

DRV11-WA

DRV11-WA DMA INTERFACE
CZDRV80

AH-T975B-MC
1 OF 1 OCT 1985
COPYRIGHT © 1985

digital
MADE IN USA

1 78 M
A 1:
1

SEQ 000

C-ZDRV8-0 DRV11-WA DMA INTERFAC MACRO Y05.02 Monday 08-Apr-85 16:01 Page 2

.ENABL LC
.REM -

IDENTIFICATION

PRODUCT CODE: AC-T9748 MC
PRODUCT NAME: CZDRV80 - DRV11-WA DMA INTERFACE DIAGNOSTIC
DATE: APRIL 1985
MAINTAINER: COMPUTER SPECIAL SYSTEMS/ISG

COPYRIGHT (C) 1985
DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.

THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A SINGLE
COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION OF THE ABOVE
COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER COPIES THEREOF, MAY NOT
BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON EXCEPT FOR
USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE TERMS. TITLE
TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DEC.

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND
SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.

DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIARELITY OF ITS SOFTWARE
ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.

TABLE OF CONTENTS

40	1.0	ABSTRACT
41	2.0	REQUIREMENTS
42	2.1	EQUIPMENT
43	2.2	STORAGE
44	3.0	LOADING PROCEDURE
45	4.0	STARTING PROCEDURE
46	4.1	PROGRAM START
47	5.0	SOFTWARE SWITCH REGISTER
48	5.1	OPTIONS
49	5.2	CONTROL
50	6.0	ERROR REPORTING
51	6.1	ERROR COMMENT
52	6.2	ERROR DATA
53	7.0	MISCELLANEOUS
54	7.1	DRV11-WA BUS & VECTOR ADDRESS MODIFICATION
55	7.2	XXDP/APT NOTES
56	7.3	POWER FAIL
57	7.4	MULTIPLE DRV11-WA INTERFACE TESTING
58	7.5	RESTRICTIONS
59	8.0	EXECUTION TIME
60	9.0	PROGRAM TEST DESCRIPTIONS
61	9.1	GENERAL
62	9.2	REGISTER TESTS
63	9.3	BYTE ADDRESSING TESTS
64	9.4	'FNCT' TO 'STAT' WRAP AROUND TEST
65	9.5	READY INTERRUPT TEST
66	9.6	NPR DATA TRANSFER TESTS
67	9.7	MAINT MODE NPR DATA TRANSER TESTS
68	9.8	BURST & NON BURST MODE TESTS
69	9.9	'NEX' ERROR CONDITION TEST
70	9.10	Q22 MODE NPR TEST IN MAINT MODE
71	10.0	LISTING

1.0 ABSTRACT

This program was originally written by Hide Nakagawa and has been modified by Kevin Liston (KJL001) and Robert Lockridge (RJL001).

The DRV11-WA diagnostic program is a series of test designed to test all logic functions and data paths accessible with the loop back cable inserted in the user I/O connectors. total program control is accomplished thru the console terminal via the ODT/Console microcode and the provisions of Section 5 of this document. If the system also includes an "REV11"(DMA Refresh), the DMA refresh MUST BE DISABLED and cpu refresh must be enabled.

2.0 REQUIREMENTS

2.1 EQUIPMENT

1. PDP-11/03, PDP-11/23, PDP-11/23+, MICRO/PDP-11, 11/73 Computer
2. DLV11 with I/O type terminal
3. DRV11-WA with loop back cable

2.2 STORAGE

The program uses the lower 8K of memory.

3.0 LOADING PROCEDURE

Load the program using XXDP+ or any standard absolute loader.

4.0 STARTING PROCEDURE

1. Make sure the maintenance loop back cable is inserted in the I/O connectors on the DRV11-WA module.
2. Make sure the device bus addresses agree with the default values defined in section 7.1 if no, change location(s) as desired via the 'address/' odt command.
3. The program will respond by typing the software switch register contents and allowing the user to change its contents by entering octal switch register data terminated by a carriage return - see section 5.0 for switch register options.

Example:

.R ZDRV80

MD C-ZDRVB-0 DRV11-WA DMA INTERFACE DIAG

SWR = 000000 NEW = xxxxxx

```

136      5.0  SOFTWARE SWITCH REGISTER
137      -----
138      5.1  OPTIONS
139
140          SWITCH          OCTAL          FUNCTION
141          -----
142          SW15:1          100000          HALT ON ERROR
143          SW14:1          040000          LOOP ON TEST
144          SW13:1          020000          INHIBIT ERROR TYPEOUTS
145          SW12:1          010000          ENABLE DRV11-WA Q22 MODE TEST
146          SW11:1          004000          INHIBIT ITERATIONS
147          SW10:1          002000          BELL ON ERROR
148          SW09:1          001000          LOOP ON ERROR
149          SW08:1          0004XX          LOOP ON TEST IN SWR <7-0>
150
151      5.2  CONTROL
152
153      1.  The software switch register 'SWREG'(LOC. 176) can be
154          changed by using the console ODT facilities.
155
156      2.  The software switch register can be changed under program
157          control by typing the "Control & G" keys. This
158          keyboard operation will print out the current contents
159          and accept new octal switch register data terminated
160          with a carriage return.
161
162      3.  Once the ODT mode has been entered because of an
163          error condition with bit 15 set (halt on error), step #2
164          above is of no value, so resort to step #1 to alter the
165          software switch register if desired before typing
166          P' (continue).
167
168      4.  If the program is performing reset instructions, several
169          'Control & G' commands may be necessary to be acknowledge
170          by the program.
171
172      6.0  ERROR REPORTING
173      -----
174      6.1  ERROR COMMENT
175
176          All errors are accompanied with an English language descriptive
177          comment as to the type of failure. Further qualification of
178          the error can be obtained if needed from the comment at the
179          error PC or from the test itself.
180
181      6.2  ERROR DATA
182
183          *ERRPC          Listing address where the error was detected
184          *TSTNUM          Test number where the error occurred
185          BUSADR          DRV11-WA bus reg address of concerned operation
186          EXPCT          Data that was expected
187          RCVD           Data that was received
188          ADRS           Memory address of data transfer on error
189
190          *ALWAYS REPORTED

```

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
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246

7.0 MISCELLANEOUS

7.1 DRV11-WA bus & vector address modification

Modify location '\$BASE' if base bus address is not 172410.
Modify location '\$VECT1' if vector address is not 124.

*NOTE: Use the LSI-11 ODT facilities to modify these locations
after program load. No vector assignment above 774 should be
allowed.

7.2 XXDP/APT NOTES

This diagnostic is chainable under XXDP (RE. 7.5)(Requires 8K or more).
This diagnostic does support "APT" and has run under it.

7.3 POWER FAIL

A power failure will cause a restart message on power up, at
which time the program is restarted (only on systems with
non-volatile memory and with appropriate hardware).

7.4 MULTIPLE DRV11-WA INTERFACE TESTING

This program will "auto-size" the number of DRV11-WA's connected.
This diagnostic will test sequentially up to 8 DRV11-WA interfaces
with contiguous bus addresses.

*NOTE: Each of the DRV11-WA's found will be tested using the setting
of SW12 as the default. Therefore, if you want to test more
than one DRV11-WA, *ALL* DRV11-WA's must be in either Q18 mode
(SW10 of E40 = 0 (off) and SW12 = 0) or Q22 mode
(SW10 of E40 = 1 (on) and SW12 = 1). If you mix DRV11-WA's
set for Q18 and Q22 mode, the diagnostic will test the modules
using the setting for SW12 as the default and the module(s)
with SW10 of E40 in conflict with SW12 will fail the
diagnostic.

7.5 RESTRICTIONS

If the system also includes an "REV11"(DMA Refresh), the DMA
refresh must be disabled and CPU refresh must be enabled.

8.0 EXECUTION TIME

Execution time with one DRV11 WA is listed below. An end pass
message indicates all tests have completed on all selected units.

	Q18 MODE	Q22 MODE
NO ITERATIONS	3 sec	5 sec
WITH ITERATIONS	15 sec	15 sec

NOTE: MEMORY IS 128KW

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
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301

9.0 PROGRAM TEST DESCRIPTIONS

9.1 GENERAL

This diagnostic contains a series of independent test designed to test logic funtions and data paths of the DRV11-WA DMA interface. A high degree of testing is accomplished w th the aid of the maintenance loop back cable provided for d'agnostic testing. A complete list of tests is available in the table of contents at the beginning of the listing. The comment field within each test can be beneficial in test understanding.

9.2 REGISTER TESTS

The following registers are read/write & reset tested:

1. WORD COUNT
2. BUFFER ADDRESS
3. EXTENDED BUFFER ADDRESS
4. COMMAND/STATUS
5. DATA BUFFER

9.3 BYTE ADDRESSING TESTS

1. COMMAND/STATUS
2. DATA BUFFER

9.4 'FNCT' to 'STAT' wrap around test

9.5 READY Interrupt test

9.6 NPR Data Transfer tests

The following NPR transfers are checked for correct status.

WORD COUNT, BUFFER ADDRESS & DATA;

1. SINGLE 'DATI' XFER - FLOATING 1/0 PTRN
2. SINGLE 'DATO' XFER - FLOATING 1/0 PTRN
3. 200 'DATI' XFERS - FLOATING 1/0 PTRN
4. 200 'DATO' XFERS - FLOATING 1/0 PTRN
5. SINGLE 'DATI' XFER TO THE TTY PRINTER CSR

9.7 MAINT MODE NPR DATA TRANSFER TESTS

1. That maint mode controls 'FNCT' b'its
2. 200 maint mode xfers - checking status & data
3. 200 maint mode xfers to each 4k available memory

9.8 BURST & NON-BURST MODE TESTS

1. That cpu is locked out in burst mode
2. That cpu is not locked out in non-burst mode


```

303          9.9      'NEX' error condition test
304
305          9.10     Q22 Mode NPR test in maint mode
306
307              1.   Test if MMU available
308              2.   Size memory map
309              3.   Do NPR data transfers to each 8K available memory
310
311          10.0     LISTING
312          -----
313          -
314
315              .NLIST  MEB
316              .NLIST  CND
317
318          167400    $SWR=167400
319          000001    $TN=1
320          172410    ABASE= 172410      ;BASE DRV11WA BUS ADRS EQUATE
321          000124    AVECT1= 000124    ;BASE DRV11WA VECTOR ADRS EQUATE -
322          000001    ADEVM= 1          ;DEFAULT TO ONE DRV11WA
323          106427    MTPS= 106427     ;INSTR EQUATE THAT MOVES BYTE TO PSW
324          000000    UP = 0           ;CODE FOR UPWARDS MAP IN MEM MGMT PDR'S
325          000006    RW = 6           ;CODE FOR READ/WRITE IN MEM MGMT PDR'S
326
327          ;* MISCELLANEOUS DEFINITIONS
328
329          017777    MASK4K= 17777     ;MASK FOR 4K ADDRESS BANK BOUNDARY
330          037777    MASK8K= 37777    ;MASK FOR 8K ADDRESS BANK BOUNDARY
331          000007    MFPT= 7          ;MOVE FROM PROCESSOR TYPE TO R0
332          000100    .*=100
333          000104    .WORD 104,200,2  ;IGNORE IT'S INTERRUPT -JUST DO & RTI
334          000200    .*=1000
335          000002
336
337          001000
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352

```


APT MAILBOX-ETABLE

354

001174		.EVEN			
001174	000000	\$MAIL:		;;APT MAILBOX	
001176	000000	\$MSGTY: .WORD	AMSGTY	;;MESSAGE TYPE CODE	
001200	000000	\$FATAL: .WORD	AFATAL	;;FATAL ERROR NUMBER	
001202	000000	\$TESTN: .WORD	ATESTN	;;TEST NUMBER	
001204	000000	\$PASS: .WORD	APASS	;;PASS COUNT	
001206	000000	\$DEVCT: .WORD	ADEVCT	;;DEVICE COUNT	
001210	000000	\$UNIT: .WORD	AUNIT	;;I/O UNIT NUMBER	
001212	000000	\$MSGAD: .WORD	AMSGAD	;;MESSAGE ADDRESS	
001214		\$MSGLG: .WORD	AMSGLG	;;MESSAGE LENGTH	
001214	000	\$ETABLE:		;;APT ENVIRONMENT TABLE	
001215	000	\$ENV: .BYTE	AENV	;;ENVIRONMENT BYTE	
001216	000000	\$ENVM: .BYTE	AENVM	;;ENVIRONMENT MODE BITS	
001220	000000	\$SWREG: .WORD	ASWREG	;;APT SWITCH REGISTER	
001222	000000	\$USWR: .WORD	AUSWR	;;USER SWITCHES	
		\$CPUOP: .WORD	ACPUOP	;;CPU TYPE,OPTIONS	
		;		BITS 15-11=CPU TYPE	
		;		11/04=01,11/05=02,11/20=03,11/40=04,11/45=05	
		;		11/70=06,PDQ=07,Q=10	
		;		BIT 10=REAL TIME CLOCK	
		;		BIT 9=FLOATING POINT PROCESSOR	
		;		BIT 8=MEMORY MANAGEMENT	
001224	000	\$MAMS1: .BYTE	AMAMS1	;;HIGH ADDRESS,M.S. BYTE	
001225	000	\$MTYP1: .BYTE	AMTYP1	;;MEM. TYPE,BLK#1	
		;		MEM.TYPE BYTE -- (HIGH BYTE)	
		;		900 NSEC CORE=001	
		;		300 NSEC BIPOLAR=002	
		;		500 NSEC MOS=003	
001226	000000	\$MADR1: .WORD	AMADR1	;;HIGH ADDRESS,BLK#1	
		;		MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE	
001230	000	\$MAMS2: .BYTE	AMAMS2	;;HIGH ADDRESS,M.S. BYTE	
001231	000	\$MTYP2: .BYTE	AMTYP2	;;MEM. TYPE,BLK#2	
001232	000000	\$MADR2: .WORD	AMADR2	;;MEM.LAST ADDRESS,BLK#2	
001234	000	\$MAMS3: .BYTE	AMAMS3	;;HIGH ADDRESS,M.S.BYTE	
001235	000	\$MTYP3: .BYTE	AMTYP3	;;MEM. TYPE,BLK#3	
001236	000000	\$MADR3: .WORD	AMADR3	;;MEM.LAST ADDRESS,BLK#3	
001240	000	\$MAMS4: .BYTE	AMAMS4	;;HIGH ADDRESS,M.S.BYTE	
001241	000	\$MTYP4: .BYTE	AMTYP4	;;MEM. TYPE,BLK#4	
001242	000000	\$MADR4: .WORD	AMADR4	;;MEM.LAST ADDRESS,BLK#4	
001244	000124	\$VECT1: .WORD	AVECT1	;;INTERRUPT VECTOR#1,BUS PRIORITY#1	
001246	000000	\$VECT2: .WORD	AVECT2	;;INTERRUPT VECTOR#2BUS PRIORITY#2	
001250	172410	\$BASE: .WORD	ABASE	;;BASE ADDRESS OF EQUIPMENT UNDER TEST	
001252	000001	\$DEVM: .WORD	ADEVM	;;DEVICE MAP	
001254	000000	\$CDW1: .WORD	ACDW1	;;CONTROLLER DESCRIPTION WORD#1	
001256	000000	\$CDW2: .WORD	ACDW2	;;CONTROLLER DESCRIPTION WORD#2	
001260	000000	\$DDW0: .WORD	ADDW0	;;DEVICE DESCRIPTOR WORD#0	
001262	000000	\$DDW1: .WORD	ADDW1	;;DEVICE DESCRIPTOR WORD#1	
001264	000000	\$DDW2: .WORD	ADDW2	;;DEVICE DESCRIPTOR WORD#2	
001266	000000	\$DDW3: .WORD	ADDW3	;;DEVICE DESCRIPTOR WORD#3	
001270	000000	\$DDW4: .WORD	ADDW4	;;DEVICE DESCRIPTOR WORD#4	
001272	000000	\$DDW5: .WORD	ADDW5	;;DEVICE DESCRIPTOR WORD#5	
001274	000000	\$DDW6: .WORD	ADDW6	;;DEVICE DESCRIPTOR WORD#6	
001276	000000	\$DDW7: .WORD	ADDW7	;;DEVICE DESCRIPTOR WORD#7	
001300	000000	\$DDW8: .WORD	ADDW8	;;DEVICE DESCRIPTOR WORD#8	
001302	000000	\$DDW9: .WORD	ADDW9	;;DEVICE DESCRIPTOR WORD#9	
001304	000000	\$DDW10: .WORD	ADDW10	;;DEVICE DESCRIPTOR WORD#10	
001306	000000	\$DDW11: .WORD	ADDW11	;;DEVICE DESCRIPTOR WORD#11	

APT MAILBOX-ETABLE

001310	000000	\$DDW12: .WORD	ADDW12	::DEVICE DESCRIPTOR WORD#12
001312	000000	\$DDW13: .WORD	ADDW13	::DEVICE DESCRIPTOR WORD#13
001314	000000	\$DDW14: .WORD	ADDW14	::DEVICE DESCRIPTOR WORD#14
001316	000000	\$DDW15: .WORD	ADDW15	::DEVICE DESCRIPTOR WORD#15
001320		\$ETEND:		

ERROR POINTER TABLE

```
.SBTTL  ERROR POINTER TABLE
;*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
```

001320

;ERRTB:

355

;ERROR 1

356

EM1

;REGISTER TIMEOUT ERROR

357 001320 023372

DH1

;ERRPC TSTNUM BUSADR EXPCT RCVD

358 001322 024355

DT1

;ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT

359 001324 024510

360 001326 000000

0

361

;ERROR 2

362

EM2

;REGISTER READ/WRITE ERROR

363 001330 023421

DH1

;ERRPC TSTNUM BUSADR EXPCT RCVD

364 001332 024355

DT1

;ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT

365 001334 024510

366 001336 000000

0

367

;ERROR 3

368

EM3

;BUS RESET ERROR

369 001340 023453

DH1

;ERRPC TSTNUM BUSADR EXPCT RCVD

370 001342 024355

DT1

;ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT

371 001344 024510

372 001346 000000

0

373

;ERROR 4

374

EM4

;FNCT BITS FAILED TO SET STAT BITS

375 001350 023473

DH1

;ERRPC TSTNUM BUSADR EXPCT RCVD

376 001352 024355

DT1

;ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT

377 001354 024510

378 001356 000000

0

379

;ERROR 5

380

EM5

;READY INTERRUPT FAILURE

381 001360 023537

DH1

;ERRPC TSTNUM BUSADR EXPCT RCVD

382 001362 024355

DT1

;ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT

383 001364 024510

384 001366 000000

0

385

;ERROR 6

386

EM6

;READY CLR OR SET ERROR

387 001370 023567

DH1

;ERRPC TSTNUM BUSADR EXPCT RCVD

388 001372 024355

DT1

;ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT

389 001374 024510

390 001376 000000

0

391

;ERROR 7

392

EM7

;STATUS ERROR ON XFER

393 001400 023616

DH1

;ERRPC TSTNUM BUSADR EXPCT RCVD

394 001402 024355

DT1

;ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT

395 001404 024510

396 001406 000000

0

397

;ERROR 10

398

EM10

;WORD COUNT ERROR ON XFER

399 001410 023643

DH1

;ERRPC TSTNUM BUSADR EXPCT RCVD

400 001412 024355

ERROR POINTER TABLE

401	001414	024510	DT1	;\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT
402	001416	000000	0	
403				
404			;ERROR 11	
405	001420	023674	EM11	;BUFFER ADRS ERROR ON XFER
406	001422	024355	DH1	;ERRPC TSTNUM BUSADR EXPCT RCVD
407	001424	024510	DT1	;\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT
408	001426	000000	0	
409				
410			;ERROR 12	
411	001430	023726	EM12	;DATA ERROR FROM MEMORY
412	001432	024417	DH2	;ERRPC TSTNUM BUSADR ADRS EXPCT RCVD
413	001434	024524	DT2	;\$ERRPC TSTNUM \$BDADR \$GDADR \$GDDAT \$BDDAT
414	001436	000000	0	
415				
416			;ERROR 13	
417	001440	023755	EM13	;DATA ERROR TO MEMORY
418	001442	024417	DH2	;ERRPC TSTNUM BUSADR ADRS EXPCT RCVD
419	001444	024524	DT2	;\$ERRPC TSTNUM \$BDADR \$GDADR \$GDDAT \$BDDAT
420	001446	000000	0	
421				
422			;ERROR 14	
423	001450	024002	EM14	;SINGLE CYCLE OFF DID NOT LOCK OUT CPU
424	001452	024464	DH3	;ERRPC TSTNUM BUSADR
425	001454	024542	DT3	;\$ERRPC TSTNUM \$BDADR
426	001456	000000	0	
427				
428			;ERROR 15	
429	001460	024050	EM15	;SINGLE CYCLE ON LOCKED OUT CPU
430	001462	024464	DH3	;ERRPC TSTNUM BUSADR
431	001464	024542	DT3	;\$ERRPC TSTNUM \$BDADR
432	001466	000000	0	
433				
434			;ERROR 16	
435	001470	024236	EM16	;NEX LOGIC ERROR
436	001472	024355	DH1	;ERRPC TSTNUM BUSADR EXPCT RCVD
437	001474	024510	DT1	;\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT
438	001476	000000	0	
439				
440			;ERROR 17	
441	001500	024256	EM17	;CYCLE FAILED TO CLK DBR (IN)
442	001502	024355	DH1	;ERRPC TSTNUM BUSADR EXPCT RCVD
443	001504	024510	DT1	;\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT
444	001506	000000	0	
445				
446			;ERROR 20	
447	001510	024315	EM20	;DATA ERROR FROM I/O PAGE (XCSR)
448	001512	024417	DH2	;ERRPC TSTNUM BUSADR ADRS EXPCT RCVD
449	001514	024524	DT2	;\$ERRPC TSTNUM \$BDADR \$GDADR \$GDDAT \$BDDAT
450	001516	000000	0	
451				
452			;;*****	
453			.SBTTL USER DEFINED TAGS	
454			;\$THE FOLLOWING TAGS ARE USER DEFINED	
455			;;*****	
456				
457	001520	000000	\$VERPC: .WORD 0	;VIRTUAL PC LOCATION FOR ERROR TYPEOUT ROUTINE

USER DEFINED TAGS

458 001522 000000	LMAD: .WORD 0	;LAST CONTIGUOUS MEMORY ADDRESS (+2)
459 001524	MEMMAP:	;MEMORY MAP - EACH BIT CORRESPONDS TO 8K
460		
461 001524 000000	.WORD 0	;1ST WORD MAP...128KW
462 001526 000000	.WORD 0	;2ND WORD MAP...256KW
463 001530 000000	.WORD 0	;3RD WORD MAP...384KW
464 001532 000000	.WORD 0	;4TH WORD MAP...512KW
465 001534 000000	.WORD 0	;5TH WORD MAP...640KW
466 001536 000000	.WORD 0	;6TH WORD MAP...768KW
467 001540 000000	.WORD 0	;7TH WORD MAP...896KW
468 001542 000000	.WORD 0	;8TH WORD MAP...1024KW
469 001544 000000	.WORD 0	;9TH WORD MAP...1152KW
470 001546 000000	.WORD 0	;10TH WORD MAP...1280KW
471 001550 000000	.WORD 0	;11TH WORD MAP...1408KW
472 001552 000000	.WORD 0	;12TH WORD MAP...1536KW
473 001554 000000	.WORD 0	;13TH WORD MAP...1664KW
474 001556 000000	.WORD 0	;14TH WORD MAP...1792KW
475 001560 000000	.WORD 0	;15TH WORD MAP...1920KW
476 001562 000000	.WORD 0	;16TH WORD MAP...2048KW
477		
478 001564	TSTMAP:	;TEST MAP - WHICH BANKS ARE SELECTED FOR TEST
479		
480 001564 000000	.WORD 0	;1ST WORD MAP...128KW
481 001566 000000	.WORD 0	;2ND WORD MAP...256KW
482 001570 000000	.WORD 0	;3RD WORD MAP...384KW
483 001572 000000	.WORD 0	;4TH WORD MAP...512KW
484 001574 000000	.WORD 0	;5TH WORD MAP...640KW
485 001576 000000	.WORD 0	;6TH WORD MAP...768KW
486 001600 000000	.WORD 0	;7TH WORD MAP...896KW
487 001602 000000	.WORD 0	;8TH WORD MAP...1024KW
488 001604 000000	.WORD 0	;9TH WORD MAP...1152KW
489 001606 000000	.WORD 0	;10TH WORD MAP...1280KW
490 001610 000000	.WORD 0	;11TH WORD MAP...1408KW
491 001612 000000	.WORD 0	;12TH WORD MAP...1536KW
492 001614 000000	.WORD 0	;13TH WORD MAP...1664KW
493 001616 000000	.WORD 0	;14TH WORD MAP...1792KW
494 001620 000000	.WORD 0	;15TH WORD MAP...1920KW
495 001622 000000	.WORD 0	;16TH WORD MAP...2048KW
496		
497 001624	SAVTST:	;SAVED TEST MAP - USED DURING FIRST PASS ONLY
498		; TEST EACH BANK ONCE.
499 001624 000000	.WORD 0	;1ST WORD MAP...128KW
500 001626 000000	.WORD 0	;2ND WORD MAP...256KW
501 001630 000000	.WORD 0	;3RD WORD MAP...384KW
502 001632 000000	.WORD 0	;4TH WORD MAP...512KW
503 001634 000000	.WORD 0	;5TH WORD MAP...640KW
504 001636 000000	.WORD 0	;6TH WORD MAP...768KW
505 001640 000000	.WORD 0	;7TH WORD MAP...896KW
506 001642 000000	.WORD 0	;8TH WORD MAP...1024KW
507 001644 000000	.WORD 0	;9TH WORD MAP...1152KW
508 001646 000000	.WORD 0	;10TH WORD MAP...1280KW
509 001650 000000	.WORD 0	;11TH WORD MAP...1408KW
510 001652 000000	.WORD 0	;12TH WORD MAP...1536KW
511 001654 000000	.WORD 0	;13TH WORD MAP...1664KW
512 001656 000000	.WORD 0	;14TH WORD MAP...1792KW
513 001660 000000	.WORD 0	;15TH WORD MAP...1920KW
514 001662 000000	.WORD 0	;16TH WORD MAP...2048KW

USER DEFINED TAGS

```

515      .LIST      SEQ
516      .NLIST    MEB
517      .NLIST    CND
518 001664 000000  $MMAP: .WORD 0 ;MEMMAP ADDRESS ENTRY
519 001666 000000  $TMAP: .WORD 0 ;TSTMAP ADDRESS ENTRY
520 001670 000000  $TTST: .WORD 0 ;TEST MAP BIT
521 001672 000000  $TBAR: .WORD 0 ;DRVBAR
522 001674 000000  $TBAE: .WORD 0 ;DRVBAE
523 001676 000000  FLG30K: .WORD 0 ;30K MEMORY FLAG
524 001700 000000  TEMP: .WORD 0 ;TEMPORARY STORAGE
525 001702 000000  LSIFLG: .WORD 0 ;LSI-11 PROCESSOR FLAG
526 001704 000000  LSIQ22: .WORD 0 ;22-BIT CPU SUPPORT FLAG
527 001706 000000  FSTPAS: .WORD 0 ;FIRST PASS FLAG ;KJL001
528 001710 000000  MMAVA: .WORD 0 ;MEMORY MANAGEMENT AVAILABLE FLAG.
529                                     ;BIT 0 = 1 MMU AVAILABLE
530                                     ;BIT 15 = 1 22 BIT ADDRESSING AVAILABLE
531
532      ;DRV11WA      BUS REGISTER ADDRESS POINTERS
533
534 001712 172410  DRVWCR: 172410 ;WORD COUNT
535 001714 172412  DRVBAR: 172412 ;BUFFER ADDRESS
536 001716 172414  DRVCSR: 172414 ;COMMAND/STATUS
537 001720 172416  DRVDBR: 172416 ;DATA BUFFERS
538 001722 172412  DRVBAE: 172412 ;EXTENDED BUFFER ADDRESS ;KJL001
539
540      ;DRV11WA      VECOTR ADDRESS POINTERS
541
542 001724 000124  DRVCT0: 124 ;READY, INDEX & INCOMPLETE DATIO VECTOR
543 001726 000126  DRVCT2: 126 ;NEW PSW ON INTR
544
545      ;COMMON PROGRAM LOCATION(S)
546
547 001730 000000  TSTNUM: 0 ;CONTAINS TEST NUMBER ON ERROR
548 001732 000001  DMAP: 1 ;DEVICE MAP - EACH BIT SET TESTS ONE DRV11WA
549 001734 000000  CORSZ: 0 ;CONTAINS 1ST NON-EXISTANT MEM ADRS
550 001736 025224  DBUFP: DBUF ;CONTAINS CURRENT 4K NPR BUFFER ADRS
551
552      .SBTTL      PROGRAM START
553
554 001740 000005  START: RESET ;INITIALIZE DRV11WA BEFORE TESTING ;KJL001
555      .SBTTL      INITIALIZE THE COMMON TAGS
556      ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
557      MOV $CMTAG,R6 ;FIRST LOCATION TO BE CLEARED
558      CLR (R6)+ ;CLEAR MEMORY LOCATION
559      CMP $SWR,R6 ;;DONE?
560      BNE .-6 ;LOOP BACK IF NO
561      MOV $STACK,SP ;SETUP THE STACK POINTER
562      ;;INITIALIZE A FEW VECTORS
563      MOV $SCOPE,$IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
564      MOV $340,$IOTVEC+2 ;;LEVEL 7
565      MOV $ERROR,$EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
566      MOV $340,$EMTVEC+2 ;;LEVEL 7
567      MOV $TRAP,$TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
568      MOV $340,$TRAPVEC+2;LEVEL 7
569      MOV $PWRDN,$PWRVEC ;;POWER FAILURE VECTOR
570      MOV $340,$PWRVEC+2 ;;LEVEL 7
571      CLR $TIMES ;INITIALIZE NUMBER OF ITERATIONS
572      CLR $ESCAPE ;CLEAR THE ESCAPE ON ERROR ADDRESS

```

INITIALIZE THE COMMON TAGS

```

002052 112737 000001 001115      MOV      #1,$ERMAX      ;;ALLOW ONE ERROR PER TEST
002060 012737 002060 001106      MOV      #,$LPADR      ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
002066 012737 002066 001110      MOV      #,$LPERR      ;;SETUP THE ERROR LOOP ADDRESS
                                ;;SIZE FOR A HARDWARE SWITCH REGISTER, IF NOT FOUND OR IT IS
                                ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
002074 013746 000004              MOV      @ERRVEC,-(SP)      ;;SAVE ERROR VECTOR
002100 012737 002134 000004      MOV      #30000,$ERRVEC      ;;SET UP ERROR VECTOR
002106 012737 177570 001140      MOV      @DSWR,SWR      ;;SETUP FOR A HARDWARE SWICH REGISTER
002114 012737 177570 001142      MOV      @DDISP,DISPLAY      ;;AND A HARDWARE DISPLAY REGISTER
002122 022777 177777 177010      CMP      #-1,$SWR      ;;TRY TO REFERENCE HARDWARE SWR
002130 001012                    BNE      30002$      ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                ;;AND THE HARDWARE SWR IS NOT = -1
002132 000403                    BR      30001$      ;;BRANCH IF NO TIMEOUT
002134 012716 002142      30000$: MOV      #30001$,(SP)      ;;SET UP FOR TRAP RETURN
002140 000002                    RTI
002142 012737 000176 001140      30001$: MOV      @SWREG,SWR      ;;POINT TO SOFTWARE SWR
002150 012737 000174 001142      MOV      @DISPREG,DISPLAY
002156 012637 000004      30002$: MOV      (SP)+,@ERRVEC      ;;RESTORE ERROR VECTOR
002162 005037 001202      CLR      $PASS      ;;CLEAR PASS COUNT
002166 132737 000200 001215      BITB     @APTSIZE,$ENVM      ;;TEST USER SIZE UNDER APT
002174 001403                    BEQ      30003$      ;;YES,USE NON-APT SWITCH
002176 012737 001216 001140      MOV      @SWREG,SWR      ;;NO,USE APT SWITCH REGISTER
002204      30003$:
556 002204 004737 023226      JSR      PC,SETREG      ;;SET UP THE DEFAULT REGISTER ADDRESSES
557 002210 005737 001706      TST      FSTPAS      ;;IS THIS THE FIRST PASS??
558 002214 001402                    BEQ      1$      ;;BR IF YES
559 002216 004737 012474      JSR      PC,RSTVEC      ;;SET UP VECTORS
560 002222 012737 022750 022746 1$: MOV      @WORK1,VECTOR      ;;SET UP THE TABLE VECTOR ADDRESS
561 002230 005737 022744      TST      $SIZE      ;;HAVE WE SIZED FOR # OF DRV11's YET?
562 002234 001024                    BNE      2$      ;;BR IF YES
563 002236 013737 000176 022736      MOV      176,SAV176      ;;SAVE THE SOFTWARE SWR
564 002244 013737 000200 022740      MOV      200,SAV200      ;;SAVE ADDRESS 200
565 002252 013737 000202 022742      MOV      202,SAV202      ;;SAVE ADDRESS 202
566 002260 004737 022366      JSR      PC,DRVSIZ      ;;SIZE THE AVAILABLE DRV11's
567 002264 013737 022736 000176      MOV      SAV176,176      ;;RESTORE THE SOFTWARE SWREG
568 002272 013737 022740 000200      MOV      SAV200,200      ;;RESTORE ADDRESS 200
569 002300 013737 022742 000202      MOV      SAV202,202      ;;RESTORE ADDRESS 202
570 002306 000416      2$: BR      START1      ;;SKIP CONTROL-C RESTART
571 002310 000005      CSTART: RESET      ;;CONTROL-C RESTART HERE
572 002312 004737 023216      JSR      PC,WSTTIM      ;;WASTE SOME TIME SO WE DON'T GET A " "
573 002316 004737 012474      JSR      PC,RSTVEC      ;;RESTORE THE VEC FOR UNIT UNDER TEST
574 002322 005037 001706      CLR      FSTPAS      ;;INITIALIZE FIRST PASS FLAG AGAIN
575 002326 004737 023226      JSR      PC,SETREG      ;;RESET DEFAULT REGISTER ADDRESSES
576 002332 004737 012474      JSR      PC,RSTVEC      ;;RESTORE THE VEC FOR DEFAULT
577 002336 005037 001202      CLR      $PASS      ;;CLEAR THE PASS COUNT
578 002342 104406      GTSWR      ;;CHECK SW SWITCH REGISTER
579 002344 004737 023226      START1: JSR      PC,SETREG      ;;RESET DEFAULT REGISTER ADDRESSES
580 002350 012706 001100      MOV      @STACK,SP      ;;RESET THE STACK POINTER
581 002354 012737 022750 022746      MOV      @WORK1,VECTOR      ;;SET UP THE VECTOR TABLE ADDRESS
582
583
.SBTTL TYPE PROGRAM NAME
;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
002362 005227 177777      INC      #-1      ;;FIRST TIME?
002366 001052                    BNE      30004$      ;;BRANCH IF NO
002370 104401 002436      TYPE      ,30005$      ;;TYPE ASCIZ STRING
.SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
002374 005737 000042      TST      @42      ;;ARE WE RUNNING UNDER XXDP/ACT?

