

.REM \$

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42

IDENTIFICATION

PRODUCT CODE: AC-A913B-MC  
PRODUCT NAME: CZKCFB0 BITSTUFF MD L TSTS  
DATE: AUGUST 1980  
MAINTAINER: DIAGNOSTICS-MERRIMACK

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

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED UNDER A LICENSE AND MAY ONLY BE USED OR COPIED IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE.

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

COPYRIGHT (C) 1977, 1980 BY DIGITAL EQUIPMENT CORPORATION

43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91

1. ABSTRACT

THE FUNCTION OF THE KMC11 DIAGNOSTICS IS TO VERIFY THAT THE OPTION OPERATES ACCORDING TO SPECIFICATIONS. THE DIAGNOSTICS VERIFY THAT THERE ARE NO MALFUNCTIONS AND THE ALL OPERATIONS OF THE KMC11 ARE CORRECT IN ITS ENVIRONMENT.

PARAMETERS MUST BE SET UP TO ALERT THE DIAGNOSTICS TO THE KMC11 CONFIGURATION. THESE PARAMETERS ARE CONTAINED IN THE STATUS TABLE AND ARE GENERATED IN TWO WAYS: 1) MANUAL INPUT - THE OPERATOR ANSWERS QUESTIONS. 2) AUTOSIZING - THE PROGRAM DETERMINES THE PARAMETERS AUTOMATICALLY.

CZKCF TESTS THE KMC-11 LINE UNIT (M8201 OR M8202). IT PERFORMS WRITE/READ TESTS ON THE KMC LINE UNIT REGISTERS. IT CHECKS FOR PROPER TRANSMITTER, RECEIVER, AND BCC OPERATION IN BITSTUFF MODE. THE MODEM SIGNALS ARE ALSO CHECKED. DZKCF REQUIRES A KMC MICRO-PROCESSOR (M8204) TO RUN. FOR BEST DIAGNOSIS A TURN-AROUND CONNECTOR SHOULD BE INSTALLED, HOWEVER THE DIAGNOSTIC WILL RUN WITHOUT IT (SOME TESTS ARE SKIPPED).

CURRENTLY THERE ARE FOUR OFF LINE DIAGNOSTICS THAT ARE TO BE RUN IN SEQUENCE TO INSURE THAT IF AN ERROR SHOULD OCCUR IT WILL BE DETECTED AT AN EARLY STAGE.

NOTE: ADDITIONAL DIAGNOSTICS MAY BE ADDED IN THE FUTURE.

THE FOUR DIAGNOSTICS ARE:

1. DZKCC [REV] BASIC W/R AND MICRO-PROCESSOR TESTS
2. DZKCD [REV] JUMP AND MAIN MEMORY TESTS (HEAT TEST TAPE)
3. DZKCE [REV] DDCMP LINE UNIT TESTS
4. DZKCF [REV] BITSTUFF LINE UNIT TESTS
5. DZKCA [REV] KMC11 CPU MICRO-DIGNOSTICS.

NOTE: NAMES MAY CHANGE AS UPDATES OCCUR.

2. REQUIREMENTS

2.1 EQUIPMENT

ANY PDP11 FAMILY CPU (EXCEPT AN LSI-11) WITH MINIMUM 8K MEMORY  
ASR 33 (OR EQUIVALENT)  
KMC11-AN IOP (M8204)  
KMC11-DA OR KMC11-MD OR KMC11-MA

92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134

2.2 STORAGE

PROGRAM WILL USE ALL 8K OF MEMORY EXCEPT WHERE ABL AND BOOTSTRAP LOADER RESIDE. LOCATIONS 2100 THRU 2300; CONTAIN THE "STATUS TABLE" INFORMATION WHICH IS GENERATED AT START OF DIAGNOSTICS BY MANUAL INPUT (QUESTIONS) OR AUTOMATICALLY (AUTO-SIZING). THIS AREA IS AN OVERLAY AREA AND SHOULD NOT BE ALTERED BY THE OPERATOR.

3. LOADING PROCEEDURE

3.1 METHOD

ALL PROGRAMS ARE IN ABSOLUTE FORMAT AND ARE LOADED USING THE ABSOLUTE LOADER. NOTE: IF THE DIAGNOSTICS ARE ON A MEDIA SUCH AS DISK ,MAGTAPE,DECTAPE, OR CASSETTE; FOLLOW INSTRUCTIONS FOR THE MONITOR WHICH HAS BEEN PROVIDED ON THAT SPECIFIC MEDIA.

ABSOLUTE LOADER STARTING ADDRESS \*500

MEMORY \* SIZE

|     |     |
|-----|-----|
| 4K  | 17  |
| 8K  | 37  |
| 12K | 57  |
| 16K | 77  |
| 20K | 117 |
| 24K | 137 |
| 28K | 157 |

3.1.1 PLACE ADDRESS OF ABS LOADER INTO SWITCH REGISTER.  
(ALSO PLACE 'HALT' SW UP)

3.1.2 DEPRESS 'LOAD ADDRESS' KEY ON CONSOLE AND RELEASE.

3.1.3 DEPRESS 'START KEY' ON CONSOLE AND RELEASE (PROGRAM SHOULD NOW BE LOADING INTO CPU)

135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187

4. STARTING PROCEEDURE

- A. SET SWITCH REGISTER TO 000200
- B. DEPRESS 'LOAD ADDRESS' KEY AND RELEASE
- C. SET SWR TO ZERO FOR 'AUTO SIZING' OR SWR BIT0=1 FOR MANUAL INPUT (QUESTIONS) OR SWR BIT7=1 TO USE EXISTING PARAMETERS SET UP BY A PREVIOUS START OR A PREVIOUSLY RUN KMC11 DIAGNOSTIC.
- D. DEPRESS 'START KEY' AND RELEASE. THE PROGRAM WILL TYPE MAINDEC NAME AND PROGRAM NAME (IF THIS WAS THE FIRST START UP OF THE PROGRAM) AND ALSO THE FOLLOWING:

MAP OF KMC11 STATUS

| PC     | CSR    | STAT1  | STAT2  | STAT3  |
|--------|--------|--------|--------|--------|
| 002100 | 160010 | 045310 | 177777 | 000000 |
| 002110 | 160020 | 045320 | 177777 | 000000 |

THE PROGRAM WILL TYPE 'R' AND PROCEED TO RUN THE DIAGNOSTIC. THE ABOVE IS ONLY AN EXAMPLE. THIS WOULD INDICATE THE STATUS TABLE STARTING AT ADD. 2100 IN THE PROGRAM. IN THIS EXAMPLE THE TABLE CONTAINS THE INFORMATION AND STATUS OF TWO KMC11'S. THE STATUS TABLE MUST BE VERIFIED BY THE USER IF AUTO SIZING IS DONE. FOR INFORMATION OF STATUS TABLE SEE SECTION 8.4 FOR HELP.

IF THE DIAGNOSTIC WAS STARTED WITH SW00=1 INDICATING MANUAL PARAMETER INPUT THEN THE FOLLOWING SHOWS AN EXAMPLE OF THE QUESTIONS ASKED AND SOME EXAMPLE ANSWERS:  
HOW MANY KMC11'S TO BE TESTED?1

01  
CSR ADDRESS?160010  
VECTOR ADDRESS?310  
BR PRIORITY LEVEL? (4,5,6,7)?5  
WHICH LINE UNIT? IF NONE TYPE 'N', IF M8201 TYPE '1', IF M8202 TYPE '2'?1  
IS THE LOOP BACK CONNECTOR ON?Y  
(IF M8201 AND LOOP BACK CONNECTOR ON THEN YOU GET THE NEXT QUESTION)  
WHICH MODEM TYPE, TYPE 'D' FOR KMC11-DA (RS232),OR TYPE 'F' FOR KMC11-FA (V.35) ? D  
SWITCH PAC#1 (DDCMP LINE#)?377  
SWITCH PAC#2 (BM873 BOOT ADD)?377  
FOLLOWING THE QUESTIONS THE STATUS MAP IS PRINTED OUT AS DESCRIBED ABOVE, THE INFORMATION IN THE MAP REFLECTS THE ANSWERS TO THE QUESTIONS. IF THE DIAGNOSTIC WAS STARTED WITH SW00=0 AND SW07=0 (AUTO-SIZING) THEN NO QUESTIONS ARE ASKED AND ONLY THE STATUS-MAP IS PRINTED OUT. IF AUTO-SIZING IS USED THE STATUS INFORMATION MUST BE VERIFIED TO BE CORRECT (MATCH THE HARDWARE). IF IT DOES NOT MATCH THE HARDWARE THE DIAGNOSTIC MUST BE RESTARTED WITH SW00=1 AND THE QUESTIONS ANSWERED.

188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215

4.1 CONTROL SWITCH SETTINGS

SW 15 SET: HALT ON ERROR  
SW 14 SET: LOOP ON CURRENT TEST  
SW 13 SET: INHIBIT ERROR PRINT OUT  
SW 12 SET: INHIBIT TYPE OUT ABELL ON ERROR.  
SW 11 SET: INHIBIT ITERATIONS. (QUICK PASS)  
SW 10 SET: ESCAPE TO NEXT TEST ON ERROR  
SW 09 SET: LOOP WITH CURRENT DATA  
SW 08 SET: CATCH ERROR AND LOOP ON IT  
SW 07 SET: USE PREVIOUS STATUS TABLE.  
SW 06 SET: HALT IN ROMCLK ROUTINE BEFORE CLOCKING  
MICRO-PROCESSOR  
SW 05 SET: RESERVED  
SW 04 SET: RESERVED  
SW 03 SET: RESELECT KMC11'S DESIRED ACTIVE  
SW 02 SET: LOCK ON SELECTED TEST  
SW 01 SET: RESTART PROGRAM AT SELECTED TEST  
SW 00 SET: BUILD NEW STATUS TABLE FROM QUESTIONS. (IF SW07=0  
AND SW00=0 A NEW STATUS TABLE IS BUILT BY  
AUTO-SIZING)

SWITCH 06 AND 08-15 ARE DYNAMIC AND CAN BE CHANGED AS NEEDED  
WHILE THE DIAGNOSTIC IS RUNNING. SWITCHES 00-03 AND SWITCH 07  
ARE STATIC, AND ARE USED ONLY ON STARTING OR RESTARTING THE  
DIAGNOSTIC.

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  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258

4.1.2 SWITCH REGISTER OPTIONS (AT START UP)

SW 01 RESTART PROGRAM AT SELECTED TEST. IT IS STRONGLY SUGGESTED THAT AT LEAST ONE PASS HAS BEEN MADE BEFORE TRYING TO SELECT A TEST, THE REASON BEING IS THAT THE PROGRAM HAS TO CLEAR AREAS AND SET UP PARAMETERS. WHEN THIS SWITCH IS USED THE DIAGNOSTIC WILL ASK TEST NO.? ANSWER BY TYPING THE NUMBER OF THE TEST DESIRED AND CARRIGE RETURN TO BEGIN EXECUTION AT THE SELECTED TEST.

SW 02 LOCK ON SELECTED TEST. THIS SWITCH WHEN USED WITH SW01 WILL CAUSE THE PROGRAM TO CONSTANTLY LOOP ON THE SELECTED TEST. HITTING ANY KEY ON THE CONSOLE WILL LET IT ADVANCE TO THE NEXT TEST AND LOOP UNTIL A KEY IS HIT AGAIN. IF SW02=0 WHEN SW01 IS USED, THE PROGRAM WILL BEGIN AT THE SELECTED TEST AND CONTINUE NORMAL OPERATIONS.

