711 *155-6053 *255-8856 *257-8893 426-13346 *429-13443 478-15242
ST	= 177776	#498-16113
STACK	= 002000	#111-4688 111-4689 137-5541 137-5552 141-5736 153-5989 189-7312
START	003656	137-5549 137-5551 #144-5799 377-12042
START1	000300	137-5545 #137-5548
START2	000310	137-5546 #137-5550
START3	000200	#137-5545 517-16655
STAR27	024164	238-8448 #238-8453
STOPOK	002416	#139-5698 *174-7038 462-14773 *462-14774 *486-15629
STRIPE	002364	#139-5686 *296-9913 296-9917 296-9936 *297-9967 297-9967
SUBAAA	004646	149-5931 #151-5944
SUBAAB	004776	#153-5974
SUBAAI	012450	169-6794 #170-6867
SUBAAP	014170	182-7148 #184-7161
SUBAAR	013374	173-6956 #174-7038
SUBAAS	011402	166-6519 #167-6703
SUCCE5	002332	#139-5670 *211-7722 *211-7748 211-7754 *395-12501 395-12506 *395-12508
SUPDOA	002260	#139-5648 *220-8030 *220-8052 *220-8078 *222-8099 *222-8102 *222-8131 *222-8149 *224-8166
		*224-8176 *224-8183 *226-8194 *226-8209 *226-8218 *229-8243 *229-8252 *229-8257 *232-8293
		*232-8295 *232-8298 *232-8300 *233-8326 *233-8334 *234-8360 *234-8369 *234-8388 *236-8411
		*236-8415 *238-8450 *238-8466 *238-8472 *238-8495 *238-8501 *240-8522 *242-8570 *244-8626
		*244-8639 *246-8662 *246-8690 *246-8695 *246-8701 *246-8722 *249-8761 *251-8788 257-8895
		*328-10857

CZMSPB CREATED BY MACRO ON 6-JUL-82 AT 10:46

PAGE 23
CREF

SYMBOL CROSS REFERENCE
SYMBOL VALUE

TAG77\$ 062230
TAG78\$ 062236
TAG79\$ 062316
TAG9\$ 011672
TBG4\$ 027306
TCFIG1 041740
TCFIG2 042100
TCFIG3 042234
TCNFI 041666
TEMP 002432

TESTAD 002410

TESTMO 002550

TIME 002336
TIMEOU 042570

TKVEC = 000060

TMFLAG 002132
TOOMAN 002404
TOTCSR 002222

TRACE 006206
TRAPVE = 000034
TSTBAN 012312
TSTDAT 002244

TSTRD1 043242
TSTREA = 104510
TST1 005524
TST2 011406
TST3 011572
TST4 012560
TST5 014744
TST6 015016
TYPDS = 104405

REFERENCES

470-15003 #473-15101
470-15004 #473-15107
470-15005 #473-15122
169-6758 #169-6762
257-8878 #257-8880
#335-10965 335-10986
#335-10995 335-11016
#335-11025 335-11039
331-10902 331-10913
#139-5704 *166-6549
395-12494
#139-5696 162-6293
*166-6681 *211-7726
213-7804 *219-7976
275-9321 275-9362
285-9656 285-9665
458-14662
#141-5731 *145-5844
242-8563 244-8611
421-13228 421-13244
#139-5672
145-5871 147-5897
384-12202 384-12214
#111-4761 331-10883
*390-12356 *390-12357
*393-12484 397-12524
#139-5606 *201-7521
#139-5694 *458-14657
#139-5636 *157-6127
406-12658 426-13350
#159-6190 *164-6438
#111-4760 *147-5889
169-6774 #169-6837
#139-5645 *273-9194
*273-9210 273-9214
275-9315 *275-9318
277-9408 *277-9411
*283-9535 *283-9536
283-9558 283-9561
*285-9641 285-9646
285-9674 *287-9701
*303-10079 *303-10081
305-10120 *316-10393
344-11215 344-11217
#110-4680 213-7821
#157-6074
#167-6705
#169-6742
#172-6890
#187-7253
#187-7270
#110-4604