```

GET VALUE FOR SOFTWARE SWITCH REGISTER

```

002400 001012      BNE 30006$      ;;BRANCH IF YES
002402 123727 001214 000001    CMPB $ENV,$1      ;;ARE WE RUNNING UNDER APT?
002410 001406      BEQ 30006$      ;;BRANCH IF YES
002412 023727 001140 000176    CMP SWR,$SWREG      ;;SOFTWARE SWITCH REG SELECTED?
002420 001005      BNE 30007$      ;;BRANCH IF NO
002422 104406      GTSWR          ;;GET SOFT-SWR SETTINGS
002424 000403      BR 30007$
002426 112737 000001 001134 30006$: MOVB $1,$AUTOB      ;;SET AUTO-MODE INDICATOR
002434 000427      30007$:
002434 000427      BR 30004$      ;;GET OVER THE ASCIZ
002514      ;;30005$:      .ASCIZ <CRLF>* MD C-ZDRVB-0 DRV11-WA DMA INTERFACE DIAG *<CRLF>
002514      30004$:
584      ;;*****
585      ;;LET'S SEE HOW MUCH MEMORY WE HAVE
586      ;;*****
587
588 002514 012700 020000      CORSZR: MOV $020000,R0      ;USE R0 TO LOOK
589 002520 012737 002532 00C004    MOV $2,$ERRVEC      ;SET UP TIME OUT RETURN ADRS
590 002526 005720      1$: TST (R0).      ;TAKE A LOOK
591 002530 000776      BR 1$      ;UNTIL TIMEOUT
592 002532 042700 017777      2$: BIC $017777,R0      ;POINT TO 1ST NON-EXSISTANT 4K BLK
593 002536 010037 001734      MOV R0,CORSZ      ;SAVE FOR LATER
594 002542 012737 000006 000004    MOV $ERRVEC+2,$ERRVEC ;RESTORE VECTOR
595 002550 012737 025224 001736      MOV $DBUF,CBUF      ;INITIALIZE TO LOWEST 4K
596 002556 012737 000000 001206      MOV $0,$UNIT      ;SET UP UNIT COUNT
597 002564 013737 001252 001732      MOV $DEVM,DMAP      ;SET THE $ & POSITION OF DRV11WA'S
598 002572 042737 177400 001732      BIC $177400,DMAP      ;UP TO 8 ONLY
599 002600 001406      BEQ RESTR      ;SO CONTINUE AS IF SOMETHING WAS SELECTED
600 002602 032737 000001 001732      BIT $1,DMAP      ;IS 1ST DRV11WA SELECTED?
601 002610 001002      BNE RESTR      ;BR IF SO
602 002612 00C137 012044      JMP NXDEV1      ;NO - SO ADVANCE BASE DRV11WA ADDRESSES
603 002616 000005      RESTR: RESET      ;CLEAR THE REGISTERS
604 002620 004737 023216      JSR PC,WSTTIM      ;WASTE SOME TIME SO WE DON'T GET A " " ;RJL001
605 002624 005737 001706      1$: TST FSTPAS      ;TEST IF FIRST PASS OR AFTER CONTROL C ;RJL001
606 002630 001007      BNE 2$      ;BRANCH IF NOT ;KJL001
607 002632 104401 024107      TYPE, WARN      ;TELL THE OPERATOR TO TURN OFF DMA REFRESH ;KJL001
608 002636 004737 013560      JSR PC,ASKQ22      ;ASK IF CPU SUPPORTS 22-BIT ADDRESSING ;KJL001
609 002642 012737 000001 001706      MOV $1,FSTPAS      ;NO LONGER FIRST PASS ;KJL001
610 002650 106427 000200      2$: MTPS $PR4      ;SET PRIORITY TO HIGHEST LEVEL
611 002654 012706 001100      MOV $STACK,SP      ;ALWAYS RESET STACK PTR
612 002660 013737 001206 001204      MOV $UNIT,$DEVCT      ;LOAD APT COUNTER
613 002666 013700 001206      MOV $UNIT,R0      ;MAKE AN INDEX
614 002672 006300      ASL R0      ;VALUE
615 002674 013760 001712 001260      MOV DRVWCR,$DDWO(R0) ;SAVE THE BUS ADDRESS
616 002702 022737 000001 001252      CMP $1,$DEVM      ;ONLY ONE DEVICE? ;RJL001
617 002710 001405      BEQ MMCK      ;BR IF YES ;RJL001
618 002712 104401 025124      TYPE .UUT      ;TYPE UNIT UNDER TEST MESSAGE ;RJL001
619 002716 013746 001206      MOV $UNIT,-(SP)      ;GET THE UNIT NUMBER ;RJL001
620 002722 104405      TYPDS      ;TYPE THE UNIT NUMBER ;RJL001
621
622      ;;*****
623      ;;*****
624      .SBTTL 18/22 BIT ADDRESSING TEST
625      ;;*****
626      ;* CHECK IF MEMORY MANAGEMENT IS AVAILABLE. IF AVAILABLE, SET IT UP.
627      ;* INITIALIZE THE MEMMAP TABLE
628      ;;*****

```

18/22 BIT ADDRESSING TEST

```

629 002724 012700 001524      MMCK:  MOV    #MEMMAP,RO      ;LOAD RO WITH MEMMAP TABLE ADDRESS
630 002730 012701 000020      MOV    #16.,R1        ;LOAD COUNTER
631 002734 004737 013700      JSR    PC,KBDSRV      ;TEST FOR "+C"                      ;KJL001
632 002740 005020      1$:  CLR    (R0)+        ;CLEAR MEMMAP TABLE ENTRY
633 002742 077102      SOB    R1,1$          ;DECREMENT COUNTER 16. TIMES
634                                ;IF COUNTER NOT = 0 THEN CLEAR NEXT ENTRY
635 002744 005037 001702      CLR    LSIFLG        ;INIT LSI-11 /2/QUAD PROCESSOR FLAG
636 002750 012737 002762 000010  MOV    #2$,RESVEC      ;FIND OUT IF LSI-11/2/QUAD
637 002756 000007      MFPT                ;MFPT INSTRUCTION WILL CAUSE TRAP
638                                ;ON LSI-11/2
639 002760 003404      BR     10$            ;11/23 OR LATER WILL BRANCH
640 002762 062706 000004      2$:  ADD    #4,SP        ;LSI-11 RETURN, CORRECT STACK
641 002766 005237 001702      INC    LSIFLG        ;AND SET LSI-11/2 FLAG
642                                ;*****
643                                ;* CHECK IF MEMORY MANAGEMENT IS AVAILABLE AND SET IT UP IF IT IS
644                                ;*****
645 002772 004737 013700      10$: JSR    PC,KBDSRV      ;TEST FOR "+C"                      ;KJL001
646 002776 005037 001710      CLR    MMAVA        ;CLEAR KT AND 22 BIT ADDRESSING FLAG
647 003002 012737 003142 000004  MOV    #NONKT,#ERRVEC    ;SET UP TIMEOUT TRAP VECTOR
648 003010 005037 177572      CLR    #SRO        ;CLR MEM MGMT STATUS REGISTER
649                                ; WILL TRAP IF NO MMU AVAILABLE                      ;KJL004
650 003014 004737 012326      JSR    PC,MMINIT      ;MEM MGMT INITIALIZATION ROUTINE
651 003020 052737 000001 001710  BIS    #BIT0,MMAVA    ;SET MEM MGMT AVAILABLE FLAG
652 003026 005737 023002      TST    $FLAG        ;TEST IF FIRST PASS ON UNIT # 0
653 003032 001002      BNE    11$            ;SKIP TYPE OUT                      ;KJL003
654 003034 104401 024552      TYPE, MMAMES        ;GO PRINT OUT THE FOLLOWING MESSAGE.
655 003040      11$:                                ;'KT11 AVAILABLE'
656                                ;*****
657                                ;* CHECK IF 22 BIT SYSTEM AVAILABLE AND SET IT UP IF IT IS
658                                ;*****
659 003040 004737 013700      JSR    PC,KBDSRV      ;TEST FOR "+C"                      ;KJL001
660 003044 012737 003114 000004  MOV    #20$,#ERRVECT    ;SET UP FOR TIME OUT VECTOR
661 003052 005037 000000      CLR    #0          ;CLEAR BASE LOCATION 0
662 003056 012737 010000 172344  MOV    #10000,#KIPAR2    ;SET PAR2 TO BASE LOC 140000
663 003064 052737 000020 172516  BIS    #BIT4,#SR3      ;TURN ON 22 BIT ADDRESSING
664 003072 012737 177777 040000  MOV    #-1,#40000      ;NOW WRITE TO BASE LOC 200000 (128KW + 2)
665                                ; WILL CAUSE TRAP IF 22-BIT SYSTEM                      ;KJL004
666                                ; WITH 128KW OR LESS OF MEMORY                      ;KJL004
667 003100 005737 000000      TST    #0          ;IF LOC 0 = 0 (WILL ONLY GET HERE
668                                ; IF 18-BIT SYSTEM WITH                      ;KJL004
669                                ; 128KW OR LESS OF MEMORY)                      ;KJL004
670                                ; OR IF 22-BIT SYSTEM WITH                      ;KJL004
671                                ; MORE THAN 128KW OF MEMORY;KJL004
672 003104 001403      BEQ    20$            ; THEN SYSTEM SUPPORTS 22 BIT ADDRESSING
673 003106 005037 172516      CLR    #SR3        ; ELSE 18 BIT SYSTEM, DISABLE 22 BIT ADDR
674 003112 000464      BR     KTSIZ        ; AND GO SIZE MEMORY
675
676                                ;*****
677                                ;* TIME OUT TRAP TO HERE OR MEMORY EXISTS AT 128KW + 2
678                                ;*
679                                ;* THIS ROUTINE WILL BE EXECUTED IF THE SYSTEM SUPPORTS 22 BIT
680                                ;* ADDRESSING. IT IS ENTERED VIA A TRAP (IF THE SUSTEM SUPPORTS
681                                ;* 22-BIT ADDRESSING BUT DOESN'T HAVE MORE THAN 128KW OF MEMORY) OR
682                                ;* VIA A BRANCH STATEMENT (IF THE SYSTEM SUPPORTS 22-BIT ADDRESSING
683                                ;* AND DOES HAVE MORE THAN 128KW OF MEMORY.
684                                ;*
685                                ;*****

```

18/22 BIT ADDRESSING TEST

```

686 003114 004737 013700 20$: JSR PC,KBDSRV ;TEST FOR "tC" ;KJL001
687 003120 052737 100000 001710 BIS #BIT15,MMAVA ;SET 22 BIT FLAG
688 003126 005737 023002 TST $FLAG ;:TEST IF FIRST PASS ON UNIT # 0 ;KJL003
689 003132 001002 BNE 25$ ;:SKIP TYPE OUT IF NOT FIRST PASS
690 003134 104401 024617 TYPE, AVAL22 ;GO PRINT OUT THE FOLLOWING MESSAGE.
691 ;'22 BIT ADR AVAILABLE"
692 003140 000451 25$: BR KTSIZ ;GO SIZE MEMORY
693
694 ;:*****
695 ;* THIS ROUTINE WILL MAP MEMORY IN 8KW SEGMENTS. SUPPORTS ONLY THE SIZING
696 ;* OF 16 BIT ADDRESSING WITHOUT MEM MGMT SUPPORT
697 ;* STORAGE USED:
698 ;* R0 = MEMMAP POINTER ... LO 128KW
699 ;* R2 = ADDRESS POINTER
700 ;* R3 = BANK POINTER ... LO 128KW
701 ;* FLAG30K = 30K MEMORY FLAG
702 ;* LITERALS:
703 ;* MASK8K = 37777
704 ;:*****
705 003142 NONKT:
706 003142 012706 001100 MOV #STACK,SP ;SET-UP THE STACK
707 003146 012700 001524 MOV #MEMMAP,R0 ;SET UP MEMORY MAP PTR TO LO 128KW
708 003152 005002 CLR R2 ;SET ADDRESS PTR TO 0
709 003154 012703 000001 MOV #1,R3 ;SET UP 8KW BANK POINTER
710 003160 012737 003206 000004 MOV #2,$#ERRVEC ;SET UP TIME OUT VECTOR
711 003166 004737 013700 1$: JSR PC,KBDSRV ;TEST FOR "tC" ;KJL001
712 003172 011222 MOV (R2),(R2)+ ;READ AND WRITE ALL MEMORY
713 003174 032702 037777 BIT #MASK8K,R2 ;IF NOT 8KW BOUNDARY
714 003200 001372 BNE 1$ ; THEN CHECK NEXT LOCATION
715 003202 050310 BIS R3,(R0) ; ELSE SET BANK FLAG IN MEMMAP
716 003204 000422 BR 3$ ; AND DO SOME MORE
717
718 ;:*****
719 ;* TIMEOUT TRAPS TO HERE
720 ;:*****
721 003206 004737 013700 2$: JSR PC,KBDSRV ;TEST FOR "tC" ;KJL001
722 003212 062706 000004 ADD #4,SP ;RESTORE STACK POINTER
723 003216 022702 160000 CMP #160000,R2 ;IF NOT 28KW BOUNDARY
724 003222 001001 BNE 20$ ; THEN BRANCH
725 003224 000405 BR 21$ ; ELSE SET UP POINTERS
726 003226 022702 170000 20$: CMP #170000,R2 ;IF NOT 30KW BOUNDARY
727 003232 001004 BNE 22$ ; THEN BRANCH
728 003234 005237 001676 INC FLG30K ; ELSE SET 30KW MEMORY FLAG
729 003240 050310 21$: BIS R3,(R0) ;SET BANK FLAG IN MEMMAP
730 003242 000407 BR 4$ ;BRANCH ALL DONE
731 003244 052702 037777 22$: BIS #MASK8K,R2 ;POINT TO LAST ADDRESS OF 8KW BANK
732 003250 005202 INC R2 ;POINT TO 1ST ADDRESS OF NEXT BANK
733 003252 106303 3$: ASLB R3 ;UPDATE BANK POINTER
734 003254 032703 000020 BIT #BIT4,R3 ;IF NOT DONE WITH 32KW
735 003260 001742 BEQ 1$ ; THEN TRY SOME MORE
736 003262 000506 4$: BR DISMAP ;GO TYPE OUT MAP
737
738 ;:*****
739 ;* THIS ROUTINE WILL MAP MEMORY IN 8KW SEGMENTS.
740 ;* MEMORY MANAGEMENT REGISTERS KIPAR2 AND KIPAR3 ARE USED TO MAP
741 ;* THE 8KW BANKS OF MEMORY.
742 ;* IF MEMORY EXISTS NEXT TO THE I/O PAGE (I.E. 760000 OR 17760000)
743 ;* THEN THE LAST BANK WILL BE ACKNOWLEDGED AS EXISTING.

```

18/22 BIT ADDRESSING TEST

```

743      ;*      STORAGE USED:
744      ;*      R0      = MEMMAP POINTER
745      ;*      R2      = ADDRESS POINTER
746      ;*      R3      = BANK POINTER
747      ;*      KIPAR2   = MAPPED TO 1ST 4KW OF PRESENT 8KW BANK
748      ;*      KIPAR3   = MAPPED TO 2ND 4KW OF PRESENT 8KW BANK
749      ;*      LITERALS USED:
750      ;*      MASK8K   = MASK OF 8KW (37777)
751      ;*****
752 003264 012706 001100 KTSIZ: MOV    #STACK,SP      ;SET-UP THE STACK
753 003270 012700 001524      MOV    #MEMMAP,R0      ;SET-UP MEMMAP PTR TO FIRST ENTRY
754 003274 012702 040000      MOV    #40000,R2      ;INIT VIRTUAL ADDRESS TO 0 MAPPED THRU PAR2
755 003300 005037 172344      CLR     @KIPAR2      ;INIT PAR2 TO LOC 0
756 003304 012737 000200 172346 MOV    #200,@KIPAR3 ;SET PAR3 TO 2ND 4KW BANK
757 003312 012737 003352 000004 MOV    #3,@ERRVEC ;LOAD TIME OUT VECTOR
758 003320 012703 000001      1$: MOV    #BIT0,R3      ;SET-UP 8KW BANK POINTER
759 003324 105777 175614      2$: TSTB   @TKS      ;HAS A KEY BEEN HIT ;RJL001
760 003330 100002      BPL     22$      ;BR IF NOT ;RJL001
761 003332 004737 013710      JSR     PC,KBDSR1      ;TEST FOR "+C" ;RJL001
762 003336 011222      22$: MOV    (R2),(R2)+      ;READ AND WRITE ALL MEMORY
763 003340 032702 037777      BIT     #MASK8K,R2      ;IF NOT 8KW BOUNDARY
764 003344 001367      BNE     2$      ; THEN TRY SOME MORE
765 003346 050310      BIS      R3,(R0)      ; ELSE SET BANK FLAG IN MEMMAP
766 003350 000426      BR       5$      ;AND GO UPDATE VARIABLES AND CONTINUE
767
768      ;*****
769      ;*      TIMEOUT TRAPS TO HERE
770      ;*****
771 003352 004737 013700      3$: JSR     PC,KBDSRV      ;TEST FOR "+C" ;KJL001
772 003356 062706 000004      ADD     #4,SP      ;RESTORE STACK POINTER
773 003362 022702 060000      CMP     #60000,R2      ;IF NOT POSSIBLY THE I/O PAGE
774 003366 001017      BNE     5$      ; THEN GO TEST SOME MORE
775 003370 005737 001710      TST     MAVA      ; ELSE IF 22 BIT ADDRESSING (BIT 15 SET)
776 003374 100406      BMI     4$      ; THEN GO SEE IF 2M I/O PAGE
777 003376 022737 007600 172346 CMP     #7600,@KIPAR3 ; ELSE IF NOT I/O BOUNDARY FOR 18 BITS
778 003404 001010      BNE     5$      ; THEN GO UPDATE VARIABLES AND TRY SOME MORE
779 003406 050310      BIS      R3,(R0)      ; ELSE SET BANK EXISTS IN MEMMAP
780 003410 000433      BR       7$      ; AND GO TYPE MEMORY MAP
781 003412 022737 177600 172346 4$: CMP     #177600,@KIPAR3 ;IF NOT 2M I/O BOUNDARY
782 003420 001002      BNE     5$      ; THEN GO TRY SOME MORE SIZING
783 003422 050310      BIS      R3,(R0)      ; ELSE SET BANK IN MEMMAP
784 003424 000425      BR       7$      ; AND GO TYPE MEMORY MAP
785 003426 062737 000400 172344 5$: ADD     #400,@KIPAR2 ;UPDATE MAP TO NEXT
786 003434 062737 000400 172346 ADD     #400,@KIPAR3 ; 8KW BANK
787 003442 012702 040000      MOV    #40000,R2      ;RESTORE ADDRESS POINTER TO 1ST ADDRESS OF THIS BANK
788 003446 005737 001710      TST     MAVA      ;IF 22 BIT ADDRESSING (BIT 15 SET)
789 003452 100403      BMI     6$      ; THEN GO TEST SOME MORE
790 003454 006303      ASL     R3      ; ELSE 18 BIT ADDR. = 1 WORD
791 003456 001322      BNE     2$      ; IF NOT END OF 18 BIT ADDR
792      ; THEN GO SIZE SOME MORE
793 003460 000407      BR       7$      ; ELSE ALL DONE 18 BIT SIZING.
794 003462 006303      6$: ASL     R3      ;UPDATE BANK POINTER
795 003464 001317      BNE     2$      ;IF NOT END OF THIS MEMMAP ENTRY, THEN CONTINUE
796 003466 062700 000002      ADD     #2,R0      ; ELSE UPDATE TO NEXT MEMMAP ENTRY
797 003472 022700 001564      CMP     #MEMMAP+40,R0 ;IF NOT END OF MEMMAP TABLE
798 003476 001310      BNE     1$      ; THEN GO SIZE SOME MORE
799      ; ELSE ALL DONE SIZING

```


18/22 BIT ADDRESSING TEST

```

800 003500          7$:          ; FALL THROUGH AND GO TYPE OUT MEMORY MAP
801
802                ;;*****
803                ;* ROUTINE WILL TYPE OUT MEMMAP, LOAD TEST MAP (SAVTST) AND CHECKS
804                ;* TO INSURE LOWEST 16KW OF MEMORY IS AVAILABLE FOR TEST TO RUN
805                ;* STORAGE LOCATIONS:
806                ;*      R0      = MEMMAP POINTER ... LO 128KW
807                ;*      R1      = COUNTER
808                ;*      R2      = SAVTST POINTER ... LO 128KW
809                ;;*****
810
811 003500 005737 023002 DISMAP: TST      $FLAG      ;TEST IF FIRST PASS ON UNIT # 0      ;KJL003
812 003504 001042          BNE      TST1          ;SKIP MEMORY MAP TYPEOUT IF NOT FIRST PASS
813 003506 012737 000001 023002      MOV      #1,$FLAG      ;SET THE UNIT PASS FLAG      ;KJL003
814 003514 012700 001524          MOV      #MEMMAP,R0      ;LOAD R0 WITH MEMMAP ADR
815 003520 104401 024657          TYPE,    MEMMES        ;GO PRINT OUT THE FOLLOWING MESSAGE.
816                                     ;'MEMORY MAP'
817 003524 004737 015772          JSR      PC,TYPEMAP      ;GO TYPE THE MAP
818 003530 104401 001171          TYPE,    $CRLF          ;GO PRINT OUT THE FOLLOWING MESSAGE.
819                                     ;'<CR><LF>'
820 003534 012702 001624          MOV      #SAVTST,R2      ;LOAD ADR OF SAVTST TABLE TO BE CLEARED
821 003540 012700 001524          MOV      #MEMMAP,R0      ;LOAD ADR OF MEMMAP TAB'E
822 003544 012701 000020          MOV      #16,R1          ;LOAD COUNTER
823 003550 004737 013700          1$:      JSR      PC,KBDSRV      ;TEST FOR "+C"      ;KJL001
824 003554 005012          CLR      (R2)          ;CLEAR SAVTST TABLE ENTRY
825 003556 012022          MOV      (R0)+,(R2)+        ;LOAD SAVTST FROM MEMMAP
826 003560 077105          SOB      R1,1$          ;DECREMENT CTR 16 TIMES
827 003562 013700 001524          MOV      MEMMAP,R0      ;LOAD R0 WITH MAP OF 1ST 128KW
828 003566 042700 177774          BIC      #177774,R0      ;MASK ALL BUT BOTTOM 16KW
829 003572 022700 000003          CMP      #3,R0          ;IF BOTTOM 16KW IS ALL THERE
830 003576 001405          BEQ      TST1          ;THEN GO RUN
831 003600 005037 177572          CLR      #SRO          ;DISABLE MMU
832 003604 104401 024712          TYPE,    INSUFF        ;GO PRINT OUT THE FOLLOWING MESSAGE.
833                                     ;'FIRST 16KW OF MEMORY NOT ALL THERE!'
834 003610 000000          HALT          ;FATAL ERROR HALT
835                                     ;MEMORY IS NOT CONFIGURED TO RUN THIS PROGRAM
836
837                ;;*****
838                ;*TEST 1      TEST THAT ALL DRV11WA REGS ARE ACCESSIBLE
839                ;;*****
840          003612 000240          TST1:    <NOP>
841          003614 012737 003630 001106      MOV      #10,$LPADR      ;;SET SCOPE LOOP ADDRESS
842          003622 012737 000001 001200      MOV      #1,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
843
844 838
845 003630 012737 000001 001102 10$:      MOV      #1,$TSTNM      ;SET TO TEST #1
846 003636 012737 003672 001110      MOV      #1,$LPERR      ;SET UP SCOPE LOOP ADRS
847 003644 005037 001124          CLR      $GDDAT          ;NO DATA COMPARE
848 003650 005037 001126          CLR      $BDDAT          ;NO DATA COMPARE
849 003654 012737 003714 000004          MOV      #2,$@ERRVEC      ;SET UP TIMEOUT RETURN ADRS
850 003662 013700 001712          MOV      DRVWCR,R0      ;SET UP 1ST DRV11 BUS ADRS
851 003666 012701 000004          MOV      #4,R1          ;SET UP REG COUNT
852 003672 004737 013700          1$:      JSR      PC,KBDSRV      ;TEST FOR "+C"      ;KJL001
853 003676 010037 001122          MOV      R0,$BADDR      ;SET UP CURRENT DRV BUS ADRS
854 003702 005710          TST      (R0)          ;SEE IF THERE
855 003704 005720          TST      (R0)+        ;BUMP TO NEXT
856 003706 005301          DEC      R1          ;COUNT 4 OF THEM
857 003710 001403          BEQ      3$          ;BR IF ALL DONE

```

T1 TEST THAT ALL DRV11WA REGS ARE ACCESSIBLE

```

852 003712 000767      BR      1$      ;TRY NEXT
853 003714 022626      2$: CMP      (SP)+,(SP)+ ;FIX STACK SINCE NO RTI
854 003716 104001      ERROR+1 ;BUS ADRS INDICATED DID NOT RESPOND
855 003720 012737 000006 000004 3$: MOV      @ERRVEC+2,@ERRVEC ;RESTORE LOC 4
856
857      ;*****
      ;*TEST 2      TEST THAT THE WORD COUNT REG IS WRITE/READABLE (FLOAT 0 COM PTRN)
      ;*****
      TST2: SCOPE
858      003726 000004
859 003730 012737 003754 001110      MOV      #1$,$LPERR ;SET UP SCOPE LOOP ADRS
860 003736 013737 001712 001122      MOV      DRVWCR,$BDADR ;SET UP WC REG ADRS
861 003744 005000      CLR      RO ;RO SAYS SHIFT PTRN WHEN 0
862 003746 012737 177776 001124      MOV      #-2,$GDDAT ;FLOAT 0 RIGHT TO LEFT
863 003754 004737 013700      1$: JSR      PC,KBDSRV ;TEST FOR "+C" ;KJL001
864 003760 013777 001124 175724      MOV      $GDDAT,@DRVWCR ;LD WC
865 003766 017737 175720 001126      MOV      @DRVWCR,$BDDAT ;READ IT BACK
866 003774 023737 001124 001126      CMP      $GDDAT,$BDDAT ;CORRECT?
867 004002 001401      BEQ      2$ ;BR IF SO
868 004004 104002      ERROR+2 ;WORD COUNT WRITE/READ FAILURE
869 004006 005137 001124      2$: COM      $GDDAT ;COMPELEMENT ZERO
870 004012 005100      COM      RO ;RO SAYS SHIFT LEFT WHEN = 0
871 004014 001357      BNE      1$ ;TRY THE COMPLEMENT IF RO NOT 0
872 004016 006337 001124      ASL      $GDDAT ;COMPLEMENT WAS DONE = NOW SHIFT ZERO LEFT
873 004022 005237 001124      INC      $GDDAT ;KEEP LSB SET
874 004026 103752      BCS      1$ ;AGAIN TILL ALL PATTERNS DONE
875
876      ;*****
      ;*TEST 3      TEST THAT THE BUFFER ADDRESS REG IS WRITE/READABLE (FLOAT 0 COM PTRN)
      ;*****
      TST3: SCOPE
877      004030 000004
878 004032 012737 004056 001110      MOV      #1$,$LPERR ;SET UP SCOPE LOOP ADRS
879 004040 013737 001714 001122      MOV      DRVBAR,$BDADR ;SET UP BA REG ADRS
880 004046 005000      CLR      RO ;RO SAYS SHIFT PTRN WHEN 0
881 004050 012737 177774 001124      MOV      #-4,$GDDAT ;FLOAT 0 RIGHT TO LEFT
882 004056 004737 013700      1$: JSR      PC,KBDSRV ;TEST FOR "+C" ;KJL001
883 004062 005777 175624      TST      @DRVWCR ;CLR BAE FLAG TO ENSURE BAR ACCESS NEXT ;KJL001
884 004066 013777 001124 175620      MOV      $GDDAT,@DRVBAR ;LD BAR
885 004074 005777 175612      TST      @DRVWCR ;CLR BAE FLAG TO ENSURE BAR ACCESS NEXT ;KJL001
886 004100 017737 175610 001126      MOV      @DRVBAR,$BDDAT ;READ IT BACK
887 004106 042737 000001 001126      BIC      #BIT00,$BDDAT ;DON'T WANT BIT00
888 004114 023737 001124 001126      CMP      $GDDAT,$BDDAT ;CORRECT?
889 004122 001401      BEQ      2$ ;BR IF SO
890 004124 104002      ERROR+2 ;BUS ADRS WRITE/READ FAILURE
891 004126 005137 001124      2$: COM      $GDDAT ;COMPLEMENT ZERO
892 004132 042737 000001 001124      BIC      #BIT00,$GDDAT ;BIT 00 NOT INVOLVED
893 004140 005100      COM      RO ;RO SAYS SHIFT LEFT WHEN = 0
894 004142 001345      BNE      1$ ;TRY THE COMPLEMENT IF RO NOT 0
895 004144 006337 001124      ASL      $GDDAT ;COMPLEMENT WAS DONE - NOW SHIFT ZERO LEFT
896 004150 103004      BCC      TST4 ;NEXT TEST IF AIT IS DONE
897 004152 062737 000002 001124      ADD      #2,$GDDAT ;KEEP ADDR LSB SET
898 004160 000736      BR      1$ ;AGAIN TILL ALL PATTERNS DONE
899
900      ;*****
      ;*TEST 4      TEST THAT THE EXTENDED ADDRESS REG IS WRITE/READABLE (FLOAT 0 COM PTRN)
      ;*****

```

T4 TEST THAT THE EXTENDED ADDRESS REG IS WRITE/READABLE (FLOAT

```

004162 000004          TST4:  SCOPE
901
902 004164 032777 010000 174746      BIT    #SW12,$SWR      ;TEST IF Q22 MODE
903 004172 001474          BEQ     TST5      ;;NEXT TEST IF DRV11-WA NOT SET FOR Q22
904 004174 012737 004222 001110      MOV     #1,$LPERR    ;SET UP SCOPE LOOP ADRS
905 004202 013737 001714 001122      MOV     DRVBAR,$BDADR  ;SET UP BA REG ADRS
906 004210 005000          CLR     R0        ;RO SAYS SHIFT PTRN WHEN 0
907 004212 012701 177776      MOV     #-2,R1      ;FLOAT 0 RIGHT TO LEFT
908 004216 005777 175470          TST     @DRVWCR      ;CLR BAE FLAG TO ENSURE BAR ACCESS NEXT ;KJL001
909 004222 004737 013700          1$:  JSR     PC,KBDSRV    ;TEST FOR "+C" ;KJL001
910 004226 010137 001124          MOV     R1,$GDDAT    ;COPY TO WORK AREA
911 004232 013777 001124 175454      MOV     $GDDAT,@DRVBAR ;LD BAR
912 004240 005137 001124          COM     $GDDAT      ;COMPLEMENT TO ZERO
913 004244 013777 001124 175450      MOV     $GDDAT,@DRVBAE ;LD BAE
914 004252 010137 001124          MOV     R1,$GDDAT    ;SET EXPECTED DATA
915 004256 042737 000001 001124      BIC     #BIT00,$GDDAT ;BIT00 NOT INVOLVED
916 004264 017737 175424 001126      MOV     @DRVBAR,$BDDAT ;READ BAR BACK
917 004272 042737 000001 001126      BIC     #BIT00,$BDDAT ;DON'T WANT BIT00
918 004300 023737 001124 001126      CMP     $GDDAT,$BDDAT ;CORRECT?
919 004306 001401          BEQ     2$              ;BR IF SO
920 004310 104002          ERROR+2      ;BUS ADRES WRITE/READ FAILURE
921 004312 010137 001124          2$:  MOV     R1,$GDDAT    ;SET EXPECTED DATA
922 004316 005137 001124          COM     $GDDAT      ;COMPLEMENT ZERO
923 004322 042737 177700 001124      BIC     #177700,$GDDAT ;DON'T WANT UPPER BITS
924 004330 017737 175366 001126      MOV     @DRVBAE,$BDDAT ;READ BAE BACK
925 004336 023737 001124 001126      CMP     $GDDAT,$BDDAT ;CORRECT?
926 004344 001401          BEQ     3$              ;BR IF SO
927 004346 104002          ERROR+2      ;BUS ADRES WRITE/READ FAILURE
928 004350 005101          3$:  COM     R1              ;COMPLEMENT ZERO
929 004352 005100          COM     R0              ;RO SAYS SHIFT LEFT WHEN = 0
930 004354 001322          BNE     1$              ;TRY THE COMPLEMENT IF RO NOT 0
931 004356 006301          ASL     R1              ;COMPLEMENT WAS DONE - NOW SHIFT ZERO LEFT
932 004360 103001          BCC     TST5          ;;NEXT TEST IF AIT IS DONE
933 004362 000717          BR      1$              ;AGAIN TILL ALL PATTERNS DONE
934
;;*****
;*TEST 5      TEST THAT THE DATA BUFFER REG IS WRITE/READABLE (FLOAT 0 COM PTRN)
;;*****
004364 000004          TST5:  SCOPE
935
936 004366 012737 004412 001110      MOV     #1,$LPERR    ;SET UP SCOPE LOOP ADRS
937 004374 013737 001720 001122      MOV     DRVDBR,$BDADR  ;SET UP DB REG ADRS
938 004402 005000          CLR     R0        ;RO SAYS SHIFT PTRN WHEN 0
939 004404 012737 177776 001124      MOV     #-2,$GDDAT    ;FLOAT 0 RIGHT TO LEFT
940 004412 004737 013700          1$:  JSR     PC,KBDSRV    ;TEST FOR "+C" ;KJL001
941 004416 013777 001124 175274      MOV     $GDDAT,@DRVDBR ;LD DB
942 004424 017737 175270 001126      MOV     @DRVDBR,$BDDAT ;READ IT BACK
943 004432 023737 001124 001126      CMP     $GDDAT,$BDDAT ;CORRECT?
944 004440 001401          BEQ     2$              ;BR IF SO
945 004442 104002          ERROR+2      ;DATA BUFFER WRITE/READ FAILURE (LOOP BACK)
946 004444 005137 001124          2$:  COM     $GDDAT    ;COMPLEMENT ZERO
947 004450 005100          COM     R0              ;RO SAYS SHIFT LEFT WHEN = 0
948 004452 001357          BNE     1$              ;TRY THE COMPLEMENT IF RO NOT 0
949 004454 006337 001124          ASL     $GDDAT    ;COMPLEMENT WAS DONE - NOW SHIFT ZERO LEFT
950 004460 005237 001124          INC     $GDDAT    ;KEEP LSB SET
951 004464 103752          BCS     1$              ;AGAIN TILL ALL PATTERNS DONE
952
953
;;*****

```

T6 TEST THAT THE DATA BUFFER REG IS BYTE ADDRESSABLE

```

; *TEST 6 TEST THAT THE DATA BUFFER REG IS BYTE ADDRESSABLE
; *****
TST6: SCOPE
954 004466 004737 013700 1$: JSR PC,KBDSRV ;TEST FOR "+C" ;KJL001
955 004470 004737 013700 MOV DRVDBR,R0 ;GET ON REG ADRS
956 004474 013700 001720 MOV R0,$GDADR ;SET UP DB REG ADRS
957 004500 010037 001120 CLR (R0) ;ZERO DATA BUFFER RFG
958 004504 005010 MOV #177177,$GDDAT ;LD EXPECTED
959 004506 012737 177177 001124 MOV #77776,$BDDAT ;SEND DATA FROM "BDDAT"
960 004514 012737 077776 001126 BISB $BDDAT,1(R0) ;LOAD HI BYTE DB
961 004522 153760 001126 000001 BISB $BDDAT+1,(R0) ;LOAD LO BYTE DB
962 004530 153710 001127 MOV (R0),$BDDAT ;READ IT BACK
963 004534 011037 001126 CMP $GDDAT,$BDDAT ;CORRECT?
964 004540 023737 001124 001126 BEQ TST7 ;NEXT TEST IF SO
965 004546 001401 ERROR+2 ;DATA ERROR ON BYTE ADDRESSING THE DATA BUFFER REG
966 004550 104002
967
968

```

```

; *****
; *TEST 7 TEST THAT RESET CLEARS WORD COUNT, BUS ADDRESS & DATA REGS
; *****
TST7: SCOPE
969 004552 000004 MOV #10,$TIMES ;DO 10 ITERATIONS
970 004554 012737 000010 001160 JSR PC,KBDSRV ;TEST FOR "+C" ;KJL001
971 004562 004737 013700 CLR $GDDAT ;LD EXPECTED
972 004566 005037 001124 MOV #-1,$DRVWCR ;SET ALL BITS - WC REG
973 004572 012777 177777 175112 MOV #-1,$DRVDBR ;SET ALL BITS - DB OUT REG
974 004600 012777 177777 175112 MOV #-2,$DRVBAR ;SET ALL BITS - BUS ADRS REG
975 004606 012777 177776 175100 BIT $SW12,$SWR ;TEST IF DRV11-WA IS IN Q22 MODE (S10-E40 ON)
976 004614 032777 010000 174316 BEQ 10$ ;BR IF DRV11-WA IS IN Q18 MODE (S10-E40 OFF)
977 004622 001403 MOV #-1,$DRVBAE ;SET ALL BITS - EXTENDED BUS ADRS REG
978 004624 012777 177777 175070 10$: RESET ;DO A BUS RESET
979 004632 000005 MOV $DRVWCR,$BDDAT ;READ WC REG
980 004634 017737 175052 001126 BEQ 1$ ;BR IF CLEARED
981 004642 001404 MOV DRVWCR,$BDADR ;SET UP WC REG ADRS
982 004644 013737 001712 001122 ERROR+3 ;RESET FAILED TO CLR WC REG
983 004652 104003 1$: MOV $DRVBAR,$BDDAT ;READ BUS ADRS REG
984 004654 017737 175034 001126 BIC $BIT00,$BDDAT ;DON'T WANT BIT00
985 004662 042737 000001 001126 BEQ 2$ ;BR IF CLEARED
986 004670 001404 MOV DRVBAR,$BDADR ;SET UP BA REG ADRS
987 004672 013737 001714 001122 ERROR+3 ;RESET FAILED TO CLR BUS ADRS REG
988 004700 104003 2$: BIT $SW12,$SWR ;TEST IF DRV11-WA IS IN Q22 MODE (S10-E40 ON)
989 004702 032777 010000 174230 BEQ 3$ ;BR IF IF DRV11-WA IS IN Q18 MODE (S10-E40 OFF)
990 004710 001410 MOV $DRVBAE,$BDDAT ;READ BAE ADRS REG
991 004712 017737 175004 001126 BEQ 3$ ;BR IF CLEARED
992 004720 001404 MOV DRVBAE,$BDADR ;SET UP BAE REG ADRS
993 004722 013737 001722 001122 ERROR+3 ;RESET FAILED TO CLR EXTENDED BUS ADDRESS REG
994 004730 104003 3$: MOV $DRVDBR,$BDDAT ;READ DATA BUFFER REG
995 004732 017737 174762 001126 BEQ TST10 ;NEXT TEST IF CLRED
996 004740 001404 MOV DRVDBR,$BDADR ;SET UP DB REG ADRS
997 004742 013737 001720 001122 ERROR+3 ;RESET FAILED TO CLR DATA BUFFER OUT REG
998 004750 104003
999

```

```

; *****
; *TEST 10 TEST THAT THE CONTROL/STATUS REG IS WRITE/READABLE (COUNT PTRN)
; *****
TST10: SCOPE
004752 000004 MOV #10,$TIMES ;DO 10 ITERATIONS
004754 012737 000010 001160

```

T10 TEST THAT THE CONTROL/STATUS REG IS WRITE/READABLE (COUNT

```

1000
1001 004762 106427 000200      MTPS      #PR'      ;DON'T WANT ANY INTERRUPTS
1002 004766 004537 012454      JSR        R5,SETVEC    ;SET UP INTR RETURN ADRS IN CASE
1003 004772 005130              3$              ;RETURN TO 3$ ON ILLEGAL INTR
1004 004774 013737 005014 001110  MOV        1$,$LPERR    ;SET UP SCOPE LOOP ADRS
1005 005002 013737 001716 001122  MOV        DRVCSR,$BDADR ;SET UP CSR ADRS
1006 005010 012700 160000      MOV        #160000,R0    ;START AT 0 - HI BITS FOR NOISE
1007 005014 004737 013700      1$: JSR        PC,KBDSRV    ;TEST FOR "+C"                ;KJL001
1008 005020 010037 001124      MOV        R0,$GDDAT    ;LD EXPECTED
1009 005024 042737 167201 001124  BIC        #167201,$GDDAT ;MASK TO WRITEABLE BITS
1010 005032 032777 010000 174100  BIT        #SW12,$SWR    ;TEST IF DRV11-WA IS IN Q22 MODE (S10-E40 ON)
1011 005040 001403              BEQ        10$          ;BR IF DRV11-WA IS IN Q18 MODE (S10-E40 OFF)
1012 005042 042737 167261 001124  BIC        #167261,$GDDAT ;MASK XAD17 & XAD16
1013 005050 010077 174642      10$: MOV        R0,$DRVCSR    ;LD CSR
1014 005054 017737 174636 001126  MOV        $DRVCSR,$BDDAT ;READ IT BACK
1015 005062 042737 007200 001126  BIC        #7200,$BDDAT    ;DON'T LOOK AT STAT & RDY BITS
1016 005070 032777 010000 174042  BIT        #SW12,$SWR    ;TEST IF DRV11-WA IS IN Q22 MODE (S10-E40 ON)
1017 005076 001403              BEQ        20$          ;BR IF DRV11-WA IS IN Q18 MODE (S10-E40 OFF)
1018 005100 042737 007260 001126  BIC        #7260,$BDDAT    ;DON'T LOOK AT XAD17 & XAD16
1019 005106 023737 001124 001126  20$: CMP        $GDDAT,$BDDAT ;CORRECT?
1020 005114 001401              BEQ        2$            ;BR IF SO
1021 005116 104002              ERROR+2          ;CONTROL/STATUS REG WRITE/READ FAILURE
1022 005120 062700 000002      2$: ADD        #2,R0        ;ADVANCE COUNT PATTERN
1023 005124 001333              BNE        1$            ;WRITE NEXT PATTERN IF NOT ALL TESTED
1024 005126 000413              BR         4$            ;GO RESTORE VECTOR
1025 005130 022626      3$:  CMP        (SP)+,(SP)+    ;FIX STACK - SHOULD NOT HAVE INTR'ED
1026 005132 052737 000200 001124  BIS        #200,$GDDAT    ;CORRECT EXPECTED
1027 005140 017737 174552 001126  MOV        $DRVCSR,$BDDAT ;READ CSR
1028 005146 042737 007000 001126  BIC        #7000,$BDDAT    ;DON'T WANT 'STAT' BITS
1029 005154 104005              ERROR+5          ;CPU FAILED TO LOCK OUT DRV11WA INTR REQ
1030 005156 004737 012474      4$: JSR        PC,RSTVEC    ;GO RESTORE VECTOR
1031
1032      ;*****
      ;*TEST 11      TEST THAT RESET CLEARS ALL WRITEABLE BITS & SET READY IN CSR
      ;*****
      TST11:  SCOPE
      MOV        #10,$TIMES      ;DO 10 ITERATIONS
1033
1034 005172 004737 013700      1$: JSR        PC,KBDSRV    ;TEST FOR "+C"                ;KJL001
1035 005176 013737 001716 001122  MOV        DRVCSR,$BDADR    ;SET UP CSR ADRS
1036 005204 012737 000200 001124  MOV        #200,$GDDAT    ;LD EXPECTED
1037 005212 012777 177776 174476  MOV        #-2,$DRVCSR    ;LD ALL CSR BITS
1038 005220 000005              RESET          ;DO A BUS RESET
1039 005222 017737 174470 001126  MOV        $DRVCSR,$BDDAT    ;READ CSR
1040 005230 023737 001124 001126  CMP        $GDDAT,$BDDAT    ;CORRECT?
1041 005236 001401              BEQ        TST12        ;NEXT TEST IF CSR CLRED & READY SET
1042 005240 104003              ERROR+3          ;RESET FAILED TO SET UP THE CSR
1043
1044      ;*****
      ;*TEST 12      TEST THAT THE CSR IS BYTE ADDRESSABLE
      ;*****
      TST12:  SCOPE
      MOV        #1,$TIMES
1045
1046 005244 004737 013700      JSR        PC,KBDSRV    ;TEST FOR "+C"                ;KJL001
1047 005250 013700 001716      MOV        DRVCSR,R0    ;SET CSR ADRS
1048 005254 010037 001122      MOV        R0,$BDADR    ;SET UP CSR ADRS
1049 005260 005010      CLR        (R0)          ;ZERO CSR

```

T12 TEST THAT THE CSR IS BYTE ADDRESSABLE

```

1050 005262 012737 010300 001124      MOV      #10300,$GDDAT      ;LD EXPECTED
1051 005270 012737 040026 001126      MOV      #40026,$BDDAT      ;SEND DATA FROM "BDDAT" - USE MAIN + IE
1052 005276 153760 001126 000001      BISB     $BDDAT,1(R0)      ;LOAD HI BYTE CSR
1053 005304 153710 001127              BISB     $BDDAT+1,(R0)      ;LOAD LO BYTE CSR
1054 005310 011037 001126              MOV      (R0),$BDDAT      ;READ IT BACK
1055 005314 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;CORRECT?
1056 005322 001401              BEQ      1$              ;BR IF SO
1057 005324 104002              ERROR+2      ;DATA ERROR ON BYTE ADDRESSIGN THE CSR
1058 005326 005010              1$: CLR      (R0)              ;ZERO CSR BEFORE ADVANCING
1059
1060
;;*****
;*TEST 13      TEST THAT THE 3 "FNCT" BITS CONTROL THE 3 "STAT"BITS (COUNT PTRN)
;;*****
TST13:  SCOPE
1061 005330 000004
1062 005332 012737 005360 001110      MOV      #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
1063 005340 013737 001716 001122      MOV      DRVCSR,$BDADR      ;SET UP CSR ADRS
1064 005346 012737 007000 001124      MOV      #7000,$GDDAT      ;LD EXPECTED
1065 005354 012700 000016              MOV      #16,R0          ;R0 CONTAINS "FNCT" BITS WRITTEN
1066 005360 004737 013700              1$: JSR      PC,KBDSRV      ;TEST FOR "+C"
1067 005364 010077 174326              MOV      R0,$DRVCSR      ;WRITE INTO "FNCT" BITS
1068 005370 017737 174322 001126      MOV      $DRVCSR,$BDDAT      ;READ BACK THRU "STAT" BITS
1069 005376 042737 170777 001126      BIC      #170777,$BDDAT      ;MASK TO "STAT" BITS ONLY
1070 005404 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;CORRECT?
1071 005412 001401              BEQ      2$              ;BR IF SO
1072 005414 104004              ERROR+4      ;"FNCT" BITS FAILED TO SET "STAT" BITS (LOOP BACK)
1073 005416 162737 001000 001124      2$: SUB      #1000,$GDDAT      ;CHANGE TO NEXT EXPECTED
1074 005424 162700 000002              SUB      #2,R0          ;DECREASE COUNT PATTERN
1075 005430 100353              BPL      1$              ;DO AGAIN UNTIL 0 TESTED
1076
1077
;;*****
;*TEST 14      TEST THAT READY SET WILL CAUSE AN INTERRUPT AT LEVEL 0
;;*****
TST14:  SCOPE
1078 005432 000004
1079 005434 012737 000010 001160      MOV      #10,$TIMES      ;;DO 10 ITERATIONS
1080 005442 004737 013700              JSR      PC,KBDSRV      ;TEST FOR "+C"
1081 005446 106427 000200              MTPS     #PR4          ;DONT WANT INTR YET
1082 005452 000005              RESET          ;SET THE READY FLAG BY INIT
1083 005454 012777 005534 174242      MOV      #1$,$DRVCTO      ;SET UP PREMATURE INTR RETURN ADRS
1084 005462 013737 001716 001122      MOV      DRVCSR,$BDADR      ;SET UP CSR ADRS
1085 005470 012737 000300 001124      MOV      #300,$GDDAT      ;LD EXPECTED (READY + IE)
1086 005476 106427 000000              MTPS     #PRO          ;ALLOW AN INTR
1087 005502 021616              CMP      (SP),(SP)      ;STALL
1088 005504 012777 005550 174212      MOV      #2$,$DRVCTO      ;SET UP EXPECTED INTR RETRUN ADRS
1089 005512 052777 000100 174176      BIS      #BIT6,$DRVCSR      ;ENABLE THE EXPECTED INTERRUPT
1090 005520 021616              CMP      (SP),(SP)      ;STALL
1091 005522 017737 174170 001126      MOV      $DRVCSR,$BDDAT      ;SET THE CSR
1092 005530 104005              ERROR+5      ;READY FAILED TO CAUSE AN INTERRUPT
1093 005532 000417              BR      3$              ;GO RESTORE VECTOR
1094 005534 022626              1$: CMP      (SP)+,(SP)+      ;SHOULD NEVER GET HERE - IE NOT WORKING?
1095 005536 017737 174154 001126      MOV      $DRVCSR,$BDDAT      ;GET THE CSR
1096 005544 104005              ERROR+5      ;READY INTERRUPTED WITHOUT THE IE BIT
1097 005546 000411              BR      3$              ;GO RESTORE VECTOR
1098 005550 022626              2$: CMP      (SP)+,(SP)+      ;FIX STACK SINCE NO RETURN
1099 005552 017737 174140 001126      MOV      $DRVCSR,$BDDAT      ;READ STATUS
1099 005560 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;CORRECT?