SW 03 RESELECT KMC11'S DESIRED ACTIVE. PLEASE NOTE THAT A MESSAGE IS TYPED OUT FOR SETTING THE SWITCH REGISTER EQUAL TO KMC11'S ACTIVE. THIS MEANS IF THE SYSTEM HAS FOUR KMC11S; BITS 00,01,02,03 WILL BE SET IN LOC 'KMACTV' FROM THE SWITCH REGISTER. USING THIS SWITCH(SW00) ALTERS THAT LOCATION;THEREFORE IF FOUR KMC11S ARE IN THE SYSTEM \*\*\*DO NOT\*\*\* SET SWITCHS GREATER THAN SW 03 IN THE UP POSITION. THIS WOULD BE A FATAL ERROR. DO NOT SELECT MORE ACTIVE KMC11S THAN THERE IS INFORMATION ON IN THE STATUS TABLE.

METHOD: A: LOAD ADDRESS 200  
B: START WITH SW 00=1  
C: PROGRAM WILL TYPE MESSAGE  
D: SET A SWITCH FOR EACH KMC DESIRED ACTIVE.  
EXAMPLE: IF YOU HAVE 4 KMC'S BUT ONLY WANT TO RUN THE FIRST AND THE LAST SET SWR BITS 0 AND 3 = 1. PRESS CONTINUE  
E: NUMBER (IF VALID) WILL BE IN DATA LIGHTS (EXCLUDING 11/05)  
F: SET WITH ANY OTHER SWITCH SETTINGS DESIRED. PRESS CONTINUE.

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  
302  
303  
304  
305  
306

#### 4.1.3 DYNAMIC SWITCHES

##### ERROR SWITCHES

1. SW 12 DELETE PRINT OUT/BELL ON ERROR.
2. SW 13 DELETE ERROR PRINTOUT.
3. SW 15 HALT ON THE ERROR.
4. SW 08 GOTO BEGINNING OF THE TEST(ON ERROR).
5. SW 10 GOTO NEXT TEST(ON ERROR).

##### SCOPE SWITCHES

1. SW06 HALT IN ROMCLK ROUTINE BEFORE CLOCKING MICRO-PROCESSOR INSTRUCTION. THIS ALLOWS THE OPERATOR TO SCOPE A MICRO-PROCESSOR INSTRUCTION IN THE STATIC STATE BEFORE IT IS CLOCKED. HIT CONTINUE TO RESUME RUNNING.
2. SW09 (IF ENABLED BY 'SCOPI') ON AN ERROR; IF AN '\*' IS PRINTED IN FRONT OF THE TEST NO. (EX. \*TEST NO. 10 ) SW09 IS INCORPORATED IN THAT TEST AND THEREFORE SW09 IS USUALLY THE BEST SWITCH FOR THE SCOPE LOOP (SW14=0, SW10=0, SW09=1, SW08=0). IF SW09 IS NOT ENABLED; AND THERE IS A HARD ERROR (CONSTANT); SW08 IS BEST. (SW14=1,0, SW10=0, SW09=0, SW08=1). FOR INTERMITTENT ERRORS; SW14=1 WILL LOOP ON TEST REGARDLESS OF ERROR OR NOT ERROR. (SW14=1, SW10=0, SW09=0, SW08=1,0)
3. SW11 INHIBIT INTERACTIONS.
4. SW14 LOOP ON CURRENT TEST.

#### 4.2 STARTING ADDRESS

STARTING ADDRESS IS AT 000200 THERE ARE NO OTHER STARTING ADDRESSES FOR THE KMC11 DIAGNOSTICS. (SEE SECTION 4.0)

NOTE: IF ADDRESS 000042 IS NON-ZERO THE PROGRAM ASSUMES IT IS UNDER ACT11 OR XXDP CONTROL AND WILL ACT ACCORDINGLY AFTER ALL AVAILABLE KMC11'S ARE TESTED THE PROGRAM WILL RETURN TO 'XXDP' OR 'ACT-11'.

#### 5. OPERATING PROCEDURE

WHEN PROGRAM IS INITIALLY STARTED MESSAGES AS DESCRIBED IN SECTION 4.0 WILL BE PRINTED, AND PROGRAM WILL BEGIN RUNNING THE DIAGNOSTIC

307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354

5.2 PROGRAM AND/OR OPERATOR ACTION

THE TYPICAL APPROACH SHOULD BE

1. HALT ON ERROR (VIA SW 15=1) WHEN EVER AN ERROR OCCURS.
2. CLEAR SW 15.
3. SET SW 14: (LOOP ON THIS TEST)
4. SET SW 13: (INHIBIT ERROR PRINT OUT)

THE TEST NUMBER AND PC WILL BE TYPED OUT AND POSSIBLY AN ERROR MESSAGE (THIS DEPENDS ON THE TEST) TO GIVE THE OPERATOR AN IDEA AS TO THE SOURCE OF THE PROBLEM. IF IT IS NECESSARY TO KNOW MORE INFORMATION CONCERNING THE ERROR REPORT; LOOK IN THE LISTING FOR THAT TEST NUMBER WHICH WAS TYPED OUT AND THEN NOTE THE PC OF THE ERROR REPORT THIS WAY THE EXACT FUNCTION OF THE TEST CAN BE DETERMINED.

6. ERRORS

AS DESCRIBED PREVIOUSLY THERE WILL ALWAYS BE A TEST NUMBER AND PC TYPED OUT AT THE TIME OF AN ERROR (PROVIDING SW 13=0 AND SW 12=0). IN MOST CASES ADDITIONAL INFORMATION WILL BE SUPPLIED IN THE THE ERROR MESSAGE TO GIVE THE OPERATOR AN INDICATION OF THE ERROR.

6.2 ERROR RECOVERY

IF FOR SOME REASON THE KMC11 SHOULD 'HANG THE BUS' (GAIN CONTROL OF BUS SO THAT CONSOLE MANUAL FUNCTIONS ARE INHIBITED) AN INIT OR POWER DOWN/UP IS NECESSARY FOR OPERATOR TO REGAIN CONTROL OF CPU. IF THIS SHOULD HAPPEN; LOOK IN LOCATION 'TSTNM' (ADDRESS 1202) FOR THE NUMBER OF THE TEST THAT WAS RUNNING AT THE TIME OF THE CATASTROPHIC ERROR. IN THIS WAY THE OPERATOR WILL HAVE AN IDEA AS TO WHAT THE KMC11 WAS DOING AT THE TIME OF THE ERROR.

7. RESTRICTIONS

7.1 STARTING RESTRICTIONS

SEE SECTION 4. (PLEASE)  
STATUS TABLE SHOULD BE VERIFIED REGARDLESS OF HOW PROGRAM WAS STARTED. ALSO IT IS IMPORTANT TO USE THIS LISTING ALONG WITH THE INFORMATION PRINTED ON THE TTY TO COMPLETELY ISOLATE PROBLEMS.



355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407

7.2 OPERATING RESTRICTIONS

THE FIRST TIME A KMC11 DIAGNOSTIC IS LOADED INTO CORE AND RUN THE STATUS TABLE MUST BE SET UP. THIS IS DONE BY MANUAL INPUT (SW00=1) OR BY AUTOSIZING (SW00=0 AND SW07=0). THEREAFTER HOWEVER THE STATUS TABLE NEED NOT BE SETUP BY SUBSEQUENT RESTARTS OR EVEN LOADING THE NEXT KMC DIAGNOSTIC BECAUSE THE STATUS TABLE IS OVERLAYED. THE CURRENT PARAMETERS IN THE STATUS TABLE ARE USED WHEN SW07=1 ON START UP.

7.3 HARDWARE CONFIGURATION RESTRICTIONS

KMC11 IOP(M8204)- JUMPER W1 MUST BE IN,

LINE UNIT(M8201)- JUMPERS W1, W2, AND W4 MUST BE IN. JUMPERS W3, AND W5 MUST BE OUT. SW8 OF E26 MUST BE IN THE ON POSITION.

LINE UNIT (M8202)- JUMPER W1 MUST BE IN. SW8 OF E26 MUST BE IN THE OFF POSITION.

8. MISCELLANEOUS

8.1 EXECUTION TIME

ALL KMC11 DEVICE DIAGNOSTICS WILL GIVE AN 'END PASS' MESSAGE (PROVIDING NO ERRORS AND SW12=0) WITHIN 4 MINS. THIS IS ASSUMING SW11=1 (DELETE ITERATIONS) IS SET TO GIVE THE FASTEST POSSIBLE EXECUTION. THE ACTUAL EXECUTION TIME DEPENDS GREATLY ON THE PDP11 CPU CONFIGURATION AND THE AMOUNT OF MEMORY IN THE SYSTEM.

8.2 PASS COMPLETE

NOTE: EVERY TIME THE PROGRAM IS STARTED; THE TESTS WILL RUN AS IF SW11 (DELETE ITERATIONS) WAS UP (=1). THIS IS TO 'VERIFY NO HARD ERRORS' AS SOON AS POSSIBLE. THEREFORE THE FIRST PASS -EACH TIME PROGRAM IS STARTED- WILL BE A 'QUICK PASS' UNTIL ALL KMC11'S IN SYSTEM ARE TESTED. WHEN THE DIAGNOSTIC HAS COMPLETED A PASS THE FOLLOWING IS AN EXAMPLE OF THE PRINT OUT TO BE EXPECTED.

END PASS CZKCF CSR: 175000 VEC: 0300 PASSES: 000001  
ERRORS: 000000

NOTE: THE PASS COUNT AND ERROR COUNTS ARE CUMMULITIVE FOR EACH KMC11 THAT IS RUNNING, AND ARE SET TO ZERO ONLY WHEN THE DIAGNOSTIC IS STARTED. THEREFORE AFTER AN OVERNIGHT RUN FOR EXAMPLE, THE TOTAL PASSES AND ERRORS FOR EACH KMC11 SINCE THE DIAGNOSTIC WAS STARTED ARE REFLECTED IN PASSES: AND ERRORS:.

408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463

8.4 KEY LOCATIONS

LPADR (1206) CONTAINS THE ADDRESS WHERE PROGRAM WILL RETURN WHEN ITERATION COUNT IS REACHED OR IF LOOP ON TEST IS ASSERTED.

NEXT (1442) CONTAINS THE ADDRESS OF THE NEXT TEST TO BE PERFORMED.

TSTNM (1202) CONTAINS THE NUMBER OF THE TEST NOW BEING PERFORMED.

RUN (1500) THE BIT IN 'RUN' ALWAYS POINTS TO THE KMC11 CURRENTLY BEING TESTED. EXAMPLE: (RUN) 1500/000000001000000 MEANS THAT KMC11 NO.06 IS THE KMC11 NOW RUNNING.

KMC00-KMC17  
KMST00-KMST17  
(2100)-(2300)

THESE LOCATIONS CONTAIN THE INFORMATION NEEDED TO TEST UP TO 16 (DECIMAL) KMC11S SEQUENTIALY. THEY CONTAIN THE CSR, VECTOR AND STATUS CONCERNING THE CONFIGURATION OF EACH KMC11.

KMACTV (1470)

EACH BIT SET IN THIS LOCATION INDICATES THAT THE ASSOCIATED KMC11 WILL BE TESTED IN TURN. EXAMPLE: (KMACTV) 1470/000000000011111 MEANS THAT KMC11 NO. 00,01,02,03,04 WILL BE TESTED. EXAMPLE: (KMACTV) 1470/000000000010001 MEANS THAT KMC11 NO. 00,04 WILL BE TESTED.

KMCSR (2066)

CONTAINS THE CSR OF THE CURRENT KMC11 UNDER TEST.

8.4A 'STATUS TABLE' (2100-2300)

THE TABLE IS FILLED BY AUTO SIZING OR BY THE MANUAL PARAMETER INPUT (QUESTIONS) AS DESCRIBED PREVIOUSLY. ALSO IF DESIRED BY USER; THE LOCATIONS MAY BE ALTERED BY HAND (TOGGLED IN) TO SUIT THE SPECIFIC CONFIGURATION.