169-6797 169-6834 169-6856
472-15055
473-15093
472-15068
331-10923 #333-10947
166-6605 166-6607 *186-7220 186-7237 *186-7238 *395-12492 *395-12493
*162-6294 *162-6295 166-6493 *166-6523 *166-6524 *166-6669 *166-6680
211-7727 *211-7730 *211-7732 *211-7751 211-7751 213-7803
*219-7977 224-8165 224-8182 273-9211 273-9212 273-9298 273-9299
275-9379 277-9390 279-9471 283-9556 283-9563 283-9567 283-9618
287-9720 303-10061 390-12313 *390-12369 *390-12370 *390-12374 *390-12436
169-6782 169-6799 169-6816 169-6842 174-6962 213-7805
257-8888 344-11229 377-12074 386-12241 388-12281 388-12299
423-13274 423-13286
164-6457 167-6732 169-6792 #339-11117 382-12186 382-12190
388-12308
*331-10885 *331-10886 *331-10928 *331-10928 390-12313 390-12313
*390-12434 393-12450 393-12450 *393-12462 *393-12463 *393-12484
*397-12536 *397-12537 *397-12557 *397-12557
*205-7604 *207-7644 373-11981
*466-14931
163-6356 163-6388 166-6494 348-11274 350-11309 354-11365
478-15212
164-6446 *164-6452 *164-6456 *402-12647 *404-12653 462-14755
273-9198 *273-9199 273-9200 273-9201 273-9202 *273-9208
273-9222 273-9228 *273-9287 *273-9288 *275-9313 *275-9314
275-9324 275-9327 *277-9397 *277-9398 *277-9401 *277-9402
*277-9415 *277-9417 277-9419 277-9421 *279-9452 *279-9453
*283-9540 283-9546 *283-9549 *283-9551 *283-9553 *283-9555
283-9575 *283-9599 *283-9600 *285-9636 *285-9637 *285-9640
*285-9651 *285-9653 *285-9655 285-9658 285-9660 285-9667
*303-10056 *303-10057 303-10058 *303-10074 *303-10076 *303-10078
*303-10085 *303-10087 *303-10089 *303-10090 *303-10092 305-10118
453-14558 453-14563 458-14663 458-14665 509-16416 509-16416
#344-11228
273-9248 273-9261 275-9337 275-9349
184-7201 184-7205 184-7209 184-7211 189-7309 395-12494 395-12498 395-12513

CZMSPB CREATED BY MACRO ON 6-JUL-82 AT 10:46

PAGE 25
CREF

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES
WARN2		027526	#261-8962 *327-10842
WARN3		027542	#261-8967 *327-10843
WARN4		027566	#261-8979 *327-10844
WARN5		027602	#261-8984 *327-10845
WARN6		041320	#328-10863
WARN6A		041260	#328-10856
WARN6B		041312	328-10855 #328-10862
WARN7		024142	#238-8450
WASDBE	=	104500	#110-4672
WASSBE	=	104476	#110-4670
WAS1DB	=	104501	#110-4673 174-6972 174-6985 174-7000 174-7015 277-9425
WAS1SB	=	104477	#110-4671
WHICHC		053730	382-12180 384-12195 #406-12657
WOOPEN		055740	430-13492 430-13495 430-13497 432-13513 432-13536 #432-13544 432-13547
WOOPS		055372	425-13328 #430-13485
WOOPSA		055770	*430-13490 *430-13491 430-13492 432-13531 432-13532 432-13535 #432-13547
WOOPUP		055556	430-13492 430-13492 430-13493 430-13495 430-13495 430-13495 #432-13512 432-13536 432-13537
			432-13547
WORST		002566	#141-5738 *147-5875 *197-7447 *199-7484 *203-7567 *207-7650 371-11958
XOCHAR		056332	#434-13601 *434-13635 *434-13638 *434-13639 434-13640 *434-13644 *434-13645 434-13646 483-15459
			483-15461 *483-15462
XXDPCH		002352	#139-5678 *153-5994 466-14921
ZEROS		002334	#139-5671 145-5835 296-9907 296-9911
\$APTHD		066004	137-5543 #493-15978
\$AUTO		002060	#139-5585 *153-5985 *153-5990 155-6062 483-15488
\$BANK		002011	#139-5565 *463-14811
\$BASE		065760	#492-15950
\$BELL		002641	#141-5764 393-12478 397-12551 465-14851
\$CACHF		042732	#341-11165 496-16046
\$CACHN		042706	#341-11158 496-16045
\$CBCSR		043372	#346-11261 496-16089
\$CBREG		044362	#355-11441 496-16104
\$CB1CS		043414	#346-11266 496-16090
\$CDW1		065764	#492-15953
\$CDW2		065766	#492-15954
\$CHARC		056520	*434-13589 *434-13599 *434-13657 #434-13662
\$CHKDI		043766	#352-11353 496-16097
\$CHK1D		044002	#352-11358 496-16098
\$CKSWR		063602	#483-15453 496-16029
\$CLRCS		043744	#352-11344 496-16095
\$CLR1C		043756	#352-11348 496-16096
\$CMTAG		002000	#139-5558 144-5806
\$CMTGE		002542	#139-5725 144-5808
\$CNTLC		064774	483-15502 486-15624 #486-15679
\$CNTLG		065006	483-15490 #486-15681
\$CNTLK		064202	483-15481 #483-15536
\$CNTLU		065001	483-15507 486-15652 #486-15680
\$CPUOP		065732	#492-15923
\$CRLF		002646	#141-5765 184-7198 331-10905 331-10908 395-12514 434-13588 468-14950 468-14977 468-14986
			478-15211 478-15232 478-15247 478-15262 478-15270 483-15516 485-15543 486-15657
\$DBLK		063572	482-15406 482-15436 #482-15444
\$DB20		065564	473-15114 #491-15870