```


T14 TEST THAT READY SET WILL CAUSE AN INTERRUPT AT LEVEL 0

```

1100 005566 001401      BEQ      3$          ;BR IF SO
1101 005570 104005      ERROR+5          ;INCORRECT STATUS ON READY INTR
1102 005572 004737 012474 3$: JSR      PC,RSTVEC ;GO RESTORE VECTOR
1103
1104      ;*****
      ;*TEST 15      TEST THAT GO CLRS READY & FNCT 2 WILL SET IT
      ;*****
      TST15: SCOPE

1105      005576 000004
1106 005600 004737 013700      JSR      PC,KBDSRV ;TEST FOR "+C" ;KJL001
1107 005604 106427 000200      MTPS     #PR4 ;DONT WANT ANY INTRs
1108 005610 013737 001716 001122      MOV     DRVCSR,#BDADR ;SET UP CSR ADRS
1109 005616 005037 001124      CLR      $GDDAT ;EXPECT 0
1110 005622 012777 000001 174066      MOV     #1,$DPVCSR ;SET GO WHICH SHOULD CLR READY
1111 005630 017737 174062 001126      MOV     @DRVCSR,$BDDAT ;READ THE CSR
1112 005636 023737 001124 001126      CMP     $GDDAT,$BDDAT ;CORRECT?
1113 005644 001401      BEQ      1$          ;BR IF SO
1114 005646 104006      ERROR+6          ;THE GO BIT FAILED TO CLR READY
1115 005650 052777 000004 174040 1$: BIS     #4,@DRVCSR ;FNCT 2 SHOULD SET READY
1116 005656 012737 002204 001124      MOV     #2204,$GDDAT ;LD EXPECTED
1117 005664 017737 174026 001126      MOV     @DRVCSR,$BDDAT ;GET CSR
1118 005672 023737 001124 001126      CMP     $GDDAT,$BDDAT ;CORRECT?
1119 005700 001401      BEQ      TST16      ;NEXT TEST IF SET
1120 005702 104006      ERROR+6          ;FNCT 2 (VIA INITV2 VIA ATTN) FAILED TO SET READY
1121
1122      ;*****
      ;*TEST 16      TEST THAT READY CONTROLS 'BAR' BIT00 (VIA A00)
      ;*****
      TST16: SCOPE

1123      005704 000004
1124 005706 004737 013700      JSR      PC,KBDSRV ;TEST FOR "+C" ;KJL001
1125 005712 012777 000004 173776      MOV     #4,@DRVCSR ;SET READY
1126 005720 013737 001714 001122      MOV     DRVBAR,$BDADR ;SET UP BAR ADRS
1127 005726 012737 000001 001124      MOV     #1,$GDDAT ;EXPECT LSB OF BAR
1128 005734 005777 173752      TST     @DRVWCR ;CLR BAE FLAG TO ENSURE BAR ACCESS NEXT ;KJL001
1129 005740 005077 173750      CLR     @DRVBAR ;CLR BAR
1130 005744 005777 173742      TST     @DRVWCR ;CLR BAE FLAG TO ENSURE BAR ACCESS NEXT ;KJL001
1131 005750 017737 173740 001126      MOV     @DRVBAR,$BDDAT ;READ BAR
1132 005756 023737 001124 001126      CMP     $GDDAT,$BDDAT ;CORRECT?
1133 005764 001401      BEQ      1$          ;BR IF SO
1134 005766 104002      ERROR+2          ;A00 FAILED TO READ A ONE (SO TIED TO RDY)
1135 005770 012777 000001 173720 1$: MOV     #1,@DRVCSR ;SET GO(CLRs BAR BIT00)
1136 005776 017737 173712 001126      MOV     @DRVBAR,$BDDAT ;READ BAR
1137 006004 001403      BEQ      2$          ;BR IF ZERO
1138 006006 005037 001124      CLR     $GDDAT ;EXPECTED ZERO
1139 006012 104002      ERROR+2          ;WHEN RDY CLEARED-A00 FAILED TO READ & ZERO
1140 006014 012777 000004 173674 2$: MOV     #4,@DRVCSR ;INSURE RDY SET BEFORE ADVANCING
1141
1142      ;*****
      ;*TEST 17      TEST THAT 'CYCLE' WILL CLOCK THE DBR (IN)
      ;*****
      TST17: SCOPE

1143      006022 000004
1144 006024 004737 013700      JSR      PC,KBDSRV ;TEST FOR "+C" ;KJL001
1145 006030 013737 001720 001122      MOV     DRVDBR,$BDADR ;SET UP DBR ADRS
1146 006036 012737 125252 001124      MOV     #125252,$GDDAT ;LD EXPECTED
1147 006044 013777 001124 173646      MOV     $GDDAT,@DRVDBR ;LD DBR WITH #125252

```

T17 TEST THAT 'CYCLE' WILL CLOCK THE DBR (IN)

```

1148 006052 012777 000400 173636      MOV    #400, @DRVCSR      ;SET CYCLE - SHOULD CLK DBR (IN)
1149 006060 012777 052525 173632      MOV    #52525, @DRVDBR    ;CHANGE DBR (OUT) DATA - SHOULD NOT AFFECT (IN)
1150 006066 017737 173626 001126      MOV    @DRVDBR, @BDDAT    ;READ DBR
1151 006074 023737 001124 001126      CMP     $GDDAT, $BDDAT    ;CORRECT?
1152 006102 001401          BEQ     1$      ;BR IF SO
1153 006104 104017          ERROR+17      ;CYCLE DID NOT LATCH DBR (IN) DATA
1154 006106 042777 000400 173602 1$:   BIC     #400, @DRVCSR    ;CLEAR CYCLE BIT
1155 006114 012737 052525 001124      MOV    #52525, $GDDAT    ;NOW EXPECT #52525
1156 006122 017737 173572 001126      MOV    @DRVDBR, @BDDAT    ;READ DBR
1157 006130 023737 001124 001126      CMP     $GDDAT, @BDDAT    ;CORRECT?
1158 006136 001401          BEQ     TST20      ;NEXT TEST IF SO
1159 006140 104002          ERROR+2      ;DBR FAILED TO READ WHEN CYCLE CLEARED (NORMAL)
1160
1161

```

```

;*****
;*TEST 20      TEST SINGLE "DATI" NPR TRANSFERS (FLOATINGS & COMPLEMENT PATRN)
;*****
TST20:  SCOPE

```

```

1162 006142 000004
1163 006144 012737 006166 001110      MOV    #1$, $LPERR      ;SET UP SCOPE LOOP ADRS
1164 006152 004537 012454          JSR     R5, SETVEC      ;GO SET UP INTERRUPT RETURN
1165 006156 006266          2$      ;RETURN TO 2$ ON INTR
1166 006160 012700 177776          MOV    #-2, R0      ;FLOAT ZERO RIGHT TO LEFT
1167 006164 005001          CLR     R1      ;R1 CONTROLS DATA SHIFTINGS
1168 006166 004737 013700 1$:   JSR     PC, KBDSRV      ;TEST FOR "+C" ;KJL001
1169 006172 012777 177777 173512      MOV    #-1, @DRVWCR      ;DO ONE XFER
1170 006200 012777 025224 173506      MOV    @DBUF, @DRVBAR    ;GET DATA WORD FROM "DBUF"
1171 006206 010037 025224          MOV    R0, DBUF      ;SET UP MEM DATA
1172 006212 012777 000101 173476      MOV    #101, @DRVCSR     ;SET IE & GO (C0, C1 = ZERO)
1173 006220 052777 000400 173470      BIS    #400, @DRVCSR     ;SET CYCLE
1174 006226 106427 000000          MTPS    @PRO      ;ENABLE THE INTR
1175 006232 013737 001716 001122      MOV    DRVCSR, @BDADR    ;SET UP CSR ADRS
1176 006240 012737 000700 001124      MOV    #700, $GDDAT     ;LD EXPECTED
1177 006246 017737 173444 001126      MOV    @DRVCSR, @BDDAT    ;READ THE CSR
1178 006254 042777 000100 173434      BIC    #100, @DRVCSR     ;CLR IE
1179 006262 104005          ERROR+5      ;WCO FAILED TO INTERRUPT (CHECK FOR WCO)
1180 006264 000442          BR     4$      ;GO RESTORE VECTOR
1181 006266 022626 2$:   CMP     (SP)+, (SP)+      ;INTR RETURNS HERE - FIX STACK SINCE NO RTI
1182 006270 004537 012520          JSR     R5, CKSTAT      ;GO CHECK STATUS
1183 006274 000700          700      ;CSR STATUS EXPECTED
1184 006276 000001          1$      ;# OF XFERS
1185 006300 104007          ERROR+7      ;RETURN HERE IF STATUS ER - EXPECTED CYCLE, READY & IE
1186 006302 000433          BR     4$      ;GO RESTORE VECTOR
1187 006304 104010          ERROR+10      ;RETURN HERE IF WC ER - EXPECTED 0
1188 006306 000431          BR     4$      ;GO RESTORE VECTOR
1189 006310 104011          ERROR+11      ;RETURN HERE IF BAR ER - SHOULD = DBUF+2
1190 006312 000427          BR     4$      ;GO RESTORE VECTOR
1191 006314 012737 025224 001120      MOV    @DBUF, $GADR      ;RETURN HERE IF OK - SET UP XFER ADRS
1192 006322 010037 001124          MOV    R0, $GDDAT      ;LD EXPECTED
1193 006326 017737 173366 001126      MOV    @DRVDBR, @BDDAT    ;READ DATA XFERED
1194 006334 023737 001124 001126      CMP     $GDDAT, @BDDAT    ;CORRECT?
1195 006342 001405          BEQ     3$      ;BR IF SO
1196 006344 013737 001720 001122      MOV    DRVDBR, @BDADR    ;SET UP DBR ADRS
1197 006352 104012          ERROR+12      ;DATA ER - DBR CONTAINS WRONG DATA
1198 006354 000406          BR     4$      ;GO RESTORE VECTOR
1199 006356 005100 3$:   COM     R0      ;RETURN HERE ON GOOD DATA - NOW COM PATRN
1200 006360 005101          COM     R1      ;KEEP TRACK OF COMPLEMENT
1201 006362 001301          BNE     1$      ;DO COMPLEMENT OF THIS FLOATING ZERO IF 0

```

T20 TEST SINGLE 'DATI' NPR TRANSFERS (FLOATINGS & COMPLEMENT P

```

1202 006364 006300      ASL      R0      ;WAS DONE - NOW SHIFT ZERO LEFT
1203 006366 005200      INC      R0      ;KEEP LSB SET
1204 006370 103676      BCS      1$      ;AGAIN TILL ZERO BIT IN CARRY
1205 006372 004737 012474 4$: JSR      PC,RSTVEC ;GO RESTORE VECTOR
1206                                     ;*****
;*TEST 21      TEST SINGLE "DATO" NPR TRANSFERS (FLOATING 0 COMPLEMENT PARTN)
;*****
006376 000004      TST21: SCOPE

1207
1208 006400 012737 006422 001110      MOV      #1,$LPERR      ;SET UP SCOPE LOOP ADRS
1209 006406 004537 012454      JSR      R5,SETVEC      ;GO SET UP INTERRUPT RETURN
1210 006412 006514      2$      ;RETURN TO 2$ ON INTR
1211 006414 012700 177776      MOV      #-2,R0      ;FLOAT ZERO RIGHT TO LEFT
1212 006420 005001      CLR      R1      ;R1 CONTROLS DATA SHIFTING
1213 006422 004737 013700      1$: JSR      PC,KBDSRV      ;TEST FOR "+C"
1214 006426 012777 177777 173256      MOV      #-1,$DRVWCR      ;DO ONE XFER ;KJL001
1215 006434 012777 025224 173252      MOV      $DBUF,$DRVBAR      ;WRITE DATA WORD TO "DBUF"
1216 006442 010077 173252      MOV      R0,$DRVDBR      ;SET UP DATA IN DBR
1217 006446 012777 000103 173242      MOV      #103,$DRVCSR      ;SET IE, GO & FNCT1 (C0 = 0, C1 = 1)
1218 006454 052777 000400 173234      BIS      #400,$DRVCSR      ;SET CYCLE
1219 006462 106427 000000      MTPS     $PRO      ;ENABLE THE INTR
1220 006466 013737 001716 001122      MOV      DRVCSR,$BDADR      ;SET UP CSR ADRS
1221 006474 012737 001702 001124      MOV      #1702,$GDDAT      ;LD EXPECTED
1222 006502 017737 173210 001126      MOV      $DRVCSR,$BDDAT      ;READ THE CSR
1223 006510 104005      ERROR-5      ;WCO FAILED TO INTERRUPT (CHECK FOR WCO)
1224 006512 000442      BR      4$      ;GO RESTORE VECTOR
1225 006514 022626      2$: CMP      (SP)+,(SP)+      ;INTR RETURNS HERE - FIX STACK SINCE NO RTI
1226 006516 004537 012520      JSR      R5,CKSTAT      ;GO CHECK STATUS
1227 006522 001702      1702      ;CSR STATUS EXPECTED
1228 006524 000001      1      ;# OF XFERS
1229 006526 104007      ERROR-7      ;RETURN HERE IF STATUS ER - EXPECTED STAT C,
1230                                     ;CYCLE, READY, IE & FNCT 1
1231 006530 000433      BR      4$      ;GO RESTORE VECTOR
1232 006532 104010      ERROR-10      ;RETURN HERE IF WC ER - EXPECTED 0
1233 006534 000431      BR      4$      ;GO RESTORE VECTOR
1234 006536 104011      ERROR-11      ;RETURN HERE IF BAR ER - SHOULD = DBUF+2
1235 006540 000427      BR      4$      ;GO RESTORE VECTOR
1236 006542 012737 025224 001120      MOV      $DBUF,$GDADR      ;RETRUN HERE IF OK - SET UP XFER ADRS
1237 006550 010037 001124      MOV      R0,$GDDAT      ;LD EXPECTED
1238 006554 013737 025224 001126      MOV      DBUF,$BDDAT      ;GET DATE XFERED
1239 006562 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;CORRECT?
1240 006570 001405      BEQ      3$      ;BR IF SO
1241 006572 013737 001720 001122      MOV      DRVDBR,$BDADR      ;SET UP DBR ADRS
1242 006600 104013      ERROR-13      ;DATA ER - MEM CONTAINS WRONG DATA
1243 006602 000406      BR      4$      ;GO RESTORE VECTOR
1244 006604 005100      3$: COM      R0      ;RETURN HERE ON GOOD DATA - NOW COM PATRN
1245 006606 005101      COM      R1      ;KEEP TRACK OF COMPLEMENT
1246 006610 001304      BNE      1$      ;DO COMPELMENT OF THIS FLOATING ZERO IF 0
1247 006612 006300      ASL      R0      ;COMPLEMENT WAS DONE - NOW SHIFT ZERO LEFT
1248 006614 005200      INC      R0      ;KEEP LSB SET
1249 006616 103701      BCS      1$      ;AGAIN TILL ZERO BIT IN CARRY
1250 006620 004737 012474      4$: JSR      PC,RSTVEC      ;GO RESTORE VECTOR
1251                                     ;*****
;*TEST 22      TEST 200 "DATI" NPR TRANSFERS (BURST MODE)
;*****
006624 000004      TST22: SCOPE
006626 012737 000010 001160      MOV      #10,$TIMES      ;DO 10 ITERATIONS

```

T22 TEST 200 "DATI" NPR TRANSFERS (BURST MODE)

SEQ 0028

```

1252
1253 006634 004737 0137C0      JSR      PC,KBDSRV      ;TEST FOR "+C"
1254 006640 004537 0124F4      JSR      R5,SETVEC      ;GO SET UP INTERRUPT RETURN      ;KJL001
1255 006644 006732              1$      ;RETURN TO 1$ ON INTR
1256 006646 004737 013140      JSR      PC,LDBUF      ;GO LOAD BUFFER WITH COMPLEMENTING PATRN
1257 006652 012777 177470 173032 MOV      #-200.,@DRVWCR ;LOAD WC REG - WILL DO 200 XFERS
1258 006660 012777 025224 173026 MOV      @DBUF,@DRVBAR ;SET UP CURRENT ADRS
1259 006666 012702 000101      MOV      #101,R2      ;SET IE & GO IN R2 (C0 = 0, C1 = 0)      ;KJL002
1260 006672 004737 013652      JSR      PC,TIMER      ;WAIT FOR INTERRUPT      ;KJL002
1261 006676 013737 001716 001122 MOV      DRVCSR,$BDADR ;SET UP CSR ADRS (ONLY HERE IF ERROR)
1262 006704 012737 000700 001124 MOV      #700,$GDDAT ;LD EXPECTED
1263 006712 017737 173000 001126 MOV      @DRVCSR,$BDDAT ;READ THE CSR
1264 006720 042777 000100 172770 BIC      #100,@DRVCSR ;CLR INTR ENABLE
1265 006726 104005              ERROR+5 ;WCO FAILED TO INTERRUPT (SINGLE CYCLE ON COULD CAUSE THIS)
1266 006730 000434              BR      2$      ;GO RESTORE VECTOR
1267 006732 022626              1$: CMP      (SP)+,(SP)+ ;INTR RETURNS HERE - FIX STACK SINCE NO RTI
1268 006734 004537 012520      JSR      R5,CKSTAT ;GO CHECK STATUS
1269 006740 000700              700 ;CSR STATUS EXPECTED
1270 006742 000310              200. ;# OF XFERS
1271 006744 104007              ERROR+7 ;RETURN HERE IF STATUS ER - EXPECTED CYCLE, READY & IE
1272 006746 000425              BR      2$      ;GO RESTORE VECTOR
1273 006750 104010              ERROR+10 ;RETURN HERE IF WC ER - EXPECTED 0
1274 006752 000423              BR      2$      ;GO RESTORE VECTOR
1275 006754 104011              ERROR+11 ;RETURN HERE IF BAR ER - SHOULD = DBUF+620
1276 006756 000421              BR      2$      ;GO RESTORE VECTOR
1277 006760 012737 026042 001120 MOV      @DBUF+616,$GDADR ;OK - SET UP LAST XFER ADRS WHERE #70707 SHOULD BE
1278 006766 012737 070707 001124 MOV      #70707,$GDDAT ;LD EXPECTED
1279 006774 017737 172720 001126 MOV      @DRVDBR,$BDDAT ;DBR SHOULD HAVE LAST DATUM
1280 007002 023737 001124 001126 CMP      $GDDAT,$BDDAT ;CORRECT?
1281 007010 001404              BEQ      2$      ;BR IF SO
1282 007012 013737 001720 001122 MOV      DRVDBR,$BDADR ;SET UP DBR ADRS
1283 007020 104012              ERROR+12 ;DATA ER - DBR DIJ NOT CONTAIN EXPECTED LAST XFER
1284 007022 004737 012474      2$: JSR      PC,RSTVEC ;GO RESTORE VECTOR
1285
1286

```

```

*****
*TEST 23      TEST 200 "DATO" NPR TRANSFERS (BURST MODE)
*****

```

```

007026 000004
007030 012737 000010 001160 TST23: SCOPE
MOV      #10,$TIMES      ;;DO 10 ITERATIONS
1287
1288 007036 004737 013700      JSR      PC,KBDSRV      ;TEST FOR "+C"
1289 007042 004537 012454      JSR      R5,SETVEC      ;GO SET UP INTERRUPT RETURN      ;KJL001
1290 007046 007136              1$      ;RETURN TO 1$ ON INTR
1291 007050 012777 177470 172634 MOV      #-200.,@DRVWCR ;WORD WC REQ - WILL DO 200 XFER'S
1292 007056 012777 025224 172630 MOV      @DBUF,@DRVBAR ;SET UP CURRENT ADRS
1293 007064 012777 177377 172626 MOV      #177377,@DRVDBR ;THIS WILL BE WRITTEN TO MEM
1294 007072 012702 000103      MOV      #103,R2      ;SET IE, FNCT 1 & GO IN R2 (C0 = 0, C1 = 1) ;KJL002
1295 007076 004737 013652      JSR      PC,TIMER      ;WAIT FOR INTERRUPT      ;KJL002
1296 007102 013737 001716 001122 MOV      DRVCSR,$BDADR ;SET UP CSR ADRS (ONLY HERE IF ERROR)
1297 007110 012737 001702 001124 MOV      #1702,$GDDAT ;LD EXPECTED
1298 007116 017737 172574 001126 MOV      @DRVCSR,$BDDAT ;READ THE CSR
1299 007124 042777 000100 172564 BIC      #100,@DRVCSR ;CLR INTR ENABLE
1300 007132 104005              ERROR+5 ;WCO FAILED TO INTERRUPT (SINGL CYCL ON COULD CAUSE THIS)
1301 007134 000416              BR      2$      ;GO RESTORE VECTOR
1302 007136 022626              1$: CMP      (SP)+,(SP)+ ;INTR RETURNS HERE - FIX STACK SINCE NO RTI
1303 007140 004537 012520      JSR      R5,CKSTAT ;GO CHECK STATUS
1304 007144 001702              1702 ;CSR STATUS EXPECTED

```

T23 TEST 200 "DATO" NPR TRANSFERS (BURST MODE)

Address	Hex	Dec	Label	Instruction	Comment
1305	007146	000310			
1306	007150	104007			
1307					
1308	007152	000407			
1309	007154	104010			
1310	007156	000405			
1311	007160	104011			
1312	007162	000403			
1313	007164	004737	013272		
1314	007170	104013			
1315	007172	004737	012474		
1316					
1317					

Address	Hex	Dec	Label	Instruction	Comment
200.					
ERROR*7					
BR	2\$				
ERROR*10					
BR	2\$				
ERROR*11					
BR	2\$				
JSR	PC,CKDAT				
ERROR*13					
2\$: JSR	PC,RSTVEC				

Comment
;# OF XFERS
;RETURN HERE IF STATUS ER - EXPECTED STAT C.
;CYCLE, READY, IE & FNCT 1
;GO RESTORE VECTOR
;RETURN HERE IF WC ER - EXPECTED 0
;GO RESTORE VECTOR
;RETURN HERE IF BAR ER - SHOULD = DBUF*620
;GO RESTORE VECTOR
;RETURN HERE IF OK - NOW GO CHECK DATA
;RETURN HERE IF DATA ER - DBR CONTAINS WRONG DATA
;RETURN HERE IF DATA CHECK OK - GO RESTORE VECTOR

Comment

```

;*****
;*TEST 24      TEST THAT THE CPU IS LOCKED OUT WITH SINGLE CYCLE OFF

```

• • • • •

	007176	000004		TST24:	SCOPE			
	007200	012737	000010	001160	MOV	#10,\$TIMES	::CO 10 ITERATIONS	
1318								
1319	007206	013737	001716	001122	MOV	DRVCSR,\$BDADR	;SET UP CSR ADRS	
1320	007214	004537	012454		JSR	R5,SETEVC	;GO SET UP INTR RETURN	
1321	007220	007316			3\$		;RETURN TO 3\$ ON INTR	
1322	007222	012700	000010		MOV	#10,R0	;DO EIGHT 200 WORD XFER'S	
1323	007226	005037	001126		CLR	\$BDDAT	;USE \$BDDAT AS A COUNTER	
1324	007232	004737	013700		JSR	PC,KBDSRV	;TEST FOR "+C"	
1325	007236	012777	177470	172446	MOV	#-200.,@DRVWCR	;DO 200 XFERS (DATI'S)	:KJL001
1326	007244	012777	025224	172442	MOV	@DBUF,@DRVBAR	;FROM DBUF	
1327	007252	012702	000101		MOV	#101,R2	;SET IE & GO IN R2 (SINGLE CYCLE LOW)	:KJL002
1328	007256	004737	013652		JSR	PC,TIMER	;GO WAIT FOR INTERRUPT OR XFER COMPLETE	:KJL002
1329	007262	005237	001124		2\$: INC	\$GDDAT	;START COUNTING - SHOULD NEVER GET HERE	
1330	007266	001375			BNE	2\$	;UNTIL 64K	
1331	007270	012737	000700	001124	MOV	#700,\$GDDAT	;LD EXPECTED	
1332	007276	017737	172414	001126	MOV	@DRVCSR,\$BDDAT	;READ STATUS	
1333	007304	042777	000100	172404	BIC	#100,@DRVCSR	;CLR IE	
1334	007312	104005			ERROR+5		;NO INTERRUPT ON 200 DATI'S	
1335	007314	000407			BR	4\$	;GO RESTORE VECTOR	
1336	007316	022626			3\$: CMP	(SP)+,(SP)+	;FIX STACK SINCE NO RTI	
1337	007320	005300			DEC	R0	;DONE 8 TIMES?	
1338	007322	001343			BNE	1\$	;BR IF NOT	
1339	007324	005737	001126		TST	\$BDDAT	;SHOULD STILL BE ZERO	
1340	007330	001401			BEQ	4\$	;BR IF SO	
1341	007332	104014			ERROR+14		;BURST MODE (SINGLE CYCLE=0) FAILS TO LOCK OUT CPU	
1342	007334	004737	012474		4\$: JSR	PC,RSTVEC	;SO RESTORE VECTOR	
1343					:	*	*	

```

;*****
;*TEST 25      TEST THAT THE CPU IS NOT LOCKED OUT WITH SINGLE CYCLE ON

```

;;*****

ADDRESS	INSTR	OPERANDS	COMMENT
007340	000004		
007342	012737	000010 001160	TST25: SCOPE
1344		MOV	#10,\$TIMES ;:DO 10 ITERATIONS
1345	007350	013737 001716 001122	MOV DRVCSR,\$BDADR ;SET UP CSR ADRS
1346	007356	004537 012454	JSR R5,SETVEC ;GO SET UP INTR RETURN
1347	007362	007474	3\$;RETURN TO 3\$ ON INTR
1348	007364	012700 000010	MOV #10,R0 ;DO EIGHT 200 WORD XFER'S
1349	007370	012737 000000 001126	MOV #0,\$BDDAT ;USE \$BDDAT AS A COUNTER
1350	007376	004737 013700	1\$: JSR PC,KBDSRV ;TEST FOR "+C"
1351	007402	012777 177470 172302	MOV #200,@DRVWCR ;DO 200 XFERS (DATI'S)
1352	007410	012777 025224 172276	MOV @DBUF,@DRVBAR ;FROM DBUF
1353	007416	106427 000000	MTPS #PRO ;ALLOW AN INTR

T25 TEST THAT THE CPU IS NOT LOCKED OUT WITH SINGLE CYCLE ON

```

1354 007422 012777 000111 172266      MOV      #111,@DRVCSR      ;SET IE, FNCT3 & GO (SINGLE CYCLE HIGH)
1355 007430 052777 000400 172260      BIS      #400,@DRVCSR      ;SET CYCLE
1356 007436 000240                NOP                ;FREEBEE
1357 007440 005237 001126                2$: INC      $BDDAT      ;START COUNTING
1358 007444 001375                BNE      2$          ;UNTIL 64K - SHOULD INTR BEFORE OVERFLOW
1359 007446 012737 004710 001124      MOV      #4710,$GDDAT    ;LD EXPECTED
1360 007454 017737 172236 001126      MOV      @DRVCSR,$BDDAT  ;READ STATUS
1361 007462 042777 000100 172226      BIC      #100,@DRVCSR    ;CLR IE
1362 007470 104005                ERROR+5      ;NO INTERRUPT ON 200 DATI'S (WITH SINGLE CYCLE)
1363 007472 000423                BR      5$          ;GO RESTORE VECTOR
1364 007474 022626                3$: CMP      (SP)+,(SP)+    ;FIX STACK SINCE NO RTI
1365 007476 005300                DEC      R0          ;DONE 8 TIMES?
1366 007500 001336                BNE      1$          ;BR IF NOT
1367 007502 022737 000000 001126      CMP      #0,$BDDAT      ;$BDDAT SHOULD HAVE BEEN COUNTED
1368 007510 103401                BCS      4$          ;BR IF SO
1369 007512 104015                ERROR+15     ;CPU APPEARED LOCKED OUT WITH SINGLE CYCLE SET
1370 007514 017737 172176 001126      4$: MOV      @DRVCSR,$BDDAT  ;READ STAUTS
1371 007522 012737 004710 001124      MOV      #4710,$GDDAT  ;LD EXPECTED
1372 007530 023737 001124 001126      CMP      $GDDAT,$BDDAT ;CORRECT?
1373 007536 001401                BEQ      5$          ;BR IF SO
1374 007540 104007                ERROR+7      ;STATUS INCORRECT ON XFER WITH SINGLE CYCLE SET
1375 007542 004737 012474      5$: JSR      PC,RSTVEC    ;GO RESTORE VECTOR
1376
1377

```

```

;*****
;*TEST 26      TEST THAT MAINT MODE CONTROLS FNCT BITS, XFER DIR & SINGLE CYCLE
;*****
TST26: SCOPE

```

```

007546 000004
007550 012737 000200 001160      MOV      #200,$TIMES    ;DO 200 ITERATIONS
1378
1379
1380
1381
1382
1383 007556 004537 012454                JSR      R5,SETVEC      ;GO SET UP INTR RETURN
1384 007562 007710                2$          ;RETURN TO 2$ ON INTR
1385 007564 004737 013216                JSR      PC,LDBUF1      ;GO SET UP DBUF (SPECIAL COM PATTERN)
1386 007570 012737 011702 007716      MOV      #11702,3$     ;3$ CONTAINS EXPECTED STATUS
1387 007576 012737 000001 007720      MOV      #1,4$         ;4$ CONTAINS THE CURRENT XFER NO # (MAX 8)
1388 007604 004737 013700                1$: JSR      PC,KBDSRV    ;TEST FOR "+C" ;KJL001
1389 007610 106427 000000                MTPS     #PRO          ;ALLOW INTR
1390 007614 005777 172072                TST      @DRVWCR        ;CLR BAE FLAG TO ENSURE BAR ACCESS NEXT ;KJL001
1391 007620 012777 025224 172066      MOV      @DBUF,@DRVBAR ;SET UP CURRENT ADRS
1392 007626 013777 007720 172056      MOV      4$,@DRVWCR    ;SET XFER #
1393 007634 005477 172052                NEG      @DRVWCR        ;NEGATE FOR WC
1394 007640 012777 010101 172050      MOV      #10101,@DRVCSR ;SET UP MAINT, IE & GO
1395 007646 052777 000400 172042      BIS      #400,@DRVCSR  ;SET CYCLE
1396 007654 013737 001716 001122      MOV      DRVCSR,$BADDR ;SET UP CSR ADRS
1397 007662 013737 007716 001124      MOV      3,$GDDAT      ;LD EXPECTED
1398 007670 017737 172022 001126      MOV      @DRVCSR,$BDDAT ;READ STATUS
1399 007676 042777 000100 172012      BIC      #100,@DRVCSR  ;DISABLE IE
1400 007704 104005                ERROR+5      ;NO INTR ON XFER (IN MAINT MODE)
1401 007706 000440                BR      6$          ;GO RESTORE VECTOR
1402 007710 022626                2$: CMP      (SP)+,(SP)+    ;RETURN HERE ON INTR - FIX STACK SINCE NO RTI
1403 007712 004537 012520                JSR      R5,CKSTAT     ;GO CHECK STATUS
1404 007716 011702                3$: 11702          ;THIS LOCATION WILL CONTAIN EXPECTED STATUS
1405 007720 000001                4$: 1              ;THIS LOCATION WILL CONTAIN CURRENT XFER # (MAX 8)
1406 007722 104007                ERROR+7      ;RETURN HERE IF STATUS ER - WILL EXPECT