THE EXAMPLE STATUS MAP SHOWN BELOW CONTAINS INFORMATION FOR TWO KMC11'S. THE TABLE CAN CONTAIN UP TO 16 KMC11'S. FOLLOWING THE MAP IS A DESCRIPTION OF THE BITS FOR EACH MAP ENTRY

MAP OF KMC11 STATUS  
-----

| PC     | CSR    | STAT1  | STAT2  | STAT3  |
|--------|--------|--------|--------|--------|
| --     | ---    | -----  | -----  | -----  |
| 002100 | 160010 | 045310 | 177777 | 000000 |
| 002110 | 160020 | 016320 | 000000 | 000000 |

464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489

EACH MAP ENTRY CONTAINS 4 WORDS WHICH CONTAIN THE STATUS INFORMATION FOR 1 KMC11. THE PC SHOWS WHERE IN CORE MEMORY THE FIRST OF THE 4 WORDS IS. IN THE EXAMPLE ABOVE THE FIRST KMC'S STATUS IS IN LOCATIONS, 2100, 2102, 2104, AND 2106. THE SECOND KMC STATUS IS LOCATED AT 2110, 2112, 2114, AND 2116. THE INFORMATION CONTAINED IN EACH 4 WORD ENTRY IS DEFINED AS FOLLOWS:

CSR: CONTAINS KMC11 CSR ADDRESS

STAT1: BITS 00-08 IS KMC11 VECTOR ADDRESS  
BIT14=1 TURNAROUND CONNECTOR IS ON  
BIT14=0 NO TURNAROUND CONNECTOR  
BIT13=0 LINE UNIT IS AN M8201  
BIT13=1 LINE UNIT IS AN M8202  
BIT12=1 NO LINE UNIT  
BITS 09-11 IS KMC11 BR PRIORITY LEVEL

STAT2: LOW BYTE IS SWITCH PAC#1 (DDCMP LINE NUMBER)  
HIGH BYTE IS SWITCH PAC#2 (BM873 BOO: ADD)

STAT3: BIT2=0 KMC11-DA  
BIT2=1 KMC11-FA

490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544

8.5 METHOD OF AUTO SIZING

8.5.1 FINDING THE CONTROL STATUS REGISTER.

THE AUTO-SIZING ROUTINE FINDS A KMC11 AS FOLLOWS: IT STARTS AT ADDRESS 160000 AND TESTS ALL ADDRESS IN INCREMENTS OF 10 UP TO AND INCLUDING ADDRESS 167760. IF THE ADDRESS DOES NOT TIME OUT, THE FOLLOWING IS DONE, THE FIRST CRAM ADDRESS IS WRITTEN TO A 125252 THEN IT IS READ BACK. IF IT CONTAINS A -1 OR 125252 A KMC11 HAS BEEN FOUND, IF NOT, THE ADDRESS IS UPDATED BY 10 AND THE SEARCH CONTINUES. A -1 INDICATES A KMC11 WITH NO CRAM, A 125252 INDICATES A KMC11 WITH CRAM FURTHER TESTS ARE PERFORMED AT THIS POINT TO DETERMINE WHICH LINE UNIT, IF ANY, IS INSTALLED, IF A LOOP-BACK CONNECTOR IS INSTALLED AND VARIOUS SWITCH SETTINGS ON THE LINE UNIT. THIS IS WHY THE STATUS TABLE MUST BE VERIFIED BY THE USER AND IF ANY OF THE INFORMATION DOES NOT AGREE WITH THE HARDWARE THE DIAGNOSTIC MUST BE RESTARTED AND THE QUESTIONS MUST BE ANSWERED. ALL KMC11'S IN THE SYSTEM WILL BE FOUND BY THE AUTO-SIZER. IF IT DOES NOT FIND A KMC11 THE DIAGNOSTIC MUST BE RESTARTED AND THE QUESTIONS ANSWERED.

8.5.2 FINDING THE VECTOR AND BR LEVEL

THE VECTOR AREA (ADDRESS 300-776) IS FILLED WITH THE INSTRUCTION IOT AND '+2' (NEXT ADDRESS). THE PROCESSOR STATUS IS STARTED AT 7 AND THE KMC IS PROGRAMMED TO INTERRUPT. THE PS IS LOWERED BY 1 UNTIL THE KMC INTERRUPTS, A DELAY IS MADE AND IF NO INTERRUPT OCCURES AT PS LEVEL 3 (BECAUSE OF A BAD KMC11) THE PROGRAM ASSUMES VECTOR ADDRESS 300 AT BR LEVEL 5 AND THE PROBLEM SHOULD BE FIXED IN THE DIAGNOSTIC. ONCE THE PROBLEM IS FIXED; THE PROGRAM SHOULD BE RE-SETUP AGAIN TO GET CORRECT VECTOR. IF AN INTERRUPT OCCURED; THE ADDRESS TO WHICH THE KMC11 INTERRUPTED TO IS PICKED UP AND REPORTED AS THE VECTOR. NOTE: IF THE VECTOR REPORTED IS NOT THE VECTOR SET UP BY YOU; THERE IS A PROBLEM AND AUTO SIZING SHOULD NOT BE DONE.

8.6 SOFTWARE SWITCH REGISTER

IF THE DIAGNOSTIC IS RUN ON AN 11/04 OR OTHER CPU WITHOUT A SWITCH REGISTER THEN A SOFTWARE SWITCH REGISTER IS USED TO ALLOW USER THE SAME SWITCH OPTIONS AS DESCRIBED PREVIOUSLY. IF THE HARDWARE SWITCH REGISTER DOES NOT EXIST OR IF ONE DOES AND IT CONTAINS ALL ONES (177777) THIS SOFTWARE SWITCH REGISTER IS USED.

CONTROL:  
TO OBTAIN CONTROL AT ANY ALLOWABLE TIME DURING EXECUTION OF THE DIAGNOSTIC THE OPERATOR TYPES A CTRL G ON THE CONSOLE TERMINAL KEYBOARD. AS SOON AS THE CTRL G IS RECOGNIZED, BY THE DIAGNOSTIC, THE FOLLOWING MESSAGE WILL BE DISPLAYED:

SWR=XXXXXX NEW?

545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585

WHERE XXXXXX IS THE CURRENT CONTENTS OF THE SOFTWARE SWITCH REGISTER IN OCTAL. THE SOFTWARE CONTROL ROUTINE WILL THEN AWAIT OPERATOR ACTION. AT WHICH TIME THE OPERATOR IS REQUIRED TO TYPE ONE OR MORE OF THE LEGAL CHARACTERS: 1) 0 - 7, 2) LINE FEED(<LF>), 3) CARRIAGE RETURN(<CR>), OR 4) CONTROL-U (CTRL U). NO CHECK IS MADE FOR LEGALITY. IF THE INPUT CHARACTER IS NOT A <LF>, <CR>, OR CTRL U IT IS ASSUMED TO BE AN OCTAL DIGIT.

TO CHANGE THE CONTENTS OF THE SSR THE OPERATOR SIMPLY TYPES THE NEW DESIRED VALUE IN OCTAL - LEADING ZEROS NEED NOT BE TYPED. AND TERMINATES THE INPUT STRING WITH A <CR> OR <LF> DEPENDING ON THE PROGRAM ACTION DESIRED AS DESCRIBED BELOW. THE INPUT VALUE WILL BE TRUNCATED TO THE LAST 6 DIGITS TYPED. AT LEAST ONE DIGIT MUST BE TYPED ON ANY GIVEN INPUT STRING PRIOR TO THE TERMINATOR BEFORE A CHANGE TO THE SSR WILL OCCUR.

WHEN THE INPUT STRING IS TERMINATED WITH A <CR> THE DIAGNOSTIC WILL CONTINUE EXECUTION FROM THE POINT AT WHICH IT WAS INTERRUPTED. IF A <CR> IS THE ONLY THING TYPED THE PROGRAM WILL CONTINUE WITHOUT CHANGING THE SSR. THE <LF> DIFFERS FROM THE <CR> BY RESTARTING THE PROGRAM AS IF IT WERE RESTARTED AT ADDRESS 200.

IF A CTRL U IS TYPED AT ANY POINT IN THE INPUT STRING PRIOR TO THE TERMINATOR THE INPUT VALUE WILL BE DISREGARDED AND THE PROMPT DISPLAYED (SWR = XXXXXX NEW?).

TO SET THE SSR FOR THE STARTING SWITCHES, FIRST LOAD THE DIAGNOSTIC, THEN HIT CTRL G, THEN START THE DIAGNOSTIC.

\*\*\*\*\*

NOTE:FOR IPG'S LINE UNIT M8202-YE USERS.

CABLE DATA TEST:[TEST 60, TEST 61]

THESE TESTS WON'T RUN RELIABLY ON LINE UNITS WITHOUT TERMINATING RESISTENCE.

586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641

APT/ACT/XXDP/SLIDE

-----  
THIS DIAGNOSTIC IS APT/ACT/XXDP/SLIDE COMPATIBLE USER WOULD BE  
ABLE TO RUN IT UNDER APT/ACT/XXDP ENVIRONMENT.

NOTE: FOR MANUFACTURING PURPOSE ONLY ITS DESCRIBED HOW TO RUN  
UNDER APT ENVIRONMENT.

\*\*\*\*\*  
ETABLE SETTING FOR APT TO RUN UNDER APT

-----  
FIRST PASS TIME:

LONGEST TEST TIME:

ADDITIONAL TEST TIME:

ALL THE ABOVE PARAMETERS ARE DEPENDENT ON PARTICULAR  
DIAGNOSTICS AND SHOULD BE LOADED AT THE TIME OF SETTING  
ETABLE.THERE IS NO DEFAULT TIME SET UP.

SOFTWARE ENVIRONMENT:001 ENVIRONMENT MODE:200

SWITCH 1:-SHOULD BE USED AS NORMAL SWITCH REGISTER.

SWITCH 2:-NOT USED.

CPU OPTIONS:-NOT USED.

MEMORY TYPE 1:-BITS<2:4>:=BITS <12:14> OF STAT1 OF DEV:0.

MAXIMUM ADDRESS:-BITS<17:19>:=BITS<12:14> OF STAT1 OF DEV:1

BITS<2:4>:=BITS <12:14> OF STAT1 OF DEV:2

BITS<10:12>:=BITS<12:14> OF STAT1 OF DEV:3

IN THE SAME MANNER

MEMORY TYPE 2 MAXIMUM ADDRESS:-GETS STAT1<12:14> OF DEVICE  
4,5,6,7.

MEMORY TYPE 3 MAXIMUM ADDRESS:-GETS STAT1<12:14> OF DEVICE  
8,9,10,11.

MEMORY TYPE 4 MAXIMUM ADDRESS:-GETS STAT1<12:14> OF DEVICE  
12,13,14,15.

INTERRUPT VECTOR 1:FIRST DEVICE RECEIVE VECTOR.

642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686

REST OF THE DEVICE(KMC'S) VECTOR SHOULD BE SET UP SEQUENTIALLY  
IN INCREMENTS OF 10.

BUS PRIORITY:KMC'S PRIORITY(SHOULD BE SAME FOR ALL KMC'S UNDER  
TEST).

INTERRUPT VECTOR 2:NOT USED.

BUS PRIORITY:NOT USED.

BASE ADDRESS:FIRST DEVICE CSR ADDRESS.

REST SHOULD FOLLOW SEQUENTIALLY  
IN INCREMENTS OF 10.

DEVICE MAP:AS DESCRIBED IN APT MANUAL.

CONTROLLER SPECIFIC CODE 1:-NO. OF DEVICES UNDER TEST.