CZMSPB CREATED BY MACRO ON 6-JUL-82 AT 10:46

PAGE 26
CREF

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES
\$DDW0		065770	149-5909 #493-15965
\$DDW1		065772	#493-15966
\$DDW2		065774	#493-15967
\$DDW3		065776	#493-15968
\$DDW4		066000	#493-15969
\$DDW5		066002	#493-15970
\$DEENE		042676	#341-11154 496-16041
\$DEVCT		065714	*189-7340 *462-14749 *462-14751 #492-15910
\$DEVN		065762	#492-15951
\$DOAGA		015350	189-7311 189-7313 #189-7347
\$DOAGN		015244	#189-7323
\$DOWN		054762	#426-13381 426-13382
\$DTBL		063562	482-15409 #482-15440
\$ECCDI		043270	#346-11237 496-16085
\$ECCIN		043316	#346-11245 496-16087
\$ECC1D		043304	#346-11241 496-16086
\$ECC1I		043332	#346-11249 496-16088
\$ENASB		043344	#346-11253 496-16099
\$ENATS		043360	#346-11257 496-16100
\$ENDAD		015234	137-5534 153-5991 154-6002 #189-7319
\$ENERG		042666	#341-11150 496-16040
\$ENV		065724	144-5800 153-5984 #492-15915
\$ENVN		065725	153-5978 153-5981 #492-15919
\$EOP		015070	#189-7295
\$ERFLG		002012	#139-5566 462-14796 *462-14802 *465-14845
\$ERRGE		044112	#355-11387 496-16103
\$ERROR		060566	147-5887 #465-14841
\$ERRTB		066326	468-14965 #500-16130
\$ERRTY		061370	465-14896 #468-14949
\$ERTTL		002616	#141-5750 189-7301 395-12498 395-12500 *465-14853 *465-14855 466-14921
\$ESCAP		002360	#139-5684 *462-14805 466-14910 466-14912
\$ETABL		065724	#492-15914
\$ETEND		066004	#493-15974 493-15984
\$EXHAL		047670	#377-12041
\$ES	=	000001	#157-6125
\$FATAL		065706	*465-14887 #492-15907
\$FILLC		002640	#141-5763 434-13592
\$FILLS		002355	#139-5681 *399-12570
\$GTSWR		063756	#483-15491 496-16028
\$HALT		061152	#466-14905
\$HALT2		066060	#494-16010
\$HIBTS		066004	#493-15979
\$HIOCT		065204	386-12224 388-12266 *487-15723 #487-15734
\$ILLUP		055364	425-13313 428-13391 #429-13470 429-13471
\$INVAL		044062	#354-11378 496-16102
\$ITEMB		002013	#139-5567 *465-14862 465-14864 465-14874 *465-14875 468-14952
\$KERNE		042656	#341-11146 496-16039
\$KMAP		045112	#361-11603 496-16043
\$LF		002647	#141-5767 486-15667
\$LOADC		042750	#343-11175 496-16048
\$LPADR		002610	#141-5747 255-8835 *255-8845 255-8846 *255-8862 257-8868 *257-8878 257-8879 *257-8899 *462-14800 *462-14803 462-14807