```

```

;KJL005
;KJL005
;KJL005

```

;KJL001

;KJL001

T26 TEST THAT MAINT MODE CONTROLS FNCT BITS, XFER DIR & SINGLE

```

1407
1408
1409 007724 000431          BR      6$          ;MAINT, COUNT INCREASE OF FNCT & STAT BITS,
1410 007726 104010          ERROR+10        ;CYCLE, READY & IE
1411 007730 000427          BR      6$          ;GO RESTORE VECTOR
1412 007732 104011          ERROR+11        ;RETURN HERE IF WC ER - SHOULD BE 0
1413 007734 000425          BR      6$          ;GO RESTORE VECTOR
1414 007736 062737 001002 007716  ADD     #1002,3$      ;GO RESTORE VECTOR
1415 007744 032737 020000 007716  BIT      #BIT13,3$    ;RETURN HERE IF OK - ADVANCE EXPECTED STATUS LOC
1416 007752 001403          BEQ      5$          ;LOOK FOR OVERFLOW
1417 007754 012737 010700 007716  MOV      #10700,3$    ;BR IF NOT
1418 007762 005237 007720 5$: INC      4$          ;FNCT & STAT BITS SHOULD BE ZERO THIS TIME
1419 007766 022737 000011 007720  CMP      #11,4$      ;ADVANCE CURRENT XFER #
1420 007774 001303          BNE      1$          ;HAVE 10 XFERS BEEN DONE
1421 007776 004537 013344          JSR      R5,CKDAT1    ;BR IF NOT
1422 010002 000010          10              ;NOW GO CHECK DATA
1423 010004 104013          ERROR+13        ;# OF XFER'S TO CHECK
1424 010006 000240          NOP              ;RETURN HERE IF DATA ER - (WITH MAINT SET)
1425 010010 004737 012474 6$: JSR      PC,RSTVEC    ;RESTORE VECTOR NEXT
1426
1427
;*****
;*TEST 27 TEST THAT A DATI FROM A NON-EXISTANT BUS ADRS SETS 'NEX'
;*****
TST27: SCOPE
MOV      #100,$TIMES      ;;DO 100 ITERATIONS
1428
1429 010024 012700 000002          MOV      #2,R0          ;R0 WHEN ZERO SAYS CLR 'NEX' WITH RESET
1430 010030 004737 013700 1$: JSR      PC,KBDSRV        ;TEST FOR "+C"
1431 010034 004537 012454          JSR      R5,SETVEC        ;GO SET UP INTERRUPT RETURN ;KJL001
1432 010040 010150          2$          ;RETURN TO 2$ ON TIMEOUT INTR
1433 010042 012777 177777 171642 MOV      #-1,@DRVWCR    ;SET UP FOR ONE XFER'S
1434 010050 012777 160000 171636 MOV      #160000,@DRVBAR ;SET UP BAR TO 160000 (RESERVED)
1435 010056 106427 000000          MTPS     #PRO          ;ALLOW INTR
1436 010062 032777 010000 171050 BIT      #SW12,@SWR      ;TEST IF DRV11-WA IS IN Q22 MODE (S10-E40 ON)
1437 010070 001403          BEQ      10$         ;BR IF DRV11-WA IS IN Q18 MODE (S10-E40 OFF)
1438 010072 012777 000077 171622 MOV      #000077,@DRVBAR ;SET XAD 21,20,19,18,17,16
1439 010100 012777 000161 171610 10$: MOV      #161,@DRVCSR    ;SET IE, XAD 17,16 & GO
1440 010106 052777 000400 171602 BIS      #400,@DRVCSR    ;SET CYCLE
1441 010114 013737 001716 001122 MOV      @DRVCSR,$BDADR   ;SET UP CSR ADRS - SHOULD NEVER GET HERE
1442 010122 012737 140760 001124 MOV      #140760,$GDDAT   ;LD EXPECTED
1443 010130 017737 171562 001126 MOV      @DRVCSR,$BDDAT   ;GIVE THEM THE STATUS
1444 010136 042777 000100 171552 BIC      #100,@DRVCSR    ;CLR IE
1445 010144 104016          ERROR+16        ;NEX FAILED TO CAUSE AN INTERRUPT
1446 010146 000526          BR      7$          ;GO RESTORE VECTOR
1447 010150 022626          2$: CMP      (SP)+,(SP)+    ;SHOULD INTR RETURN HERE - FIX STACK
1448 010152 017737 171540 001126 MOV      @DRVCSR,$BDDAT   ;READ THE CSR
1449 010160 012737 140760 001124 MOV      #140760,$GDDAT   ;LD EXPECTED
1450 010166 023737 001124 001126 CMP      $GDDAT,$BDDAT   ;CORRECT?
1451 010174 001405          BEQ      3$          ;BR IF SO
1452 010176 013737 001716 001122 MOV      @DRVCSR,$BDADR   ;SET UP CSR ADRS
1453 010204 104016          ERROR+16        ;STATUS ER - EXPECTED
1454
1455 010206 000506          BR      7$          ;ER, NEX, CYCLE, READY, IE, XAD17 & XAD16
1456 010210 017737 171476 001126 3$: MOV      @DRVWCR,$BDDAT    ;GO RESTORE VECTOR
1457 010216 012737 000000 001124 MOV      #0,$GDDAT      ;READ WORD COUNT
1458 010224 023737 001124 001126 CMP      $GDDAT,$BDDAT   ;** SHOULD HAVE 0
1459 010232 001405          BEQ      4$          ;CORRECT?
;BR IF SO

```


T27 TEST THAT A DATI FROM A NON-EXISTANT BUS ADRS SETS 'NEX'

```

1460 010234 013737 001712 001122      MOV      DRVWCR,$BDADR      ;SET UP WCR ADRS
1461 010242 104010                      ERROR+10      ;** WC WAS NOT INCREMENTED ON A TIMEOUT ER
1462 010244 000467                      BR          7$      ;GO RESTORE VECTOR
1463 010246 005777 171440          4$:  TST      @DRVWCR      ;CLR BAE FLAG TO ENSURE BAR ACCESS NEXT ;KJL001
1464 010252 017737 171436 001126      MOV      @DRVBAR,$BDDAT      ;READ BUFFER ADRS
1465 010260 042737 000001 001126      BIC      @BIT00,$BDDAT      ;** DON'T NEED BIT00
1466 010266 012737 160002 001124      MOV      @160002,$GDDAT      ;** SHOULD HAVE INCREMENTED
1467 010274 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;CORRECT?
1468 010302 001405                      BEQ      5$      ;BR IF SO
1469 010304 013737 001714 001122      MOV      DRVBAR,$BDADR      ;SET UP BAR ADRS
1470 010312 104011                      ERROR+11      ;** BAR WAS NOT INCREMENTED ON A TIMEOUT ER
1471 010314 000443                      BR          7$      ;GO RESTORE VECTOR
1472 010316 005300          5$:  DEC      R0      ;KEEP TRACK ON HOW TO CLR
1473 010320 001422                      BEQ      6$      ;BR IF CLR BY RESET
1474 010322 042777 040000 171366      BIC      @40000,@DRVCSR      ;WRITE NEX TO ZERO
1475 010330 017737 171362 001126      MOV      @DRVCSR,$BDDAT      ;READ CSR
1476 010336 012737 000760 001124      MOV      @760,$GDDAT      ;LD EXPECTED
1477 010344 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;CORRECT?
1478 010352 001626                      BEQ      1$      ;BR IF SO + REPEAT TEST FOR RESET TEST
1479 010354 013737 001716 001122      MOV      DRVCSR,$BDADR      ;SET UP CSR ADRS
1480 010362 104016                      ERROR+16      ;'NEX' FAILED TO WRITE TO ZERO
1481 010364 000417                      BR          7$      ;GO RESTORE VECTOR
1482 010366 000005          6$:  RESET      ;ISSUE BUS RESET
1483 010370 017737 171322 001126      MOV      @DRVCSR,$BDDAT      ;READ THE CSR
1484 010376 012737 000200 001124      MOV      @200,$GDDAT      ;EXPECT ONLY READY
1485 010404 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;CORRECT?
1486 010412 001404                      BEQ      7$      ;BR IF SO
1487 010414 013737 001716 001122      MOV      DRVCSR,$BDADR      ;SET UP CSR ADRS
1488 010422 104016                      ERROR+16      ;RESET FAILED TO CLR 'NEX'
1489 010424 004737 012474          7$:  JSR      PC,RSTVEC      ;GO RESTORE VECTOR
1490                                     ;*****
;*****
;*TEST 30      TEST 200 NPR TRANSFERS IN MAINT MODE
;*****
;*****
TST30:  SCOPE
MOV      @10,$TIMES      ;DO 10 ITERATIONS

1491
1492 010440 004737 013700      JSR      PC,KBDSRV      ;TEST FOR "+C" ;KJL001
1493 010444 004537 012454      JSR      R5,SETVEC      ;GO SET UP INTR RETURN
1494 010450 010566          2$:  ;RETURN TO 2$ ON INTR
1495 010452 004737 013216      JSR      PC,LDBUF1      ;GO SET UP DBUF (SPECIAL COM PATTERN)
1496 010456 005777 171230      TST      @DRVWCR      ;CLR BAE FLAG TO ENSURE BAR ACCESS NEXT ;KJL001
1497 010462 012777 025224 171224      MOV      @DBUF,@DRVBAR      ;SET UP CURRENT ADRS
1498 010470 012777 177470 171214      MOV      @-200,@DRVWCR      ;SET UP PCR 200 XFER'S
1499 010476 106427 000000      MTPS     @PRO      ;ALLOW INTR
1500 010502 012777 010101 171206      MOV      @10101,@DRVCSR      ;SET MAINT, IE & GO (DATI)
1501 010510 052777 000400 171200      BIS      @400,@DRVCSR      ;SET CYCLE
1502 010516 012737 000000 001126      MOV      @0,$BDDAT      ;SET UP A COUNTER
1503 010524 005237 001126          1$:  INC      $BDDAT      ;COUNT AWAY
1504 010530 001375                      BNE      1$      ;WAIT TILL DONE - SHOULD INTR BEFORE OVFL0
1505 010532 013737 001716 001122      MOV      DRVCSR,$BDADR      ;SET UP CSR ADRS
1506 010540 012737 010700 001124      MOV      @10700,$GDDAT      ;LD EXPECTED
1507 010546 017737 171144 001126      MOV      @DRVCSR,$BDDAT      ;READ STATUS
1508 010554 042777 000100 171134      BIC      @100,@DRVCSR      ;DISABLE IE
1509 010562 104005                      ERROR+5      ;NO INTR AFTER 200 MAINT MODE XFER'S
1510 010564 000420          2$:  BR          3$      ;GO RESTORE VECTOR
1511 010566 022626                      CMP      (SP)+,(SP)+      ;RETURN HERE ON INTR - FIX STACK SINCE NO RTI
1512 010570 004537 012520          JSR      R5,CKSTAT      ;GO CHECK STATUS

```

T30 TEST 200 NPR TRANSFERS IN MAINT MODE

```

1513 010574 010700          10700          ;EXPECTED STATUS
1514 010576 000310          200.          ;# OF XFERS
1515 010600 104007          ERROR+7        ;RETURN HERE IF STATUS ER - EXPECTED MAINT.
1516                                     ;CYCLE, READY & IE
1517 010602 000411          BR 3$          ;GO RESTORE VECTOR
1518 010604 104010          ERROR+10        ;RETURN HERE IF WC ER - SHOULD BE 0
1519 010606 000407          BR 3$          ;GO RESTORE VECTOR
1520 010610 104011          ERROR+11        ;RETURN HERE IF BAR ER - SHOULD = DBUF+620
1521 010612 000405          BR 3$          ;GO RESTORE VECTOR
1522 010614 004537 013344    JSR R5,CKDAT1 ;RETURN HERE IF OK - NOW GO CHECK DATA
1523 010620 000310          200.          ;# OF XFER'S TO CHECK
1524 010622 104013          ERROR+13        ;RETURN HERE IF DATA ER - (WITH MAINT SET)
1525 010624 000240          NOP            ;RESTORE VECTOR NEXT
1526 010626 004737 012474    3$: JSR PC,RSTVEC ;GO RESTORE VECTOR
1527
1528
;*****
;*TEST 31 TEST A 200 WORD MAINT MODE XFER TO EACH ADDITIONAL AVAILABLE 4K
;*****
TST31: SCOPE
MOV 45,$TIMES ;DO 5 ITERATIONS
1529
1530 010642 004537 012454    JSR R5,SETVEC ;GO SET UP INTR RETURN
1531 010646 011014          3$          ;RETURN TO 3$ ON INTR
1532 010650 012737 020000 001736    MOV 420000,DBUFP ;GET LOWEST BUFFER ADRS
1533 010656 004737 013700          1$: JSR PC,KBDSRV ;TEST FOR "+C" ;KJL001
1534 010662 062737 020000 001736    ADD 420000,DBUFP ;POINT TO NEXT 4K
1535 010670 023737 001734 001736    CMP CORSZ,DBUFP ;IS THE 4K THERE?
1536 010676 001467          BEQ 4$          ;BR IF NOT
1537 010700 004737 013216    JSR PC,LDBUF1 ;GO SET UP SPECIAL COMPLEMENT PATTERN
1538 010704 005777 171002          TST 8DRVWCR ;CLR BAE FLAG TO ENSURE BAR ACCESS NEXT ;KJL001
1539 010710 013777 001736 170776    MOV DBUFP,8DRVBAR ;SET UP BUFFER ADRS
1540 010716 012777 177470 170766    MOV 4-200.,8DRVWCR ;SET UP FOR 200 XFER'S
1541 010724 106427 000000          MTPS 4PRO ;ALLOW INTR
1542 010730 012777 010101 170760    MOV 410101,8DRVCSR ;SET MAINT, IE & GO
1543 010736 052777 000400 170752    BIS 4400,8DRVCSR ;SET CYCLE
1544 010744 012737 000000 001126    MOV 40,$BDDAT ;SET UP A COUNTER
1545 010752 005237 001126          2$: INC 4BDDAT ;COUNT AWAY
1546 010756 001375          BNE 2$          ;SHOULD ALWAYS INTR FROM THIS LOOP
1547 010760 013737 001716 001122    MOV 8RVCSR,$BDADR ;SET UP CSR ADRS
1548 010766 012737 010700 001124    MOV 410700,$GDDAT ;LD EXPECTED
1549 010774 017737 170716 001126    MOV 8DRVCSR,$BDDAT ;READ CSR
1550 011002 042777 000100 170706    BIC 4100,8DRVCSR ;DISABLE IE
1551 011010 104005          ERROR+5        ;NO INTR AFTER 200 MAINT MODE XFER'S
1552 011012 000421          BR 4$          ;GO RESTORE VECTOR
1553 011014 022626          3$: CMP (SP)+,(SP)+ ;RETURN HERE ON INTR - FIX STACK SINCE NO RTI
1554 011016 004537 012520    JSR R5,CKSTAT ;GO CHECK STATUS
1555 011022 010700          10700          ;EXPECTED STATUS
1556 011024 000310          200.          ;# OF XFER'S
1557 011026 104007          ERROR+7        ;RETURN HERE IF STATUS ER - EXPECTED MAINT.
1558                                     ;CYCLE, READY & IE
1559 011030 000412          BR 4$          ;GO RESTORE VECTOR
1560 011032 104010          ERROR+10        ;RETURN HERE IF WC ER - SHOULD = 0
1561 011034 000410          BR 4$          ;GO RESTORE VECTOR
1562 011036 104011          ERROR+11        ;RETURN HERE IF BAR ER - SHOULD = DBUF+620
1563 011040 000406          BR 4$          ;GO RESTORE VECTOR
1564 011042 004537 013344    JSR R5,CKDAT1 ;RETURN HERE IF OK - NOW GO CK DATA
1565 011046 000310          200.          ;# OF XFER'S TO CK

```

T31 TEST A 200 WORD MAINT MODE XFER TO EACH ADDITIONAL AVAILAB

```

1566 011050 104013          ERROR+13          ;RETURN HERE IF DATA ER
1567 011052 000401          BR      4$          ;GO RESTORE VECTOR
1568 011054 000700          BR      1$          ;TRY NEXT BANK
1569 011056 012737 025224 001736 4$: MOV      $DBUF,DBUFP ;RESTORE BUFFER ADRS TO LOWEST 4K
1570 011064 004737 012474          JSR      PC,RSTVEC ;GO RESTORE VECTOR
1571
1572          ;;*****
          ;*TEST 32      TEST THE ADDRESS (I/O) ABILITY TO THE TTY PRINTER CSR
          ;;*****
          TST32: SCOPE
011070 000004
1573
1574 011072 004737 013700          JSR      PC,KBDSRV          ;TEST FOR "+C"
1575 011076 005737 001704          TST      LSIQ22          ;TEST IF THE SYSTEM IS SUPPORTS 22-BIT ;KJL001
1576 011102 001114          BNE      TST33          ;;SKIP DMA TEST IN I/O PAGE IF SYSTEM SUPPORTS Q22
1577 011104 004537 012454          JSR      R5,SETVEC          ;GO SET UP INTR RETURN
1578 011110 011232          1$          ;RETURN TO 1$ ON INTR
1579 011112 012777 177777 170572 MOV      #-1,$DRVWCR          ;SET UP WC - 1 XFER
1580 011120 013737 001150 001736 MOV      $TPS,CBUIP          ;SET UP BUFFER ADRS TO PRINTER CSR ADRS
1581 011126 005077 170566          CLR      $DRVDDBR          ;ZERO THE DBR
1582 011132 013777 001150 170554 MOV      $TPS,$DRVBAR          ;SET UP BUFFER ADRS - FROM PRINTER CSR
1583 011140 106427 000000          MTPS     $PRO          ;ALLOW INTR
1584 011144 032777 010000 167766 BIT      $SW12,$SWR          ;TEST IF DRV11-WA IS IN Q22 MODE (S10-E40 ON)
1585 011152 001403          BEQ      10$          ;BR IF DRV11-WA IS IN Q18 MODE (S10-E40 OFF)
1586 011154 012777 000077 170540 MOV      $000077,$DRVBAB          ;SET XAD 21,20,19,18,17,16
1587 011162 012777 000161 170526 10$: MOV      $161,$DRVCSR          ;SET IE, XAD17 & XAD16, & GO
1588 011170 052777 000400 170520 BIS      $400,$DRVCSR          ;SET CYCLE
1589 011176 013737 001716 001122 MOV      DRVCSR,$BDADR          ;SET UP CSR ADRS
1590 011204 012737 000760 001124 MOV      $760,$GDDAT          ;LD EXPECTED STATUS
1591 011212 017737 170500 001126 MOV      $DRVCSR,$BDDAT          ;READ THE CSR
1592 011220 042777 000100 170470 BIC      $100,$DRVCSR          ;DISABLE IE
1593 011226 104005          ERROR+5          ;NO INTR ON 1 WD XFER FROM XCSR
1594 011230 000434          BR      2$          ;GO RESTORE VECTOR
1595 011232 022626          1$: CMP      (SP)+,(SP)+          ;INTR RETURNS HERE - FIX STACK SINCE NO RTI
1596 011234 004537 012520          JSR      R5,CKSTAT          ;GO CK STATUS
1597 011240 000760          760          ;CSR EXPECTED STATUS
1598 011242 000001          1          ;# OF XFER'S
1599 011244 104007          ERROR+7          ;RETURN HERE IF STATUS ER - EXPECTED CYCLE,
1600          ;XAD17 & XAD16, RDY & IE
1601 011246 000425          BR      2$          ;GO RESTORE VECTOR
1602 011250 104010          ERROR+10          ;RETURN HERE IF WC ER - EXPECTED #
1603 011252 000423          BR      2$          ;GO RESTORE VECTOR
1604 011254 104011          ERROR+11          ;RETURN HERE IF BAR ER - SHOULD = XCSR+2
1605 011256 000421          BR      2$          ;GO RESTORE VECTOR
1606 011260 017737 167664 001124 MOV      $TPS,$GDDAT          ;RETURN HERE IF OK - GET XCSR CONTENTS
1607 011266 017737 170426 001126 MOV      $DRVDDBR,$BDDAT          ;READ DBR
1608 011274 023737 001124 001126 CMP      $GDDAT,$BDDAT          ;CORRECT?
1609 011302 001407          BEQ      2$          ;BR IF SO
1610 011304 013737 001720 001122 MOV      DRVDDBR,$BDADR          ;SET UP DBR ADRS
1611 011312 013737 001150 001120 MOV      $TPS,$GDADR          ;SET ADRS OF DATA (XCSR)
1612 011320 104020          ERROR+20          ;DATA ER FROM TTY PRINTER CSR
1613 011322 012737 025224 001736 2$: MOV      $DBUF,DBUFP          ;RESTORE BUFFER ADRS TO LOWEST 4K
1614 011330 004737 012474          JSR      PC,RSTVEC          ;GO RESTORE VECTOR
1615          ;;*****
          ;*TEST 33      TEST A 200 WORD MAINT MODE XFER WITH MMU
          ;;*****
          TST33: SCOPE
011334 000004
011336 012737 000005 001160          MOV      $5,$TIMES          ;;DO 5 ITERATIONS

```

T33 TEST A 200 WORD MAINT MODE XFER WITH MMU

```

1616
1617 011344 032777 010000 167566      BIT      #SW12,@SWR      ;TEST IF DRV11-WA IS IN Q22 MODE (S10-E40 ON)
1618 011352 001555                      BEQ      4$          ;BR IF DRV11-WA IS IN Q18 MODE (S10-E40 OFF)
1619 011354 032737 000001 001710      BIT      @BIT0,MMAVA  ;MMU AVAILABLE ?
1620 011362 001551                      BEQ      4$          ;BR IF MMU NOT AVAILABLE
1621 011364 005737 001710              TST      MMAVA        ;IS 22 ADDRESSING AVAILABLE? (BIT15 SET)
1622 011370 100003                      BPL      10$         ;BR IF NOT AVAILABLE
1623 011372 052737 000020 172516      BIS      @BIT4,@#SR3  ;TURN ON 22 BIT ADDRESSING
1624 011400 012737 000001 177572 10$: MOV      @BIT0,@#SRO  ;ENABLE MEMORY MANAGEMENT
1625 011406 004537 012454              JSR      R5,SETVEC    ;GO SET UP INTR RETURN
1626 011412 011644                      3$          ;RETURN TO 3$ ON INTR
1627 011414 012737 040000 001736      MOV      #40000,DBUFP ;POINT TO KIPAR2
1628 011422 012700 001564              MOV      @TSTMAP,R0   ;SET TSTMAP START ADDRESS
1629 011426 012701 000020              MOV      #16.,R1      ;SET COUNTER = 16.
1630 011432 005020                      11$: CLR      (R0)+      ;CLEAR TSTMAP
1631 011434 077102                      SOB      R1,11$      ;BR IF NOT END
1632 011436 012737 001524 001664      MOV      @MEMMAP,$MMAP ;SET MEMMAP ADRS TO $MMAP
1633 011444 012737 001564 001666      MOV      @TSTMAP,$TMAP ;SET TSTMAP ADRS TO $TMAP
1634 011452 012737 000002 001670      MOV      @BIT1,$TTST  ;SET TEST BANK BIT TO $TTST
1635 011460 012737 000400 172344      MOV      #400,@#KIPAR2 ;SET PAR2 TO 8K + 2
1636 011466 012737 000400 172346      MOV      #400,@#KIPAR3 ;SET PAR3 TO 12K + 2
1637 011474 012737 040000 001672      MOV      #40000,$TBAE ;SET $TBAE TO 8K + 2
1638 011502 012737 000000 001674      MOV      #0,$TBAE     ;SET $TBAE TO 8K + 2
1639 011510 004737 013700              1$: JSR      PC,KBDSRV   ;TEST FOR "C"
1640 011514 004737 013216              JSR      PC,LDBUF1    ;GO SET UP SPECIAL COMPLEMENT PATTERN ;KJL001
1641 011520 005777 170166              TST      @DRVWCR      ;CLR BAE FLAG TO ENSURE BAR ACCESS NEXT ;KJL001
1642 011524 013777 001672 170162      MOV      $TBAE,@DRVBAR ;SET UP BUFFER ADRS
1643 011532 013777 001674 170162      MOV      $TBAE,@DRVBAE ;SET UP EXTENDED BUFFER ADDRESS
1644 011540 012777 177470 170144      MOV      #-200.,@DRVWCR ;SET UP FOR 200 XFER'S
1645 011546 106427 000000              MTPS     #PRO        ;ALLOW INTR
1646 011552 012777 010101 170136      MOV      #10101,@DRVCSR ;SET MAINT, IE & GO
1647 011560 052777 000400 170130      BIS      #400,@DRVCSR  ;SET CYCLE
1648 011566 012737 000000 001126      MOV      #0,$BDDAT     ;SET UP A COUNTER
1649 011574 005237 001126              2$: INC      $BDDAT    ;COUNT AWAY
1650 011600 001375                      BNE      2$          ;SHOULD ALWAYS INTR FROM THIS LOOP
1651 011602 013737 001716 001122      MOV      DRVCSR,$BDAAR ;SET UP CSR ADRS
1652 011610 012737 010700 001124      MOV      #10700,$GDDAT ;LD EXPECTED
1653 011616 017737 170074 001126      MOV      @DRVCSR,$BDDAT ;READ CSR
1654 011624 042737 000060 001126      BIC      #60,$BDDAT    ;CLEAR XAD17, XAD16
1655 011632 042777 000100 170056      BIC      #100,@DRVCSR  ;DISABLE IE
1656 011640 104005                      ERROR+5 ;NO INTR AFTER 200 MAINT MODE XFER'S
1657 011642 000421                      BR       4$          ;GO RESTORE VECTOR
1658 011644 022626                      3$: CMP      (SP)+,(SP)+ ;RETURN HERE ON INTR - FIX STACK SINCE NO RTI
1659 011646 004537 012704              JSR      R5,CKSTA1    ;GO CHECK STATUS
1660 011652 010700 10700              200.          ;EXPECTED STATUS
1661 011654 000310                      200.          ;# OF XFER'S
1662 011656 104007                      ERROR+7 ;RETURN HERE IF STATUS ER - EXPECTED MAINT.
1663
1664 011660 000412                      BR       4$          ;CYCLE, READY & IE
1665 011662 104010                      ERROR+10 ;GO RESTORE VECTOR
1666 011664 000410                      BR       4$          ;RETURN HERE IF WC ER - SHOULD = 0
1667 011666 104011                      ERROR+11 ;GO RESTORE VECTOR
1668 011670 000406                      BR       4$          ;RETURN HERE IF BAR ER - SHOULD = DBUFP+620
1669 011672 004537 013452              JSR      R5,CKDAT2    ;GO RESTORE VECTOR
1670 011676 000310 200.              200.          ;RETURN HERE IF OK - NOW GO CK DATA
1671 011700 104013                      ERROR+13 ;# OF XFER'S TO CK
1672 011702 000401                      BR       4$          ;RETURN HERE IF DATA ER
                                ;GO RESTORE VECTOR

```

T33 TEST A 200 WORD MAINT MODE XFER WITH MMU

```

1673 011704 000411          BR      5$      ;ALL GOOD RETURN HERE
1674 011706 012737 000000 177572 4$: MOV    $0,$SR0      ;DISABLE MEMORY MANAGEMENT
1675 011714 012737 025224 001736      MOV    $DBUF,$DBUF    ;RESTORE BUFFER ADRS TO LOWEST 4K
1676 011722 004737 012474          JSR     PC,$STVEC    ;GO RESTORE VECTOR
1677 011726 000445          BR      NXDEV    ;GO NEXT DEVICE
1678 011730 006337 001670          ASL     $TTST    ;UPDATE BANK POINTER
1679 011734 013777 001670 167724      MOV    $TTST,$TMAP    ;SET NEXT TEST BANK
1680 011742 001011          BNE     6$      ;IF NOT END OF THIS MEMMAP ENTRY CONTINUE
1681 011744 012737 000001 001670      MOV    $BIT0,$TTST    ;SET NEW BANK POINTER
1682 011752 062737 000002 001664      ADD    $2,$MMAP    ;UPDATE TO NEXT MEMMAP ENTRY
1683 011760 062737 000002 001666      ADD    $2,$TMAP    ;UPDATE TO NEXT TSTMAP ENTRY
1684 011766 033777 001670 167670 6$: BIC     $TTST,$MMAP    ;IS MEMORY AVAILABLE
1685 011774 001744          BEQ     4$      ;NO, BR TO END OF TEST
1686 011776 013700 001736      MOV    $DBUF,$R0    ;SET BUFFER START ADDRESS
1687 012002 012701 000310      MOV    $200,$R1    ;SET COUNT = 200.
1688 012006 005020          7$: CLR     (R0)+    ;CLEAR MEMORY
1689 012010 077102          SOB     R1,7$    ;BR IF NOT DONE
1690 012012 062737 000400 172344      ADD    $400,$KIPAR2    ;SET NEXT 8K
1691 012020 062737 000400 172346      ADD    $400,$KIPAR3    ;SET NEXT 8K
1692 012026 062737 040000 001672      ADD    $40000,$TBAE    ;SET NEXT 8K
1693 012034 005537 001674      ADC     $TBAE    ;ADD CARRY INTO HIGH ORDER
1694 012040 000623          BR      1$      ;TRY NEXT BANK
1695          ;*****
1696          ;DON'T REPORT 'END OF PASS' UNTIL ALL SELECTED DRV11'S HAVE BEEN TESTED
1697          ;*****
1698 012042 000004          NXDEV: SCOPE
1699 012044 000241          NXDEV1: CLC
1700 012046 004737 013700      1$: JSR     PC,$BDSRV    ;CLR CARRY
1701 012052 006037 001732      ROR     DMAP    ;TEST FOR "+C" ;KJL001
1702 012056 001443          BEQ     2$      ;LOOK FOR NEXT
1703 012060 062737 000010 001712      ADD    $10,$DRVWCR    ;BR IF ALL TESTED ;RJL001
1704 012066 062737 000010 001714      ADD    $10,$DRVBAR    ;OFFSET BASE BUS ADRS TO NEXT DRV11WA
1705 012074 062737 000010 001716      ADD    $10,$DRVCSR    ;
1706 012102 062737 000010 001720      ADD    $10,$DRVDBR    ;
1707 012110 062737 000010 001722      ADD    $10,$DRVBAE    ;
1708 012116 062737 000002 022746      ADD    $2,$VECTOR    ;INCREMENT THE TABLE ADDRESS ;KJL001
1709 012124 017737 010616 001724      MOV    $VECTOR,$DRVCT0 ;SET UP NEXT VECTOR (FROM VECTOR TABLE) ;RJL001
1710 012132 017737 010610 001726      MOV    $VECTOR,$DRVCT2 ;SET UP NEXT VECTOR (FROM VECTOR TABLE) ;RJL001
1711 012140 062737 000002 001726      ADD    $2,$DRVCT2    ;MAKE THE 2ND VECTOR CORRECT ;RJL001
1712 012146 005237 001206          INC     $UNIT    ;COUNT DEVICE
1713 012152 032737 000001 001732      BIT     $1,$DMAP    ;IS IT SELECTED?
1714 012160 001731          BEQ     NXDEV1    ;BR IF NOT
1715 012162 000137 002616          JMP     RESTRT    ;TEST NEXT
1716 012166 012737 022750 022746 2$: MOV    $WORK1,$VECTOR ;RESET VECTOR TABLE ADDRESS ;RJL001
1717          ;FALL THROUGH TO END OF PASS ;RJL001
1718          .SBTTL END OF PASS ROUTINE
          ;*****
          ;*INCREMENT THE PASS NUMBER ($PASS)
          ;*TYPE "END PASS $XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
          ;*IF THERES A MONITOR GO TO IT
          ;*IF THERE ISN'T JUMP TO START1
          $EOP:
          SCOPE
          CLR     $TSTNM    ;;ZERO THE TEST NUMBER
          CLR     $TIMES    ;;ZERO THE NUMBER OF ITERATIONS
          INC     $PASS     ;;INCREMENT THE PASS NUMBER
          BIC     $100000,$PASS ;;DON'T ALLOW A NEG. NUMBER

```

END OF PASS ROUTINE

```

012220 005327          DEC      (PC)+          ;;LOOP?
012222 000001      $EOPCT: .WORD      1
012224 003024          BGT      $DOAGN          ;;YES
012226 012737          MOV      (PC)+,@(PC)+    ;;RESTORE COUNTER
012230 000001      $ENDCT: .WORD      1
012232 012222          $EOPCT
012234 104401 012310    TYPE      , $ENDMG          ;;TYPE "END PASS #"
012240 013746 001202    MOV      $PASS,-(SP)      ;;SAVE $PASS FOR TYPEOUT
012244 104405          TYPDS          ;;GO TYPE--DECIMAL ASCII WITH SIGN
012246 104401 012302    TYPE      , $ENDLF          ;;TYPE A CR LF
012252 104401 012305    TYPE      , $ENULL          ;;TYPE A NULL CHARACTER
012256 013700 000042    $GET42: MOV      @42,R0      ;;GET MONITOR ADDRESS
012262 001405          BEQ      $DOAGN          ;;BRANCH IF NO MONITOR
012264 000005          RESET          ;;CLEAR THE WORLD
012266 004710      $ENDAD: JSR      PC,(R0)          ;;GO TO MONITOR
012270 000240          NOP          ;;SAVE ROOM
012272 000240          NOP          ;;FOR
012274 000240          NOP          ;;ACT11
012276          $DOAGN:
012276 000137          JMP      @ (PC)+          ;;RETURN
012300 002344      $RTNAD: .WORD      START1
012302          015      012      000      $ENDLF: .ASCIZ  <15><12>          ;;CR LF
012305          377      377      000      $ENULL: .BYTE   -1,-1,0          ;;NULL CHARACTER STRING
012310          015      012      011      $ENDMG: .ASCIZ  <15><12>/      END PASS #/
012313          105      116      104
012316          040      120      101
012321          123      123      040
012324          043      000

1719          ;;*****
1720          .SBTTL  START OF PROGRAM SUBROUTINE AREA
1721          ;;*****
1722
1723          ;;*****
1724          .SBTTL  MEMORY MANAGEMENT AND ADDRESSING SUBROUTINES.
1725          ;*  SET UP ALL THE MEM MGMT REGISTERS FOR NORMAL OPERATION.
1726          ;*  THE PROGRAM IS POINTED TO BY PARS 0 AND 1.
1727          ;*  THE MEMORY UNDER TEST IS POINTED BY PARS 2 AND 3.
1728          ;*  THE DEVICE ADDRESS AREA IS POINTED TO BY PAR 7.
1729          ;*  PARS 4, 5, AND 6 ARE UNUSED.
1730          ;;*****
1731 012326      MMINIT:
1732 012326 012737 077406 172300      MOV      #200-1*400+UP+RW,@KIPDR0      ;SET KIPDR0 = RW UP 200 BLOCKS
1733 012334 012737 077406 172302      MOV      #200-1*400+UP+RW,@KIPDR1      ;SET KIPDR1 = RW UP 200 BLOCKS
1734 012342 012737 077406 172304      MOV      #200-1*400+UP+RW,@KIPDR2      ;SET KIPDR2 = RW UP 200 BLOCKS
1735 012350 012737 077406 172306      MOV      #200-1*400+UP+RW,@KIPDR3      ;SET KIPDR3 = RW UP 200 BLOCKS
1736 012356 005037 172310      CLR      @KIPDR4
1737 012362 005037 172312      CLR      @KIPDR5
1738 012366 005037 172314      CLR      @KIPDR6
1739 012372 012737 077406 172316      MOV      #200-1*400+UP+RW,@KIPDR7      ;SET KIPDR7 = RW UP 200 BLOCKS
1740 012400 005037 172340      CLR      @KIPAR0      ;MAP PAR0 INTO BANK0
1741 012404 012737 000200 172342      MOV      #200,@KIPAR1      ;MAP PAR1 INTO BANK1
1742 012412 005037 172344      CLR      @KIPAR2      ;MAP PAR2 INTO BANK0
1743 012416 005037 172346      CLR      @KIPAR3
1744 012422 005037 172350      CLR      @KIPAR4
1745 012426 005037 172352      CLR      @KIPAR5
1746 012432 005037 172354      CLR      @KIPAR6
1747 012436 012737 177600 172356      MOV      #177600,@KIPAR7      ;MAP PAR7 INTO I/O BANK

```


MEMORY MANAGEMENT AND ADDRESSING SUBROUTINES.

```

1748 012444 012737 000001 177572 1$:  MOV    #1,00SRO      ;ENABLE MEMORY MANAGEMENT
1749 012452 000207                RTS    PC              ;RETURN
1750
1751
1752      ;;*****
1753      .SBTTL SET UP INTERRUPT RETURN ADDRESS
1754      ;THIS ROUTINE SETS THE PRIORITY LEVEL FOR NO INTERRUPT =
1755      ;SETS UP THE DRV11WA INTERRUPT TO RETURN ON INTERRUPT
1756      ;TO THE ADDRESS INDICATED ((#5)) BY THE CALL +2
1757 012454 106427 000200      SETVEC: MTPS   #PR4      ;SET UP FOR NO INTERRUPT
1758 012460 012577 167240      MOV    (R5)+,00DRVCT0 ;SET UP INTR RETURN ADRS
1759 012464 012777 000200 167234      MOV    #200,00DRVCT2 ;KEEP PRIORITY LEVEL AT TOP ON INTR
1760 012472 000205                RTS    R5              ;EXIT
1761
1762      ;;*****
1763      .SBTTL RESTORE INTERRUPT VECTOR TO HALT
1764      ;THIS ROUTINE CLEARS THE DRV11WA CSR - RESTORES THE DRV11WA
1765      ;INTERRUPT VECTOR TO A HALT - RAISES PRIORITY LEVEL
1766      ;;*****
1767 012474 005077 167216      RSTVEC: CLR    00DRVCSR      ;CLR STATUS & CONTROL
1768 012500 013777 001726 167216      MOV    DRVCT2,00DRVCT0 ;POINT VECTOR TO HALT
1769 012506 005077 167214      CLR    00DRVCT2      ;SET UP HALT
1770 012512 106427 000200      MTPS   #PR4      ;RAISE PRIORITY LEVEL
1771 012516 000207                RTS    PC              ;EXIT
1772
1773      ;;*****
1774      .SBTTL CHECK STATUS OF ALL DATA TRANSFERS FOR ERRORS
1775      ;THIS ROUTINE CHECKS ON ALL DATA TRANSFERS FOR:CORRECT STATUS,
1776      ;CORRECT WORD COUNT & CORRECT BUFFER ADDRESS - THE EXPECTED DATA IS
1777      ;SUPPLIED IN THE CALL +2 & +4 - THE RETURN IS TO +22 IF NO ERRORS
1778      ;DETECTED - IF AN ERROR IS DETECTED THE RETURN IS TO THE APPROPRIATE ERROR EMT
1779      ;;*****
1780 012520 017737 167172 001126      CKSTAT: MOV    00DRVCSR,#BDDAT ;READ THE STATUS
1781 012526 042777 000100 167162      BIC    #100,00DRVCSR ;DISABLE THE IE BIT
1782 012534 012537 001124      MOV    (R5)+,#GDDAT ;SET UP EXPECTED STATUS
1783 012540 023737 001124 001126      CMP    #GDDAT,#BDDAT ;CORRECT?
1784 012546 001406      BEQ    1$ ;BR IF SO
1785 012550 013737 001716 001122      MOV    DRVCSR,#BDADR ;SET UP CSR ADRS
1786 012556 062705 000002      ADD    #2,R5 ;POINT TO THE CSR ER
1787 012562 000205      RTS    R5 ;EXIT HERE ON STATUS ERROR
1788 012564 017737 167122 001126 1$: MOV    00DRVWCR,#BDDAT ;SET WC
1789 012572 001410      BEQ    2$ ;BR IF ZERO
1790 012574 013737 001712 001122      MOV    DRVWCR,#BDADR ;SET UP WCR ADRS
1791 012602 005037 001124      CLR    #GDDAT ;EXPECTED 0
1792 012606 062705 000006      ADD    #6,R5 ;POINT TO THE WCR ER
1793 012612 000205      RTS    R5 ;EXIT HERE ON WCR ER
1794 012614 011537 001124      2$: MOV    (R5),#GDDAT ;SET XFER ?
1795 012620 006337 001124      ASL    #GDDAT ;CONVERT TO WORD
1796 012624 063737 001736 001124      ADD    DBUFP,#GDDAT ;SET XFER ?
1797 012632 005777 167054      TST    00DRVWCR ;CLR BAE FLAG TO ENSURE BAR ACCESS NEXT ;KJL001
1798 012636 017737 167052 001126      MOV    00DRVBAR,#BDDAT ;GET BAR
1799 012644 042737 000001 001126      BIC    #BIT00,#BDDAT ;DON'T WANT BIT00
1800 012652 023737 001124 001126      CMP    #GDDAT,#BDDAT ;CORRECT?
1801 012660 001406      BEQ    3$ ;BR IF SO
1802 012662 013737 001714 001122      MOV    DRVBAR,#BDADR ;SET UP BAR ADRS
1803 012670 062705 000012      ADD    #12,R5 ;POINT TO BAR ER
1804 012674 000205      RTS    R5 ;EXIT HERE ON BAR ER