CONTROLLER SPECIFIC CODE 2:-NOT USED.

DEVICE DESCRIPTOR WORD 0:STAT2 OF FIRST DEVICE.

. .

. .

TO

. .

. .

DEVICE DESCRIPTOR WORD 15:STAT2 OF 16TH DEVICE.(KMC)

## 9.0 HISTORY

THIS DIAGNOSTIC WAS UPDATED TO DETECT FOR THE CONDITION OF V.35  
AND M8201. IN THIS CONIFURATION, RING WILL NOT BE LOOPED BACK  
AND SHOULD NOT BE TESTED FOR.

\$

```
687 .TITLE CZKCF
688 :*COPYRIGHT (C) 1976
689 :*DIGITAL EQUIPMENT CORP.
690 :*MAYNARD, MASS. 01754
691 :*
692 :*PROGRAM BY DINESH GORADIA
693 :*
694 :*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
695 :*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
696 :*
697
698
699
700
701
702 :*CZKCF BIT STUFF MD L TSTS
703 :*COPYRIGHT 1976, DIGITAL EQUIPMENT CORP., MAYNARD, MASS. 01754
704 :-----
705
706 :STARTING PROCEDURE
707 :LOAD PROGRAM
708 :LOAD ADDRESS 000200
709 :SWR=0 AUTOSIZE KMC11
710 :SW07=1 USE CURRENT KMC11 PARAMETERS
711 :SW00=1 INPUT NEW KMC11 PARAMETERS
712 :PRESS START
713 :PROGRAM WILL TYPE "CZKCF BIT STUFF MD L TSTS"
714 :PROGRAM WILL TYPE STATUS MAP
715 :PROGRAM WILL TYPE "R" TO INDICATE THAT TESTING HAS STARTED
716 :AT THE END OF A PASS, PROGRAM WILL TYPE PASS COMPLETE MESSAGE
717 :AND THEN RESUME TESTING
718 :SUBSEQUENT RESTARTS WILL NOT TYPE PROGRAM TITLE
719
720 .SBTTL BASIC DEFINITIONS
721
722 :*INITIAL ADDRESS OF THE STACK POINTER *** 1200 ***
723 STACK= 1200
724 001200 .EQUIV EMT,ERROR ;;BASIC DEFINITION OF ERROR CALL
725 .EQUIV IOT,SCOPE ;;BASIC DEFINITION OF SCOPE CALL
726
727 :*MISCELLANEOUS DEFINITIONS
728 HT= 11 ;;CODE FOR HORIZONTAL TAB
729 LF= 12 ;;CODE FOR LINE FEED
730 CR= 15 ;;CODE FOR CARRIAGE RETURN
731 CRLF= 200 ;;CODE FOR CARRIAGE RETURN-LINE FEED
732 PS= 177776 ;;PROCESSOR STATUS WORD
733 .EQUIV PS,PSW
734 STKLMT= 177774 ;;STACK LIMIT REGISTER
735 PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
736 DSWR= 177570 ;;HARDWARE SWITCH REGISTER
737 DDISP= 177570 ;;HARDWARE DISPLAY REGISTER
738
739 :*GENERAL PURPOSE REGISTER DEFINITIONS
740 R0= X0 ;;GENERAL REGISTER
741 R1= X1 ;;GENERAL REGISTER
742 R2= X2 ;;GENERAL REGISTER
```



CZKCF MACY11 30A(1052) 08-JUL-80 08:27 PAGE 18  
 CZKCF.P11 08-JUL-80 08:24 BASIC DEFINITIONS

SEQ 0017

|     |        |     |    |                    |
|-----|--------|-----|----|--------------------|
| 743 | 000003 | R3= | X3 | ::GENERAL REGISTER |
| 744 | 000004 | R4= | X4 | ::GENERAL REGISTER |
| 745 | 000005 | R5= | X5 | ::GENERAL REGISTER |
| 746 | 000006 | R6= | X6 | ::GENERAL REGISTER |
| 747 | 000007 | R7= | X7 | ::GENERAL REGISTER |
| 748 | 000006 | SP= | X6 | ::STACK POINTER    |
| 749 | 000007 | PC= | X7 | ::PROGRAM COUNTER  |

750  
 751 ;\*PRIORITY LEVEL DEFINITIONS  
 752 000000 PPO= 0 ::PRIORITY LEVEL 0  
 753 000040 PR1= 40 ::PRIORITY LEVEL 1  
 754 000100 PR2= 100 ::PRIORITY LEVEL 2  
 755 000140 PR3= 140 ::PRIORITY LEVEL 3  
 756 000200 PR4= 200 ::PRIORITY LEVEL 4  
 757 000240 PR5= 240 ::PRIORITY LEVEL 5  
 758 000300 PR6= 300 ::PRIORITY LEVEL 6  
 759 000340 PR7= 340 ::PRIORITY LEVEL 7

760  
 761 ;\*'SWITCH REGISTER' SWITCH DEFINITIONS  
 762 100000 SW15= 100000  
 763 040000 SW14= 40000  
 764 020000 SW13= 20000  
 765 010000 SW12= 10000  
 766 004000 SW11= 4000  
 767 002000 SW10= 2000  
 768 001000 SW09= 1000  
 769 000400 SW08= 400  
 770 000200 SW07= 200  
 771 000100 SW06= 100  
 772 000040 SW05= 40  
 773 000020 SW04= 20  
 774 000010 SW03= 10  
 775 000004 SW02= 4  
 776 000002 SW01= 2  
 777 000001 SW00= 1  
 778 .EQUIV SW09,SW9  
 779 .EQUIV SW08,SW8  
 780 .EQUIV SW07,SW7  
 781 .EQUIV SW06,SW6  
 782 .EQUIV SW05,SW5  
 783 .EQUIV SW04,SW4  
 784 .EQUIV SW03,SW3  
 785 .EQUIV SW02,SW2  
 786 .EQUIV SW01,SW1  
 787 .EQUIV SW00,SW0

788  
 789 ;\*DATA BIT DEFINITIONS (BIT00 TO BIT15)  
 790 100000 BIT15= 100000  
 791 040000 BIT14= 40000  
 792 020000 BIT13= 20000  
 793 010000 BIT12= 10000  
 794 004000 BIT11= 4000  
 795 002000 BIT10= 2000  
 796 001000 BIT09= 1000  
 797 000400 BIT08= 400  
 798 000200 BIT07= 200

CZKCF MACY11 30A(1052) 08-JUL-80 08:27 PAGE 19  
 CZKCF.P11 08-JUL-80 08:24 BASIC DEFINITIONS

SEQ 0018

|     |        |                   |
|-----|--------|-------------------|
| 799 | 000100 | BIT06= 100        |
| 800 | 000040 | BIT05= 40         |
| 801 | 000020 | BIT04= 20         |
| 802 | 000010 | BIT03= 10         |
| 803 | 000004 | BIT02= 4          |
| 804 | 000002 | BIT01= 2          |
| 805 | 000001 | BIT00= 1          |
| 806 |        | .EQUIV BIT09,BIT9 |
| 807 |        | .EQUIV BIT08,BIT8 |
| 808 |        | .EQUIV BIT07,BIT7 |
| 809 |        | .EQUIV BIT06,BIT6 |
| 810 |        | .EQUIV BIT05,BIT5 |
| 811 |        | .EQUIV BIT04,BIT4 |
| 812 |        | .EQUIV BIT03,BIT3 |
| 813 |        | .EQUIV BIT02,BIT2 |
| 814 |        | .EQUIV BIT01,BIT1 |
| 815 |        | .EQUIV BIT00,BIT0 |

|     |        |                                     |                                      |
|-----|--------|-------------------------------------|--------------------------------------|
| 817 |        | ;*BASIC "CPU" TRAP VECTOR ADDRESSES |                                      |
| 818 | 000004 | ERRVEC= 4                           | :: TIME OUT AND OTHER ERRORS         |
| 819 | 000010 | RESVEC= 10                          | :: RESERVED AND ILLEGAL INSTRUCTIONS |
| 820 | 000014 | TBITVEC=14                          | :: "T" BIT                           |
| 821 | 000014 | TRTVEC= 14                          | :: TRACE TRAP                        |
| 822 | 000014 | BPTVEC= 14                          | :: BREAKPOINT TRAP (BPT)             |
| 823 | 000020 | IOTVEC= 20                          | :: INPUT/OUTPUT TRAP (IOT) **SCOPE** |
| 824 | 000024 | PWRVEC= 24                          | :: POWER FAIL                        |
| 825 | 000030 | EMTVEC= 30                          | :: EMULATOR TRAP (EMT) **ERROR**     |
| 826 | 000034 | TRAPVEC=34                          | :: "TRAP" TRAP                       |
| 827 | 000060 | TKVEC= 60                           | :: TTY KEYBOARD VECTOR               |
| 828 | 000064 | TPVEC= 64                           | :: TTY PRINTER VECTOR                |
| 829 | 000240 | PIRQVEC=240                         | :: PROGRAM INTERRUPT REQUEST VECTOR  |

|     |  |                           |
|-----|--|---------------------------|
| 830 |  |                           |
| 831 |  |                           |
| 832 |  |                           |
| 833 |  |                           |
| 834 |  | : INSTRUCTION DEFINITIONS |
| 835 |  | :-----                    |

|     |        |                |                                    |
|-----|--------|----------------|------------------------------------|
| 836 |        |                |                                    |
| 837 | 005746 | PUSH1SP=5746   | : DECREMENT PROCESSOR STACK 1 WORD |
| 838 | 005726 | POP1SP=5726    | : INCREMENT PROCESSOR STACK 1 WORD |
| 839 | 010046 | PUSHRO=10046   | : SAVE R0 ON STACK                 |
| 840 | 012600 | POPRO=12600    | : RESTORE R0 FROM STACK            |
| 841 | 024646 | PUSH2SP=24646  | : DECREMENT STACK TWICE            |
| 842 | 022626 | POP2SP=22626   | : INCREMENT STACK TWICE            |
| 843 |        | .EQUIV EMT,HLT | : BASIC DEFINITION OF ERROR CALL   |
| 844 |        |                |                                    |
| 845 |        |                |                                    |
| 846 |        |                |                                    |

```

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

```

```

:*****
-----
;TRAPCATCAER FOR ILLEGAL INTERRUPTS
;THE STANDARD "TRAP CATCHER" IS PLACED
;BETWEEN ADDRESS 0 TO ADDRESS 776.
;IT LOOKS LIKE "PC+2 HALT".
-----
:*****
.=0
000000 000000 000000
      .WORD 0,0
;STANDARD INTERRUPT VECTORS
-----
.=20
000020 004134      $SCOPE      ; SCOPE LOOP HANDLER.
000022 000340      PR7          ; SERVICE AT LEVEL 7.
000024 007126      $PWRDN      ;POWER FAIL HANDLER
000026 000340      PR7          ;SERVICE AT LEVEL 7
000030 006512      $ERROR      ;ERROR HANDLER
000032 000340      PR7          ;SERVICE AT LEVEL 7
000034 006414      $TRAP      ;GENERAL HANDLER DISPATCH SERVICE
000036 000340      PR7          ;SERVICE AT LEVEL 7
.SBTTL ACT11 HOOKS
:*****
;HOOKS REQUIRED BY ACT11
      $SVPC=.          ;SAVE PC
      .=46             ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
000046 004070      $ENDAD      ;;2)SET LOC.52 TO ZERO
      .=52             ;; RESTORE PC
000052 000000      .WORD 0
      .=$SVPC
.=174
000174 000000      DISPREG:0      ;SOFTWARE DISPLAY REGISTER
000176 000000      SWREG: 0      ;SOFTWARE SWITCH REGISTER
.=200
000200 000137 002402      JMP      .START      ;GO TO START OF PROGRAM
.=1000
001000 005200 055103 041513      MTITLE: .ASCII <200><12>/CZKCF/<200>
(2) 001010 044502 020124 052123      .ASCII /BIT STUFF MD L TSTS/<200>
(2)
177570      DSWR      =      177570
177570      DDISP     =      177570

```

894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949

001200  
001200 000000  
001202 000  
001203 000  
001204 000000  
001206 000000  
001210 000000  
001212 000000  
001214 000  
001215 001  
001216 000000  
001220 000000  
001222 000000  
001224 000000  
001226 000000  
001230 000000  
001232 000000  
001234 000  
001235 000  
001236 000000  
001240 177570  
001242 177570  
001244 177560  
001246 177562  
001250 177564  
001252 177566  
001254 000  
001255 002  
001256 012  
001257 000  
001260 000000  
001262 000000  
001264 000000  
001266 000000  
001270 000000  
001272 000000  
001274 000000  
001276 000000  
001300 000000  
001302 000000  
001304 000000  
001306 000000  
001310 000000  
001312 077  
001313 015  
001314 000012

.SBTTL COMMON TAGS

\*\*\*\*\*  
\*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS  
\*USED IN THE PROGRAM.