CZMSPB CREATED BY MACRO ON 6-JUL-82 AT 10:46

PAGE 28
CREF

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES
\$PER21		057204	#453-14576 496-16065
\$PER22		057224	#453-14581 496-16066
\$PER23		057242	#453-14586 496-16067
\$PER24		057260	#453-14591 496-16068
\$PER25		054016	#421-13226 496-16069
\$PER26		057450	#456-14628 496-16070
\$PER27		057470	#456-14632 496-16071
\$PER30		054244	#423-13271 496-16072
\$PER31		057660	#458-14671 496-16073
\$PER32		057756	#460-14689 496-16074
\$PER33		060024	#460-14698 496-16075
\$PER34		060104	#460-14709 496-16076
\$PER35		060136	#460-14718 496-16077
\$PER36		060172	#460-14726 496-16078
\$PER37		060222	#460-14733 496-16079
\$PER40		060226	#460-14736 496-16080
\$PWRDN		054412	147-5891 #425-13300 429-13463
\$PWRUP		054766	426-13380 #428-13387 432-13543
\$QUES		002645	#141-5765 483-15534 486-15660
\$RAND		065470	#489-15841
\$RDCHR		064322	#486-15586 496-16031
\$RDDEC		065206	#487-15749 496-16034
\$RDLIN		064452	#486-15615 496-16032
\$RDOCT		065036	#487-15698 496-16033
\$READC		043044	#343-11193 496-16049
\$RESRE		065432	#488-15822 496-16037
\$SAVRE		065374	#488-15811 496-16036
\$SAVR6		055370	*426-13378 428-13393 *428-13394 *428-13395 #429-13472 432-13526
\$SCOPE		060232	147-5885 #462-14749
\$STN	=	000001	#157-6074 #167-6705 #169-6742 #172-6890 #187-7253 #187-7270
\$SVLAD		060466	462-14788 462-14797 #462-14803
\$SWR	=	163000	#108-4564 157-6074 167-6705 169-6742 172-6890 187-7253 187-7270
\$SWREG		065726	153-5987 #492-15921
\$TESTN		065710	*465-14848 470-15030 #492-15908
\$TKB		002632	#141-5760 331-10887 331-10927 390-12358 390-12435 393-12464 393-12483 397-12538 397-12556
			434-13638 434-13644 483-15466 483-15498 485-15570 486-15590 486-15596
\$TKS		002630	#141-5759 331-10889 331-10926 390-12360 390-12433 393-12466 393-12482 397-12540 397-12555
			434-13636 434-13642 483-15464 483-15496 485-15568 486-15588 486-15594
\$TN	=	000007	#108-4565 157-6074 157-6074 #157-6074 167-6705 167-6705 #167-6705 169-6742 169-6742
			#169-6742 172-6890 172-6890 #172-6890 187-7253 187-7253 #187-7253 187-7270 187-7270
			#187-7270
\$TPB		002636	#141-5762 434-13651
\$TPFLG		002356	#139-5682 *153-5979 434-13571
\$TPS		002634	#141-5761 434-13633
\$TRAP		066020	147-5889 #494-15994
\$TRAP2		066042	#494-16005 496-16020
\$TRPAD		066062	494-15999 #496-16020
\$STSM		066010	#493-15981
\$STRD		043064	#344-11200 496-16101
\$TTYIN		064750	486-15617 486-15618 486-15640 486-15658 486-15672 #486-15676
\$TYPDS		063356	#482-15399 496-16025
\$TYPE		056206	#434-13571 496-16021