```


CHECK STATUS OF ALL DATA TRANSFERS FOR ERRORS

```

1805 012676 062705 000016      3#:  ADD    #16,R5      ;ALL OK - POINT TO GOOD EXIT
1806 012702 000205              RTS    R5      ;EXIT HERE IF NO ERRORS
1807
1808
1809      ;*****
1810      ;THIS ROUTINE CHECKS ON ALL DATA TRANSFERS FOR:CORRECT STATUS,
1811      ;CORRECT WORD COUNT & CORRECT BUFFER ADDRESS - THE EXPECTED DATA IS
1812      ;SUPPLIED IN THE CALL +2 & +4 - THE RETURN IS TO +22 IF NO ERRORS
1813      ;DETECTED - IF AN ERROR IS DETECTED THE RETURN IS TO THE APPROPRIATE ERROR EMT
1814      ;*****
1814 012704 017737 167006 001126 CKSTA1: MOV    @DRVCSR,#BDDAT ;READ THE STATUS
1815 012712 042777 000100 166776      BIC    #100,@DRVCSR ;DISABLE THE IE BIT
1816 012720 012537 001124      MOV    (R5)+,#GDDAT ;SET UP EXPECTED STATUS
1817 012724 042737 000060 001126      BIC    #60,#BDDAT ;CLEAR XAD17, XAD16
1818 012732 023737 001124 001126      CMP    @GDDAT,#BDDAT ;CORRECT?
1819 012740 001406      BEQ    1# ;BR IF SO
1820 012742 013737 001716 001122      MOV    DRVCSR,#BDADR ;SET UP CSR ADRS
1821 012750 062705 000002      ADD    #2,R5 ;POINT TO THE CSR ER
1822 012754 000205      RTS    R5 ;EXIT HERE ON STATUS ERROR
1823 012756 017737 166730 001126 1#: MOV    @DRVWCR,#BDDAT ;SET WC
1824 012764 001410      BEQ    2# ;BR IF ZERO
1825 012766 013737 001712 001122      MOV    DRVWCR,#BDADR ;SET UP WCR ADRS
1826 012774 005037 001124      CLR    @GDDAT ;EXPECTED 0
1827 013000 062705 000006      ADD    #6,R5 ;POINT TO THE WCR ER
1828 013004 000205      RTS    R5 ;EXIT HERE ON WCR ER
1829 013006 005777 166700      2#: TST    @DRVWCR ;CLR BAE FLAG TO ENSURE BAR ACCESS NEXT ;KJL001
1830 013012 011537 001124      MOV    (R5),#GDDAT ;SET XFER ?
1831 013016 006337 001124      ASL    @GDDAT ;CONVERT TO WORD
1832 013022 013700 001672      MOV    $TBAR,R0 ;GET START ADDRESS
1833 013026 060037 001124      ADD    R0,#GDDAT ;GET EXPECTED ADDRESS
1834 013032 005537 001674      ADC    $TBAE ;ADD CARRY INTO BAE
1835 013036 017737 166652 001126      MOV    @DRVBAR,#BDDAT ;GET BAR
1836 013044 042737 000001 001126      BIC    #BIT00,#BDDAT ;DON'T WANT BIT00
1837 013052 023737 001124 001126      CMP    @GDDAT,#BDDAT ;CORRECT?
1838 013060 001406      BEQ    3# ;BR IF SO
1839 013062 013737 001714 001122      MOV    DRVBAR,#BDADR ;SET UP BAR ADRS
1840 013070 062705 000012      5#: ADD    #12,R5 ;POINT TO BAR ER
1841 013074 000205      RTS    R5 ;EXIT HERE ON BAR ER
1842 013076 013737 001674 001124 3#: MOV    $TBAE,#GDDAT ;SET EXPECTED BAE
1843 013104 017737 166612 001126      MOV    @DRVBAE,#BDDAT ;GET BAE
1844 013112 023737 001124 001126      CMP    @GDDAT,#BDDAT ;CORRECT?
1845 013120 001404      BEQ    4# ;BR IF SO
1846 013122 013737 001714 001122      MOV    DRVBAR,#BDADR ;SET UP BAR ADRS
1847 013130 000757      BR      5# ;BR BAE ERROR EXIT
1848 013132 062705 000016      4#: ADD    #16,R5 ;ALL OK - POINT TO GOOD EXIT
1849 013136 000205      RTS    R5 ;EXIT HERE IF NO ERRORS
1850      ;*****
1851      ;.SBTTL LOAD DATA BUFFER WITH FLOATING PATTERN
1852      ;THIS ROUTINE LOADS 'DBUF' WITH A FLOATING ZERO/ONE PATTERN
1853      ;FOR 199 LOCATIONS - THE LAST LOCATION IS LOADED WITH THE
1854      ;#70707 WHICH SHOULD BE THE DATA WORD AVAILABLE IN THE DBR
1855      ;AT THE COMPLETION OF A 200 WORD TRANSFER
1856      ;*****
1857 013140 012703 025224      LDBUF: MOV    @DBUF,R3 ;SET BUFFER ADRS
1858 013144 012704 177776      1#: MOV    #177776,R4 ;SET UP FLOATING ZERO PATRN
1859 013150 004737 013700      2#: JSR    PC,KBDSRV ;TEST FOR "+C"
1860 013154 010423      MOV    R4,(R3)+ ;LOAD IT (FLOATING 0)
1861 013156 005104      COM    R4 ;MAKE INTO FLOATING 1

```

;KJL001

LOAD DATA BUFFER WITH FLOATING PATTERN

```

1862 013160 022703 026042      CMP    #DBUF+616,R3      ;AT END OF BUFFER?
1863 013164 001003              BNE     4$              ;BR IF NOT
1864 013166 012713 070707      3$:   MOV    #70707,(R3)      ;LOAD LAST DATVAR(SPECIAL)
1865 013172 000207              RTS     PC              ;SET OUT
1866 013174 010423              4$:   MOV    R4,(R3)+        ;LOAD IT (FLOATING 1)
1867 013176 022703 026042      CMP    #DBUF+616,R3      ;AT END OF BUFFER?
1868 013202 001771              BEQ     3$              ;BR IF SO
1869 013204 005104              COM     R4              ;BACK TO FLOATING ZERO
1870 013206 006304              ASL     R4              ;SHIFT LEFT
1871 013210 005204              INC     R4              ;KEEP LSS SET
1872 013212 103354              BCC     1$              ;GO RESET FLOATING PATRN
1873 013214 000755              BR      2$              ;GO LOAD NEXT PATRN
1874
1875
1876      ;*****
1877      ;THIS ROUTINE LOADS "DBUF" WITH A UNIQUE FLOATING ZERO/ONE PATTERN
1878      ;(:17776,0,1,0,17775,0,2,0,17773,0,4,0,17767,0,10,0 ETC.).
1879      ;IT IS USED WITH MAINT BIT SET (DATI/DATO SEQUENCE). 200 LOCATIONS
1880      ;ARE LOADED WITH THIS PATTERN.
1881      ;*****
1881 013216 013703 001736      LDBUF1: MOV    DBUF,R3      ;SET BUFFER ADRS
1882 013222 010305              MOV     R3,R5      ;SAVE IN R5
1883 013224 062705 000620      ADD     #620,R5      ;POINT TO END OF BUFFER
1884 013230 012704 177776      1$:   MOV    #17776,R4      ;SET UP FLOATING ZERO PATRN
1885 013234 004737 013700      2$:   JSR     PC,KBDSRV      ;TEST FOR "+C"
1886 013240 010423              MOV     R4,(R3)+      ;LOAD IT (FLOATING 0)
1887 013242 005023              CLR     (R3)+      ;ZERO NEXT
1888 013244 005104              COM     R4              ;SET UP FLOATING :
1889 013246 010423              MOV     R4,(R3)+      ;LOAD IT
1890 013250 005023              CLR     (R3)+      ;ZERO NEXT
1891 013252 020503              CMP     R5,R3      ;200 LOCS DONE?
1892 013254 001001              BNE     3$              ;BR IF NOT
1893 013256 000207              RTS     PC              ;SET OUT
1894 013260 005104              3$:   COM     R4              ;BACK TO FLOATING ZERO
1895 013262 006304              ASL     R4              ;SHIFT LEFT
1896 013264 005204              INC     R4              ;KEEP LSS SET
1897 013266 103360              BCC     1$              ;GO RESET FLOATING PATRN
1898 013270 000761              BR      2$              ;GO FLOAT NEXT PATRN
1899
1900      ;*****
1901      ;.SBTTL CHECK DATA ON 'DATO' TRANSFERS FOR ERRORS
1902      ;THIS ROUTINE CHECKS 200 LOCATIONS IN "DBUF" FOR GOOD TRANSFERED
1903      ;DATA (#177577) ON 'DATO' TRANSFERS - IF AN ERROR IS DETECTED
1904      ;THE RETURN IS TO CALL +2 - IF NO ERROR THE RETRUN IS TO CALL +4
1905      ;*****
1905 013272 012701 025224      CKDAT: MOV    #DBUF,R1      ;SET BUFFER ADRS
1906 013276 004737 013700      1$:   JSR     PC,KBDSRV      ;TEST FOR "+C"
1907 013302 022721 177377      CMP     #177377,(R1)+      ;DATA OK?
1908 013306 001410              BEQ     2$              ;BR IF SO
1909 013310 013737 001720 001122      MOV     DRVDBR,$BADDR      ;SET UP DBR ADRS
1910 013316 014137 001126      MOV     -(R1),$BDDAT      ;SET ACTUAL DATA XFERED
1911 013322 010137 001120      MOV     R1,$GDADR      ;SET MEMORY ADRS
1912 013326 000207              RTS     PC              ;RETURN TO ERROR
1913 013330 022701 026044      2$:   CMP    #DBUF+620,R1      ;AT END OF 'DBUF'?
1914 013334 001360              BNE     1$              ;BR IF MORE
1915 013336 062716 000002      ADD     #2,(SP)      ;ADJUST STACK FOR GOOD RETURN
1916 013342 000207              RTS     PC              ;SET OUT
1917
1918      ;*****

```

CHECK DATA ON MAINT MODE' TRANSFERS FOR ERRORS

```

1919 .SBTTL CHECK DATA ON 'MAINT MODE' TRANSFERS FOR ERRORS
1920 ;THIS ROUTINE CHECK 200 LOCATIONS IN 'DBUF' FOR GOOD TRANSFERED
1921 ;DATA (177776,177776,1,1,177775,177776,2,2,177773,177773,ETC)
1922 ;ON MAINT MODE TRANSFERS - THE NUMBER OF CHECKS REQUIRED IS INDICATED
1923 ;BY THE CALL +2 - IF AN ERROR IS DETECTED THE RETURN IS TO CALL +4 =
1924 ;IF NO ERROR THE RETURN IS TO CALL +10
1925 ;:*****
1926 013344 012500 CKDAT1: MOV (R5)+,R0 ;SET # OF CHECKS
1927 013346 013702 001736 MOV DBUFP,R2 ;SET BUFFER ADRS
1928 013352 012701 177776 1$: MOV #177776,R1 ;SET UP FLOATING ZERO PATRN
1929 013356 005003 CLR R3 ;R3 SAYS WHEN TO SHIFT PATRN
1930 013360 004737 013700 2$: JSR PC,KBDSRV ;TEST FOR "+C" ;KJL001
1931 013364 020122 CMP R1,(R2)+ ;DATA OK?
1932 013366 001010 BNE 3$ ;BR IF NOT
1933 013370 020122 CMP R1,(R2)+ ;DATA WRITTEN OK?
1934 013372 001006 BNE 3$ ;BR IF NOT
1935 013374 162700 000002 SUB #2,R0 ;ACCOUNT FOR TWO ADR'S
1936 013400 003015 BGT 4$ ;BR IF MORE
1937 013402 062705 000004 ADD #4,R5 ;ADJUST FOR GOOD RETURN
1938 013406 000205 RTS R5 ;EXIT
1939 013410 014237 001126 3$: MOV -(R2),#BDDAT ;SET BAD DATA
1940 013414 010237 001120 MOV R2,#GDADR ;GET MEM ADRS
1941 013420 010137 001124 MOV R1,#GDDAT ;LO EXPECTED DATA
1942 013424 013737 001720 001122 MOV DRVDBR,#BDADR ;SET UP DBR ADRS
1943 013432 000205 RTS R5 ;RETURN TO ERROR
1944 013434 005101 4$: COM R1 ;NOW EXPECT COMPLEMENT
1945 013436 005103 COM R3 ;TIME TO SHIFT?
1946 013440 001347 BNE 2$ ;BR IF NOT
1947 013442 006301 ASL R1 ;SHIFT LEFT
1948 013444 005201 INC R1 ;KEEP LSS SET
1949 013446 103341 BCC 1$ ;GO RESET FLOATING LPATRN
1950 013450 000743 BR 2$ ;GO NEXT

1951 ;:*****
1952 ;THIS ROUTINE CHECK 200 LOCATIONS IN 'DBUF' FOR GOOD TRANSFERED
1953 ;DATA (177776,177776,1,1,177775,177776,2,2,177773,177773,ETC)
1954 ;ON MAINT MODE TRANSFERS - THE NUMBER OF CHECKS REQUIRED IS INDICATED
1955 ;BY THE CALL +2 - IF AN ERROR IS DETECTED THE RETURN IS TO CALL +4 =
1956 ;IF NO ERROR THE RETURN IS TO CALL +10
1957 ;:*****
1958 013452 012500 CKDAT2: MOV (R5)+,R0 ;SET # OF CHECKS
1959 013454 013702 001736 MOV DBUFP,R2 ;SET BUFFER ADRS
1960 013460 012701 177776 1$: MOV #177776,R1 ;SET UP FLOATING ZERO PATRN
1961 013464 005003 CLR R3 ;R3 SAYS WHEN TO SHIFT PATRN
1962 013466 004737 013700 2$: JSR PC,KBDSRV ;TEST FOR "+C" ;KJL001
1963 013472 020122 CMP R1,(R2)+ ;DATA OK?
1964 013474 001010 BNE 3$ ;BR IF NOT
1965 013476 020122 CMP R1,(R2)+ ;DATA WRITTEN OK?
1966 013500 001006 BNE 3$ ;BR IF NOT
1967 013502 162700 000002 SUB #2,R0 ;ACCOUNT FOR TWO ADR'S
1968 013506 003015 BGT 4$ ;BR IF MORE
1969 013510 062705 000004 ADD #4,R5 ;ADJUST FOR GOOD RETURN
1970 013514 000205 RTS R5 ;EXIT
1971 013516 014237 001126 3$: MOV -(R2),#BDDAT ;SET BAD DATA
1972 013522 010237 001120 MOV R2,#GDADR ;GET MEM ADRS
1973 013526 010137 001124 MOV R1,#GDDAT ;LO EXPECTED DATA
1974 013532 013737 001720 001122 MOV DRVDBR,#BDADR ;SET UP DBR ADRS

```

CHECK DATA ON 'MAINT MODE' TRANSFERS FOR ERRORS

```

1976 013540 000205
1977 013542 005101
1978 013544 005103
1979 013546 001347
1980 013550 006301
1981 013552 005201
1982 013554 103341
1983 013556 000743
1984
1985
1986
1987
1988
1989
1990
1991 013560 104401 025042
1992 013564 104411
1993 013566 012603
1994 013570 105713
1995 013572 001402
1996 013574 005203
1997 013576 000774
1998 013600 114300
1999 013602 122700 000131
2000 013606 001412
2001 013610 122700 000171
2002 013614 001407
2003 013616 122700 000116
2004 013622 001410
2005 013624 122700 000156
2006 013630 001405
2007 013632 000752
2008 013634 012737 000001 001704
2009 013642 000402
2010 013644 005037 001704
2011 013650 000207
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022 013652 005003
2023 013654 106427 000000
2024 013660 010277 166032
2025 013664 052777 000400 166024
2026 013672 005303
2027 013674 001376
2028 013676 000207
2029
2030
2031
2032

      RTS      R5      ;RETURN TO ERROR
4$:    COM      R1      ;NOW EXPECT COMPLEMENT
      COM      R3      ;TIME TO SHIFT?
      BNE      2$      ;BR IF NOT
      ASL      R1      ;SHIFT LEFT
      INC      R1      ;KEEP LSS SET
      BCC      1$      ;GO RESET FLOATING LPATRN
      BR       2$      ;GO NEXT

;*****
;      .SBTTL  ASK USER CPU TYPE
;THIS ROUTINE ASKS THE USER IF THE SYSTEM SUPPORTS 22-BIT ADDRESSING
;*****
;KJL001
;KJL001
;KJL001

ASKQ22: TYPE,    Q22CPU      ;ASK OPERATOR IF CPU SUPPORTS 22-BIT ADDRESSING
      RDLIN      ;WAIT INPUT
      MOV      (SP)+,R3      ;GET ADDRESS OF FIRST CHARACTER
1$:    TSTB      (R3)      ;TEST CHARACTER IS NULL
      BEQ      10$      ;YES NULL
      INC      R3      ;POINT NEXT CHARACTER
      BR       1$      ;TEST NEXT CHARACTER
10$:   MOVB      -(R3),R0      ;GET LAST CHARACTER
      CMPB      #131,R0      ;"Y" ?
      BEQ      2$      ;YES
      CMPB      #171,R0      ;"y" ?
      BEQ      2$      ;YES
      CMPB      #116,R0      ;"N" ?
      BEQ      3$      ;YES
      CMPB      #156,R0      ;"n" ?
      BEQ      3$      ;YES
      BR       ASKQ22      ;TRY AGAIN
2$:    MOV      #1,LSIQ22      ;SET 22-BIT CPU FLAG
      BR       4$
3$:    CLR      LSIQ22      ;CLEAR 22-BIT CPU FLAG
4$:    RTS      PC      ;RETURN

;*****
;      .SBTTL  INTERRUPT TIMER FOR BURST MODE TRANSFERS
;THIS ROUTINE IS TO SET UP A TIMER AND WAIT FOR INTERRUPT TO OCCUR
;WITHIN THE TIME FRAME.  THE INTERRUPT RETURN ADDRESS SHOULD HAVE BEEN
;SET UP PRIOR TO CALLING THIS ROUTINE.  IF NO INTERRUPT OCCURS BEFORE
;TIMER RUNS OUT, THE BURST MODE TRANSFER HAS FAILED AND CONTROL IS
;RETURNED TO LOCATION PC+2.
;*****
;KJL002
;KJL002
;KJL002
;KJL002
;KJL002
;KJL002

TIMER:  CLR      R3      ;INITIALIZE R3 AS COUNTER
      MTPS      #PRO      ;ENABLE THE INTERRUPT
      MOV      R2,@DRVCSR      ;SET BITS IN CSR
      BIS      #400,@DRVCSR      ;SET CYCLE
1$:    DEC      R3      ;DECREMENT COUNTER
      BNE      1$      ;WAIT MORE TIME
      RTS      PC      ;RETURN TO MAIN PROGRAM (NO INTERRUPT)
;KJL004
;KJL002
;KJL002
;KJL002
;KJL004
;KJL002
;KJL002

;*****
;      .SBTTL  KEYBOARD SERVICE ROUTINE
;THIS ROUTINE WILL SERVICE THE TTY KEYBOARD.
;*****
;KJL001
;KJL001

```

KEYBOARD SERVICE ROUTINE

;KJL001

SEQ 0043

```

2033      ;IF THE CHARACTER TYPED IS A "CONTROL-C" (+C), EXIT IS MADE TO THE      ;KJL001
2034      ;"CONTROL-C" RESTART ADDRESS (CSTART). IF THE CHARACTER IS CONTROL G    ;KJL001
2035      ;(+G), THE SOFTWARE SWITCH WILL BE SET UP IF IT IS ENABLED.              ;KJL001
2036      ;*****
2037 013700 105777 165240      KBDSRV: TSTB      @TKS      ;TEST KEYBOARD STATUS      ;KJL001
2038 013704 100401      BMI      KBDSR1      ;BR IF A KEY IS HIT      ;KJL001
2039 013706 000207      RTS      PC      ;EXIT      ;KJL001
2040 013710 117746 165232      KBDSR1: MOVB      @TKB,-(SP)      ;PICK UP CHARACTER TYPED      ;KJL001
2041 013714 042716 177600      BIC      @177600,(SP)      ;STRIP OFF THE JUNK      ;KJL001
2042 013720 021627 000003      CMP      (SP),#3      ;IS IT A CONTROL-C?      ;KJL001
2043 013724 001005      BNE      1$      ;BRANCH IF NO      ;KJL001
2044 013726 104401 017566      TYPE      ,CNTLC      ;TYPE A CONTROL-C (+C)      ;KJL001
2045 013732 005726      TST      (SP)+      ;POP THE STACK      ;KJL001
2046 013734 000137 002310      JMP      CSTART      ;EXIT TO CSTART      ;KJL001
2047 013740 022716 000005      1$: CMP      @5,(SP)      ;CTRL E?      ;KJL001
2048 013744 001004      BNE      100$      ;BR IF NOT      ;KJL001
2049 013746 005726      TST      (SP)+      ;POP THE STACK      ;KJL001
2050 013750 004737 020050      JSR      PC,EXAM      ;GO TO THE EXAMINE ROUTINE      ;KJL001
2051 013754 000207      RTS      PC      ;RETURN      ;KJL001
2052 013756 022726 000007      100$: CMP      @7,(SP)+      ;IS IT A CONTROL G?      ;KJL001
2053 013762 001070      BNE      10$      ;BR IF NOT      ;KJL001
2054 013764 022737 000176 001140      CMP      @SWREG,SWR      ;IS THE SOFT-SWR SELECTED?      ;KJL001
2055 013772 001064      BNE      10$      ;BRANCH IF NO      ;KJL001
2056 013774 123727 001134 000001      CMPB      $AUTOB,#1      ;ARE WE RUNNING IN AUTO-MODE?      ;KJL001
2057 014002 001460      BEQ      10$      ;BRANCH IF YES      ;KJL001
2058 014004 104401 017600      TYPE      ,CNTLG      ;ECHO THE CONTROL-G (+G)      ;KJL001
2059 014010 104401 017605      TYPE      ,MSWR      ;TYPE CURRENT CONTENTS      ;KJL001
2060 014014 013746 000176      MOV      SWREG,-(SP)      ;SAVE SWREG FOR TYPEOUT      ;KJL001
2061 014020 104402      TYPOC      ;GO TYPE--OCTAL ASCII(ALL DIGITS)      ;KJL001
2062 014022 104401 017616      TYPE      ,MNEW      ;PROMPT FOR NEW SWR      ;KJL001
2063 014026 005046      2$: CLR      -(SP)      ;CLEAR COUNTER      ;KJL001
2064 014030 005046      CLR      -(SP)      ;THE NEW SWR      ;KJL001
2065 014032 105777 165106      3$: TSTB      @TKS      ;CHAR THERE?      ;KJL001
2066 014036 100375      BPL      3$      ;IF NOT TRY AGAIN      ;KJL001
2067 014040 117746 165102      MOVB      @TKB,-(SP)      ;PICK UP CHAR      ;KJL001
2068 014044 042716 177600      BIC      @C177,(SP)      ;MAKE IT 7-BIT ASCII      ;KJL001
2069 014050 021627 000003      CMP      (SP),#3      ;IS IT A CONTROL-C?      ;KJL001
2070 014054 001006      BNE      5$      ;BRANCH IF NOT      ;KJL001
2071 014056 104401 017566      TYPE      ,CNTLC      ;YES, ECHO CONTROL-C (+C)      ;KJL001
2072 014062 062706 000006      ADD      @6,SP      ;CLEAN UP STACK      ;KJL001
2073 014066 000137 002310      4$: JMP      CSTART      ;CONTROL-C RESTART      ;KJL001
2074 014072 021627 000025      5$: CMP      (SP),#25      ;IS IT A CONTROL-U?      ;KJL001
2075 014076 001005      BNE      7$      ;BRANCH IF NOT      ;KJL001
2076 014100 104401 017573      TYPE      ,CNTLU      ;YES, ECHO CONTROL-U (+U)      ;KJL001
2077 014104 062706 000006      6$: ADD      @6,SP      ;IGNORE PREVIOUS INPUT      ;KJL001
2078 014110 000746      BR      2$      ;LET'S TRY IT AGAIN      ;KJL001
2079 014112 021627 000015      7$: CMP      (SP),#15      ;IS IT A <CR>?      ;KJL001
2080 014116 001013      BNE      11$      ;BRANCH IF NO      ;KJL001
2081 014120 005766 000004      TST      4(SP)      ;YES, IS IT THE FIRST CHAR?      ;KJL001
2082 014124 001403      BEQ      8$      ;BRANCH IF YES      ;KJL001
2083 014126 016677 000002 165004      MOV      2(SP),@SWR      ;SAVE NEW SWR      ;KJL001
2084 014134 062706 000006      8$: ADD      @6,SP      ;CLEAR UP STACK      ;KJL001
2085 014140 104401 001171      9$: TYPE      ,CRLF      ;ECHO <CR> AND <LF>      ;KJL001
2086 014144 000207      10$: RTS      PC      ;RETURN      ;KJL001
2087 014146 004737 014440      11$: JSR      PC,$TYPEC      ;ECHO CHAR      ;KJL001
2088 014152 021627 000060      CMP      (SP),#60      ;CHAR < 0?      ;KJL001
2089 014156 002420      BLT      13$      ;BRANCH IF YES      ;KJL001

```

KEYBOARD SERVICE ROUTINE

;KJL001

```

2090 014160 021627 000067      CMP      (SP),#67      ;CHAR > 7?      ;KJL001
2091 014164 003015      BGT      13$      ;BRANCH IF YES      ;KJL001
2092 014166 042726 000060      BIC      #60,(SP)+      ;STRIP-OFF ASCII      ;KJL001
2093 014172 005766 000002      TST      2(SP)      ;IS THIS THE FIRST CHAR      ;KJL001
2094 014176 001403      BEQ      12$      ;BRANCH IF YES      ;KJL001
2095 014200 006316      ASL      (SP)      ;NO, SHIFT PRESENT      ;KJL001
2096 014202 006316      ASL      (SP)      ;CHAR OVER TO MAKE      ;KJL001
2097 014204 006316      ASL      (SP)      ;ROOM FOR NEW ONE.      ;KJL001
2098 014206 005266 000002      12$: INC      2(SP)      ;KEEP COUNT OF CHAR      ;KJL001
2099 014212 056616 177776      BIS      -2(SP),(SP)      ;SET IN NEW CHAR      ;KJL001
2100 014216 000705      BR      3$      ;GET THE NEXT ONE      ;KJL001
2101 014220 104401 001170      13$: TYPE ,#QUES      ;TYPE ?<CR><LF>      ;KJL001
2102 014224 000727      BR      6$      ;SIMULATE CONTROL-U      ;KJL001
2103
2104

```

.SBTTL TYPE ROUTINE

;*****

;*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

;*

```

014226 105737 001157      $TYPE: TSTB      $TPFLG      ;;IS THERE A TERMINAL?
014232 100002      BPL      1$      ;;BR IF YES
014234 000000      HALT      ;;HALT HERE IF NO TERMINAL
014236 000430      BR      ;;LEAVE
014240 010046      1$: MOV      R0,-(SP)      ;;SAVE R0
014242 017600 000002      MOV      @2(SP),R0      ;;GET ADDRESS OF ASCIZ STRING
014246 122737 000001 001214      CMPB      #APTENV,$ENV      ;;RUNNING IN APT MODE
014254 001011      BNE      62$      ;;NO,GO CHECK FOR APT CONSOLE
014256 132737 000100 001215      BITB      #APTPOOL,$ENVM      ;;SPOOL MESSAGE TO APT
014264 001405      BEQ      62$      ;;NO,GO CHECK FOR CONSOLE
014266 010037 014276      MOV      R0,61$      ;;SETUP MESSAGE ADDRESS FOR APT
014272 004737 014516      JSR      PC,$ATY3      ;;SPOOL MESSAGE TO APT
014276 000000      61$: .WORD      0      ;;MESSAGE ADDRESS
014300 132737 000040 001215      62$: BITB      #APTCSUP,$ENVM      ;;APT CONSOLE SUPPRESSED
014306 001003      BNE      60$      ;;YES,SKIP TYPE OUT
014310 112046      2$: MOVB      (R0)+,-(SP)      ;;PUSH CHARACTER TO BE TYPED ONTO STACK
014312 001005      BNE      4$      ;;BR IF IT ISN'T THE TERMINATOR
014314 005726      TST      (SP)+      ;;IF TERMINATOR POP IT OFF THE STACK
014316 012600      60$: MOV      (SP)+,R0      ;;RESTORE R0
014320 062716 000002      3$: ADD      #2,(SP)      ;;ADJUST RETURN PC
014324 000002      RTI      ;;RETURN
014326 122716 000011      4$: CMPB      #HT,(SP)      ;;BRANCH IF <HT>
014332 001430      BEQ      8$      ;;BRANCH IF NOT <CRLF>
014334 122716 000200      CMPB      #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
014340 001006      BNE      5$      ;;POP <CR><LF> EQUIV
014342 005726      TST      (SP)+      ;;TYPE A CR AND LF
014344 104401      TYPE
014346 001171      $CRLF

```


TYPE ROUTINE

```

014350 105037 014504          CLRB  $CHARCNT      ;;CLEAR CHARACTER COUNT
014354 000755                BR      2$          ;;GET NEXT CHARACTER
014356 004737 014440          5$: JSR    PC,$TYPEC  ;;GO TYPE THIS CHARACTER
014362 123726 001156          6$: CMPB  $FILLC,(SP)+ ;;IS IT TIME FOR FILLER CHARS.?
014366 001350                BNE    2$          ;;IF NO GO GET NEXT CHAR.
014370 013746 001154          MOV    $NULL,-(SP)  ;;GET # OF FILLER CHARS. NEEDED
                                           ;;AND THE NULL CHAR.
014374 105366 000001          7$: DECB  1(SP)    ;;DOES A NULL NEED TO BE TYPED?
014400 002770                BLT    6$          ;;BR IF NO--GO POP THE NULL OFF OF STACK
014402 004737 014440          JSR    PC,$TYPEC  ;;GO TYPE A NULL
014406 105337 014504          DECB  $CHARCNT    ;;DO NOT COUNT AS A COUNT
014412 000770                BR      7$          ;;LOOP

;HORIZONTAL TAB PROCESSOR
014414 112716 000040          8$: MOVB  #' ,(SP)  ;;REPLACE TAB WITH SPACE
014420 004737 014440          9$: JSR    PC,$TYPEC  ;;TYPE A SPACE
014424 132737 000007 014504  BITB  #7,$CHARCNT  ;;BRANCH IF NOT AT
014432 001372                BNE    9$          ;;TAB STOP
014434 005726                TST    (SP)+       ;;POP SPACE OFF STACK
014436 000724                BR      2$          ;;GET NEXT CHARACTER
014440 105777 164504          $TYPEC: TSTB  #4$TPS  ;;WAIT UNTIL PRINTER IS READY
014444 100375                BPL    $TYPEC
014446 116677 000002 164476  MOVB  2(SP),#4$TPB  ;;LOAD CHAR TO BE TYPED INTO DATA REG.
014454 122766 000015 000002  CMPB  #CR,2(SP)    ;;IS CHARACTER A CARRIAGE RETURN?
014462 001003                BNE    1$          ;;BRANCH IF NO
014464 105037 014504          CLRB  $CHARCNT    ;;YES--CLEAR CHARACTER COUNT
014470 000406                BR      $TYPEX     ;;EXIT
014472 122766 000012 000002  1$: CMPB  #LF,2(SP)  ;;IS CHARACTER A LINE FEED?
014500 001402                BEQ    $TYPEX     ;;BRANCH IF YES
014502 105227                INCB  (PC)+       ;;COUNT THE CHARACTER
014504 000000          $CHARCNT: .WORD 0        ;;CHARACTER COUNT STORAGE
014506 000207          $TYPEX: RTS    PC

2105 .SBTTL APT COMMUNICATIONS ROUTINE
;*****
014510 112737 000001 014754  $ATY1: MOVB  #1,$FFLG  ;;TO REPORT FATAL ERROR
014516 112737 000001 014752  $ATY3: MOVB  #1,$MFLG  ;;TO TYPE A MESSAGE
014524 000403                BR      $ATYC
014526 112737 000001 014754  $ATY4: MOVB  #1,$FFLG  ;;TO ONLY REPORT FATAL ERROR
014534 000000          $ATYC:
014534 010046                MOV    R0,-(SP)    ;;PUSH R0 ON STACK
014536 010146                MOV    R1,-(SP)    ;;PUSH R1 ON STACK
014540 105737 014752          TSTB  $MFLG      ;;SHOULD TYPE A MESSAGE?
014544 001450                BEQ    5$          ;;IF NOT: BR
014546 122737 000001 001214  CMPB  #APTENV,$ENV  ;;OPERATING UNDER APT?
014554 001031                BNE    3$          ;;IF NOT: BR
014556 132737 000100 001215  BITB  #APTPOOL,$ENVM  ;;SHOULD SPOOL MESSAGES?
014564 001425                BEQ    3$          ;;IF NOT: BR
014566 017600 000004          MOV    #4(SP),R0  ;;GET MESSAGE ADDR.
014572 062766 000002 000004  ADD    #2,4(SP)    ;;BUMP RETURN ADDR.
014600 005737 001174          1$: TST    $MSGTYPE  ;;SEE IF DONE W/ LAST XMISSION?
014604 001375                BNE    1$          ;;IF NOT: WAIT
014606 010037 001210          MOV    R0,$MSGAD  ;;PUT ADDR IN MAILBOX
014612 105720                2$: TSTB  (R0)+    ;;FIND END OF MESSAGE
014614 001376                BNE    2$
014616 163700 001210          SUB    $MSGAD,R0  ;;SUB START OF MESSAGE
014622 006200                ASR    R0          ;;GET MESSAGE LENGTH IN WORDS
014624 010037 001212          MOV    R0,$MSGLGT  ;;PUT LENGTH IN MAILBOX
014630 012737 000004 001174  MOV    #4,$MSGTYPE  ;;TELL APT TO TAKE MSG.

```


APT COMMUNICATIONS ROUTINE

```

014636 000413      BR      5$
014640 017637 000004 014664 3$: MOV    04(SP),4$      ;;PUT MSG ADDR IN JSR LINKAGE
014646 062766 000002 000004      ADD    02,4(SP)      ;;BUMP RETURN ADDRESS
014654 013746 177776      MOV    177776,-(SP)      ;;PUSH 177776 ON STACK
014660 004737 014226      JSR    PC,$TYPE      ;;CALL TYPE MACRO
014664 000000      4$: .WORD 0
014666      5$:
014666 105737 014754      10$: TSTB   $FFLG      ;;SHOULD REPORT FATAL ERROR?
014672 001416      BEQ    12$      ;;IF NOT: BR
014674 005737 001214      TST    $ENV      ;;RUNNING UNDER APT?
014700 001413      BEQ    12$      ;;IF NOT: BR
014702 005737 001174      11$: TST    $MSGTYPE      ;;FINISHED LAST MESSAGE?
014706 001375      BNE    11$      ;;IF NOT: WAIT
014710 017637 000004 001176      MOV    04(SP),$FATAL      ;;GET ERROR #
014716 062766 000002 000004      ADD    02,4(SP)      ;;BUMP RETURN ADDR.
014724 005237 001174      INC    $MSGTYPE      ;;TELL APT TO TAKE ERROR
014730 105037 014754      12$: CLRB   $FFLG      ;;CLEAR FATAL FLAG
014734 105037 014753      CLRB   $LFLG      ;;CLEAR LOG FLAG
014740 105037 014752      CLRB   $MFLG      ;;CLEAR MESSAGE FLAG
014744 012601      MOV    (SP)+,R1      ;;POP STACK INTO R1
014746 012600      MOV    (SP)+,R0      ;;POP STACK INTO R0
014750 000207      RTS    PC      ;;RETURN
014752 000      $MFLG: .BYTE 0      ;;MESSG. FLAG
014753 000      $LFLG: .BYTE 0      ;;LOG FLAG
014754 000      $FFLG: .BYTE 0      ;;FATAL FLAG

```

```

000200
000001
000100
000040

```

2106

```

APTSIZE=200
APTENV=001
APTSPool=100
APTCsup=040
.SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
;*****
;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
;OCTAL (ASCII) NUMBER AND TYPE IT.
;$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
;CALL:
;*      MOV    NUM,-(SP)      ;;NUMBER TO BE TYPED
;*      TYPOS      ;;CALL FOR TYPEOUT
;*      .BYTE  N      ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
;*      .BYTE  M      ;;M=1 OR 0
;*                      ;;1=TYPE LEADING ZEROS
;*                      ;;0=SUPPRESS LEADING ZEROS
;$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
;$TYPOS OR $TYPOC
;CALL:
;*      MOV    NUM,-(SP)      ;;NUMBER TO BE TYPED
;*      TYPON      ;;CALL FOR TYPEOUT
;$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
;CALL:
;*      MOV    NUM,-(SP)      ;;NUMBER TO BE TYPED
;*      TYPOC      ;;CALL FOR TYPEOUT
$TYPOS: MOV    0(SP),-(SP)      ;;PICKUP THE MODE
        MOVB   1(SP),$OFILL      ;;LOAD ZERO FILL SWITCH
        MOVB   (SP)+,$OMODE+1      ;;NUMBER OF DIGITS TO TYPE
        ADD    #2,(SP)      ;;ADJUST RETURN ADDRESS

```

```

014756 017646 000000      015201
014762 116637 000001
014770 112637 015203
014774 062716 000002

```

BINARY TO OCTAL (ASCII) AND TYPE

```

015000 000406          BR      $TYPON
015002 112737 000001 015201 $TYPOC: MOVB  #1,$OFILL      ;;SET THE ZERO FILL SWITCH
015010 112737 000006 015203      MOVB  #6,$OMODE+1      ;;SET FOR SIX(6) DIGITS
015016 112737 000005 015200 $TYPON: MOVB  #5,$OCNT      ;;SET THE ITERATION COUNT
015024 010346          MOV    R3,-(SP)      ;;SAVE R3
015026 010446          MOV    R4,-(SP)      ;;SAVE R4
015030 010546          MOV    R5,-(SP)      ;;SAVE R5
015032 113704 015203      MOVB  $OMODE+1,R4      ;;GET THE NUMBER OF DIGITS TO TYPE
015036 005404          NEG    R4
015040 062704 000006      ADD    #6,R4      ;;SUBTRACT IT FOR MAX. ALLOWED
015044 110437 015202      MOVB  R4,$OMODE      ;;SAVE IT FOR USE
015050 113704 015201      MOVB  $OFILL,R4      ;;GET THE ZERO FILL SWITCH
015054 016605 000012      MOV    12(SP),R5      ;;PICKUP THE INPUT NUMBER
015060 005003          CLR    R3      ;;CLEAR THE OUTPUT WORD
015062 006105          1$: ROL    R5      ;;ROTATE MSB INTO "C"
015064 000404          BR      3$      ;;GO DO MSB
015066 006105          2$: ROL    R5      ;;FORM THIS DIGIT
015070 006105          ROL    R5
015072 006105          ROL    R5
015074 010503          MOV    R5,R3
015076 006103          3$: ROL    R3      ;;GET LSB OF THIS DIGIT
015100 105337 015202      DECB  $OMODE      ;;TYPE THIS DIGIT?
015104 100016          BPL     7$      ;;BR IF NO
015106 042703 177770      BIC    #177770,R3      ;;GET RID OF JUNK
015112 001002          BNE     4$      ;;TEST FOR 0
015114 005704          TST    R4      ;;SUPPRESS THIS 0?
015116 001403          BEQ     5$      ;;BR IF YES
015120 005204          4$: INC    R4      ;;DON'T SUPPRESS ANYMORE 0'S
015122 052703 000060      BIS    #'0,R3      ;;MAKE THIS DIGIT ASCII
015126 052703 000040      5$: BIS    #' ,R3      ;;MAKE ASCII IF NOT ALREADY
015132 110337 015176      MOVB  R3,8$      ;;SAVE FOR TYPING
015136 104401 015176      TYPE  ,8$      ;;GO TYPE THIS DIGIT
015142 105337 015200      7$: DECB  $OCNT      ;;COUNT BY 1
015146 003347          BGT     2$      ;;BR IF MORE TO DO
015150 002402          BLT     6$      ;;BR IF DONE
015152 005204          INC    R4      ;;INSURE LAST DIGIT ISN'T A BLANK
015154 000744          BR      2$      ;;GO DO THE LAST DIGIT
015156 012605          6$: MOV    (SP)+,R5      ;;RESTORE R5
015160 012604          MOV    (SP)+,R4      ;;RESTORE R4
015162 012603          MOV    (SP)+,R3      ;;RESTORE R3
015164 016666 000002 000004      MOV    2(SP),4(SP)      ;;SET THE STACK FOR RETURNING
015172 012616          MOV    (SP)+,(SP)
015174 000002          RTI      ;;RETURN
015176 000          8$: .BYTE  0      ;;STORAGE FOR ASCII DIGIT
015177 000          .BYTE  0      ;;TERMINATOR FOR TYPE ROUTINE
015200 000          $OCNT: .BYTE  0      ;;OCTAL DIGIT COUNTER
015201 000          $OFILL: .BYTE  0      ;;ZERO FILL SWITCH
015202 000000          $OMODE: .WORD  0      ;;NUMBER OF DIGITS TO TYPE
2107 .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
;*****
;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
;SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
;NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
;BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
;REPLACED WITH SPACES.
;CALL:
;*      MOV    NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK

```

CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

```

015204          ;*      TYPDS          ;;GO TO THE ROUTINE
015204          $TYPDS:
015204 010046      MOV      R0,-(SP)          ;;PUSH R0 ON STACK
015206 010146      MOV      R1,-(SP)          ;;PUSH R1 ON STACK
015210 010246      MOV      R2,-(SP)          ;;PUSH R2 ON STACK
015212 010346      MOV      R3,-(SP)          ;;PUSH R3 ON STACK
015214 010546      MOV      R5,-(SP)          ;;PUSH R5 ON STACK
015216 012746 020200  MOV      #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
015222 016605 000020  MOV      20(SP),R5      ;;GET THE INPUT NUMBER
015226 100004      BPL      1$              ;;BR IF INPUT IS POS.
015230 005405      NEG      R5              ;;MAKE THE BINARY NUMBER POS.
015232 112766 000055 000001  MOVB     #'-,1(SP)  ;;MAKE THE ASCII NUMBER NEG.
015240 005000      CLR      R0              ;;ZERO THE CONSTANTS INDEX
015242 012703 015420      MOV      #DBLK,R3    ;;SETUP THE OUTPUT POINTER
015246 112723 000040      MOVB     #' ,(R3)+    ;;SET THE FIRST CHARACTER TO A BLANK
015252 005002      CLR      R2              ;;CLEAR THE BCD NUMBER
015254 016001 015410      MOV      $DTBL(R0),R1 ;;GET THE CONSTANT
015260 160105      SUB      R1,R5          ;;FORM THIS BCD DIGIT
015262 002402      BLT      4$              ;;BR IF DONE
015264 005202      INC      R2              ;;INCREASE THE BCD DIGIT BY 1
015266 000774      BR       3$
015270 060105      4$:      ADD      R1,R5    ;;ADD BACK THE CONSTANT
015272 005702      TST      R2              ;;CHECK IF BCD DIGIT=0
015274 001002      BNE      5$              ;;FALL THROUGH IF 0
015276 105716      TSTB     (SP)            ;;STILL DOING LEADING 0'S?
015300 100407      BMI      7$              ;;BR IF YES
015302 106316      5$:      ASLB     (SP)      ;;MSD?
015304 103003      BCC      6$              ;;BR IF NO
015306 116663 000001 177777  MOVB     1(SP),-1(R3) ;;YES--SET THE SIGN
015314 052702 000060      6$:      BIS      #'0,R2  ;;MAKE THE BCD DIGIT ASCII
015320 052702 000040      7$:      BIS      #' ,R2  ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
015324 110223      MOVB     R2,(R3)+    ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
015326 005720      TST      (R0)+        ;;JUST INCREMENTING
015330 020027 000010      CMP      R0,#10    ;;CHECK THE TABLE INDEX
015334 002746      BLT      2$              ;;GO DO THE NEXT DIGIT
015336 003002      BGT      8$              ;;GO TO EXIT
015340 010502      MOV      R5,R2          ;;GET THE LSD
015342 000764      BR       6$              ;;GO CHANGE TO ASCII
015344 105726      8$:      TSTB     (SP)+    ;;WAS THE LSD THE FIRST NON-ZERO?
015346 100003      BPL      9$              ;;BR IF NO
015350 116663 177777 177776  MOVB     -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
015356 105013      9$:      CLRB     (R3)      ;;SET THE TERMINATOR
015360 012605      MOV      (SP)+,R5      ;;POP STACK INTO R5
015362 012603      MOV      (SP)+,R3      ;;POP STACK INTO R3
015364 012602      MOV      (SP)+,R2      ;;POP STACK INTO R2
015366 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
015370 012600      MOV      (SP)+,R0      ;;POP STACK INTO R0
015372 104401 015420      TYPE      ,#DBLK    ;;NOW TYPE THE NUMBER
015376 016666 000002 000004  MOV      2(SP),4(SP) ;;ADJUST THE STACK
015404 012616      MOV      (SP)+,(SP)
015406 000002      RTI
015410 023420      $DTBL: 10000.
015412 001750      1000.
015414 000144      100
015416 000012      10.
015420          $DBLK: .BLKW 4
          .SBTTL  ERROR HANDLER ROUTINE

```

ERROR HANDLER ROUTINE

```

;*****
;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 $WRCK ON ERROR
;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
;SW15=1      HALT ON ERROR
;SW13=1      INHIBIT ERROR TYPEOUTS
;SW10=1      BELL ON ERROR
;SW09=1      LOOP ON ERROR
;CALL
;*
;ERROR:  ERROR  N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER
015430
015430 104407          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
015432 104407          CKSWR          ;
015434 105237 001103  7$:  INCB      $ERFLG      ;;SET THE ERROR FLAG
015440 001775          BEQ          7$          ;;DON'T LET THE FLAG GO TO ZERO
015442 013777 001102 163472  MOV      $TSTNM,$DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
015450 032777 002000 163462  BIT      $BIT10,$SWR      ;;BELL ON ERROR?
015456 001402          BEQ          1$          ;;NO - SKIP
015460 104401 001164          TYPE      , $BELL      ;;RING BELL
015464 005237 001112  1$:  INC      $ERTTL      ;;COUNT THE NUMBER OF ERRORS
015470 011637 001116          MOV      (SP), $ERRPC    ;;GET ADDRESS OF ERROR INSTRUCTION
015474 162737 000002 001116  SUB      $2, $ERRPC
015502 117737 163410 001114  MOV      $ERRPC, $ITEMB    ;;STRIP AND SAVE THE ERROR ITEM CODE
015510 032777 020000 163422  BIT      $BIT13,$SWR      ;;SKIP TYPEOUT IF SET
015516 001004          BNE          20$          ;;SKIP TYPEOUTS
015520 004737 015620          JSR      PC, $WRCK      ;;GO TO USER ERROR ROUTINE
015524 104401 001171          TYPE      , $CRLF
015530
015530 122737 000001 001214  20$:  CMP      $APTENV, $ENV      ;;RUNNING IN APT MODE
015536 001007          BNE          2$          ;;NO, SKIP APT ERROR REPORT
015540 113737 001114 015552  MOV      $ITEMB, 21$      ;;SET ITEM NUMBER AS ERROR NUMBER
015546 004737 014526          JSR      PC, $ATY4      ;;REPORT FATAL ERROR TO APT
015552 000          21$:  .BYTE      0
015553 000          .BYTE      0
015554 000777          22$:  BR          22$          ;;APT ERROR LOOP
015556 005777 163356  2$:  TST      $SWR          ;;HALT ON ERROR
015562 100002          BPL          3$          ;;SKIP IF CONTINUE
015564 000000          HALT          ;;HALT ON ERROR!
015566 104407          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
015570 032777 001000 163342  3$:  BIT      $BIT09,$SWR      ;;LOOP ON ERROR SWITCH SET?
015576 001402          BEQ          4$          ;;BR IF NO
015600 013716 001110          MOV      $LPERR,(SP)      ;;FUDGE RETURN FOR LOOPING
015604 005737 001162  4$:  TST      $ESCAPE      ;;CHECK FOR AN ESCAPE ADDRESS
015610 001402          BEQ          5$          ;;BR IF NONE
015612 013716 001162          MOV      $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
015616
015616 000002  5$:  RTI          ;;RETURN
2109
2110
2111
2112
2113 015620 113737 001102 001730 $WRCK:  MOV      $TSTNM, $TSTNM    ;;SET UP TEST # ON ER
2114 015626 004737 015636          JSR      PC, $ERRTYP    ;;GO TYPE ERROR
2115 015632 104407          CKSWR          ;;GO LOOK FOR SWR CHANGE
2116 015634 000207          RTS      PC          ;;RETURN TO ERROR HANDLER
2117
.SBTTL  ERROR MESSAGE TYPEOUT ROUTINE

```

ERROR MESSAGE TYPEOUT ROUTINE

```

;*****
;*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:
015636      015636 104401 001171      TYPE      , $CRLF      ;; "CARRIAGE RETURN" & "LINE FEED"
015642      015642 010046      MOV      R0, -(SP)      ;; SAVE R0
015644      015644 005000      CLR      R0      ;; PICKUP THE ITEM INDEX
015646      015646 153700 001114      BISB     @ $ITEMB, R0
015652      015652 001004      BNE      1$      ;; IF ITEM NUMBER IS ZERO, JUST
                                ;; TYPE THE PC OF THE ERROR
015654      015654 013746 001116      MOV      $ERRPC, -(SP)      ;; SAVE $ERRPC FOR TYPEOUT
                                ;; ERROR ADDRESS
015660      015660 104402      TYP0C     ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
015662      015662 000426      BR       6$      ;; GET OUT
015664      015664 005300      1$: DEC      R0      ;; ADJUST THE INDEX SO THAT IT WILL
015666      015666 006300      ASL      R0      ;; WORK FOR THE ERROR TABLE
015670      015670 006300      ASL      R0
015672      015672 006300      ASL      R0
015674      015674 062700 001320      ADD      @ $ERRTB, R0      ;; FORM TABLE POINTER
015700      015700 012037 015710      MOV      (R0)+, 2$      ;; PICKUP "ERROR MESSAGE" POINTER
015704      015704 001404      BEQ      3$      ;; SKIP TYPEOUT IF NO POINTER
015706      015706 104401      TYPE     ;; TYPE THE "ERROR MESSAGE"
015710      015710 000000      2$: .WORD   0      ;; "ERROR MESSAGE" POINTER GOES HERE
015712      015712 104401 001171      TYPE     , $CRLF      ;; "CARRIAGE RETURN" & "LINE FEED"
015716      015716 012037 015726      3$: MOV      (R0)+, 4$      ;; PICKUP "DATA HEADER" POINTER
015722      015722 001404      BEQ      5$      ;; SKIP TYPEOUT IF 0
015724      015724 104401      TYPE     ;; TYPE THE "DATA HEADER"
015726      015726 000000      4$: .WORD   0      ;; "DATA HEADER" POINTER GOES HERE
015730      015730 104401 001171      TYPE     , $CRLF      ;; "CARRIAGE RETURN" & "LINE FEED"
015734      015734 011000      5$: MOV      (R0), R0      ;; PICKUP "DATA TABLE" POINTER
015736      015736 001004      BNE      7$      ;; GO TYPE THE DATA
015740      015740 012600      6$: MOV      (SP)+, R0      ;; RESTORE R0
015742      015742 104401 001171      TYPE     , $CRLF      ;; "CARRIAGE RETURN" & "LINE FEED"
015746      015746 000207      RTS      PC      ;; RETURN
015750      015750      7$:      MOV      @ (R0)+, -(SP)      ;; SAVE @ (R0)+ FOR TYPEOUT
015752      015752 104402      TYP0C     .. GO TYPE--OCTAL ASCII(ALL DIGITS)
015754      015754 005710      TST      (R0)      ;; IS THERE ANOTHER NUMBER?
015756      015756 001770      BEQ      6$      ;; BR IF NO
015760      015760 104401 015766      TYPE     , 8$      ;; TYPE TWO(2) SPACES
015764      015764 000771      BR       7$      ;; LOOP
015766      015766 040 040 000 8$: .ASCIZ  / /      ;; TWO(2) SPACES
                                .EVEN

```

```

2118      ;*****
2119      ;* THIS ROUTINE WILL TYPE OUT A MAP OF MEMORY BANKS.
2120      ;* UPON ENTERING THIS ROUTINE R0 WILL CONTAIN THE ADDRESS
2121      ;* OF THE TABLE TO BE TYPED OUT. THE ROUTINE REQUIRES THAT THE
2122      ;* TABLE CONTAIN 16 DECIMAL WORDS.
2123      ;* STORAGE USED:
2124      ;*      R0      = MAP TABLE ADDRESS
2125      ;*      R1      = BANK POINTER
2126      ;*      R2      = CONSECUTIVE ZERO BANK FLAG
2127      ;*      R3      = HIGH MEMORY ADR ... LO 16 BITS
2128      ;*      R4      = HIGH MEMORY ADR ... HI 6 BITS
2129      ;*      LMEMLO  = LOW MEMORY ADR ... LO 16 BITS
2130      ;*      LMEMHI  = LOW MEMORY ADR ... HI 6 BITS

```

ERROR MESSAGE TYPEOUT ROUTINE

```

2131          ;*          BNKTAB  = ADDRESS OF TABLE TO BE PROCESSED
2132          ;*          ENDBKT   = LAST ADDRESS + 2 OF TABLE TO BE PROCESSED
2133          ;*          FLG30K   = 30K MEMORY FLAG
2134          ;*****
2135 015772 010037 016424          TYPMAP: MOV    R0,#BNKTAB      ;SAVE ADDRESS OF TABLE
2136 015776 012737 000040 016426 MOV    #40,#ENDBKT      ;LOAD END BANK WITH 40
2137 016004 060037 016426          ADD    R0,#ENDBKT      ;GET LAST ADR + 2 INTO ENDBKT
2138 016010 005710          1$:   TST    (R0)            ;ANY MEMORY IN THIS TABLE ENTRY
2139 016012 001010          BNE    2$                    ; THEN GO PRINT OUT LOCATIONS
2140 016014 005720          TST    (R0)+                ; ELSE POINT TO NEXT TABLE ENTRY
2141 016016 023700 016426          CMP    #ENDBKT,R0      ;IF NOT END OF TABLE
2142 016022 001372          BNE    1$                    ; THEN TRY AGAIN
2143 016024 104401 025017          TYPE,  NOMEM          ;GO PRINT OUT THE FOLLOWING MESSAGE.
2144          ;'NO MEMORY FOUND'
2145 016030 000137 016416          JMP    13$            ;EXIT
2146 016034 013700 016424          2$:   MOV    #BNKTAB,R0    ;RESTORE MAP TABLE ADDRESS
2147 016040 010146          MOV    R1,-(SP)              ;;PUSH R1 ON STACK
2148 016042 010246          MOV    R2,-(SP)              ;;PUSH R2 ON STACK
2149 016044 010346          MOV    R3,-(SP)              ;;PUSH R3 ON STACK
2150 016046 010446          MOV    R4,-(SP)              ;;PUSH R4 ON STACK
2151 016050 005037 016420          CLR    LMEMLO          ;INIT LOW MEM ADR ... LO 16 BITS
2152 016054 005037 016422          CLR    LMEMHI          ;INIT LOW MEM ADR ... HI 6 BITS
2153 016060 005037 001700          CLR    TEMP           ;INIT LAST BANK IN MAP FLAG
2154 016064 012703 177777          MOV    #-1,R3         ;INIT HIGH MEM ADR ... LO 16 BITS
2155 016070 010304          MOV    R3,R4                ;INIT HIGH MEM ADR ... HI 6 BITS
2156 016072 012702 000001          MOV    #1,R2          ;INIT CONSECUTIVE ZERO BANK FLAG
2157 016076 012701 000001          3$:   MOV    #BIT0,R1    ;INIT BANK POINTER
2158 016102 030110          4$:   BIT    R1,(R0)          ;IF THIS 8K BANK IS NOT PRESENT
2159 016104 001405          BEQ    5$                    ; THEN GO PRINTOUT MAP
2160 016106 062703 040000          ADD    #40000,R3       ; ELSE UPDATE HIGH MEM TO TOP OF THIS BANK
2161 016112 005504          ADC     R4
2162 016114 005002          CLR    R2
2163 016116 000517          BR     11$
2164 016120 005702          5$:   TST    R2              ;CLEAR CONSECUTIVE ZERO BANK FLAG
2165 016122 001101          BNE    10$                   ;GO CHECK FOR NEXT BANK
2166 016124 005202          INC     R2                    ;IF CONSECUTIVE ZERO BANK
2167 016126 022703 177777          CMP    #177777,R3      ; THEN DO NOT PRINT MESSAGE
2168 016132 001023          BVC     9$                    ; ELSE UPDATE CONSECUTIVE ZERO BANK FLAG
2169 016134 005737 001710          BIT    #MAVA          ;IF NOT 32K, 128K OR 2M BOUNDARY
2170 016140 100413          BMI     7$                    ; THEN GO PRINT MESSAGE
2171 016142 001006          BNE    6$                    ; ELSE IF 22-BIT ADDRESSING
2172 016144 005737 001676          TST    FLG30K          ; THEN GO WORK ON 22-BIT ADDRESSING
2173 016150 001412          BEQ    8$                    ; OR GO WORK ON 18 BIT ADDRESSING
2174 016152 162703 010000          SUB    #10000,R3       ; ELSE IF NOT 30K SYSTEM
2175 016156 000411          BR     9$                    ; THEN GO ADJ TO 28K BOUNDARY
2176 016160 022704 000003          6$:   CMP    #3,R4      ; ELSE ADJ TO 30K BOUNDARY
2177 016164 001006          BNE    9$                    ;AND GO PRINT
2178 016166 000403          BR     8$                    ;IF NOT 128K BOUNDARY
2179 016170 022704 000077          7$:   CMP    #77,R4      ; THEN GO PRINT
2180 016174 001002          BNE    9$                    ; ELSE GO ADJ TO 124K BOUNDARY
2181 016176 152703 020000          8$:   SUB    #20000,R3   ;IF NOT 2M BOUNDARY
2182 016202          9$:          ;ADJ HIGH MEMORY TO BELOW I/O PAGE
2183 016202 010346          MOV    R3,-(SP)              ;;PUSH R3 ON STACK
2184 016204 010446          MOV    R4,-(SP)              ;;PUSH R4 ON STACK
2185 016206 013746 016420          MOV    LMEMLO,-(SP)    ;;PUSH LMEMLO ON STACK
2186 016212 013746 016422          MOV    LMEMHI,-(SP)    ;;PUSH LMEMHI ON STACK
2187 016216 104401 024675          TYPE,  FROM          ;GO PRINT OUT THE FOLLOWING MESSAGE.

```


ERROR MESSAGE TYPEOUT ROUTINE

PC	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418	Op419
----	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

ERROR MESSAGE TYPEOUT ROUTINE

2217 016420 000000
 2218 016422 000000
 2219 016424 000000
 2220 016426 000000
 2221
 2222
 2223
 2224

LMEMLO: .WORD 0 ; LOW MEMORY ADR ... LO 16 BITS
 LMEMHI: .WORD 0 ; LOW MEMORY ADR ... HI 6 BITS
 BNKTAB: .WORD 0 ; ADDRESS OF TABLE TO BE PROCESSED
 ENDBKT: .WORD 0 ; LAST ADDRESS + 2 OF TABLE

.NLIST BEX

.SBTTL SCOPE HANDLER ROUTINE

 *THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
 *AND LOAD THE TEST NUMBER(\$TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
 *AND LOAD THE ERROR FLAG (\$ERFLG) INTO DISPLAY<15:08>
 *THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
 *SW14=1 LOOP ON TEST
 *SW11=1 INHIBIT ITERATIONS
 *SW09=1 LOOP ON ERROR
 *SW08=1 LOOP ON TEST IN SWR<7:0>
 *CALL

* SCOPE ;:SCOPE=IOT

016430				\$SCOPE:		
01643C	104407			CKSWR		
016432	032777	040000	162500	1\$: BIT	\$BIT14,\$SWR	::TEST FOR CHANGE IN SOFT SWR
016440	001114			BNE	\$OVER	::LOOP ON PRESENT TEST?
						::YES IF SW14=1
016442	000416			*****START OF CODE FOR THE XOR TESTER*****		
				\$XTSTR: BR	6\$	::IF RUNNING ON THE "XOR" TESTER CHANGE
016444	013746	000004		MOV	\$ERRVEC,-(SP)	::THIS INSTRUCTION TO A "NOP" (NOP=240)
016450	012737	016470	000004	MOV	\$5,\$ERRVEC	::SAVE THE CONTENTS OF THE ERROR VECTOR
016456	005737	177060		TST	\$177060	::SET FOR TIMEOUT
016462	012637	000004		MOV	(SP)+,\$ERRVEC	::TIME OUT ON XOR
016466	000463			BR	\$SVLAD	::RESTORE THE ERROR VECTOR
016470	022626			5\$: CMP	(SP)+,(SP)+	::GO TO THE NEXT TEST
016472	012637	000004		MOV	(SP)+,\$ERRVEC	::CLEAR THE STACK AFTER A TIME OUT
016476	000423			BR	7\$	::RESTORE THE ERROR VECTOR
016500				6\$: *****END OF CODE FOR THE XOR TESTER*****		::LOOP ON THE PRESENT TEST
016500	032777	000400	162432	BIT	\$BIT08,\$SWR	::LOOP ON SPEC. TEST?
016506	001404			BEQ	2\$	::BR IF NO
016510	127737	162424	001102	CMPB	\$SWR,\$TSTNM	::ON THE RIGHT TEST? SWR<7:0>
016516	001465			BEQ	\$OVER	::BR IF YES
016520	105737	001103		2\$: TSTB	\$ERFLG	::HAS AN ERROR OCCURRED?
016524	001421			BEQ	3\$	::BR IF NO
016526	123737	001115	001103	CMPB	\$ERMAX,\$ERFLG	::MAX. ERRORS FOR THIS TEST OCCURRED?
016534	101015			BHI	3\$	::BR IF NO
016536	032777	001000	162374	BIT	\$BIT09,\$SWR	::LOOP ON ERROR?
016544	001404			BEQ	4\$	::BR IF NO
016546	013737	001110	001106	7\$: MOV	\$LPERR,\$LPADR	::SET LOOP ADDRESS TO LAST SCOPE
016554	000446			BR	\$OVER	
016556	105037	001103		4\$: CLRB	\$ERFLG	::ZERO THE ERROR FLAG
016562	005037	001160		CLR	\$TIMES	::CLEAR THE NUMBER OF ITERATIONS TO MAKE
016566	000415			BR	1\$	::ESCAPE TO THE NEXT TEST
016570	032777	004000	162342	3\$: BIT	\$BIT11,\$SWR	::INHIBIT ITERATIONS?
016576	001011			BNE	1\$	::BR IF YES
016600	005737	001202		TST	\$PASS	::IF FIRST PASS OF PROGRAM
016604	001406			BEQ	1\$	:: INHIBIT ITERATIONS
016606	005237	001104		INC	\$ICNT	::INCREMENT ITERATION COUNT
016612	023737	001160	001104	CMP	\$TIMES,\$ICNT	::CHECK THE NUMBER OF ITERATIONS MADE
016620	002024			BGE	\$OVER	::BR IF MORE ITERATION REQUIRED

SCOPE HANDLER ROUTINE

```

016622 012737 C00001 001104 1$: MOV    #1,$ICNT      ;;REINITIALIZE THE ITERATION COUNTER
016630 013737 016706 001160      MOV    $MXCNT,$TIMES  ;;SET NUMBER OF ITERATIONS TO DO
016636 105237 001102      $SVLAD: INCB   $TSTNM      ;;COUNT TEST NUMBERS
016642 113737 001102 001200      MOV    $TSTNM,$TESTN  ;;SET TEST NUMBER IN APT MAILBOX
016650 011637 001106      MOV    (SP),$LPADR      ;;SAVE SCOPE LOOP ADDRESS
016654 011637 001110      MOV    (SP),$LPERR      ;;SAVE ERROR LOOP ADDRESS
016660 005037 001162      CLR    $ESCAPE      ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
016664 112737 000001 001115      MOV    #1,$ERMAX      ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
016672 013777 001102 162242 $OVER: MOV    $TSTNM,$DISPLAY  ;;DISPLAY TEST NUMBER
016700 013716 001106      MOV    $LPADR,(SP)      ;;FUDGE RETURN ADDRESS
016704 000002      RTI                          ;;FIXES PS
016706 000050      $MXCNT: 50                    ;;MAX. NUMBER OF ITERATIONS

2225 .LIST BIN
2226 .SBTTL TTY INPUT ROUTINE
      ;*****
      .ENABL LSB
      ;*****
      ;*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.
016710 105777 162230      $CKSWR: TSTB   @TKS          ;;CHAR THERE?
016714 100144      BPL    15$                    ;;IF NO, DON'T WAIT AROUND
016716 117746 162224      MOV    @TKB,-(SP)        ;;SAVE THE CHAR
016722 042716 177600      BIC    @C177,(SP)        ;;STRIP-OFF THE ASCII
      ;
016726 021627 000023      CMP    (SP),#23          ;;IS IT A CONTROL-S?      ;KJL001
016732 001014      BNE    132$                    ;;BRANCH IF NO          ;KJL001
016734 005726      TST    (SP)+                    ;;CLEAN CHAR OFF STACK ;KJL001
016736 105777 162202      131$: TSTB   @TKS          ;;WAIT FOR A CHAR      ;KJL001
016742 100375      BPL    131$                    ;;LOOP UNTIL ITS THERE ;KJL001
016744 117746 162176      MOV    @TKB,-(SP)        ;;GET THE CHARACTER   ;KJL001
016750 042716 177600      BIC    @C177,(SP)        ;;MAKE IT 7 BIT ASCII ;KJL001
016754 022627 000021      CMP    (SP)+,#21          ;;IS IT A CONTROL-Q?      ;KJL001
016760 001366      BNE    131$                    ;;BRANCH IF NO          ;KJL001
016762 000521      BR     15$                    ;;NO, RETURN TO USER   ;KJL001
      ;
016764 022716 000003      132$: CMP    #3,(SP)      ;;IS IT A CONTORL C?      ;RJL001
016770 001003      BNE    30$                    ;;BR IF NOT          ;RJL001
016772 005726      TST    (SP)+                    ;;POP THE STACK      ;RJL001
016774 000137 002310      JMP    CSTART            ;;RESTART IF +C      ;RJL001
017000 022737 000176 001140      30$: CMP    $SWREG,SWR      ;;IS THE SOFT-SWR SELECTED? ;RJL001
017006 001402      BEQ    31$                    ;;BR IF YES          ;RJL001
017010 005726      TST    (SP)+                    ;;POP THE STACK      ;RJL001
017012 000505      BR     15$                    ;;BRANCH TO EXIT     ;RJL001
017014 022726 000007      31$: CMP    #7,(SP)+      ;;IS IT A CONTROL G?      ;RJL001
017020 001102      BNE    15$                    ;;NO, RETURN TO USER   ;RJL001
017022 123727 001134 000001      CMP    $AUTOB,#1        ;;ARE WE RUNNING IN AUTO-MODE?
017030 001476      BEQ    15$                    ;;BRANCH IF YES      ;RJL001
017032 104401 017600      TYPE    , $CNTLG          ;;ECHO THE CONTROL-G (+G)
017036 104401 017605      $GTSWR: TYPE    , $MSWR        ;;TYPE CURRENT CONTENTS
017042 013746 000176      MOV    SWREG,-(SP)        ;;SAVE SWREG FOR TYPEOUT
017046 104402      TYPOC      ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
017050 104401 017616      TYPE    , $MNEW          ;;PROMPT FOR NEW SWR
017054 005046      19$: CLR    -(SP)              ;;CLEAR COUNTER
017056 005046      CLR    -(SP)              ;;THE NEW SWR
017060 105777 162060      7$: TSTB   @TKS          ;;CHAR THERE?

```

TTY INPUT ROUTINE

```

017064 100375      BPL      7$      ;; IF NOT TRY AGAIN
017066 117746 162054 MOV8     @TKB,-(SP)  ;; PICK UP CHAR
017072 042716 177600 BIC      @C177,(SP)  ;; MAKE IT 7 BIT ASCII
017076 021627 000003 CMP      (SP),@3      ;; IS IT A CONTROL-C?
017102 001015      BNE      9$      ;; BRANCH IF NOT
017104 104401 017566 TYPE     ,CNTLC  ;; YES, ECHO CONTROL-C (+C)
017110 062706 000006 ADD      @6,SP  ;; CLEAN UP STACK
017114 123727 001135 000001 CMPB   $INTAG,@1  ;; REENABLE TTY KEYBOARD INTERRUPTS?
017122 001003      BNE      8$      ;; BRANCH IF NO
017124 012777 000100 162012 MOV     @100,@TKS  ;; ALLOW TTY KEYBOARD INTERRUPTS
017132 000137 002310 8$:      JMP     CSTART  ;; CONTROL-C RESTART
017136 021627 000025 9$:      CMP     (SP),@25  ;; IS IT A CONTROL-U?
017142 001005      BNE      10$     ;; BRANCH IF NOT
017144 104401 017573 TYPE     ,CNTLU  ;; YES ECHO CONTROL-U (+U)
017150 062706 000006 20$:     ADD     @6,SP  ;; IGNORE PREVIOUS INPUT
017154 000737      BR       19$     ;; LET'S TRY IT AGAIN
017156 021627 000015 10$:     CMP     (SP),@15  ;; IS IT A <CR>?
017162 001022      BNE      16$     ;; BRANCH IF NO
017164 005766 000004 TST      4(SP)  ;; YES, IS IT THE FIRST CHAR?
017170 001403      BEQ      11$     ;; BRANCH IF YES
017172 016677 000002 161740 MOV     2(SP),@SWR  ;; SAVE NEW SWR
017200 062706 000006 11$:     ADD     @6,SP  ;; CLEAR UP STACK
017204 104401 001171 14$:     TYPE     ,CRLF  ;; ECHO <CR> AND <LF>
017210 123727 001135 000001 CMPB   $INTAG,@1  ;; RE-ENABLE TTY KBD INTERRUPTS?
01721- 001003      BNE      15$     ;; BRANCH IF NOT
017220 012777 000100 161716 MOV     @100,@TKS  ;; RE-ENABLE TTY KBD INTERRUPTS
017226 000002 15$:      RTI      ;; RETURN
017230 004737 014440 16$:     JSR     PC,$TYPEC  ;; ECHO CHAR
017234 021627 000060      CMP     (SP),@60  ;; CHAR < 0?
017240 002420      BLT      18$     ;; BRANCH IF YES
017242 021627 000067      CMP     (SP),@67  ;; CHAR > 7?
017246 003015      BGT      18$     ;; BRANCH IF YES
017250 042726 000060      BIC     @60,(SP)+  ;; STRIP-OFF ASCII
017254 005766 000002 TST      2(SP)  ;; IS THIS THE FIRST CHAR
017260 001403      BEQ      17$     ;; BRANCH IF YES
017262 006316      ASL      (SP)  ;; NO, SHIFT PRESENT
017264 006316      ASL      (SP)  ;; CHAR OVER TO MAKE
017266 006316      ASL      (SP)  ;; ROOM FOR NEW ONE.
017270 005266 000002 17$:     INC     2(SP)  ;; KEEP COUNT OF CHAR
017274 056616 177776      BIS     -2(SP),(SP)  ;; SET IN NEW CHAR
017300 000667      BR       7$     ;; GET THE NEXT ONE
017302 104401 001170 18$:     TYPE     ,QUES  ;; TYPE ?<CR><LF>
017306 000720      BR       20$     ;; SIMULATE CONTROL-U
      .DSABL  LSB
      ;;*****
      ;;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
      ;
017310 011646      $RDCHR: MOV     (SP),-(SP)  ;; PUSH DOWN THE PC
017312 016666 000004 000002 MOV     4(SP),2(SP)  ;; SAVE THE PS
017320 105777 161620 1$:      TSTB   @TKS  ;; WAIT FOR
017324 100375      BPL      1$      ;; A CHARACTER
017326 117766 161614 000004 MOV8     @TKB,4(SP)  ;; READ THE TTY
017334 042766 177600 000004 BIC      @C<177>,4(SP)  ;; GET RID OF JUNK IF ANY

```

TTY INPUT ROUTINE

```

017342 026627 000004 000023      CMP      4(SP),#23      ;;IS IT A CONTROL-S?
017350 001013      BNE      3$      ;;BRANCH IF NO
017352 105777 161566      2$:      TSTB      0$TKS      ;;WAIT FOR A CHARACTER
017356 100375      BPL      2$      ;;LOOP UNTIL ITS THERE
017360 117746 161562      MOV      0$TKB,-(SP)      ;;GET CHARACTER
017364 042716 177600      BIC      0+C177,(SP)      ;;MAKE IT 7-BIT ASCII
017370 022627 000021      CMP      (SP)+,#21      ;;IS IT A CONTROL-Q?
017374 001366      BNE      2$      ;;IF NOT DISCARD IT
017376 000750      BR      1$      ;;YES, RESUME
017400 026627 000004 000140 3$:      CMP      4(SP),#140      ;;IS IT UPPER CASE?
017406 002407      BLT      4$      ;;BRANCH IF YES
017410 026627 000004 000175      CMP      4(SP),#175      ;;IS IT A SPECIAL CHAR?
017416 003003      BGT      4$      ;;BRANCH IF YES
017420 042766 000040 000004      BIC      040,4(SP)      ;;MAKE IT UPPER CASE
017426 000002      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
017430 010346      $RDLIN: MOV      R3,-(SP)      ;;SAVE R3
017432 012703 017556 1$:      MOV      0$TTYIN,R3      ;;GET ADDRESS
017436 022703 017566 2$:      CMP      0$TTYIN+8.,R3      ;;BUFFER FULL?
017442 101415      BLOS      4$      ;;BR IF YES
017444 104410      RDCHR      ;;GO READ ONE CHARACTER FROM THE TTY
017446 112613      MOV      (SP)+,(R3)      ;;GET CHARACTER
017450 122713 000003      CMP      03,(R3)      ;;IS IT A CONTROL-C?
017454 001005      BNE      10$      ;;BRANCH IF NO
017456 104401 017566      TYPE      , $CNTLC      ;;TYPE A CONTROL-C (+C)
017462 012603      MOV      (SP)+,R3      ;;RESTORE R3
017464 000137 002310      JMP      CSTART      ;;GOTO CONTROL-C RESTART
017470 122713 000177 10$:      CMP      0177,(R3)      ;;IS IT A RUBOUT
017474 001003      BNE      3$      ;;SKIP IF NOT
017476 104401 001170 4$:      TYPE      , $QUES      ;;TYPE A '?'
017502 000753      BR      1$      ;;CLEAR THE BUFFER AND LOOP
017504 111337 017554 3$:      MOV      (R3),9$      ;;ECHO THE CHARACTER
017510 104401 017554      TYPE      ,9$
017514 122723 000015      CMP      015,(R3)+      ;;CHECK FOR RETURN
017520 001346      BNE      2$      ;;LOOP IF NOT RETURN
017522 105063 177777      CLRB      -1(R3)      ;;CLEAR RETURN (THE 15)
017526 104401 001172      TYPE      , $LF      ;;TYPE A LINE FEED
017532 012603      MOV      (SP)+,R3      ;;RESTORE R3
017534 011646      MOV      (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
017536 016666 000004 000002      MOV      4(SP),2(SP)      ;;FIRST ASCII CHARACTER ON IT
017544 012766 017556 000004      MOV      0$TTYIN,4(SP)
017552 000002      RTI      ;;RETURN
017554 000      9$:      .BYTE      0      ;;STORAGE FOR ASCII CHAR. TO TYPE
017555 000      .BYTE      0      ;;TERMINATOR
017556      $TTYIN: .BLKB      8.      ;;RESERVE 8 BYTES FOR TTY INPUT
017566 136 103 015 $CNTLC: .ASCIZ /+C/<15><12>      ;;CONTROL "C"
017573 136 125 015 $CNTLU: .ASCIZ /+U/<15><12>      ;;CONTROL "U"
017600 136 107 015 $CNTLG: .ASCIZ /+G/<15><12>      ;;CONTROL "G"
017605 015 012 123 $MSWR: .ASCIZ <15><12>/SWR = /
017616 040 040 116 $MNEW: .ASCIZ / NEW = /
      .EVEN

```

2227

;;; CONTROL-C RESTART ENABLED

;KJL001

POWER DOWN AND UP ROUTINES

2228

.SBTTL POWER DOWN AND UP ROUTINES

;*****

;POWER DOWN ROUTINE

017630	012737	017774	000024	\$PWRDN: MOV	\$ILLUP,@PWRVEC	::SET FOR FAST UP
017636	012737	000340	000026	MOV	#340,@PWRVEC+2	::PRIO:7
017644	010046			MOV	R0,-(SP)	::PUSH R0 ON STACK
017646	010146			MOV	R1,-(SP)	::PUSH R1 ON STACK
017650	010246			MOV	R2,-(SP)	::PUSH R2 ON STACK
017652	010346			MOV	R3,-(SP)	::PUSH R3 ON STACK
017654	010446			MOV	R4,-(SP)	::PUSH R4 ON STACK
017656	010546			MOV	R5,-(SP)	::PUSH R5 ON STACK
017660	017746	161254		MOV	@SWR,-(SP)	::PUSH @SWR ON STACK
017664	010637	020000		MOV	SP,\$SAVR6	::SAVE SP
017670	012737	017702	000024	MOV	\$PWRUP,@PWRVEC	::SET UP VECTOR
017676	000000			HALT		
017700	000776			BR	.-2	::HANG UP

;*****

;POWER UP ROUTINE

017702	012737	017774	000024	\$PWRUP: MOV	\$ILLUP,@PWRVEC	::SET FOR FAST DOWN
017710	013706	020000		MOV	\$SAVR6,SP	::GET SP
017714	005037	020000		CLR	\$SAVR6	::WAIT LOOP FOR THE TTY
017720	005237	020000		1\$: INC	\$SAVR6	::WAIT FOR THE INC
017724	001375			BNE	1\$	::OF WORD
017726	012677	161206		MOV	(SP)+,@SWR	::POP STACK INTO @SWR
017732	012605			MOV	(SP)+,R5	::POP STACK INTO R5
017734	012604			MOV	(SP)+,R4	::POP STACK INTO R4
017736	012603			MOV	(SP)+,R3	::POP STACK INTO R3
017740	012602			MOV	(SP)+,R2	::POP STACK INTO R2
017742	012601			MOV	(SP)+,R1	::POP STACK INTO R1
017744	012600			MOV	(SP)+,R0	::POP STACK INTO R0
017746	012737	017630	000024	MOV	\$PWRDN,@PWRVEC	::SET UP THE POWER DOWN VECTOR
017754	012737	000340	000026	MOV	#340,@PWRVEC+2	::PRIO:7
017762	104401			TYPE		::REPORT THE POWER FAILURE
017764	020002			\$PWRMG: .WORD	PWRMSG	::POWER FAIL MESSAGE POINTER
017766	012716			MOV	(PC)+,(SP)	::RESTART AT RESTR
017770	002616			\$PWRAD: .WORD	RESTR	::RESTART ADDRESS
017772	000002			RTI		
017774	000000			\$ILLUP: HALT		::THE POWER UP SEQUENCE WAS STARTED
017776	000776			BR	.-2	::BEFORE THE POWER DOWN WAS COMPLETE
020000	000000			\$SAVR6: 0		::PUT THE SP HERE