.=1200

SCMTAG: .WORD 0  
\$TSTNM: .BYTE 0  
\$ERFLG: .BYTE 0  
\$ICNT: .WORD 0  
\$LPADR: .WORD 0  
\$LPERR: .WORD 0  
\$ERTTL: .WORD 0  
\$ITEMB: .BYTE 0  
\$ERMAX: .BYTE 1  
\$ERRPC: .WORD 0  
\$GDADR: .WORD 0  
\$BDADR: .WORD 0  
\$GDDAT: .WORD 0  
\$BDDAT: .WORD 0  
\$AUTOB: .BYTE 0  
\$INTAG: .BYTE 0  
\$SWR: .WORD DSWR  
\$DISPLAY: .WORD DDISP  
\$TKS: 177560  
\$TKB: 177562  
\$TPS: 177564  
\$TPB: 177566  
\$NULL: .BYTE 0  
\$FILLS: .BYTE 2  
\$FILLC: .BYTE 12  
\$TPFLG: .BYTE 0  
\$REGAD: .WORD 0  
\$REG0: .WORD 0  
\$REG1: .WORD 0  
\$REG2: .WORD 0  
\$REG3: .WORD 0  
\$REG4: .WORD 0  
\$REG5: .WORD 0  
\$TMP0: .WORD 0  
\$TMP1: .WORD 0  
\$TMP2: .WORD 0  
\$TMP3: .WORD 0  
\$TMP4: .WORD 0  
\$TIMES: 0  
\$QUES: .ASCII /?/  
\$CRLF: .ASCII <15>  
\$LF: .ASCII <12>

;;START OF COMMON TAGS  
;;CONTAINS THE TEST NUMBER  
;;CONTAINS ERROR FLAG  
;;CONTAINS SUBTEST ITERATION COUNT  
;;CONTAINS SCOPE LOOP ADDRESS  
;;CONTAINS SCOPE RETURN FOR ERRORS  
;;CONTAINS TOTAL ERRORS DETECTED  
;;CONTAINS ITEM CONTROL BYTE  
;;CONTAINS MAX. ERRORS PER TEST  
;;CONTAINS PC OF LAST ERROR INSTRUCTION  
;;CONTAINS ADDRESS OF 'GOOD' DATA  
;;CONTAINS ADDRESS OF 'BAD' DATA  
;;CONTAINS 'GOOD' DATA  
;;CONTAINS 'BAD' DATA  
;;RESERVED--NOT TO BE USED  
;;AUTOMATIC MODE INDICATOR  
;;INTERRUPT MODE INDICATOR  
;;ADDRESS OF SWITCH REGISTER  
;;ADDRESS OF DISPLAY REGISTER  
;;TTY KBD STATUS  
;;TTY KBD BUFFER  
;;TTY PRINTER STATUS REG. ADDRESS  
;;TTY PRINTER BUFFER REG. ADDRESS  
;;CONTAINS NULL CHARACTER FOR FILLS  
;;CONTAINS # OF FILLER CHARACTERS REQUIRED  
;;INSERT FILL CHARS. AFTER A 'LINE FEED'  
;;'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)  
;;CONTAINS THE ADDRESS FROM  
;;WHICH (\$REG0) WAS OBTAINED  
;;CONTAINS ((\$REGAD)+0)  
;;CONTAINS ((\$REGAD)+2)  
;;CONTAINS ((\$REGAD)+4)  
;;CONTAINS ((\$REGAD)+6)  
;;CONTAINS ((\$REGAD)+10)  
;;CONTAINS ((\$REGAD)+12)  
;;USER DEFINED  
;;USER DEFINED  
;;USER DEFINED  
;;USER DEFINED  
;;USER DEFINED  
;;MAX. NUMBER OF ITERATIONS  
;;QUESTION MARK  
;;CARRIAGE RETURN  
;;LINE FEED

\*\*\*\*\*  
.SBTTL APT MAILBOX-ETABLE

950  
951  
952  
953 001316  
954 001316 000000  
955 0C1320 000000  
956 001322 000000  
957 001324 000000  
958 001326 000000  
959 001330 000000  
960 001332 000000  
961 001334 000000  
962 001336  
963 001336 002  
964 001337 000  
965 001340 000000  
966 001342 000000  
967 001344 000000  
968  
969  
970  
971  
972  
973  
974 001346 000  
975 001347 000  
976  
977  
978  
979  
980 001350 000000  
981  
982 001352 000  
983 001353 000  
984 001354 000000  
985 001356 000  
986 001357 000  
987 001360 000000  
988 001362 000  
989 001363 000  
990 001364 000000  
991 001366 000000  
992 001370 000000  
993 001372 000000  
994 001374 000000  
995 001376 000000  
996 001400 000000  
997 001402 000000  
998 001404 000000  
999 001406 000000  
1000 001410 000000  
1001 001412 000000  
1002 001414 000000  
1003 001416 000000  
1004 001420 000000  
1005 001422 000000

\*\*\*\*\*

```

.EVEN
$MAIL:          ;;APT MAILBOX
$MSGTY: .WORD   AMSGTY ;;MESSAGE TYPE CODE
$FATAL: .WORD   AFATAL ;;FATAL ERROR NUMBER
$TESTN: .WORD   ATESTN ;;TEST NUMBER
$PASS:   .WORD   APASS  ;;PASS COUNT
$DEVCT:  .WORD   ADEVCT ;;DEVICE COUNT
$UNIT:   .WORD   AUNIT  ;;I/O UNIT NUMBER
$MSGAD:  .WORD   AMSGAD ;;MESSAGE ADDRESS
$MSGLG:  .WORD   AMSGLG ;;MESSAGE LENGTH
$ETABLE:          ;;APT ENVIRONMENT TABLE
$ENV:     .BYTE   AENV   ;;ENVIRONMENT BYTE
$ENVM:    .BYTE   AENVM  ;;ENVIRONMENT MODE BITS
$$SWREG: .WORD   ASWREG ;;APT SWITCH REGISTER
$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
$MAMS1:  .BYTE   AMAMS1 ;;HIGH ADDRESS,M.S. BYTE
$MTYP1:  .BYTE   AMTYP1 ;;MEM. TYPE,BLK#1
          ;;MEM.TYPE BYTE -- (HIGH BYTE)
          ;;900 NSEC CORE=001
          ;;300 NSEC BIPOLAR=002
          ;;500 NSEC MOS=003
$MADR1:  .WORD   AMADR1 ;;HIGH ADDRESS,BLK#1
          ;;MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
$MAMS2:  .BYTE   AMAMS2 ;;HIGH ADDRESS,M.S. BYTE
$MTYP2:  .BYTE   AMTYP2 ;;MEM.TYPE,BLK#2
$MADR2:  .WORD   AMADR2 ;;MEM.LAST ADDRESS,BLK#2
$MAMS3:  .BYTE   AMAMS3 ;;HIGH ADDRESS,M.S.BYTE
$MTYP3:  .BYTE   AMTYP3 ;;MEM.TYPE,BLK#3
$MADR3:  .WORD   AMADR3 ;;MEM.LAST ADDRESS,BLK#3
$MAMS4:  .BYTE   AMAMS4 ;;HIGH ADDRESS,M.S.BYTE
$MTYP4:  .BYTE   AMTYP4 ;;MEM.TYPE,BLK#4
$MADR4:  .WORD   AMADR4 ;;MEM.LAST ADDRESS,BLK#4
$VECT1:  .WORD   AVECT1 ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
$VECT2:  .WORD   AVECT2 ;;INTERRUPT VECTOR#2BUS PRIORITY#2
$BASE:   .WORD   ABASE  ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
$DEVM:   .WORD   ADEVM  ;;DEVICE MAP
$CDW1:   .WORD   ACDW1  ;;CONTROLLER DESCRIPTION WORD#1
$CDW2:   .WORD   ACDW2  ;;CONTROLLER DESCRIPTION WORD#2
$DDW0:   .WORD   ADDW0  ;;DEVICE DESCRIPTOR WORD#0
$DDW1:   .WORD   ADDW1  ;;DEVICE DESCRIPTOR WORD#1
$DDW2:   .WORD   ADDW2  ;;DEVICE DESCRIPTOR WORD#2
$DDW3:   .WORD   ADDW3  ;;DEVICE DESCRIPTOR WORD#3
$DDW4:   .WORD   ADDW4  ;;DEVICE DESCRIPTOR WORD#4
$DDW5:   .WORD   ADDW5  ;;DEVICE DESCRIPTOR WORD#5
$DDW6:   .WORD   ADDW6  ;;DEVICE DESCRIPTOR WORD#6
$DDW7:   .WORD   ADDW7  ;;DEVICE DESCRIPTOR WORD#7
$DDW8:   .WORD   ADDW8  ;;DEVICE DESCRIPTOR WORD#8

```

1006 001424 000000  
 1007 001426 000000  
 1008 001430 000000  
 1009 001432 000000  
 1010 001434 000000  
 1011 001436 000000  
 1012 001440 000000  
 1013  
 1014  
 1015 001442  
 1016  
 1017  
 1018  
 1019  
 1020 001442 000000  
 1021 001444 000000  
 1022  
 1023  
 1024  
 1025 001446 000000  
 1026 001450 000000  
 1027 001452 000000  
 1028 001454 000000  
 1029 001456 000000  
 1030 001460 000000  
 1031 001462 000000  
 1032 001464 000001  
 1033 001466 000000  
 1034 001470 000001  
 1035 001472 000001  
 1036 001474 000001  
 1037 001476 000001  
 1038 001500 000000  
 1039  
 1040 001502 002072  
 1041 001504 002276  
 1042  
 1043  
 1044  
 1045 001506 000  
 1046 001510 001510  
 1047 001511 000  
 1048 001511 000  
 1049  
 1050

\$DDW9: .WORD ADDW9 ;;DEVICE DESCRIPTOR WORD#9  
 \$DDW10: .WORD ADDW10 ;;DEVICE DESCRIPTOR WORD#10  
 \$DDW11: .WORD ADDW11 ;;DEVICE DESCRIPTOR WORD#11  
 \$DDW12: .WORD ADDW12 ;;DEVICE DESCRIPTOR WORD#12  
 \$DDW13: .WORD ADDW13 ;;DEVICE DESCRIPTOR WORD#13  
 \$DDW14: .WORD ADDW14 ;;DEVICE DESCRIPTOR WORD#14  
 \$DDW15: .WORD ADDW15 ;;DEVICE DESCRIPTOR WORD#15

\$ETEND:

PROGRAM CONTROL PARAMETERS

-----  
 NEXT: .WORD 0 ; ADDRSS OF NEXT TEST TO BE EXECUTED  
 LOCK: .WORD 0 ; ADDRESS FOR LOCK CURRENT DATA

PROGRAM VARIABLES

-----  
 STRTSW: .WORD 0 ; SWITCHES AT START OF PROGRAM  
 STAT: .WORD 0 ; KM STATUS WORD STORAGE  
 CLKX: .WORD 0 ;  
 MASKX: .WORD 0 ;  
 SAVSP: .WORD 0 ; STACK POINTER STORAGE  
 SAVPC: .WORD 0 ; PROGRAM COUNTER STORAGE  
 ZERO: .WORD 0 ;  
 ONE: .WORD 1 ;  
 MEMLIM: .WORD 0 ; HIGHEST LOCATION FOR NPR'S  
 KMACTV: .BLKW 1 ; KMC11 SELECTED ACTIVE  
 KMNUM: .BLKW 1 ; OCTAL NUMBER OF KMC11'S  
 SAVACT: .BLKW 1 ; ORIGINAL ACTIVE DEVICES.  
 SAVNUM: .BLKW 1 ; WORKABLE NUMBER.  
 RUN: .WORD 0 ; POINTER TO RUNNING DEVICES  
 .EVEN  
 CREAM: .WORD KM.MAP-6 ; TABLE POINTER  
 MILK: .WORD CNT.MAP-4 ; TABLE POINTER

PROGRAM CONTROL FLAGS

-----  
 INIFLG: .BYTE 0 ; PROGRAM INITIALIZING FLAG  
 .EVEN  
 LOKFLG: .BYTE 0 ; LOCK ON CURRENT TEST FLAG  
 QV.FLG: .BYTE 0 ; QUICK VERIFY FLAG  
 ; ON FIRST PASS OF EACH KMC11 ITERATIONS WILL BE SJPPRES  
 .EVEN

```

1051 .SBTTL ERROR POINTER TABLE
1052
1053 ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
1054 ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FGUND IN
1055 ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
1056 ;*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
1057 ;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
1058
1059 ;* EM ;;POINTS TO THE ERROR MESSAGE
1060 ;* DH ;;POINTS TO THE DATA HEADER
1061 ;* DT ;;POINTS TO THE DATA
1062 ;* DF ;;POINTS TO THE DATA FORMAT
1063
1064
1065 001512 $ERRTB:
1066 .EVEN
1067 ;* DF ;; DOES NOT APPLY IN THIS DIAGNOSTIC.
1068 001512 000000 0
1069 001514 000000 0
1070 001516 000000 0
1071 001520 032062 EM1
1072 001522 033224 DH1 ; ERROR 1
1073 001524 033550 DT1
1074 001526 032120 EM2
1075 001530 033245 DH2 ; ERROR 2
1076 001532 033562 DT2
1077 001534 032163 EM3
1078 001536 033303 DH3 ; ERROR 3
1079 001540 033600 DT3
1080 001542 032227 EM4
1081 001544 033341 DH4 ; ERROR 4
1082 001546 033612 DT4
1083 001550 032227 EM4
1084 001552 033401 DH5 ; ERROR 5
1085 001554 033624 DT5
1086 001556 032271 EM5
1087 001560 033303 DH3 ; ERROR 6
1088 001562 033642 DT6
1089 001564 032321 EM6
1090 001566 033433 DH6 ; ERROR 7
1091 001570 033624 DT5
1092 001572 032340 EM7
1093 001574 000000 0 ; ERROR 10
1094 001576 000000 0
1095 001600 032365 EM10
1096 001602 000000 0 ; ERROR 11
1097 001604 000000 0
1098 001606 032570 EM11
1099 001610 033303 DH3 ; ERROR 12
1100 001612 033600 DT3
1101 001614 032617 EM12
1102 001616 033433 DH6 ; ERROR 13
1103 001620 033624 DT5
1104 001622 032643 EM13
1105 001624 033433 DH6 ; ERROR 14
1106 001626 033624 DT5
    
```

|      |        |        |      |            |
|------|--------|--------|------|------------|
| 1107 | 001630 | 032703 | EM14 |            |
| 1108 | 001632 | 033303 | DH3  | ; ERROR 15 |
| 1109 | 001634 | 033642 | DT6  |            |
| 1110 | 001636 | 032753 | EM15 |            |
| 1111 | 001640 | 033341 | DH4  | ; ERROR 16 |
| 1112 | 001642 | 033612 | DT4  |            |
| 1113 | 001644 | 032774 | EM16 |            |
| 1114 | 001646 | 000000 | 0    | ; ERROR 17 |
| 1115 | 001650 | 000000 | 0    |            |
| 1116 | 001652 | 033910 | EM17 |            |
| 1117 | 001654 | 000000 | 0    | ; ERROR 20 |
| 1118 | 001656 | 000000 | 0    |            |
| 1119 | 001660 | 032570 | EM11 |            |
| 1120 | 001662 | 033433 | DH6  | ; ERROR 21 |
| 1121 | 001664 | 033624 | DT5  |            |
| 1122 | 001666 | 033054 | EM20 |            |
| 1123 | 001670 | 033303 | DH3  | ; ERROR 22 |
| 1124 | 001672 | 033660 | DT7  |            |
| 1125 | 001674 | 033075 | EM21 |            |
| 1126 | 001676 | 033303 | DH3  | ; ERROR 23 |
| 1127 | 001700 | 033600 | DT3  |            |
| 1128 | 001702 | 033054 | EM20 |            |
| 1129 | 001704 | 000000 | 0    | ; ERROR 24 |
| 1130 | 001706 | 000000 | 0    |            |
| 1131 | 001710 | 033112 | EM22 |            |
| 1132 | 001712 | 033303 | DH3  | ; ERROR 25 |
| 1133 | 001714 | 033600 | DT3  |            |
| 1134 |        | 002034 |      |            |

. =2034  
.SBTTL APT PARAMETER BLOCK

```

: *****
:SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
: *****
      . $X =.      ;;SAVE CURRENT LOCATION
      . =24      ;;SET POWER FAIL TO POINT TO START OF PROGRAM
      200      ;;FOR APT START UP
      . =44      ;;POINT TO APT INDIRECT ADDRESS PNTR.
      $APTHDR ;;POINT TO APT HEADER BLOCK
      . = $X      ;;RESET LOCATION COUNTER
: *****
:SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
:INTERFACE SPEC.

```

|      |        |        |                |   |
|------|--------|--------|----------------|---|
| 1140 |        | 002034 |                |   |
| 1141 |        | 000024 |                |   |
| 1142 | 000024 | 000200 |                |   |
| 1143 |        | 000044 |                |   |
| 1144 | 000044 | 002034 |                |   |
| 1145 |        | 002034 |                |   |
| 1146 |        |        |                |   |
| 1147 |        |        |                |   |
| 1148 |        |        |                |   |
| 1149 |        |        |                |   |
| 1150 | 002034 |        | \$APTHD:       |   |
| 1151 | 002034 | 000000 | \$HIBTS: .WORD | 0 ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.                           |
| 1152 | 002036 | 001316 | \$MBADR: .WORD | \$MAIL ;;ADDRESS OF APT MAILBOX (BITS 0-15)                         |
| 1153 | 002040 | 000132 | \$STMT: .WORD  | 90. ;;RUN TIM OF LONGEST TEST                                       |
| 1154 | 002042 | 000137 | \$PASTM: .WORD | 95. ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)        |
| 1155 | 002044 | 000137 | \$UNITM: .WORD | 95. ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT |
| 1156 | 002046 | 000052 | .WORD          | SETEND-\$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)                      |
| 1157 |        |        |                |   |



```

1158
1159 ;KMC11 CONTROL INDICATORS FOR CURRENT KMC11 UNDER TEST
1160 ;-----
1161
1162 002050 000000 STAT1: 0
1163 002052 000000 STAT2: 0
1164 002054 000000 STAT3: 0
1165
1166 ;KMC11 VECTOR AND REGISTER INDIRECT POINTERS
1167 ;-----
1168
1169 002056 000000 KMRVEC: 0 ; POINTER TO KMC11 RECEIVER INTERRUPT VECTOR
1170 002060 000000 KMRLVL: 0 ; POINTER TO KMC11 RECEIVER INTERRUPT SERVICE PS
1171 002062 000000 KMTVEC: 0 ; POINTER TO KMC11 TRANSMITTER INTERRUPT VECTOR
1172 002064 000000 KMTLVL: 0 ; POINTER TO KMC11 TRANSMITTER INTERRUPT SERVICE PS
1173 002066 000000 KMCSR: 0 ; POINTER TO KMC11 CONTROL STATUS REGISTER
1174 002070 000000 KMCSRH: 0 ; POINTER TO KMC11 CONTROL STATUS REGISTER HIGH BYTE.
1175 002072 000000 KMCTL: 0 ; POINTER TO KMC11 CONTROL OUT REGISTER
1176 002074 000000 KMP04: 0 ; POINTER TO KMC11 PORT REGISTER(SEL 4)
1177 002076 000000 KMP06: 0 ; POINTER TO KMC11 PORT REGISTER(SEL 6)
1178
1179 ;TEMP STORAGE
1180 ;-----
1181
1182 ;TEMP: 0
1183 ;.=.+40
1184
1185 ;KMC11 STATUS TABLE AND ADDRESS ASSIGNMENTS
1186 ;-----
1187
1188 002100 . =2100
1189 002100 KM.MAP:
1190 002100 000001 KMCRO0: .BLKW 1 ; CONTROL STATUS REGISTER FOR KMC11 NUMBER 00
1191 002102 000001 KMS100: .BLKW 1 ; VECTOR FOR KMC11 NUMBER 00
1192 002104 000001 KMS200: .BLKW 1 ; DDCMP LINE# FOR KMC11 NUMBER 00
1193 002106 000001 KMS300: .BLKW 1 ; 3RD STATUS WORD
1194
1195 002110 000001 KMCRO1: .BLKW 1 ; CONTROL STATUS REGISTER FOR KMC11 NUMBER 01
1196 002112 000001 KMS101: .BLKW 1 ; VECTOR FOR KMC11 NUMBER 01
1197 002114 000001 KMS201: .BLKW 1 ; DDCMP LINE# FOR KMC11 NUMBER 01
1198 002116 000001 KMS301: .BLKW 1 ; 3RD STATUS WORD
1199
1200 002120 000001 KMCRO2: .BLKW 1 ; CONTROL STATUS REGISTER FOR KMC11 NUMBER 02
1201 002122 000001 KMS102: .BLKW 1 ; VECTOR FOR KMC11 NUMBER 02
1202 002124 000001 KMS202: .BLKW 1 ; DDCMP LINE# FOR KMC11 NUMBER 02
1203 002126 000001 KMS302: .BLKW 1 ; 3RD STATUS WORD
1204
1205 002130 000001 KMCRO3: .BLKW 1 ; CONTROL STATUS REGISTER FOR KMC11 NUMBER 03
1206 002132 000001 KMS103: .BLKW 1 ; VECTOR FOR KMC11 NUMBER 03
1207 002134 000001 KMS203: .BLKW 1 ; DDCMP LINE# FOR KMC11 NUMBER 03
1208 002136 000001 KMS303: .BLKW 1 ; 3RD STATUS WORD
1209
1210 002140 000001 KMCRO4: .BLKW 1 ; CONTROL STATUS REGISTER FOR KMC11 NUMBER 04
1211 002142 000001 KMS104: .BLKW 1 ; VECTOR FOR KMC11 NUMBER 04
1212 002144 000001 KMS204: .BLKW 1 ; DDCMP LINE# FOR KMC11 NUMBER 04
1213 002146 000001 KMS304: .BLKW 1 ; 3RD STATUS WORD