CZMSPB CREATED BY MACRO ON 6-JUL-82 AT 10:46

PAGE 30
CREF

MACRO CROSS REFERENCE

MACRO NAME	REFERENCES									
AND	#108-4554									
ANDB	#108-4557									
BEGIN	#108-4555	#182-7063	#209-7672	#211-7721	#348-11276	#350-11311	#354-11366	#363-11630	#363-11650	#400-12614
BMOV	#426-13351	#428-13419	#478-15213							
	#126-5293	169-6759	220-8033	220-8055	220-8081	222-8105	222-8108	222-8109	222-8110	222-8134
	222-8135	222-8154	222-8155	232-8307	232-8310	232-8314	232-8317	234-8366	234-8367	234-8368
	236-8423	236-8424	236-8426	236-8431	238-8449	240-8518	242-8564	244-8620	246-8658	246-8689
	328-10862	344-11217	363-11695	365-11754	367-11813	367-11829	430-13492	430-13495		
CASE CLEAR	#108-4554	#215-7873	#380-12133							
	#108-4552	#144-5803	#184-7162	#213-7808	#213-7815	#213-7863	#240-8541	#240-8552	#273-9300	#279-9472
	#283-9619	#327-10835	#331-10900	#331-10918	#352-11344	#352-11348	#363-11687	#365-11747	#365-11764	#371-11919
	#371-11921	#371-11922	#466-14931							
CLEARB DLEFT	#108-4552									
	#135-5495	267-9064	267-9083	273-9272	273-9278	275-9369	275-9375	279-9438	279-9444	283-9586
	283-9592	287-9688	287-9694							
DOWNTD ELSE	#108-4555									
	#108-4554	153-5988	153-5993	154-6006	158-6146	159-6173	159-6180	160-6210	160-6241	160-6259
	160-6266	167-6737	170-6878	170-6883	172-6920	173-6952	174-6990	174-7005	182-7075	184-7171
	184-7174	186-7228	217-7921	217-7936	226-8206	246-8697	253-8827	291-9777	291-9813	296-9909
	296-9925	297-9952	316-10376	316-10400	323-10630	325-10714	325-10729	325-10744	325-10762	335-11005
	335-11008	344-11223	348-11293	350-11328	377-12040	382-12163	382-12171	390-12378	393-12456	397-12530
	421-13234	421-13265	434-13648	454-14603	454-14615	458-14664	460-14712	466-14928	478-15252	478-15267
	485-15574									
END	#108-4555	149-5938	149-5940	151-5953	151-5970	153-5980	153-5983	153-5995	153-5996	153-5997
	154-6008	154-6009	155-6065	155-6066	157-6109	157-6115	157-6121	158-6145	158-6148	159-6175
	159-6176	159-6179	159-6182	160-6212	160-6223	160-6233	160-6243	160-6248	160-6261	160-6263
	160-6269	160-6281	167-6739	169-6852	169-6855	170-6881	170-6885	172-6922	172-6931	173-6942
	173-6943	173-6944	173-6945	173-6946	173-6947	173-6954	174-6992	174-7007	174-7016	174-7017
	174-7018	174-7021	#182-7074	#182-7077	182-7079	182-7080	182-7081	182-7082	184-7173	184-7176
	184-7177	184-7178	184-7187	184-7191	184-7192	184-7215	186-7230	186-7241	186-7244	186-7246
	186-7247	186-7250	186-7251	187-7278	189-7308	189-7345	189-7346	191-7359	191-7360	209-7676
	209-7677	211-7746	#211-7750	211-7751	211-7752	211-7753	211-7759	211-7760	211-7762	213-7829
	213-7854	215-7914	217-7923	217-7930	217-7938	219-7982	#226-8191	#226-8200	226-8208	#226-8215
	#227-8226	#229-8240	#229-8250	#231-8265	#234-8385	238-8468	238-8474	238-8475	238-8476	238-8477
	238-8481	238-8497	238-8503	238-8504	238-8505	238-8506	240-8532	240-8533	240-8534	240-8543
	240-8550	246-8706	253-8829	273-9237	273-9238	282-9496	282-9503	282-9517	291-9779	291-9797
	291-9809	291-9816	296-9912	296-9928	296-9931	297-9947	297-9951	297-9956	297-9960	297-9961
	297-9964	297-9967	297-9968	297-9975	297-9980	315-10357	316-10378	316-10396	316-10403	317-10431
	319-10468	321-10508	321-10518	322-10554	322-10564	323-10611	323-10625	323-10642	324-10676	324-10685
	325-10716	325-10731	325-10739	325-10747	325-10761	325-10766	325-10767	326-10797	326-10807	335-11007
	335-11010	344-11225	#348-11284	348-11285	348-11287	348-11288	348-11295	#350-11319	350-11320	350-11322
	350-11323	350-11330	354-11371	354-11373	354-11374	#363-11629	#363-11643	363-11644	363-11647	363-11649
	#363-11662	363-11666	363-11667	363-11668	363-11669	363-11670	363-11677	363-11678	363-11684	363-11686
	363-11704	371-11933	371-11963	377-12043	377-12059	380-12119	380-12153	382-12165	382-12175	390-12127
	390-12337	390-12346	390-12375	390-12382	393-12458	393-12476	393-12477	395-12509	395-12515	395-12516
	395-12517	397-12532	397-12549	397-12550	#400-12621	400-12625	400-12626	400-12641	421-13231	421-13236
	421-13267	423-13277	426-13357	426-13359	426-13360	428-13425	428-13427	428-13428	434-13650	454-14605
	454-14617	458-14666	458-14673	458-14678	458-14681	458-14684	460-14714	462-14753	462-14776	462-14779
	465-14856	465-14857	465-14886	465-14891	465-14894	465-14895	465-14900	466-14909	466-14920	466-14926
	466-14927	466-14930	470-15022	478-15220	478-15222	478-15223	478-15236	478-15251	478-15254	478-15266
	478-15269	485-15555	485-15576							
FATAL	#115-4997	167-6726	167-6730	303-10073	337-11093	339-11118	339-11122	339-11125		