2229

2230	020002	PWRMSG: .ASCIZ	<15><12>/RESTARTED FROM POWER FAIL/
2231		.EVEN	

2232

2233

2575

2576

2577

2578

2579

2580

2581

2582

2583

2584

2585

2586

022366	010446		
022370	005037	022774	
022374	012737	000001	022776
022402	012704	022750	
022406	012737	172414	022732

.NLIST	MEB
.NLIST	CND

;

;*****

.SBTTL ROUTINE TO SIZE FOR DRV11 UNITS

;*****

;

DRVSIZ: MOV	R4,-(SP)	;SAVE R4
CLR	WORK2	;CLEAR THE 2ND WORK LOCATION
MOV	#1,WORK3	;SET # OF DEVICES TO 1
MOV	@WORK1,R4	;SET UP THE TABLE ADDRESS
MOV	#172414,RCSR	;SET ADDR

;RJL001

;RJL001

;RJL001

;RJL001

;RJL001

;RJL001

;RJL001

;RJL001

ROUTINE TO SIZE FOR DRV11 UNITS

;RJL001

```

2587 022414 012737 022534 000004      MOV    #3$,TRAPT4      ;SET VECTOR      ;RJL001
2588 022422 104401 023004              TYPE    ,AUTOS        ;TYPE THE AUTO SIZER MESSAGE ;RJL001
2589 022426 005777 000300              TST     @RCSR        ;GOOD ADDRESS?      ;RJL001
2590 022432 013737 022732 023000 1$:   MOV     RCSR,WORK4    ;GET THE ADDRESS     ;RJL001
2591 022440 162737 000004 023000      SUB     #4,WORK4      ;MAKE IT THE BASE ADDRESS ;RJL001
2592 022446 104401 023130              TYPE    ,DEVADR      ;TYPE THE ADDRESS MESSAGE ;RJL001
2593 022452 013746 023000              MOV     WORK4,-(SP)    ;PUT THE NUMBER ON THE STACK ;RJL001
2594 022456 104403              TYPOS    ;TYPE THE OCTAL NUMBER ;RJL001
2595 022460      006              .BYTE    6          ;PRINT 6 NUMBERS      ;RJL001
2596 022461      000              .BYTE    0          ;SUPPRESS LEADING 0's  ;RJL001
2597 022462 005237 022774              INC     WORK2        ;INCREMENT THE # OF DEVICES ;RJL001
2598 022466 022737 000001 022774      CMP     #1,WORK2      ;1ST ONE?            ;RJL001
2599 022474 001402              BEQ     102$      ;BR IF YES           ;RJL001
2600 022476 006337 022776 100$:      ASL     WORK3        ;MAKE DEVICE ENABLE BIT ;RJL001
2601 022502 053737 022776 001252 102$:  BIS     WORK3,$DEVM    ;NOW SET THE DEVICF MAP  ;RJL001
2602 022510 004737 022564              JSR     PC,GETVEC     ;GET THE VECTOR       ;RJL001
2603 022514 062737 000010 022732 2$:   ADD     #10,RCSR      ;INC ADDR            ;RJL001
2604 022522 022737 172524 022732      CMP     #172524,RCSR   ;LAST ONE DONE?      ;RJL001
2605 022530 001410              BEQ     DRDONE        ;YES--EXIT           ;RJL001
2606 022532 000735              BR      1$          ;TRY NEXT ONE        ;RJL001
2607 022534 062706 000004 3$:         ADD     #4,SP          ;POP STACK           ;RJL001
2608 022540 005737 022774              TST     WORK2        ;ARE THERE ANY UNITS?  ;RJL001
2609 022544 001002              BNE     DRDONE        ;BR IF YES           ;RJL001
2610 022546 104401 023171              TYPE    ,NODEV      ;TYPE THE "No devices found" ;RJL001
2611 022552 012604  DRDONE:  MOV     (SP)+,R4      ;RESTORE R4          ;RJL001
2612 022554 012737 177777 022744      MOV     #-1,$SIZE    ;SET THE SIZE FLAG    ;RJL001
2613 022562 000207              RTS     PC          ;EXIT                ;RJL001
2614                                     ;                  ;RJL001
2615                                     ;*****          ;RJL001
2616                                     ;.SBTTL ROUTINE TO GET DRV11-WA VECTOR ADDRESS ;RJL001
2617                                     ;*****          ;RJL001
2618                                     ;                  ;RJL001
2619 022564 010637 022734  GETVEC:  MOV     SP,SAVE$P      ;SAVE SP             ;RJL001
2620 022570 012700 000112      MOV     #112,R0      ;SET UP VECTOR AREA   ;RJL001
2621 022574 010060 177776 1$:         MOV     R0,-2(R0)    ;MOV VEC+2 TO VEC     ;RJL001
2622 022600 012720 000003      MOV     #3,(R0)+    ;MOV BPT TO VEC+2     ;RJL001
2623 022604 005720              TST     (R0)+        ;INC R0 BY 2          ;RJL001
2624 022606 022700 000776      CMP     #776,R0      ;LAST VECTOR?        ;RJL001
2625 022612 100370              BPL     1$          ;NO                   ;RJL001
2626 022614 106437 000000      MTPS    PRO        ;ALLOW INTERRUPTS     ;RJL002
2627 022620 012737 022700 000014      MOV     #FIGV,14     ;SET UP BPT VECTOR    ;RJL001
2628 022626 012737 000340 000016      MOV     #340,16      ;SET BPT PSW          ;RJL001
2629 022634 012777 010777 000070      MOV     #10777,@RCSR ;FORCE INTR           ;RJL001
2630                                     ;*****          ;RJL001
2631                                     ;.SBTTL KILL SOME TIME WHILE WAITING FOR INTERRUPT ;RJL001
2632                                     ;*****          ;RJL001
2633 022642 012701 000002  VECTIM:  MOV     #2,R1        ;SET COUNTER          ;RJL001
2634 022646 005000              CLR     R0          ;CLEAR R0             ;RJL001
2635 022650 005300 1$:         DEC     R0          ;DEC R0               ;RJL001
2636 022652 001376              BNE     1$          ;R0=0? BR IF NOT     ;RJL001
2637 022654 005301 2$:         DEC     R1          ;DEC COUNTER          ;RJL001
2638 022656 001374              BNE     1$          ;BR IF NOT 0         ;RJL001
2639 022660 042777 010777 000044      BIC     #10777,@RCSR  ;CLR INTR ENABL       ;RJL001
2640 022666 013706 022734      MOV     SAVESP,SP    ;RESTORE STACK        ;RJL001
2641 022672 104401 023076              TYPE    ,NOINTR     ;TYPE DEVICE DIDN'T INTERRUPT ;RJL001
2642 022676 000207 4$:         RTS     PC          ;EXIT                ;RJL001
2643 022700 162716 000004  FIGV:    SUB     #4,(SP)      ;FIGURE OUT THE VECTOR ;RJL001

```


KILL SOME TIME WHILE WAITING FOR INTERRUPT

;RJL001

```

2644 022704 011624      MOV      (SP),(R4)+      ;SAVE THE VALUE      ;RJL001
2645 022706 104401 023156  TYPE      ,VECADR      ;TYPE THE "Vector="      ;RJL001
2646 022712 104403      TYP0S      ;TYPE THE VECTOR ADDRESS      ;RJL001
2647 022714      003      .BYTE      3      ; 3 NUMBERS      ;RJL001
2648 022715      000      .BYTE      0      ; SUPPRESS LEADING 0's      ;RJL001
2649 022716 013706 022734  MOV      SAVESP,SP      ;RESTORE STACK      ;RJL001
2650 022722 042777 010777 000002  BIC      #10777,@RCSR      ;CLR INTR ENABL      ;RJL001
2651 022730 000207      RTS      PC      ;EXIT      ;RJL001
2652      ;
2653 022732 000000      ;RCSR: .WORD      0      ;RJL001
2654 022734 000000      SAVESP: .WORD      0      ;RJL001
2655 022736 000000      SAV176: .WORD      0      ;RJL001
2656 022740 000000      SAV200: .WORD      0      ;LOCATION TO STORE CONTENTS OF LOC 176      ;RJL001
2657 022742 000000      SAV202: .WORD      0      ;LOCATION TO STORE CONTENTS OF LOC 200      ;RJL001
2658 022744 000000      $SIZE: .WORD      0      ;LOCATION TO STORE CONTENTS OF LOC 202      ;RJL001
2659 022746 022750      VECTOR: WORK1      ;FLAG LOCATION      ;RJL001
2660 022750 000000      WORK1: .WORD      0      ;TABLE POINTER LOCATION      ;RJL001
2661 022752 000000      .WORD      0      ;LOCATIONS WHERE THE FOUND VECTORS GO      ;RJL001
2662 022754 000000      .WORD      0      ;RJL001
2663 022756 000000      .WORD      0      ;RJL001
2664 022760 000000      .WORD      0      ;RJL001
2665 022762 000000      .WORD      0      ;RJL001
2666 022764 000000      .WORD      0      ;RJL001
2667 022766 000000      .WORD      0      ;RJL001
2668 022770 000000      .WORD      0      ;RJL001
2669 022772 000000      .WORD      0      ;RJL001
2670 022774 000000      WORK2: .WORD      0      ;RJL001
2671 022776 000000      WORK3: .WORD      0      ;RJL001
2672 023000 000000      WORK4: .WORD      0      ;RJL001
2673 023002 000000      $FLAG: .WORD      0      ;RJL001
2674      .NLIST      BIN      ;RJL001
2675
2676 023004 AUTOS: .ASCII <15><12>/Auto Size found DRV11-WA'S/      ;RJL001
2677 023040      .ASCIIZ / at the following locations/<15><12>      ;RJL001
2678 023076 NOINTR: .ASCIIZ / Device didn't interrupt./      ;RJL001
2679 023130 DEVADR: .ASCIIZ <15><12>/DRV11-WA Address = /      ;RJL001
2680 023156 VECADR: .ASCIIZ / Vector = /      ;RJL001
2681 023171 NODEV: .ASCIIZ <15><12>/ No devices found./      ;RJL001
2682      .EVEN      ;RJL001
2683      .LIST      BIN      ;RJL001
2684      ;      ;KJL001
2685      ;*****
2686      .SBTTL      TIMING ROUTINE TO LOOP AFTER RESET
2687      ; THIS LOOP IS TO WASTE TIME WHILE THE RESET IS TAKING PLACE      ;KJL001
2688      ; OTHERWISE, A ' ' WILL BE DISPLAYED ON THE CONSOLE TERMINAL      ;KJL001
2689      ;*****
2690
2691 023216 005000      WSTTIM: CLR      R0      ;CLEAR DELAY COUNTER      ;KJL001
2692 023220 00530C      1$: DEC      R0      ;DECREMENT DELAY COUNTER      ;KJL001
2693 023222 001376      BNE      1$      ;BRANCH IF NOT ZERO      ;KJL001
2694 023224 000207      RTS      PC      ;RETURN      ;KJL001
2695
2696      ;*****
2697      .SBTTL      RESTORE DEFAULT REGISTER ADDRESSES ROUTINE      ;RJL001
2698      ;*****
2699
2700 023226 012700 001712      SETREG: MOV      #DRVWCR,R0      ;SET UP REG ADRS POINTERS

```


RESTORE DEFAULT REGISTER ADDRESSES ROUTINE

;RJL001

```

2701 023232 013701 001250      MOV    $BASE,R1      ;SET BASE ADRS
2702 023236 010120      1$:  MOV    R1,(R0)+      ;LOAD EM
2703 023240 062701 000002      ADD    #2,R1      ;
2704 023244 022700 001722      CMP    #DRVDBR+2,R0  ;ALL DONE?
2705 023250 001372      BNE    1$      ;BR IF NOT
2706 023252 013737 001714 001722  MOV    DRVBAR,DRVB AE  ;SET UP BAE REG ADDRESS      ;KJL001
2707 023260 012700 001724      MOV    #DRVCT0,R0  ;SET UP DRV11WA VECTOR ADRS POINTER
2708 023264 013701 001244      MOV    $VECT1,R1  ;GET BASE VECTOR ADRS
2709 023270 042701 170000      BIC    #170000,R1  ;CLR OUT PRIORITY BITS
2710 023274 010120      2$:  MOV    R1,(R0)+      ;
2711 023276 062701 000002      ADD    #2,R1      ;POINT TO NEXT
2712 023302 022700 001730      CMP    #DRVCT2+2,R0  ;ALL DONE?
2713 023306 001372      BNE    2$      ;BR IF NOT
2714 023310 000207      RTS    PC      ;RETURN
2715
2716

```

.SBTTL TRAP DECODER

```

;*****
;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.

```

```

023312 010046      $TRAP:  MOV    R0,-(SP)      ;;SAVE R0
023314 016600 000002      MOV    2(SP),R0      ;;GET TRAP ADDRESS
023320 005740      TST    -(R0)      ;;BACKUP BY 2
023322 111000      MOVB    (R0),R0      ;;GET RIGHT BYTE OF TRAP
023324 006300      ASL    R0      ;;POSITION FOR INDEXING
023326 016000 023346      MOV    $TRPAD(R0),R0  ;;INDEX TO TABLE
023332 000200      RTS    R0      ;;GO TO ROUTINE

;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
023334 011646      $TRAP2: MOV    (SP),-(SP)      ;;MOVE THE PC DOWN
023336 016666 000004 000002      MOV    4(SP),2(SP)  ;;MOVE THE PSW DOWN
023344 000002      RTI      ;;RESTORE THE PSW

```

.SBTTL TRAP TABLE

```

;THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
;BY THE "TRAP" INSTRUCTION.

```

ROUTINE

```

023346 023334      $TRPAD: .WORD  $TRAP2
023350 014226      $TYPE    ;;CALL=TYPE      TRAP+1(104401) TTY TYPEOUT ROUTINE
023352 015002      $TYPOC   ;;CALL=TYPOC     TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
023354 014756      $TYPOS   ;;CALL=TYPOS     TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
023356 015016      $TYPON   ;;CALL=TYPON     TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
023360 015204      $TYPDS   ;;CALL=TYPDS     TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
023362 017036      $GTSWR   ;;CALL=GTSWR     TRAP+6(104406) GET SOFT-SWR SETTING
023364 016710      $CKSWR   ;;CALL=CKSWR     TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR
023366 017310      $RDCHR   ;;CALL=RDCHR     TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
023370 017430      $RDLIN   ;;CALL=RDLIN     TRAP+11(104411) TTY TYPEIN STRING ROUTINE

```

;*****

.SBTTL ASCII MESSAGES

;*****

2717

2718

2719

2720

2721

.NLIST BIN

```

2722 023372 EM1:    .ASCIZ  /REGISTER TIMEOUT ERROR/
2723 023421 EM2:    .ASCIZ  "REGISTER READ/WRITE ERROR"
2724 023453 EM3:    .ASCIZ  /BUS RESET ERROR/
2725 023473 EM4:    .ASCIZ  /FNCT BITS FAILED TO SET STATUS BITS/
2726 023537 EM5:    .ASCIZ  /READY INTERRUPT FAILURE/

```

ASCII MESSAGES

```

2727 023567 EM6: .ASCIZ /READY CLR OR SET ERROR/
2728 023616 EM7: .ASCIZ /STATUS ERROR ON XFER/
2729 023643 EM10: .ASCIZ /WORD COUNT ERROR ON XFER/
2730 023674 EM11: .ASCIZ /BUFFER ADRS ERROR ON XFER/
2731 023726 EM12: .ASCIZ /DATA ERROR FROM MEMORY/
2732 023755 EM13: .ASCIZ /DATA ERROR TO MEMORY/
2733 024002 EM14: .ASCIZ /SINGLE CYCLE OFF DID NOT LOCK OUT CPU/
2734 024050 EM15: .ASCIZ /SINGLE CYCLE ON LOCKED OUT CPU/
2735 024107 WARN: .ASCII <15><12>/Please disable "REV11" memory refresh option/
2736 024165 .ASCIZ <15><12>/and enable processor memory refresh./<15><12>
2737 024236 EM16: .ASCIZ /NEX LOGIC ERROR/
2738 024256 EM17: .ASCIZ /CYCLE FAILED TO CLOCK DBR (IN)/
2739 024315 EM20: .ASCIZ "DATA ERROR FROM I/O PAGE (XCSR)"
2740 024355 DH1: .ASCIZ /ERRPC TSTNUM BUSADR EXPCT RCVD/
2741 024417 DH2: .ASCIZ /ERRPC TSTNUM BUSADR ADRS EXPCT RCVD/
2742 024464 DH3: .ASCIZ /ERRPC TSTNUM SUSADR/
2743 .EVEN
2744 024510 DT1: $ERRPC,TSTNUM,$BDADR,$GDDAT,$BDDAT,0
2745 024524 DT2: $ERRPC,TSTNUM,$BDADR,$GDADR,$GDDAT,$BDDAT,0
2746 024542 DT3: $ERRPC,TSTNUM,$BDADR,0
2747
2748 ;;*****
2749 .SBTTL STANDARD PROGRAM MESSAGES
2750 ;;*****
2751 024552 MMAMES: .ASCIZ <15><12>/KT11 (Memory Management) available/
2752 024617 AVAL22: .ASCIZ <15><12>/22 Bit Addressing available/<15><12>
2753 024657 MEMMES: .ASCIZ <15><12>/Memory Map:/
2754 024675 FROM: .ASCIZ <15><12>/From /
2755 024705 TO: .ASCIZ / To /
2756 024712 INSUFF: .ASCII <15><12>/INSUFFICIENT MEMORY TO RUN DIAGNOSTIC/
2757 024761 .ASCIZ / ... FIRST 16K NOT ALL THERE!/
2758 025017 NOMEM: .ASCIZ <15><12>/NO MEMORY FOUND./
2759 025042 Q22CPU: .ASCIZ <15><12>\Does this CPU support 22-bit addressing? [Y/N] \
2760 025124 UUT: .ASCIZ <15><12>/ Testing Unit Number /
2761 .EVEN
2762 .LIST BIN
2763
2764 025154 PATCH: .BLKW 20. ;20 WORD PATCH AREA ;RJL001
2765 ;;*****
2766 ;'DBUF' IS THE WORKING AREA IN EACH 4K MEM FOR ALL
2767 ;NPR OPERATIONS - IT IS 200 WORDS LONG
2768 ;;*****
2769
2770 025224 000000 DBUF: 0 ;1ST ADRS OF DATA BUFFER
2771 000001 .END

```

Symbol table

ABASE = 172410	AVECT2= 000000	DISMAP 003500	EXAM08 021704	LOOKIE 020404
ACDW1 = 000000	BACKEX 021524	DISPLA 001142	EXAM09 021710	LOOK1 020672
ACDW2 = 000000	BIT0 = 000001	DISPRE 000174	EXAM1 020120	LOOK2 020036
ACPUOP= 000000	BIT00 = 000001	DMAP 001732	EXAM10 021747	LOOK3 021050
ADDW0 = 000000	BIT01 = 000002	DRDONE 022552	EXAM11 021754	LSIFLG 001702
ADDW1 = 000000	BIT02 = 000004	DRVBAE 001722	EXAM12 021762	LSIQ22 001704
ADDW10= 000000	BIT03 = 000010	DRVBAR 001714	EXAM2 020104	MASK4K= 017777
ADDW11= 000000	BIT04 = 000020	DRVCSR 001716	EXAM3 021774	MASK8K= 037777
ADDW12= 000000	BIT05 = 000040	DRVCTO 001724	EXBACK 020422	MEMMAP 001524
ADDW13= 000000	BIT06 = 000100	DRVCT2 001726	EXTP1 021500	MEMMES 024657
ADDW14= 000000	BIT07 = 000200	DRVDBR 001720	EXTRAP 021470	MFPT = 000007
ADDW15= 000000	BIT08 = 000400	DRVSIZ 022366	FIDDLE 021776	MMAMES 024552
ADDW2 = 000000	BIT09 = 001000	DRVWCR 001712	FIGV 022700	MMAVA 001710
ADDW3 = 000000	BIT1 = 000002	DSWR = 177570	FIRST2 020452	MMCK 002724
ADDW4 = 000000	BIT10 = 002000	DT1 024510	FLG30K 001676	MMINIT 012326
ADDW5 = 000000	BIT11 = 004000	DT2 024524	FROM 024675	MMVEC = 000250
ADDW6 = 000000	BIT12 = 010000	DT3 024542	FSTPAS 001706	MOV272 021152
ADDW7 = 000000	BIT13 = 020000	EDEL 020366	GETVEC 022564	MPAT 021070
ADDW8 = 000000	BIT14 = 040000	EINPUT 022102 G	GOST 020414	MPATT 021064
ADDW9 = 000000	BIT15 = 100000	ELTOU 022336 G	GTSWR = 104406	MSD 021134
ADEVCT= 000000	BIT2 = 000004	EMTVEC= 000030	HT = 000011	MTPS = 106427
ADEVM = 000001	BIT3 = 000010	EM1 023372	INCE 020340	MYCC 022002
AENV = 000000	BIT4 = 000020	EM10 023643	INSUFF 024712	NODEV 023171
AENVM = 000000	BIT5 = 000040	EM11 023674	IOTVEC= 000020	NOINTR 023076
AFATAL= 000000	BIT6 = 000100	EM12 023726	JMPCHK 021432	NOMEM 025017
AMADR1= 000000	BIT7 = 000200	EM13 023755	JMPCK 020410	NONKT 003142
AMADR2= 000000	BIT8 = 000400	EM14 024002	JPCX 021244	NORUB 022026 G
AMADR3= 000000	BIT9 = 001000	EM15 024050	JPEX1 021240	NXDEV 012042
AMADR4= 000000	BNKTAB 016424	EM16 024236	KBSC = 177560	NXDEV1 012044
AMAMS1= 000000	BPTVEC= 000014	EM17 024256	KBDB = 177562	OPRINT 022322 G
AMAMS2= 000000	CHSV 021336	EM2 023421	KBDSRV 013700	PATCH 025154
AMAMS3= 000000	CHSV1 021404	EM20 024315	KBDSR1 013710	PATT 022010
AMAMS4= 000000	CHSV2 021560	EM3 023453	KIPAR0= 172340	PIRQ = 177772
AMSGAD= 000000	CKDAT 013272	EM4 023473	KIPAR1= 172342	PIRQVE= 000240
AMSGLG= 000000	CKDAT1 013344	EM5 023537	KIPAR2= 172344	PRCS = 177564
AMSGTY= 000000	CKDAT2 013452	EM6 023567	KIPAR3= 172346	PRDB = 177566
AMTYP1= 000000	CKSTAT 012520	EM7 023616	KIPAR4= 172350	PRO = 000000
AMTYP2= 000000	CKSTA1 012704	ENDBKT 016426	KIPAR5= 172352	PR1 = 000040
AMTYP3= 000000	CKSWR = 104407	ENDL 020470	KIPAR6= 172354	PR2 = 000100
AMTYP4= 000000	CONL 020474	ENDLOK 021320	KIPAR7= 172356	PR3 = 000140
APASS = 000000	CONLOK 021350	EPRINT 022216 G	KIPDR0= 172300	PR4 = 000200
APRIOR= 000000	CORCMP 020544	ERROR = 104000	KIPDR1= 172302	PR5 = 000240
APTCSU= 000040	CORCP1 020556	ERRVEC= 000004	KIPDR2= 172304	PR6 = 000300
APTENV= 000001	CORSCN 020504	ESC 022014 G	KIPDR3= 172306	PR7 = 000340
APTSIZ= 000200	CORSZ 001734	EWORK 021764	KIPDR4= 172310	PS = 177776
APTSP0= 000100	CORSZR 002514	EWORK0 021766	KIPDR5= 172312	PSW = 177776
ASKQ22 013560	CR = 000015	EWORK1 021770	KIPDR6= 172314	PWRMSG 020002
ASLE 020330	CRLF = 000200	EWORK2 021772	KIPDR7= 172316	PWRVEC= 000024
ASRAD 021306	CSTART 002310	EXAM 020050 G	KTSIZ 003264	Q22CPU 025042
ASR3 022064	CTRLE 022166 G	EXAMIN 020134	LDBUF 013140	RCSR 022732
ASWREG= 000000	DBUF 025224	EXAM01 021626	LDBUF1 013216	RDCHR = 104410
ATESTN= 000000	DBUFP 001736	EXAM02 021634	LF = 000012	RDLIN = 104411
AUNIT = 000000	DDISP = 177570	EXAM03 021642	LMAD 001522	RELOC 022012
AUSWR = 000000	DEVADR 023130	EXAM04 021650	LMEMHI 016422	RESTRT 002616
AUTOS 023004	DH1 024355	EXAM05 021656	LMEMLO 016420	RESVEC= 000010
AVAL22 024617	DH2 024417	EXAM06 021663	LOOK 020772	RETEX 020426
AVECT1= 000124	DH3 024464	EXAM07 021666	LOOKAD 022000	REXTS 021624

RSTVEC	012474	TKVEC	= 000060	\$APTHD	001000	\$ERRTB	001320	\$PWRDN	017630
RTEX1	020450	TO	024705	\$ATYC	014534	\$ERRTY	015636	\$PWRMG	017764
RW	= 000006	TPVEC	= 000064	\$ATY1	014510	\$ERTTL	001112	\$PWRUP	017702
R6	=*000006	TRAPT4	= 000004	\$ATY3	014516	\$ESCAP	001162	\$QUES	001170
R7	=*000007	TRAPVE	= 000034	\$ATY4	014526	\$ETABL	001214	\$RDCHR	017310
SAVESP	022734	TRTVEC	= 000014	\$AUTOB	001134	\$ETEND	001320	\$RDLIN	017430
SAVET4	022006	TSTMAP	001564	\$BASE	001250	\$FATAL	001176	\$RDSZ	= 000010
SAVE2	022004	TSTNUM	001730	\$BDADR	001122	\$FFLG	014754	\$RTNAD	012300
SAVTST	001624	TST1	003612	\$BDDAT	001126	\$FILLC	001156	\$SAVR6	020000
SAV176	022736	TST10	004752	\$BELL	001164	\$FILLS	001155	\$SCOPE	016430
SAV200	022740	TST11	005162	\$CDW1	001254	\$FLAG	023002	\$SETUP	= 000117
SAV202	022742	TST12	005242	\$CDW2	001256	\$GDADR	001120	\$SIZE	022744
SCNTRP	021514	TST13	005330	\$CHARC	014504	\$GDDAT	001124	\$STUP	= 177777
SCOPE	= 000004	TST14	005432	\$CKSWR	016710	\$GET42	012256	\$SVLAD	016636
SETREG	023226	TST15	005576	\$CMTAG	001100	\$GTSWR	017036	\$SVPC	= 000106
SETVEC	012454	TST16	005704	\$CM3	= 000000	\$HD	= 000003	\$SWR	= 167400
SRO	= 177572	TST17	006022	\$CNTLC	017566	\$HIBTS	001000	\$SWREG	001216
SR1	= 177574	TST2	003726	\$CNTLG	017600	\$ICNT	001104	\$SWRMK	= 000000
SR2	= 177576	TST20	006142	\$CNTLU	017573	\$ILLUP	017774	\$TBAE	001674
SR3	= 172516	TST21	006376	\$CPUOP	001222	\$INTAG	001135	\$TBAR	001672
STACK	= 001100	TST22	006624	\$CRLF	001171	\$ITEMB	001114	\$TESTN	001200
START	001740	TST23	007026	\$DBLK	015420	\$LF	001172	\$TIMES	001160
START1	002344	TST24	007176	\$DDW0	001260	\$LFLG	014753	\$TKB	001146
STKLMT	= 177774	TST25	007340	\$DDW1	001262	\$LPADR	001106	\$TKS	001144
STPCOR	020642	TST26	007546	\$DDW10	001304	\$LPERR	001110	\$TMAP	001666
SWR	001140	TST27	010014	\$DDW11	001306	\$MADR1	001226	\$TN	= 000034
SWREG	000176	TST3	004030	\$DDW12	001310	\$MADR2	001232	\$TPB	001152
SW0	= 000001	TST30	010430	\$DDW13	001312	\$MADR3	001236	\$TPFLG	001157
SW00	= 000001	TST31	010632	\$DDW14	001314	\$MADR4	001242	\$TPS	001150
SW01	= 000002	TST32	011070	\$DDW15	001316	\$MAIL	001174	\$TRAP	023312
SW02	= 000004	TST33	011334	\$DDW2	001264	\$MAMS1	001224	\$TRAP2	023334
SW03	= 000010	TST4	004162	\$DDW3	001266	\$MAMS2	001230	\$TRP	= 000012
SW04	= 000020	TST5	004364	\$DDW4	001270	\$MAMS3	001234	\$TRPAD	023346
SW05	= 000040	TST6	004466	\$DDW5	001272	\$MAMS4	001240	\$TSTM	001004
SW06	= 000100	TST7	004552	\$DDW6	001274	\$MBADR	001002	\$TSTNM	001102
SW07	= 000200	TYPDS	= 104405	\$DDW7	001276	\$MFLG	014752	\$TTST	001670
SW08	= 000400	TYPE	= 104401	\$DDW8	001300	\$MMAP	001664	\$TTYIN	017556
SW09	= 001000	TYPMAP	015772	\$DDW9	001302	\$MNEW	017616	\$TYPDS	015204
SW1	= 000002	TYP0C	= 104402	\$DEVCT	001204	\$MSGAD	001210	\$TYPE	014226
SW10	= 002000	TYP0N	= 104404	\$DEVH	001252	\$MSGLG	001212	\$TYPEC	014440
SW11	= 004000	TYP0S	= 104403	\$DOAGN	012276	\$MSGTY	001174	\$TYPEX	014506
SW12	= 010000	UP	= 000000	\$DTBL	015410	\$MSWR	017605	\$TYP0C	015002

Symbol table

. ABS.	025226	000	(RW,I,GBL,ABS,OVR)
	000000	001	(RW,I,LCL,REL,CON)

Errors detected: 0

*** Assembler statistics

Work file reads: 364

Work file writes: 282

Size of work file: 49032 Words (192 Pages)

Size of core pool: 19402 Words (74 Pages)

Operating system: RSX-11M/PLUS (Under VAX/VMS)

Elapsed time: 00:02:21.60

ZDRVB0.BIN,ZDRVB0.LIS/CR/-SP=SYSMAC/ML,ZDRVB0.MAC

SYMBOL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES
ABASE	= 172410	#8-335 9-354 9-354
ACDW1	= 000000	9-354 9-354
ACDW2	= 000000	9-354 9-354
ACPUOP	= 000000	9-354 9-354
ADDW0	= 000000	9-354 9-354
ADDW1	= 000000	9-354 9-354
ADDW10	= 000000	9-354 9-354
ADDW11	= 000000	9-354 9-354
ADDW12	= 000000	9-354 9-354
ADDW13	= 000000	9-354 9-354
ADDW14	= 000000	9-354 9-354
ADDW15	= 000000	9-354 9-354
ADDW2	= 000000	9-354 9-354
ADDW3	= 000000	9-354 9-354
ADDW4	= 000000	9-354 9-354
ADDW5	= 000000	9-354 9-354
ACDW6	= 000000	9-354 9-354
ADDW7	= 000000	9-354 9-354
ADDW8	= 000000	9-354 9-354
ADDW9	= 000000	9-354 9-354
ADEVCT	= 000000	9-354 9-354
ADEVM	= 000001	#8-337 9-354 9-354
AENV	= 000000	9-354 9-354
AENVH	= 000000	9-354 9-354
AFATAL	= 000000	9-354 9-354
AMADR1	= 000000	9-354 9-354
AMADR2	= 000000	9-354 9-354
AMADR3	= 000000	9-354 9-354
AMADR4	= 000000	9-354 9-354
AMAMS1	= 000000	9-354 9-354
AMAMS2	= 000000	9-354 9-354
AMAMS3	= 000000	9-354 9-354
AMAMS4	= 000000	9-354 9-354
AMSGAD	= 000000	9-354 9-354
AMSGLG	= 000000	9-354 9-354
AMSGTY	= 000000	9-354 9-354
AMTYP1	= 000000	9-354 9-354
AMTYP2	= 000000	9-354 9-354
AMTYP3	= 000000	9-354 9-354
AMTYP4	= 000000	9-354 9-354
APASS	= 000000	9-354 9-354
APRIOR	= 000000	9-354
APTCU	= 000040	10-2104 #10-2105
APTENV	= 000001	10-2104 10-2105 #10-2105 10-2108
APTSIZ	= 000200	10-555 #10-2105
APTSP0	= 000100	10-2104 10-2105 #10-2105
ASKQ22	013560	10-608 #10-1991 10-2007
ASLE	020330	#10-2286 10-2288
ASRAD	021306	#10-2415 10-2417
ASR3	022064	#10-2516 10-2518
ASWREG	= 000000	9-354 9-354
ATESTN	= 000000	9-354 9-354

SYMBOL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES
AUNIT	= 000000	9-354 9-354
AUSWR	= 000000	9-354 9-354
AUTOS	023004	10-2588 #10-2676
AVAL22	024617	10-690 #10-2752
AVECT1	= 000124	#8-336 9-354 9-354
AVECT2	= 000000	9-354 9-354
BACKEX	021524	10-2302 #10-2447
BIT0	= 000001	#8-334 10-651 10-758 10-1619 10-1624 10-1681 10-2154
BIT00	= 000001	#8-334 8-334 10 887 10-892 10 915 10-917 10-984 10-1465 10-1799
BIT01	= 000002	10-1836 #8-334 8-334
BIT02	= 000004	#8-334 8-334
BIT03	= 000010	#8-334 8-334
BIT04	= 000020	#8-334 8-334
BIT05	= 000040	#8-334 8-334
BIT06	= 000100	#8-334 8-334
BIT07	= 000200	#8-334 8-334
BIT08	= 000400	#8-334 8-334 10-2224
BIT09	= 001000	#8-334 8-334 10 2108 10-2224
BIT1	= 000002	#8-334 10-1634
BIT10	= 002000	#8-334 10-2108
BIT11	= 004000	#8-334 10-2224
BIT12	= 010000	#8-334
BIT13	= 020000	#8-334 10-1415 10-2108
BIT14	= 040000	#8-334 10-2224
BIT15	= 100000	#8-334 10-687
BIT2	= 000004	#8-334
BIT3	= 000010	#8-334
BIT4	= 000020	#8-334 10-663 10-734 10 1623
BIT5	= 000040	#8-334
BIT6	= 000100	#8-334 10-1088
BIT7	= 000200	#8-334
BIT8	= 000400	#8-334
BIT9	= 001000	#8-334
BNKTAB	016424	#10-2135 10-2146 #10-2219
BPTVEC	= 000014	#8-334
CHSV	021336	10-2420 #10-2422
CHSV1	021404	10-2428 #10-2430
CHSV2	021560	10-2451 #10-2453
CKDAT	013272	10-1313 #10-1905
CKDAT1	013344	10-1421 10-1522 10-1564 #10-1926
CKDAT2	013452	10-1669 #10-1959
CKSTAT	012520	10-1182 10-1226 10-1268 10-1303 10-1403 10-1512 10-1554 10-1596 #10-1780
CKSTA1	012704	10-1659 #10-1814
CKSWR	= 104407	10-2108 10-2108 10-2108 10-2115 10-2224 #10-2716
CONL	020474	10-2262 #10-2312
CONLOK	021350	10-2312 #10-2424
CORCMP	020544	#10-2327 10-2332 10-2410
CORCP1	020556	10-2328 #10-2331
CORSCN	020504	10-2274 #10-2320
CORSZ	001734	#10-549 #10-593 10-1535
CORSZR	002514	#10-588

SYMBOL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES
CR	= 000015	#8-334 10-2104 10-2104
CRLF	= 000200	#8-334 10-583 10-583
CSTART	002310	#10-571 10-2046 10-2073
CTRLE	022166	10-2246 10-2322 #10-2535
DBUF	025224	10-550 10-595 10-1170
		10-1277 10-1292 10-1326
		10-1857 10-1862 10-1867
DBUFP	001736	#10-550 *10-595 *10-1532
		*10-1627 *10-1675 10-1686
DDISP	= 177570	#8-334 9-354 10-555
DEVADR	023130	10-2592 #10-2679
DM1	024355	10-358 10-364 10-370
		10-436 10-442 #10-2740
DM2	024417	10-412 10-418 #10-2741
DM3	024464	10-424 10-430 #10-2742
DISMAP	003500	10-736 #10-811
DISPLA	001142	#9-354 *10-555 *10-555
DISPRE	000174	#8-348 10-555
DMAP	001732	#10-548 *10-597 *10-598
DRDCNE	022552	10-2605 10-2609 #10-2611
DRVBAE	001722	#10-538 10-913 10-924
		*10-1707 10-1843 *10-2706
DRVBAR	001714	#10-535 10-879 10-884
		10-986 10-1126 10-1129
		10-1326 10-1352 10-1391
		10-1642 *10-1704 10-1798
DRVCSR	001716	#10-536 10-1005 10-1013
		10-1063 10-1067 10-1068
		10-1110 10-1111 10-1115
		10-1172 10-1173 10-1175
		10-1261 10-1263 10-1264
		10-1345 10-1354 10-1355
		10-1398 10-1399 10-1439
		10-1474 10-1475 10-1479
		10-1508 10-1542 10-1543
		10-1591 10-1592 10-1646
		10-1780 10-1781 10-1785
DRVCTO	001724	#10-542 10-1082 10-1087
DRVCT2	001726	#10-543 *10-1710 *10-1711
DRVDBR	001720	#10-537 10-937 10-941
		10-1147 10-1149 10-1150
		10-1282 10-1293 10-1581
		10-2704
DRVSIZ	022366	10-566 #10-2582
DRVWCR	001712	#10-534 10-615 10-844
		10-972 10-979 10-981
		10-1325 10-1351 10-1390
		10-1496 10-1498 10-1538
		10-1790 10-1797 10-1823
DSWR	= 177570	#8-334 9-354 10-555
DT1	024510	10-359 10-365 10-371
		10-437 10-443 #10-2744