```

|      |        |        |               |   |  |
|------|--------|--------|---------------|---|--|
| 1214 |        |        |               |   |  |
| 1215 | 002150 | 000001 | KMCR05: .BLKW | 1 | :CONTROL STATUS REGISTER FOR KMC11 NUMBER 05 |
| 1216 | 002152 | 000001 | KMS105: .BLKW | 1 | :VECTOR FOR KMC11 NUMBER 05                  |
| 1217 | 002154 | 000001 | KMS205: .BLKW | 1 | :DDCMP LINE# FOR KMC11 NUMBER 05             |
| 1218 | 002156 | 000001 | KMS305: .BLKW | 1 | :3RD STATUS WORD                             |
| 1219 |        |        |               |   |  |
| 1220 | 002160 | 000001 | KMCR06: .BLKW | 1 | :CONTROL STATUS REGISTER FOR KMC11 NUMBER 06 |
| 1221 | 002162 | 000001 | KMS106: .BLKW | 1 | :VECTOR FOR KMC11 NUMBER 06                  |
| 1222 | 002164 | 000001 | KMS206: .BLKW | 1 | :DDCMP LINE# FOR KMC11 NUMBER 06             |
| 1223 | 002166 | 000001 | KMS306: .BLKW | 1 | :3RD STATUS WORD                             |
| 1224 |        |        |               |   |  |
| 1225 | 002170 | 000001 | KMCR07: .BLKW | 1 | :CONTROL STATUS REGISTER FOR KMC11 NUMBER 07 |
| 1226 | 002172 | 000001 | KMS107: .BLKW | 1 | :VECTOR FOR KMC11 NUMBER 07                  |
| 1227 | 002174 | 000001 | KMS207: .BLKW | 1 | :DDCMP LINE# FOR KMC11 NUMBER 07             |
| 1228 | 002176 | 000001 | KMS307: .BLKW | 1 | :3RD STATUS WORD                             |
| 1229 |        |        |               |   |  |
| 1230 | 002200 | 000001 | KMCR10: .BLKW | 1 | :CONTROL STATUS REGISTER FOR KMC11 NUMBER 10 |
| 1231 | 002202 | 000001 | KMS110: .BLKW | 1 | :VECTOR FOR KMC11 NUMBER 10                  |
| 1232 | 002204 | 000001 | KMS210: .BLKW | 1 | :DDCMP LINE# FOR KMC11 NUMBER 10             |
| 1233 | 002206 | 000001 | KMS310: .BLKW | 1 | :3RD STATUS WORD                             |
| 1234 |        |        |               |   |  |
| 1235 | 002210 | 000001 | KMCR11: .BLKW | 1 | :CONTROL STATUS REGISTER FOR KMC11 NUMBER 11 |
| 1236 | 002212 | 000001 | KMS111: .BLKW | 1 | :VECTOR FOR KMC11 NUMBER 11                  |
| 1237 | 002214 | 000001 | KMS211: .BLKW | 1 | :DDCMP LINE# FOR KMC11 NUMBER 11             |
| 1238 | 002216 | 000001 | KMS311: .BLKW | 1 | :3RD STATUS WORD                             |
| 1239 |        |        |               |   |  |
| 1240 | 002220 | 000001 | KMCR12: .BLKW | 1 | :CONTROL STATUS REGISTER FOR KMC11 NUMBER 12 |
| 1241 | 002222 | 000001 | KMS112: .BLKW | 1 | :VECTOR FOR KMC11 NUMBER 12                  |
| 1242 | 002224 | 000001 | KMS212: .BLKW | 1 | :DDCMP LINE# FOR KMC11 NUMBER 12             |
| 1243 | 002226 | 000001 | KMS312: .BLKW | 1 | :3RD STATUS WORD                             |
| 1244 |        |        |               |   |  |
| 1245 | 002230 | 000001 | KMCR13: .BLKW | 1 | :CONTROL STATUS REGISTER FOR KMC11 NUMBER 13 |
| 1246 | 002232 | 000001 | KMS113: .BLKW | 1 | :VECTOR FOR KMC11 NUMBER 13                  |
| 1247 | 002234 | 000001 | KMS213: .BLKW | 1 | :DDCMP LINE# FOR KMC11 NUMBER 13             |
| 1248 | 002236 | 000001 | KMS313: .BLKW | 1 | :3RD STATUS WORD                             |
| 1249 |        |        |               |   |  |
| 1250 | 002240 | 000001 | KMCR14: .BLKW | 1 | :CONTROL STATUS REGISTER FOR KMC11 NUMBER 14 |
| 1251 | 002242 | 000001 | KMS114: .BLKW | 1 | :VECTOR FOR KMC11 NUMBER 14                  |
| 1252 | 002244 | 000001 | KMS214: .BLKW | 1 | :DDCMP LINE# FOR KMC11 NUMBER 14             |
| 1253 | 002246 | 000001 | KMS314: .BLKW | 1 | :3RD STATUS WORD                             |
| 1254 |        |        |               |   |  |
| 1255 | 002250 | 000001 | KMCR15: .BLKW | 1 | :CONTROL STATUS REGISTER FOR KMC11 NUMBER 15 |
| 1256 | 002252 | 000001 | KMS115: .BLKW | 1 | :VECTOR FOR KMC11 NUMBER 15                  |
| 1257 | 002254 | 000001 | KMS215: .BLKW | 1 | :DDCMP LINE# FOR KMC11 NUMBER 15             |
| 1258 | 002256 | 000001 | KMS315: .BLKW | 1 | :3RD STATUS WORD                             |
| 1259 |        |        |               |   |  |
| 1260 | 002260 | 000001 | KMCR16: .BLKW | 1 | :CONTROL STATUS REGISTER FOR KMC11 NUMBER 16 |
| 1261 | 002262 | 000001 | KMS116: .BLKW | 1 | :VECTOR FOR KMC11 NUMBER 16                  |
| 1262 | 002264 | 000001 | KMS216: .BLKW | 1 | :DDCMP LINE# FOR KMC11 NUMBER 16             |
| 1263 | 002266 | 000001 | KMS316: .BLKW | 1 | :3RD STATUS WORD                             |
| 1264 |        |        |               |   |  |
| 1265 | 002270 | 000001 | KMCR17: .BLKW | 1 | :CONTROL STATUS REGISTER FOR KMC11 NUMBER 17 |
| 1266 | 002272 | 000001 | KMS117: .BLKW | 1 | :VECTOR FOR KMC11 NUMBER 17                  |
| 1267 | 002274 | 000001 | KMS217: .BLKW | 1 | :DDCMP LINE# FOR KMC11 NUMBER 17             |
| 1268 | 002276 | 000001 | KMS317: .BLKW | 1 | :3RD STATUS WORD                             |
| 1269 |        |        |               |   |  |

CZKCF MACY11 30A(1052) 08-JUL-80 08:27 PAGE 28  
CZKCF.P11 08-JUL-80 08:24 APT PARAMETER BLOCK

B 3

SEQ 0027

1270 002300 000000

KM.END: 000000

```

1271
1272 ;KMC11 PASS COUNT AND ERROR COUNT TABLE
1273 ;-----
1274
1275 CNT.MAP:
1276 002302 000000 PACT00: 0 ;PASS COUNT FOR KMC11 NUMBER 00
1277 002304 000000 ERCT00: 0 ;ERROR COUNT FOR KMC11 NUMBER 00
1278
1279 002306 000000 PACT01: 0 ;PASS COUNT FOR KMC11 NUMBER 01
1280 002310 000000 ERCT01: 0 ;ERROR COUNT FOR KMC11 NUMBER 01
1281
1282 002312 000000 PACT02: 0 ;PASS COUNT FOR KMC11 NUMBER 02
1283 002314 000000 ERCT02: 0 ;ERROR COUNT FOR KMC11 NUMBER 02
1284
1285 002316 000000 PACT03: 0 ;PASS COUNT FOR KMC11 NUMBER 03
1286 002320 000000 ERCT03: 0 ;ERROR COUNT FOR KMC11 NUMBER 03
1287
1288 002322 000000 PACT04: 0 ;PASS COUNT FOR KMC11 NUMBER 04
1289 002324 000000 ERCT04: 0 ;ERROR COUNT FOR KMC11 NUMBER 04
1290
1291 002326 000000 PACT05: 0 ;PASS COUNT FOR KMC11 NUMBER 05
1292 002330 000000 ERCT05: 0 ;ERROR COUNT FOR KMC11 NUMBER 05
1293
1294 002332 000000 PACT06: 0 ;PASS COUNT FOR KMC11 NUMBER 06
1295 002334 000000 ERCT06: 0 ;ERROR COUNT FOR KMC11 NUMBER 06
1296
1297 002336 000000 PACT07: 0 ;PASS COUNT FOR KMC11 NUMBER 07
1298 002340 000000 ERCT07: 0 ;ERROR COUNT FOR KMC11 NUMBER 07
1299
1300 002342 000000 PACT10: 0 ;PASS COUNT FOR KMC11 NUMBER 10
1301 002344 000000 ERCT10: 0 ;ERROR COUNT FOR KMC11 NUMBER 10
1302
1303 002346 000000 PACT11: 0 ;PASS COUNT FOR KMC11 NUMBER 11
1304 002350 000000 ERCT11: 0 ;ERROR COUNT FOR KMC11 NUMBER 11
1305
1306 002352 000000 PACT12: 0 ;PASS COUNT FOR KMC11 NUMBER 12
1307 002354 000000 ERCT12: 0 ;ERROR COUNT FOR KMC11 NUMBER 12
1308
1309 002356 000000 PACT13: 0 ;PASS COUNT FOR KMC11 NUMBER 13
1310 002360 000000 ERCT13: 0 ;ERROR COUNT FOR KMC11 NUMBER 13
1311
1312 002362 000000 PACT14: 0 ;PASS COUNT FOR KMC11 NUMBER 14
1313 002364 000000 ERCT14: 0 ;ERROR COUNT FOR KMC11 NUMBER 14
1314
1315 002366 000000 PACT15: 0 ;PASS COUNT FOR KMC11 NUMBER 15
1316 002370 000000 ERCT15: 0 ;ERROR COUNT FOR KMC11 NUMBER 15
1317
1318 002372 000000 PACT16: 0 ;PASS COUNT FOR KMC11 NUMBER 16
1319 002374 000000 ERCT16: 0 ;ERROR COUNT FOR KMC11 NUMBER 16
1320
1321 002376 000000 PACT17: 0 ;PASS COUNT FOR KMC11 NUMBER 17
1322 002400 000000 ERCT17: 0 ;ERROR COUNT FOR KMC11 NUMBER 17
1323

```

1324  
 1325  
 1326  
 1327  
 1328  
 1329

-----  
 FORMAT OF STATUS TABLE  
 -----

| 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |       |       |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------|-------|
| I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | CSR   |       |
| I  | C  | O  | N  | T  | R  | O  | L  | I  | R  | E  | G  | I  | S  | T  | E  | R     |       |
| I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  |       |       |
| I  | *  | I  | *  | I  | *  | I  | *  | I  | *  | I  | *  | I  | *  | I  | *  | I     | STAT1 |
| I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  |       |       |
| I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | STAT2 |       |
| I  | *  | B  | M  | I  | A  | D  | D  | *  | I  | *  | L  | I  | N  | E  | #  | *     |       |
| I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  |       |       |
| I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | *  | I  | *  | I     | STAT3 |
| I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  | I  |       |       |