CZMSPB CREATED BY MACRO ON 6-JUL-82 AT 10:46 PAGE 31
 MACRO CROSS REFERENCE CREF

MACRO NAME	REFERENCES									
FOR	#108-4555	149-5934	172-6910	182-7052	184-7166	184-7181	184-7188	186-7222	191-7352	211-7717
	211-7718	211-7726	211-7737	238-8453	238-8454	238-8482	238-8483	240-8525	296-9905	296-9913
	297-9973	315-10333	#348-11277	#350-11312	#354-11367	#363-11631	363-11654	393-12471	395-12502	397-12544
	400-12639	#426-13352	#428-13420	#478-15214	478-15231	478-15246	478-15261			
GOTO	#108-4556	390-12326	393-12479	397-12552	465-14893					
IF	#108-4554	151-5948	151-5963	#153-5989	153-5991	154-6002	154-6003	155-6062	155-6063	157-6107
	157-6112	158-6139	158-6143	159-6169	159-6170	159-6171	159-6177	160-6208	160-6218	160-6228
	160-6239	160-6246	160-6254	160-6257	160-6264	160-6272	160-6276	167-6735	169-6849	169-6853
	170-6874	170-6875	172-6914	172-6915	172-6916	172-6917	172-6927	172-6929	173-6934	173-6950
	174-6987	174-7002	182-7056	182-7061	#182-7064	182-7070	#184-7167	184-7168	184-7169	#184-7189
	184-7197	184-7213	186-7219	186-7226	186-7248	187-7273	189-7305	189-7336	189-7337	191-7354
	#209-7673	209-7674	211-7690	#211-7691	211-7723	211-7754	213-7822	213-7828	213-7847	213-7853
	217-7919	217-7928	217-7934	219-7980	222-8087	226-8189	226-8190	226-8192	226-8198	226-8199
	226-8201	226-8204	226-8213	226-8214	226-8216	227-8224	227-8225	227-8227	229-8238	229-8239
	229-8241	229-8247	#229-8248	229-8249	231-8263	231-8264	231-8267	234-8383	234-8384	234-8386
	238-8456	238-8463	238-8469	238-8478	238-8485	238-8492	238-8498	240-8527	240-8528	240-8535
	246-8691	246-8724	247-8731	248-8743	249-8758	250-8767	251-8777	251-8785	251-8793	251-8801
	251-8809	253-8825	273-9233	273-9235	273-9274	273-9275	275-9371	275-9372	279-9440	279-9441
	282-9494	282-9499	282-9512	283-9588	283-9589	287-9690	287-9691	291-9793	291-9804	296-9906
	#296-9920	#296-9921	#296-9922	296-9934	#297-9941	#297-9942	#297-9943	297-9945	297-9949	297-9954
	297-9958	316-10390	317-10427	319-10464	321-10504	321-10513	322-10550	322-10559	323-10605	323-10619
	324-10672	324-10681	325-10727	325-10735	325-10757	325-10758	#325-10763	326-10792	326-10802	331-10897
	335-10984	335-11002	335-11003	335-11014	335-11037	343-11177	344-11221	348-11281	348-11286	350-11316
	350-11321	354-11372	363-11626	#363-11627	363-11628	363-11633	363-11645	#363-11655	363-11656	363-11657
	363-11658	363-11659	363-11663	363-11671	363-11672	363-11680	371-11931	371-11961	375-12016	377-12038
	377-12056	#377-12057	380-12115	382-12160	382-12169	388-12273	388-12275	388-12276	390-12319	390-12324