SYMBOL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES
EXAMIN	020134	#10-2254 10-2293 10-2297 10-2310 10-2350 10-2414
EXAM01	021626	10-2248 #10-2464
EXAM02	021634	10-2399 #10-2465
EXAM03	021642	10-2377 #10-2466
EXAM04	021650	10-2304 #10-2467
EXAM05	021656	10-2347 10-2447 #10-2468
EXAM06	021663	10-2324 10-2408 10-2436 #10-2469
EXAM07	021666	10-2412 #10-2470
EXAM08	021704	10-2424 #10-2472
EXAM09	021710	10-2440 #10-2473
EXAM1	020120	#10-2251 10-2406
EXAM10	021747	#10-2474 10-2500
EXAM11	021754	10-2395 #10-2475
EXAM12	021762	10-2360 #10-2476
EXAM2	020104	#10-2248 10-2423 10-2444
EXAM3	021774	*10-2341 *10-2342 10-2343 #10-2485
EXBACK	020422	10-2264 #10-2302
EXTP1	021500	#10-2442 10-2446
EXTRAP	021470	10-2244 10-2349 #10-2440 10-2445
FIDDLE	021776	*10-2326 10-2331 #10-2486
FIGV	022700	10-2607 #10-2643
FIRST2	020452	10-2204 #10-2308
FLG30K	001676	#10-523 *10-728 10-2169
FROM	024675	10-2181 #10-2754
FSTPAS	001706	#10-527 10-557 *10-574 10-605 *10-609
GETVEC	022564	10-2602 #10-2619
GNS	= *****	8-348 8-348 10-583 10-2716 10-2716 10-2716 10-2716 10-2716 10-2716 10-2716
GOST	020414	10-2716 10-2716 10-2716 10-2716 10-2716 10-2716 10-2716 10-2716
GTSWR	= 104406	10-2266 10-2268 #10-2300
HT	= 000011	10-578 10-583 #10-2716
INCE	020340	#8-334 10-2104 10-2104
INSUFF	024712	#10-2289 10-832 #10-2756
IOTVEC	= 000020	#8-334 *10-555 *10-555
JMPCHK	021432	10-2299 #10-2434
JMPCK	020410	10-2270 #10-2299
JPCX	021244	10-2402 #10-2407
JPEX1	021240	10-2404 #10-2406
KBCS	= 177560	10-2339 #10-2494 10-2521 10-2546 10-2553
KBDB	= 177562	10-2341 #10-2495 10-2523 10-2524 10-2548 10-2555
KBDSRV	013700	10-631 10-645 10-659 10-686 10-711 10-721 10-771 10-823 10-846
		10-863 10-882 10-909 10-940 10-955 10-970 10-1007 10-1034 10-1046
		10-1066 10-1079 10-1106 10-1124 10-1144 10-1168 10-1213 10-1253 10-1288
		10-1324 10-1350 10-1388 10-1430 10-1492 10-1533 10-1574 10-1639 10-1700
		10-1859 10-1885 10-1906 10-1930 10-1963 #10-2037
KBDSR1	013710	10-761 10-2038 #10-2040
KIPAR0	= 172340	#8-339 *10-1740
KIPAR1	= 172342	#8-339 *10-1741
KIPAR2	= 172344	#8-339 *10-662 *10-755 *10-785 *10-1635 *10-1690 *10-1742
KIPAR3	= 172346	#8-339 *10-756 10-777 10-781 *10-786 *10-1636 *10-1691 *10-1743
KIPAR4	= 172350	#8-339 *10-1744

SYMBOL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES
KIPAR5	= 172352	#8-339 *10-1745
KIPAR6	= 172354	#8-339 *10-1746
KIPAR7	= 172356	#8-339 *10-1747
KIPDR0	= 172300	#8-339 *10-1732
KIPDR1	= 172302	#8-339 *10-1733
KIPDR2	= 172304	#8-339 *10-1734
KIPDR3	= 172306	#8-339 *10-1735
KIPDR4	= 172310	#8-339 *10-1736
KIPDR5	= 172312	#8-339 *10-1737
KIPDR6	= 172314	#8-339 *10-1738
KIPDR7	= 172316	#8-339 *10-1739
KTSIZ	003264	10-674 10-692 #10-752
LDBUF	013140	10-1256 #10-1857
LDBUF1	013216	10-1385 10-1495 10-1537 10-1640 #10-1881
LF	= 000012	#8-334 10-2104 10-2104
LMAD	001522	#10-458
LMEMHI	016422	*10-2149 10-2180 *10-2187 10-2189 *10-2194 *10-2200 *10-2202 *10-2218
LMEMLO	016420	*10-2148 10-2180 *10-2186 *10-2188 10-2190 *10-2194 *10-2199 *10-2201 #10-2217
LOOK	020772	10-2241 10-2298 10-2340 10-2345 #10-2366 10-2433 10-2439 10-2456
LOOKAD	022000	*10-2250 10-2327 *10-2357 10-2359 *10-2373 10-2376 10-2434 #10-2487
LOOKIE	020404	10-2253 #10-2298
LOOK1	020672	#10-2351 10-2367
LOOK2	020036	#10-2240 10-2258
LOOK3	021050	#10-2375
_SIFLG	001702	#10-525 *10-635 *10-641
LSIQ22	001704	#10-526 10-1575 *10-2008 *10-2010
MASK4K	= 017777	#8-345
MASK8K	= 037777	#8-346 10-713 10-731 10-763
MEMMAP	001524	#10-459 10-629 10-707 10-753 10-797 10-814 10-821 10-827 10-1632
MEMMES	024657	10-815 #10-2753
MFPT	= 000007	#8-347
MMAMES	024552	10-654 #10-2751
MMAVA	001710	#10-528 *10-646 *10-651 *10-687 10-775 10-788 10-1619 10-1621 10-2166
MMCK	002724	10-617 #10-629
MMINIT	012326	10-650 #10-1731
MMVEC	= 000250	#8-339
MOV272	021152	#10-2393
MPAT	021070	#10-2379
MPATT	021064	10-2365 #10-2378 10-2388
MSD	021134	10-2387 #10-2389
MTPS	= 106427	#8-338
MYCC	022002	*10-2240 *10-2247 10-2252 *10-2320 *10-2338 *10-2346 10-2351 10-2361 *10-2364 10-2366 10-2368 *10-2370 *10-2372 10-2393 *10-2397 10-2401 10-2403 *10-2405 *10-2407 *10-2426 *10-2432 *10-2435 *10-2438 *10-2442 *10-2449 *10-2455 #10-2488
NODEV	023171	10-2610 #10-2681
NOINTR	023076	10-2641 #10-2678
NOMEM	025017	10-2143 #10-2758
NOMKT	003142	10-647 #10-705
NORUB	022026 G	10-2294 #10-2510
NXDEV	012042	10-1677 #10-1698
NXDEV1	012044	10-602 #10-1699 10-1714
OPRINT	022322 G	10-2514 10-2528 10-2536 10-2539 10-2545 #10-2561

SYMBOL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES
PATCH	025154	#10-2764
PATT	022010	*10-2378 *10-2379 *10-2380 10-2381 *10-2415 #10-2491
PIRQ	= 177772	#8-334
PIRQVE	= 000240	#8-334
PRCS	= 177564	#10-2496 10-2562
PRDB	= 177566	#10-2497 *10-2561
PRO	= 000000	#8-334 10 1085 10-1174 10-1219 10-1353 10-1389 10-1435 10 1499 10-1541
		10-1583 10-1645 10-2023 10-2626
PR1	= 000040	#8-334
PR2	= 000100	#8-334
PR3	= 000140	#8-334
PR4	= 000200	#8-334 10-610 10-1001 10-1080 10-1107 10-1757 10-1770
PR5	= 000240	#8-334
PR6	= 000300	#8-334
PR7	= 000340	#8-334
PS	= 177776	#8-334 8-334
PSW	= 177776	#8-334
PWRMSG	020002	10-2228 #10-2230
PWRVEC	= 000024	#8-334 *10-555 *10-555 *10 2228 *10-2228 *10-2228 *10-2228 *10-2228
Q22CPU	025042	10-1991 #10-2759
RCSR	022732	*10-2586 10-2589 10-2590 *10-2603 10-2604 10-2629 10-2639 10-2650 #10-2653
RDCHR	= 104410	10-2226 #10-2716
RDLIN	= 104411	10-1992 #10-2716
RELOC	022012	10-2354 10-2356 10-2363 10-2371 #10-2492
RESTR	002616	10-599 10-601 #10-603 10-1715 10-2228
RESVEC	= 000010	#8-334 *10-636
RETEX	020426	10-2272 #10-2303
REXTS	021624	10-2458 #10-2460
RSTVEC	012474	10-559 10-573 10-576 10-1030 10-1102 10-1205 10-1250 10-1284 10-1315
		10-1342 10-1375 10-1425 10-1489 10-1526 10-1570 10-1614 10-1676 #10-1767
RTEX1	020450	#10-2307
RW	= 000006	#8-341 10-1732 10-1733 10-1734 10-1735 10-1739
R6	= #000006	#8-334 *10-555 *10-555 10-555
R7	= #000007	#8-334
SAVESP	022734	*10-2619 10-2640 10-2649 #10-2654
SAVET4	022006	*10-2243 10-2306 #10-2490
SAVE2	022004	*10-2374 10-2421 10-2422 10-2429 10-2430 10-2452 10-2453 10-2459 #10-2489
SAVTST	001624	#10-497 10-820
SAV176	022736	*10-563 10-567 #10-2655
SAV200	022740	*10-564 10-568 #10-2656
SAV202	022742	*10-565 10-569 #10-2657
SCNTRP	021514	10-2323 #10-2445
SCOPE	= 000004	#8-334 10 857 10-876 10-900 10-934 10-953 10-968 10-999 10-1032
		10-1044 10-1060 10-1077 10-1104 10-1122 10-1142 10-1161 10-1206 10-1251
		10-1286 10-1317 10-1343 10-1377 10-1427 10-1490 10-1528 10-1572 10-1615
		10-1698 10-1718
SETREG	023226	10-556 10-575 10-579 #10-2700
SETVEC	012454	10-1002 10-1164 10-1209 10-1254 10-1289 10-1320 10-1346 10-1383 10-1431
		10-1493 10-1530 10-1577 10-1625 #10-1757
SR0	= 177572	#8-339 *10-648 *10-831 *10-1624 *10-1674 *10-1748
SR1	= 177574	#8-339
SR2	= 177576	#8-339

SYMBOL CROSS REFERENCE

CREF V02

SEQ 0072

SYMBOL	VALUE	REFERENCES
SR3	= 172516	#8-339 *10-663 *10-673 *10-162
STACK	= 001100	#8-334 10-555 10-580 10-611 10-706 10-752
START	001740	8-348 #10-554
START1	002344	10-570 #10-579 10-1718
STKLMT	= 177774	#8-334
STPCOR	020642	10-2344 #10-2346
SWR	001140	#9-354 10-555 *10-555 10-555 *10-555 *10-555 10-583 10-902 10-975
		10-988 10-1010 10-1016 10-1436 10-1584 10-1617 10-2054 10-2083 10-2108
		10-2108 10-2108 10-2108 10-2224 10-2224 10-2224 10-2224 10-2224 10-2226
		10-2226 10-2228 10-2228 10-2228
SWREG	000176	#8-348 10-555 10-583 10-2054 10-2060 10-2226 10-2226
SW0	000001	#8-334
SW00	000001	#8-334 8-334
SW0	000002	#8-334 8-334
SW0	000004	#8-334 8-334
SW0	= 000010	#8-334 8-334
SW0	= 000020	#8-334 8-334
SW05	= 000040	#8-334 8-334
SW06	= 000100	#8-334 8-334
SW07	= 000200	#8-334 8-334
SW08	= 000400	#8-334 8-334
SW09	= 001000	#8-334 8-334
SW1	= 000002	#8-334
SW10	= 002000	#8-334
SW11	= 004000	#8-334
SW12	= 010000	#8-334 10-902 10-975 10-988 10-1010 10-1016 10-1436 10-1584 10-1617
SW13	= 020000	#8-334
SW14	= 040000	#8-334
SW15	= 100000	#8-334
SW2	= 000004	#8-334
SW3	= 000010	#8-334
SW4	= 000020	#8-334
SW5	= 000040	#8-334
SW6	= 000100	#8-334
SW7	= 000200	#8-334
SW8	= 000400	#8-334
SW9	= 001000	#8-334
TBITVE	= 000014	#8-334
TEMP	001700	#10-524 *10-2150 10-2195 *10-2211
TIMER	013652	10-1260 10-1295 10-1328 #10-2022
TKVEC	= 000060	#8-334
TO	024705	10-2191 #10-2755
TPVEC	= 000064	#8-334
TRAPT4	= 000004	10-2243 *10-2244 *10-2306 *10-2323 *10-2349 *10-2445 #10-2498 *10-2587
TRAPVE	= 000034	#8-334 *10-555 *10-555
TRIVEC	= 000014	#8-334
TSTMAP	001564	#10-478 10-1628 10-1633
TSTNUM	001730	#10-547 *10-2113 10-2744 10-2745 10-2746
TST1	003612	10-812 10-830 #10-837
TST10	004752	10-995 #10-999
TST11	005162	#10-1032
TST12	005242	10-1041 #10-1044

SYMBOL CROSS REFERENCE

CREF V02

SEQ 0073

SYMBOL	VALUE	REFERENCES
TST13	005330	#10-1060
TST14	005432	#10-1077
TST15	005576	#10-1104
TST16	005704	10-1119 #10-1122
TST17	006022	#10-1142
TST2	003726	#10-857
TST20	006142	10-1158 #10-1161
TST21	006376	#10-1206
TST22	006624	#10-1251
TST23	007026	#10-1286
TST24	007176	#10-1317
TST25	007340	#10-1343
TST26	007546	#10-1377
TST27	010014	#10-1427
TST3	004030	#10-876
TST30	010430	#10-1490
TST31	010632	#10-1528
TST32	011070	#10-1572
TST33	011334	10-1576 #10-1615
TST4	004162	10-896 #10-900
TST5	004364	10-903 10-932 #10-934
TST6	004466	#10-953
TST7	004552	10-965 #10-968
TYPDS	= 104405	10-620 10-1718 #10-2716
TYPE	= 104401	10-583 10-607 10-618 10-654 10-690 10-815 10-818 10-832 10-1718
		10-1718 10-1718 10-1991 10-2044 10-2058 10-2059 10-2062 10-2071 10-2076
		10-2085 10-2101 10-2104 10-2106 10-2107 10-2108 10-2108 10-2117 10-2117
		10-2117 10-2117 10-2117 10-2117 10-2117 10-2143 10-2181 10-2191 10-2226
		10-2226 10-2226 10-2226 10-2226 10-2226 10-2226 10-2226 10-2226 10-2226
		10-2226 10-2228 10-2588 10-2592 10-2610 10-2641 10-2645 #10-2716
TYPMAP	015772	10-817 #10-2135
TYP0C	= 104402	10-2061 10-2117 10-2117 10-2226 #10-2716
TYP0N	= 104404	#10-2716
TYP0S	= 104403	10-2189 10-2190 10-2192 10-2193 10-2591 10-2646 #10-2716
UP	= 000000	#8-340 10-1732 10-1733 10-1734 10-1735 10-1739
UUT	025124	10-618 #10-2760
VECADR	023156	10-2645 #10-2680
VECTIM	022642	#10-2633
VECTOR	022746	*10-560 *10-581 *10-1708 10-1709 10-1710 *10-1716 #10-2659
WARN	024107	10-607 #10-2735
WORK1	022750	10-560 10-581 10-1716 10-2585 10-2659 #10-2660
WORK2	022774	*10-2583 *10-2597 10-2598 10-2608 #10-2670
WORK3	022776	*10-2584 *10-2600 10-2601 #10-2671
WORK4	023000	*10-2590 *10-2591 10-2593 #10-2672
WRONG2	020500	10-2279 10-2281 10-2290 10-2296 #10-2313
WRONG3	021266	10-2313 #10-2411
WSTTIM	023216	10-572 10-604 #10-2691
XCMPO	021606	#10-2457
\$AP*HD	001000	8-353 #8-353
\$AS AT	= *****	10-2105 10-2105
\$A*IC	014534	10-2105 #10-2105
\$Y1	014510	#10-2105

SYMBOL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES
\$ATY3	014516	10-2104 #10-2105
\$ATY4	014526	#10-2105 10-2108
\$AUTOB	001134	#9-354 *10-583 10-2056 10-2226 10-2226 10-2226
\$BASE	001250	#9-354 10-2701
\$BDADR	001122	#9-354 *10-847 *10-860 *10-879 *10-905 *10-937 *10-981 *10-986 *10-992
		*10-996 *10-1005 *10-1035 *10-1048 *10-1063 *10-1083 *10-1108 *10-1126 *10-1145
		*10-1175 *10-1196 *10-1220 *10-1241 *10-1261 *10-1282 *10-1296 *10-1319 *10-1345
		*10-1396 *10-1441 *10-1452 *10-1460 *10-1469 *10-1479 *10-1487 *10-1505 *10-1547
		*10-1589 *10-1610 *10-1651 *10-1785 *10-1790 *10-1802 *10-1820 *10-1825 *10-1839
		*10-1846 *10-1909 *10-1942 *10-1975 10-2744 10-2745 10-2746
\$BDDAT	001126	#9-354 *10-842 *10-865 10-866 *10-886 *10-887 10-888 *10-916 *10-917
		10-918 *10-924 10-925 *10-942 10-943 *10-960 10-961 10-962 *10-963
		10-964 *10-979 *10-983 *10-984 *10-990 *10-994 *10-1014 *10-1015 *10-1018
		10-1019 *10-1027 *10-1028 *10-1039 10-1040 *10-1051 10-1052 10-1053 *10-1054
		10-1055 *10-1068 *10-1069 10-1070 *10-1090 *10-1094 *10-1098 10-1099 *10-1111
		10-1112 *10-1117 10-1118 *10-1131 10-1132 *10-1136 *10-1150 10-1151 *10-1156
		10-1157 *10-1177 *10-1193 10-1194 *10-1222 *10-1238 10-1239 *10-1263 *10-1279
		10-1280 *10-1298 *10-1323 *10-1332 10-1339 *10-1349 *10-1357 *10-1360 10-1367
		*10-1370 10-1372 *10-1398 *10-1443 *10-1448 10-1450 *10-1456 10-1458 *10-1464
		*10-1465 10-1467 *10-1475 10-1477 *10-1483 10-1485 *10-1502 *10-1503 *10-1507
		*10-1544 *10-1545 *10-1549 *10-1591 *10-1607 10-1608 *10-1648 *10-1649 *10-1653
		*10-1654 *10-1780 10-1783 *10-1788 *10-1798 *10-1799 10-1800 *10-1814 *10-1817
		10-1818 *10-1823 *10-1835 *10-1836 10-1837 *10-1843 10-1844 *10-1910 *10-1939
		*10-1972 10-2744 10-2745
\$BELL	001164	#9-354 10-2108 10-2108 10-2108
\$CDW1	001254	#9-354
\$CDW2	001256	#9-354
\$CHARC	014504	*10-2104 *10-2104 10-2104 *10-2104 #10-2104
\$CKSWR	016710	*10-2226 10-2716 10-2716
\$CMTAG	001100	#9-354 10-555 10-555 10-555 10-555 10-555
\$CM3	= 000000	#9-354 9-354
\$CNTLC	017566	10-2044 10-2071 10-2226 10-2226 10-2226 10-2226 #10-2226
\$CNTLG	017600	10-2058 10-2226 #10-2226
\$CNTLU	017573	10-2076 10-2226 #10-2226
\$CPUOP	001222	#9-354
\$CRLF	001171	#9-354 10-818 10-2085 10-2104 10-2104 10-2104 10-2108 10-2108 10 2108
		10-2117 10-2117 10-2117 10-2117 10-2226 10-2226 10-2226
		10-2107 10-2107 #10-2107
\$DBLK	015420	
\$DDW0	001260	#9-354 *10-615
\$DDW1	001262	#9-354
\$DDW10	001304	#9-354
\$DDW11	001306	#9-354
\$DDW12	001310	#9-354
\$DDW13	001312	#9-354
\$DDW14	001314	#9-354
\$DDW15	001316	#9-354
\$DDW2	001264	#9-354
\$DDW3	001266	#9-354
\$DDW4	001270	#9-354
\$DDW5	001272	#9-354
\$DDW6	001274	#9-354
\$DDW7	001276	#9-354

SYMBOL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES
\$DDW8	001300	#9-354
\$DDW9	001302	#9-354
\$DEVCT	001204	#9-354 *10-612
\$DEVM	001252	#9-354 10-597 10-616 *10-2601
\$DOAGN	012276	10-1718 10-1718 #10-1718
\$DTBL	015410	10-2107 #10-2107
\$ENDAD	012266	8-351 #10-1718
\$ENDCT	012230	#10-1718
\$ENDLF	012302	10-1718 #10-1718
\$ENDMG	012310	10-1718 #10-1718
\$ENULL	012305	10-1718 #10-1718
\$ENV	001214	#9-354 10-583 10-2104 10-2105 10-2105 10-2108
\$ENVM	001215	#9-354 10-555 10-2104 10-2104 10-2105
\$EOP	012174	#10-1718
\$EOPCT	012222	#10-1718 10-1718
\$ERFLG	001103	#9-354 *10-2108 10-2108 10-2108 10-2224 10-2224 10-2224 *10-2224 10-2224
\$ERMAX	001115	10-2224
\$ERROR	015430	#9-354 *10-555 10-2224 *10-2224 10-2224 10-2224
\$ERRPC	001116	10-555 #10-2108 *10-2108 10-2108 10-2108 10-2108 10-2108 10-2117 10-2744 10-2745
\$ERRTB	001320	10-2746
\$ERRTY	015636	#10-354 10-2117
\$ERTTL	001112	10-2114 #10-2117
\$ESCAP	001162	#9-354 *10-2108 10-2108 10-2108
\$ETABL	001214	#9-354 *10-555 10-2108 10-2108 10-2108 *10-2224
\$ETEND	001320	#9-354
\$FATAL	001176	8-353 #9-354
\$FFLG	014754	#9-354 *10-2105
\$FILLC	001156	*10-2105 *10-2105 10-2105 *10-2105 #10-2105
\$FILLS	001155	#9-354 10-2104 10-2104 10-2104
\$FLAG	023002	#9-354 10-652 10-688 10-811 *10-813 #10-2673
\$GDADR	001120	10-811 *10-1236 *10-1277 *10-1611 *10-1911 *10-1940 *10-1973
\$GDDAT	001124	#9-354 10-2745
		*10-841 *10-862 10-864 10-866 *10-869 *10-872 *10-873 *10-881
		10-884 10-888 *10-891 *10-892 *10-895 *10-897 *10-910 10-911 *10-912
		10-913 *10-914 *10-915 10-918 *10-921 *10-922 *10-923 10-925 *10-939
		10-941 10-943 *10-946 *10-949 *10-950 *10-959 10-964 *10-971 *10-1008
		*10-1009 *10-1012 10-1019 *10-1026 *10-1036 10-1040 *10-1050 10-1055 *10-1064
		10-1070 *10-1073 *10-1084 10-1099 *10-1109 10-1112 *10-1116 10-1118 *10-1127
		10-1132 *10-1138 *10-1146 10-1147 10-1151 *10-1155 10-1157 *10-1176 *10-1192
		10-1194 *10-1221 *10-1237 10-1239 *10-1262 *10-1278 10-1280 *10-1297 *10-1329
		*10-1331 *10-1359 *10-1371 10-1372 *10-1397 *10-1442 *10-1449 10-1450 *10-1457
		10-1458 *10-1466 10-1467 *10-1476 10-1477 *10-1484 10-1485 *10-1506 *10-1548
		*10-1590 *10-1606 10-1608 *10-1652 *10-1782 10-1783 *10-1791 *10-1794 *10-1795
		*10-1796 10-1800 *10-1816 10-1818 *10-1826 *10-1830 *10-1831 *10-1833 10-1837
		*10-1842 10-1844 *10-1941 *10-1974 10-2744 10-2745
\$GET42	012256	#10-1718
\$GTSWR	017036	#10-2226 10-2716 10-2716
\$HD	000003	8-330 8-330 8-330
\$HIBTS	001000	#8-353
\$ICNT	001104	#9-354 *10-2224 10-2224 *10-2224 10-2224 10-2224

SYMBOL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES
\$ILLUP	017774	10-2228 10-2228 #10-2228
\$INTAG	001135	#9-354 10-2226 10-2226
\$ITEMB	001114	#9-354 *10-2108 10-2108
\$LF	001172	#9-354 10-2104 10-2104
\$LFLG	014753	*10-2105 #10-2105
\$LPADR	001106	#9-354 *10-555 *10-837 *10-2224 *10-2224 10-2224
\$LPERR	001110	#9-354 *10-555 *10-840 *10-859 *10-878 *10-904 *10-936 *10-1004 *10-1062
		*10-1163 *10-1208 10-2108 10-2224 *10-2224 10-2224
\$MADR1	001226	#9-354
\$MADR2	001232	#9-354
\$MADR3	001236	#9-354
\$MADR4	001242	#9-354
\$MAIL	001174	8-353 8-353 #9-354 10-555 10-583 10-837 10-2104 10-2108 10-2224
\$MAMS1	001224	#9-354
\$MAMS2	001230	#9-354
\$MAMS3	001234	#9-354
\$MAMS4	001240	#9-354
\$MBADR	001002	#8-353
\$MFLG	014752	*10-2105 10-2105 *10-2105 #10-2105
\$MMAP	001664	#10-518 *10-1632 *10-1682 10-1684
\$MNEW	017616	10-2062 10-2226 #10-2226
\$MSGAD	001210	#9-354 *10-2105 10-2105
\$MSGLG	001212	#9-354 *10-2105
\$MSGTY	001174	#9-354 10-2105 *10-2105 10-2105 *10-2105
\$MSWR	017605	10-2059 10-2226 #10-2226
\$MTYP1	001225	#9-354
\$MTYP2	001231	#9-354
\$MTYP3	001235	#9-354
\$MTYP4	001241	#9-354
\$MXCNT	016706	10-2224 10-2224 10-2224 #10-2224
\$NULL	001154	#9-354 10-2104 10-2104
\$NWTST	= 000001	#10-837 10-837 #10-837 #10-857 10-857 #10-857 #10-876 10-876 #10-876
		#10-900 10-900 #10-900 #10-934 10-934 #10-934 #10-953 10-953 #10-953
		#10-968 10-968 #10-968 #10-999 10-999 #10-999 #10-1032 10-1032 #10-1032
		#10-1044 10-1044 #10-1044 #10-1060 10-1060 #10-1060 #10-1077 10-1077 #10-1077
		#10-1104 10-1104 #10-1104 #10-1122 10-1122 #10-1122 #10-1142 10-1142 #10-1142
		#10-1161 10-1161 #10-1161 #10-1206 10-1206 #10-1206 #10-1251 10-1251 #10-1251
		#10-1286 10-1286 #10-1286 #10-1317 10-1317 #10-1317 #10-1343 10-1343 #10-1343
		#10-1377 10-1377 #10-1377 #10-1427 10-1427 #10-1427 #10-1490 10-1490 #10-1490
		#10-1528 10-1528 #10-1528 #10-1572 10-1572 #10-1572 #10-1615 10-1615 #10-1615
\$OCNT	015200	*10-2106 *10-2106 #10-2106
\$OMODE	015202	*10-2106 *10-2106 10-2106 *10-2106 *10-2106 #10-2106
\$OVER	016672	10-2224 10-2224 10-2224 #10-2224
\$PASS	001202	#9-354 *10-555 *10-577 *10-1718 *10-1718 10-1718 10-1718 10-1718 10-2224
		10-2224 10-2224
\$PASTM	001006	#8-353
\$PWRAD	017770	#10-2228
\$PWRDN	017630	10-555 #10-2228 10-2228
\$PWRMG	017764	#10-2228
\$PWRUP	017702	10-2228 #10-2228
\$QUES	001170	#9-354 10-2101 10-2104 10-2104 10-2108 10-2108 10-2226 10-2226 10-2226
		10-2226

SYMBOL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES
\$RDCHR	017310	#10-2226 10-2716 10-2716
\$RDDEC	= *****	10-2716
\$RDLIN	017430	#10-2226 10-2716 10-2716
\$RDOCT	= *****	10-2716
\$RDSZ	= 000010	#10-2226 10-2226
\$RTNAD	012300	#10-1718
\$R2A	= *****	10-2716
\$SAVRE	= *****	10-2716
\$SAVR6	020000	*10-2228 10-2228 *10-2228 *10-2228 #10-2228
\$SCOPE	016430	10-555 #10-2224
\$SETUP	= 000117	#10-553 10-553 #10-553 10-553 #10-553 10-553 #10-553 10-553 #10-553 10-553 #10-553
		10-553 #10-553 10-555 10-555 10-555 10-555 10-555 10-555 10-583 10-583 10-583 10-1718
		10-1718 10-2108 10-2108 10-2108 10-2108 10-2108 10-2224 10-2226 10-2226
\$SIZE	022744	10-561 *10-2612 #10-2658
\$STUP	= 177777	#10-553 #10-553 10-553 #10-553 #10-553 10-553 #10-553 #10-553 10-553
		#10-553 #10-553 10-553
\$SVLAD	016636	10-2224 #10-2224
\$SVPC	= 000106	#8-351 8-351
\$SWR	= 167400	8-330 #8-330 #8-331 8-333 8-333 8-333 8-333 8-333 8-333 8-333 8-333
		8-333 8-333 9-354 9-354 9-354 10-555 10-555 10-555 10-555
		10-837 10-857 10-876 10-900 10-934 10-953 10-968 10-999 10-1032
		10-1044 10-1060 10-1077 10-1104 10-1122 10-1142 10-1161 10-1206 10-1251
		10-1286 10-1317 10-1343 10-1377 10-1427 10-1490 10-1528 10-1572 10-1615
		10-1718 10-1718 10-1718 10-1718 10-1718 10-2108 10-2108 10-2108 10-2108
		10-2108 10-2108 10-2108 10-2108 10-2108 10-2108 10-2224 10-2224 10-2224
		10-2224 10-2224 10-2224 10-2224 10-2224 10-2224 10-2224 10-2224 10-2224
		10-2224 10-2224 10-2224 10-2224 10-2224 10-2224 10-2224 10-2224 10-2224
\$SWREG	001216	10-2228
\$SWRMK	= 000000	#9-354 10-555
		8-333 8-333 8-333 8-333 8-333 8-333 8-333 8-333 8-333 8-333
		10-2224 10-2224 10-2224 10-2224 10-2224 10-2224 10-2224 10-2224 10-2224 10-2224
		10-2224
\$TBAE	001674	#10-522 *10-1638 10-1643 *10-1693 *10-1834 10-1842
\$TBAR	001672	#10-521 *10-1637 10-1642 *10-1692 10-1832
\$TESTN	001200	#9-354 *10-837 *10-2224
\$TIMES	001160	#9-354 *10-555 *10-968 *10-999 *10-1032 *10-1077 *10-1251 *10-1286 *10-1317
		*10-1343 *10-1377 *10-1427 *10-1490 *10-1528 *10-1615 *10-1718 *10-2224 10-2224
		*10-2224 10-2224 10-2224
\$TKB	001146	#9-354 10-2040 10-2067 10-2226 10-2226 10-2226 10-2226 10-2226 10-2226 10-2226
		10-2226
\$TKS	001144	#9-354 10-759 10-2037 10-2065 10-2226 10-2226 10-2226 10-2226 10-2226 10-2226
		10-2226 10-2226 10-2226
\$TMAP	001666	#10-519 *10-1633 10-1679 *10-1683
\$TN	= 000034	8-330 #8-330 #8-332 10-812 10-830 10-837 10-837 #10-837 10-837
		10-857 10-857 #10-857 10-876 10-876 #10-876 10-896 10-900 10-900
		#10-900 10-903 10-932 10-934 10-934 #10-934 10-953 10-953 #10-953
		10-965 10-968 10-968 #10-968 10-995 10-999 10-999 #10-999 10-1032
		10-1032 #10-1032 10-1041 10-1044 10-1044 #10-1044 10-1060 10-1060 #10-1060
		10-1077 10-1077 #10-1077 10-1104 10-1104 #10-1104 10-1119 10-1122 10-1122
		#10-1122 10-1142 10-1142 #10-1142 10-1158 10-1161 10-1161 #10-1161 10-1206
		10-1206 #10-1206 10-1251 10-1251 #10-1251 10-1286 10-1286 #10-1286 10-1317

SYMBOL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES
		10-1317 #10-1317 10-1343 10-1343 #10-1343 10-1377 10-1377 #10-1377 10-1427 10-1427 #10-1427 10-1490 10-1490 #10-1490 10-1528 10-1528 #10-1528 10-1572 10-1572 #10-1572 10-1576 10-1615 10-1615 #10-1615
#TPB	001152	#9-354 10-2104 10-2104 10-2104 10-2104
#TPFLG	001157	#9-354 10-2104 10-2104 10-2104 10-2104
#TPS	001150	#9-354 10-1580 10-1582 10-1606 10-1611 10-2104 10-2104 10-2104
#TRAP	023312	10-555 #10-2716
#TRAP2	023334	#10-2716 10-2716
#TRP	= 000012	#10-2716 10-2716 10-2716 10-2716 10-2716 #10-2716 10-2716 10-2716 #10-2716 10-2716 10-2716 10-2716 #10-2716 10-2716 10-2716 #10-2716 10-2716 10-2716 #10-2716 10-2716 10-2716
		#10-2716 10-2716 #10-2716 10-2716 #10-2716 10-2716 #10-2716 10-2716 #10-2716 10-2716 #10-2716 10-2716 #10-2716 10-2716 #10-2716 10-2716 #10-2716 10-2716 #10-2716 10-2716 #10-2716 10-2716
#TRPAD	023346	10-2716 #10-2716
#TSTM	001004	#8-353
#TSTM	001102	#9-354 *10-839 *10-1718 10-2108 10-2108 10-2108 10-2113 10-2224 10-2224
		*10-2224 10-2224 10-2224 10-2224 10-2224 10-2224
#TTST	001670	#10-520 *10-1634 *10-1678 10-1679 *10-1681 10-1684
#TTYIN	017556	10-2226 10-2226 10-2226 #10-2226
#TYPBN	= *****	10-2716
#TYPDS	015204	#10-2107 10-2716 10-2716
#TYPE	014226	#10-2104 10-2105 10-2716 10-2716
#YPEC	014440	10-2087 10-2104 10-2104 10-2104 #10-2104 10-2104 10-2226
#YPEX	014506	10-2104 10-2104 #10-2104
#TYPOC	015002	#10-2106 10-2716 10-2716
#TYPON	015016	10-2106 #10-2106 10-2716
#TYPOS	014756	#10-2106 10-2716
#UNIT	001206	#9-354 *10-596 10-612 10-613 10-619 *10-1712
#UNITM	001010	#8-353
#USMR	001220	#9-354
#VECT1	001244	#9-354 10-2708
#VECT2	001246	#9-354
#VERPC	001520	#10-457
#WRCK	015620	10-2108 #10-2113
#XTSTR	016442	#10-2224
#GET4	= 000000	#10-1718 10-1718
#OFILL	015201	*10-2106 *10-2106 10-2106 #10-2106
#4OCAT	= *****	10-2108 10-2224
.#ASTA	= *****	10-2105 10-2105
.#X	= 001000	#8-353 8-353

MACRO CROSS REFERENCE

CREF V02

MACRO NAME	REFERENCES
COMMEN	#8-334
ENDCOM	#8-334
ESCAPE	#8-334
GETPRI	#8-334
GETSWR	#8-326 #8-334 #10-583 10-583
MULT	#8-334
NEWTST	#8-326 #8-334 10-837 10-857 10-876 10-900 10-934 10-953 10-968 10-999
	10-1032 10-1044 10-1060 10-1077 10-1104 10-1122 10-1142 10-1161 10-1206 10-1251
	10-1286 10-1317 10-1343 10-1377 10-1427 10-1490 10-1528 10-1572 10-1615
POP	#8-334 10-2105 10-2105 10-2107 10-2194 10-2214 10-2228 10-2228
PUSH	#8-334 10-2105 10-2105 10-2105 10-2107 10-2147 10-2180 10-2228 10-2228
REPORT	#8-334
SETPRI	#8-334
SETTRA	#10-2716 10-2716 10-2716 10-2716 10-2716 10-2716 10-2716 10-2716 10-2716 10-2716
SETUP	#8-334 10-555
SKIP	#8-334 10-812 10-830 10-896 10-903 10-932 10-965 10-995 10-1041 10-1119
	10-1158 10-1576
SLASH	#8-326 #8-334
STARS	#8-322 #8-326 #8-334 8-351 8-353 8-353 8-353 9-354 9-354 9-354
	10-452 10-455 10-584 10-586 10-622 10-623 10-625 10-628 10-642 10-644
	10-656 10-658 10-676 10-685 10-694 10-704 10-718 10-720 10-737 10-751
	10-768 10-770 10-802 10-809 10-837 10-837 10-857 10-857 10-876 10-876
	10-900 10-900 10-934 10-934 10-953 10-953 10-968 10-968 10-999 10-999
	10-1032 10-1032 10-1044 10-1044 10-1060 10-1060 10-1077 10-1077 10-1104 10-1104
	10-1122 10-1122 10-1142 10-1142 10-1161 10-1161 10-1206 10-1206 10-1251 10-1251
	10-1286 10-1286 10-1317 10-1317 10-1343 10-1343 10-1377 10-1377 10-1378 10-1382
	10-1427 10-1427 10-1490 10-1490 10-1528 10-1528 10-1572 10-1572 10-1615 10-1615
	10-1695 10-1697 10-1718 10-1719 10-1721 10-1723 10-1730 10-1751 10-1756 10-1762
	10-1766 10-1773 10-1779 10-1808 10-1813 10-1850 10-1856 10-1875 10-1880 10-1899
	10-1904 10-1918 10-1925 10-1952 10-1958 10-1985 10-1989 10-2013 10-2020 10-2030
	10-2036 10-2104 10-2105 10-2106 10-2107 10-2108 10-2109 10-2112 10-2117 10-2118
	10-2134 10-2224 10-2226 10-2226 10-2226 10-2226 10-2228 10-2228 10-2504 10-2509
	10-2578 10-2580 10-2615 10-2617 10-2630 10-2632 10-2685 10-2689 10-2696 10-2698
	10-2716 10-2717 10-2719 10-2748 10-2750 10-2765 10-2768
SWRSU	#8-334 #10-555 10-555
TRMTRP	#10-2716
TYPBIN	#8-334
TYPDEC	#8-334 10-1718
TYPNAM	#8-334 10-583
TYPNUM	#8-334
TYPOCS	#8-334 10-2189 10-2190 10-2192 10-2193
TYPOCT	#8-334 10-2117 10-2117 10-2226
TYPTXT	#8-334
##CMRE	#8-354
##CMTM	#8-354
##ESCA	#8-334
##NEWT	#8-334 10-837 10-857 10-876 10-900 10-934 10-953 10-968 10-999 10-1032
	10-1044 10-1060 10-1077 10-1104 10-1122 10-1142 10-1161 10-1206 10-1251 10-1286
	10-1317 10-1343 10-1377 10-1427 10-1490 10-1528 10-1572 10-1615
##SET	#10-2716 10-2716 10-2716 10-2716 10-2716 10-2716 10-2716 10-2716 10-2716 10-2716
##SETM	#10-555 10-555
##SKIP	#8-334 10-812 10-930 10-896 10-903 10-932 10-965 10-995 10-1041 10-1119

MACRO CROSS REFERENCE

CREF V02

SEQ 0080

MACRO NAME	REFERENCES
	10-1158 10-1576
.EQUAT	08-321 8-334
.HEADE	08-321 8-330
.KT11	08-321 8-339
.SETUP	08-321 10-553
.SWRHI	08-321 8-333
.SWRLO	08-333
.\$ACT1	08-323 8-351
.\$APT8	09-354 9-354
.\$APTH	08-323 8-353
.\$APTY	08-324 10-2105
.\$CA^C	08-323 8-348
.\$CM^A	08-323 8-354
.\$EOP	08-324 10-1718
.\$ERRO	08-324 10-2108
.\$ERRT	08-325 10-2117
.\$POWE	08-325 10-2228
.\$READ	08-325 10-2226
.\$SCOP	08-325 10-2224
.\$TRAP	08-325 10-2716
.\$TYPD	08-324 10-2107
.\$TYPE	08-324 10-2104
.\$TYPC	08-324 10-2106