	390-12332	390-12333	390-12343	390-12371	390-12376	393-12454	393-12473	395-12500	395-12506	397-12528
	397-12546	400-12619	421-13227	421-13232	421-13240	421-13263	423-13273	425-13328	426-13358	428-13426
	434-13646	454-14598	#454-14599	454-14601	454-14610	#454-14611	454-14613	456-14622	458-14662	458-14671
	458-14674	458-14676	458-14679	458-14682	460-14689	460-14698	460-14709	#460-14710	462-14750	462-14773
	462-14777	462-14780	465-14842	465-14854	465-14879	465-14880	465-14889	465-14892	465-14897	466-14902
	466-14907	466-14913	466-14917	466-14921	470-15020	478-15221	478-15245	478-15260	485-15553	485-15572
IFB	#108-4557	153-5978	153-5981	153-5984	186-7223	186-7239	186-7242	291-9775	291-9811	316-10374
	316-10398	323-10627	325-10712	325-10742	395-12505					
JUMPTO	#108-4556	157-6129								
LEAVE	#108-4556	#182-7073	#182-7076	#211-7749	#348-11283	#350-11318	#363-11642	#363-11661	#400-12620	
LET	#108-4556	#157-6106	#157-6127	#158-6138	#158-6141	#158-6142	#160-6204	#160-6206	#160-6207	#160-6209
	#160-6211	#160-6213	#160-6215	#160-6216	#160-6219	#160-6220	#160-6224	#160-6225	#160-6229	#160-6230
	#160-6234	#160-6238	#160-6240	#160-6242	#160-6249	#160-6250	#160-6252	#160-6255	#160-6256	#160-6273
	#160-6274	#160-6277	#160-6278	#160-6282	#184-7170	#184-7172	#184-7175	#184-7190	#211-7740	#227-8230
	#231-8269	#238-8458	#238-8486	#249-8761	#249-8762	#251-8780	#251-8788	#251-8789	#251-8796	#251-8804
	#251-8812	#251-8813	#282-9485	#282-9486	#282-9487	#282-9489	#282-9491	#282-9492	#282-9501	#282-9510
	#282-9514	#282-9515	#291-9772	#291-9773	#291-9774	#291-9776	#291-9778	#291-9782	#291-9783	#291-9786
	#291-9788	#291-9794	#291-9795	#291-9801	#291-9806	#291-9807	#296-9907	#296-9908	#296-9910	#296-9911
	#296-9917	#296-9918	#296-9923	#296-9924	#296-9926	#296-9927	#296-9929	#296-9930	#296-9936	#296-9937
	#297-9944	#297-9948	#297-9953	#297-9957	#297-9962	#297-9963	#297-9976	#297-9979	#316-10371	#316-10372
	#316-10373	#316-10375	#316-10377	#316-10382	#316-10383	#316-10386	#316-10388	#316-10391	#316-10392	#316-10393
	#316-10394	#317-10421	#317-10422	#317-10424	#317-10426	#317-10428	#317-10429	#319-10441	#319-10444	#319-10445
	#319-10446	#319-10451	#319-10452	#319-10453	#319-10455	#319-10459	#319-10461	#319-10465	#319-10466	#319-10469
	#321-10484	#321-10489	#321-10490	#321-10497	#321-10499	#321-10505	#321-10506	#321-10509	#321-10514	#321-10515
	#321-10519	#321-10520	#322-10539	#322-10540	#322-10543	#322-10545	#322-10547	#322-10551	#322-10552	#322-10561
	#322-10562	#322-10566	#323-10580	#323-10581	#323-10583	#323-10585	#323-10588	#323-10590	#323-10591	#323-10592

CZMSPB CREATED BY MACRO ON 6-JUL-82 AT 10:46 PAGE 34
 MACRO CROSS REFERENCE CREF

MACRO NAME	REFERENCES									
THRU	#108-4555									
TO	#108-4555									
TYPDEC	#125-5257	#184-7201	#184-7205	#184-7209	#184-7211	#189-7309	#395-12494	#395-12498	#395-12513	#472-15044
TYPE	#115-5011	#154-6007	154-6021	154-6035	154-6040	154-6041	159-6153	159-6154	159-6163	159-6164
	159-6167	159-6183	159-6184	159-6188	163-6427	182-7156	#184-7198	184-7202	184-7206	184-7210
	184-7212	189-7304	189-7306	211-7755	211-7757	331-10891	331-10892	331-10893	331-10901	331-10905
	331-10906	331-10907	331-10908	331-10909	331-10910	331-10919	331-10920	331-10921	331-10922	333-10948
	333-10951	333-10953	335-10963	335-10983	335-10994	335-11013	335-11024	335-11036	335-11046	335-11054
	335-11056	363-11646	363-11683	380-12113	380-12116	380-12126	380-12131	382-12158	382-12188	384-12199
	384-12203	384-12208	384-12212	386-12221	386-12222	386-12247	386-12251	388-12263	388-12264	388-12287
	388-12291	388-12294	388-12296	388-12303	390-12315	390-12316	390-12325	390-12329	390-12334	390-12344
	390-12347	393-12452	393-12459	393-12461	393-12478	395-12497	395-12499	395-12507	395-12514	397-12526
	397-12533	397-12535	397-12551	399-12566	399-12575	399-12580	399-12586	399-12593	400-12612	400-12616
	400-12627	400-12632	402-12646	404-12652	406-12664	429-13464	434-13587	465-14851	465-14852	466-14922
	468-14950	468-14974	468-14975	468-14977	468-14984	468-14986	470-15013	470-15023	472-15054	472-15058
	473-15092	473-15101	473-15116	473-15122	473-15124	478-15210	478-15211	478-15219	478-15228	#478-15232
	478-15234	478-15244	#478-15247	478-15249	#478-15253	478-15259	#478-15262	478-15264	#478-15268	478-15270
	481-15371	482-15436	483-15481	483-15490	483-15491	483-15493	483-15502	483-15507	483-15516	483-15534
	485-15543	485-15554	485-15556	485-15558	486-15624	486-15637	486-15643	486-15648	486-15652	486-15657
	486-15658	486-15660	486-15663	486-15667	487-15728	487-15730	487-15731	487-15732	487-15787	487-15789
	487-15790	487-15791	494-16009							
TYPOCS	#123-5204	#211-7756	#386-12244	#388-12295	#388-12304	#395-12510	#472-15076	#472-15082	#473-15123	#485-15557
	#485-15559									
TYPOCT	#121-5158	468-14955	472-15038	478-15218	478-15233	478-15235	478-15248	478-15250	478-15263	478-15265
	483-15492									
UNTIL	#108-4555	157-6122	159-6166	159-6187	160-6270	160-6271	282-9519	291-9817	316-10404	323-10650
	324-10689	325-10748	325-10751							
UNTILB	#108-4557	291-9819	316-10406	319-10471	321-10523	322-10567	323-10644	323-10646	325-10750	
USER	#130-5406	363-11694	365-11753	367-11812	367-11828	426-13339	429-13446			
WHILE	#108-4554	296-9919	297-9940	297-9977						
WHILEB	#108-4557									
\$CALL	#108-4558									
\$RETUR	#108-4558	#213-7860	#213-7868	#253-8826	#253-8828	#363-11648	#363-11685	#363-11727		
\$SUBTS	#119-5119	144-5799	144-5811	145-5824	147-5874	147-5884	149-5904	149-5933	151-5944	151-5956
	153-5974	154-5999	154-6011	155-6043	155-6061	155-6068	158-6131	159-6151	160-6192	164-6430
	166-6460	167-6734	170-6867	182-7047	184-7161	186-7218	187-7284	191-7350	193-7366	195-7396
	197-7426	199-7463	201-7500	203-7538	205-7583	207-7621	209-7666	211-7686	213-7777	215-7872
	217-7918	219-7973	220-8023	220-8036	220-8058	222-8086	222-8122	222-8140	224-8162	224-8169
	224-8179	226-8188	226-8197	226-8212	227-8223	229-8237	229-8246	229-8255	231-8262	232-8275
	233-8322	233-8330	234-8340	234-8382	236-8393	238-8440	240-8511	242-8556	244-8585	246-8649
	246-8679	246-8709	247-8730	248-8742	248-8754	249-8757	250-8766	251-8776	251-8784	251-8792
	251-8800	251-8808	253-8819	253-8824	255-8832	259-8913	259-8924	259-8936	261-8950	261-8973
	263-8991	263-8999	265-9009	265-9022	265-9031	267-9045	269-9088	271-9128	273-9164	275-9305
	277-9387	282-9478	283-9526	285-9625	289-9727	291-9761	292-9829	296-9899	297-9971	299-9993
	301-10037	301-10045	303-10055	307-10149	307-10156	307-10174	307-10187	309-10195	309-10201	311-10252
	313-10272	314-10318	315-10330	316-10362	317-10415	319-10436	321-10477	322-10530	323-10571	324-10654
	325-10694	326-10770	327-10816	327-10822	328-10849	329-10868	331-10882	333-10934	339-11133	354-11363
	354-11378	355-11387	355-11441	357-11448	361-11517	363-11625	365-11730	365-11775	367-11788	369-11836
	369-11856	369-11869	369-11879	369-11890	371-11897	373-11977	373-11988	373-11996	373-12000	375-12007
	377-12036	377-12045	377-12072	379-12082	380-12111	382-12157	382-12179	384-12194	386-12218	388-12260
	390-12312	391-12443	393-12449	395-12490	397-12523	399-12564	399-12574	399-12579	399-12585	399-12592
	400-12611	400-12631	402-12645	404-12651	406-12657	421-13239	423-13280	430-13485	432-13512	454-14596