-----  
 DEFINITION OF FORMAT  
 -----

- CSR: CONTAINS KMC11 CSR ADDRESS
- STAT1: BITS 00-08 IS KMC11 VECTOR ADDRESS  
 BIT14=1 ??? TURNAROUND CONNECTOR IS ON  
 BIT14=0 NO TURNAROUND CONNECTOR  
 BIT13=0 LINE UNIT IS AN MB201  
 BIT13=1 LINE UNIT IS AN MB202  
 BIT12=1 NO LINE UNIT  
 BITS 09-11 IS KMC11 BR PRIORITY LEVEL
- STAT2: LOW BYTE IS SWITCH PAC#1 (DDCMP LINE NUMBER)  
 HIGH BYTE IS SWITCH PAC#2 (BM873 BOOT ADD)
- STAT3: BIT0=1 DO FREE RUNNING TESTS ON KMC  
 (MUST BE SET TO A ONE MANUALLY [PROGRAMS G AND H ONLY])  
 BIT2=0 DMC11-DA (RS232C)  
 BIT2=1 DMC11-FA (V.35)

CZKCF MACY11 30A(1052) 08-JUL-80 08:27 PAGE 31  
CZKCF.P11 08-JUL-80 08:24 APT PARAMETER BLOCK

E 3

SEQ 0030

8

```

1381
1382          :PROGRAM INITIALIZATION
1383          :LOCK OUT INTERRUPTS
1384          :SET UP PROCESSOR STACK
1385          :SET UP POWER FAIL VECTOR
1386          :CLEAR PROGRAM CONTROL FLAGS AND COUNTS
1387          :TYPE TITLE MESSAGE
1388
1389 002402 012737 000340 177776 .START: MOV    #340,PS          ;LOCK OUT INTERRUPTS
1390 002410 012706 001200          MOV    #STACK,SP        ;SET UP STACK
1391 002414 012737 007126 000024 MOV    #SPWRDN,@#24     ;SET UP POWER FAIL VECTOR
1392 002422 013737 001472 001476 MOV    KNUM,SAVNUM      ;SAVE NUMBER OF DEVICES IN SYSTEM.
1393 002430 005037 011544          CLR    SWFLG           ;CLEAR SOFT TIMEOUT FLAG
1394 002434 105037 001203          CLRB  $ERFLG          ;CLEAR ERROR FLAG
1395 002440 105037 001511          CLRB  QV.FLG          ;ZERO QUICK VERIFY FLAG
1396 002444 012737 002070 001502 MOV    #KM.MAP-10,CREAM;GET MAP POINTER.
1397 002452 012737 002276 001504 MOV    #CNT.MAP-4,MILK  ;GET PASS COUNT MAP POINTER
1398 002460 012737 100000 001500 MOV    #BIT15,RUN      ;POINT POINTER TO FIRST DEVICE.
1399 002466 012700 002302          MOV    #CNT.MAP,RO     ;PASS COUNT POINTER TO RO
1400 002472 005020          23$: CLR    (RO)+          ;CLEAR TABLE
1401 002474 022700 002402          CMP    #CNT.MAP+100,RO ;DONE YET?
1402 002500 001374          BNE    23$            ;KEEP GOING
1403 002502 005037 001216          CLR    $ERRPC         ;CLEAR LAST ERROR POINTER
1404 002506 012737 000001 001202 MOV    #1,$STNM        ;SET UP FOR TEST 1
1405 002514 012737 002402 001206 MOV    #.START,$LPADR  ;SET UP FOR POWER FAIL BEFORE
1406          :TESTING STARTS
1407 002522 132737 000001 001336 BITB   #1,$ENV          ; IS IT RUNNING UNDER APT?
1408 002530 001404          BEQ   3$              ; IF NOT CHECK FOR TYPE OF SWITCH REGISTER.
1409 002532 013737 001340 000176 MOV    $SWREG,SWREG    ; LOAD SOFTWARE SWITCH REG.
1410 002540 000423          BR    6$+2           ; GO SET UP SOFTWARE SWITCH REG.
1411 002542 013746 000006          3$: MOV    @#6,-(SP)     ;SAVE CURRENT VECTORS
1412 002546 013746 000004          MOV    @#4,-(SP)     ;
1413 002552 012737 002606 000004 MOV    #6$,@#4        ;SET UP FOR TIMEOUT
1414 002560 012737 177570 001240 MOV    #177570,SWR     ;SET SWR TO HARD SWR ADDRESS
1415 002566 012737 177570 001242 MOV    #177570,DISPLAY ;SET DISPLAY TO HARD SWR ADDRESS
1416 002574 022777 177777 176436 CMP    #-1,@SWR        ;REFERENCE HARDWARE SWITCH REGISTER
1417 002602 001402          BEQ   6$+2           ;IF = -1 USE SOFT SWR ANYWAY
1418 002604 000407          BR    7$              ;IF IT EXISTS AND NOT = -1 USE HARD SWR
1419 002606 022626          6$: CMP    (SP)+,(SP)+ ;ADJUST STACK
1420 002610 012737 000176 001240 MOV    #SWREG,SWR     ;POINTER TO SOFT SWR
1421 002616 012737 000174 001242 MOV    #DISPREG,DISPLAY;POINTER TO SOFT DISPLAY REG
1422 002624 012637 000004          7$: MOV    (SP)+,@#4    ;RESTORE VECTORS
1423 002630 012637 000006          MOV    (SP)+,@#6    ;
1424 002634 105737 001506          TSTB  INIFLG         ;HAS INITIALIZATION BEEN PERFORMED
1425 002640 001006          BNE   20$            ;BR IF YES
1426 002642 022737 004070 000042 CMP    #SENDAD,@#42   ;IF ACT-11 AUTOMATIC MODE, DON'T TYPE ID
1427 002650 001402          BEQ   20$            ;
1428 002652 104401 001000          TYPE  ,MTITLE        ;TYPE TITLE MESSAGE
1429 002656 004737 011340          20$: JSR    PC,CKSWR     ;CHECK FOR SOFT SWR
1430 002662 017737 176352 001446 MOV    @SWR,STRTSW    ;STORE STARTING SWITCHES
1431 002670 005737 000042          TST   @#42           ;IS IT RUNNING IN AUTO MODE?
1432 002674 001402          BEQ   +6             ;BR IF NO
1433 002676 005037 001446          CLR   STRTSW         ;IF YES, CLEAR SWITCHES
1434 002702 032737 000001 001446 BIT    #SW00,STRTSW   ;IF SW00=1, QUESTIONS ARE ASKED.
1435 002710 001012          BNE   17$            ;BR IF SW00=1
1436 002712 105737 001446          TSTB  STRTSW         ;BIT7=1??

```

```

1437 002716 100007          BPL      17$          ;BR IF SW07=0
1438 002720 005737 001470  TST      KMACTV      ;ARE ANY DEVICES SELECTED?
1439 002724 001027          BNE      16$          ;BN IF YES
1440 002726 104401 010731  TYPE    NOACT        ;NO DEVICES SELECTED.
1441 002732 000000          HALT                    ;STOP THE SHOW
1442 002734 000776          BR       -2           ;DISQUALIFY CONTINUE SWITCH
1443 002736 105737 001336  17$:  TSTB     $ENV      ; IS IT UNDER APT DUMP MODE?
1444 002742 001405          BEQ     27$          ; YES, CHECK IF APT SIZED IT?
1445 002744 132737 000001 001336  BITB     #1,$ENV      ; IS IT UNDER Q,V OR RUN MODE?
1446 002752 001012          BNE     30$          ; YES, NEEDS ONLY APT SIZING.
1447 002754 000406          BR       33$          ; NO, NEEDS REGULAR AUTO.SIZE.
1448 002756 105737 001337  27$:  TSTB     $ENVM     ; IS IT SIZED BY APT?
1449 002762 100406          BMI     30$          ; YES, NEEDS ONLY APT SIZING.
1450 002764 042737 000001 001446  BIC     #SW00,STRTSW ; SIZE ONLY IN AUTO MODE.
1451 002772 004737 012236  33$:  JSR      PC,AUTO.SIZE ; GO DO THE AUTO.SIZE.
1452 002776 000402          BR       16$          ; GO PRINT THE MAP.
1453 003000 004737 013716  30$:  JSR      PC,APT.SIZE ; GO DO THE APT SIZING.
1454 003004 105737 001506  16$:  TSTB     INIFLG     ;FIRST TIME?
1455 003010 001410          BEQ     21$          ;BR IF YES
1456 003012 105737 001446  TSTB     STRTSW      ;IF USING SAME PARAMETERS DONT TYPE MAP
1457 003016 100431          BMI     1$           ;
1458 003020 032737 000006 001446  BIT      #BIT1!BIT2,STRTSW;IS TEST NO. OR LOCK SELECTED
1459 003026 001403          BEQ     24$          ;IF NO THEN TYPE STATUS
1460 003030 000424          BR       1$           ;IF YES DO NOT TYPE STATUS
1461 003032 105137 001506  21$:  COMB     INIFLG     ;SET FLAG
1462 003036 104401 010077  24$:  TYPE     .XHEAD     ;TYPE HEADER
1463 003042 012704 002100  MOV      #KM.MAP,R4   ;SET POINTER
1464 003046 010437 001276  5$:  MOV      R4,$TMP0    ;SET ADDRESS
1465 003052 012437 001300  MOV      (R4)+,$TMP1  ;SET CSR
1466 003056 001411          BEQ     1$           ;ALL DONE IF ZERO
1467 003060 012437 001302  MOV      (R4)+,$TMP2  ;SET STAT1
1468 003064 012437 001304  MOV      (R4)+,$TMP3  ;SET STAT2
1469 003070 012437 001306  MOV      (R4)+,$TMP4  ;SET STAT3
1470 003074 104416          CONVRT                    ;TYPE OUT STATUS MAP
1471 003076 011206          XSTATQ                    ;
1472 003100 000762          BR       5$           ;
1473 003102 012700 002100  1$:  MOV      #KM.MAP,R0  ;R0 POINTS TO STATUS TABLE
1474
1475  ;*****
1476  ;*AUTO SIZE TEST
1477  ;*THIS TEST VERIFYS THAT THE KMC11S AND/OR KMC11S ARE AT THE CORRECT FLOATING
1478  ;*ADDRESSES FOR YOUR SYSTEM. IF THIS TEST FAILS. IT IS NOT A HARDWARE ERROR.
1479  ;*CHECK THE ADDRESSES OF ALL FLOATING DEVICES (DJ,DH,DQ,DU,DUP,LK,DMC,DZ,KMC).
1480  ;*IF THERE ARE NO OTHER FLOATING DEVICES BEFORE THE KMC11, THE FIRST
1481  ;* KMC11 IS 760110. NO DEVICE SHOULD EVER BE AT
1482  ;*ADDRESS 760000.
1483  ;*****
1484
1485 003106 013746 000004          MOV      @#4,-(SP)    ;SAVE LOC 4
1486 003112 013746 000006          MOV      @#6,-(SP)    ;SAVE LOC 6
1487 003116 005037 000006          CLR      @#6          ;CLEAR VEC+2
1488 003122 005037 001302          CLR      $TMP2        ;CLEAR FLAG
1489 003126 011037 002066  AUSTRT: MOV      (R0),KMCSR ;GET NEXT KMC CSR
1490 003132 001510          BEQ     AUDONE        ;BR IF DONE
1491 003134 012737 003240 000004  2$:  MOV      #NODEV,@#4  ;SET UP FOR TIMEOUT
1492 003142 012703 000010  3$:  MOV      #10,R3      ;R3 IS COUNT OF DEVICES BEFORE KMC

```



```

1493 003146 012702 003342      4$:  MOV      #DEVTAB,R2      ;R2 IS DEVICE TABLE PONTER
1494 003152 012701 160010      MOV      #160010,R1      ;START WITH ADDRESS 160010
1495 003156 005711      FLOAT: TST      (R1)      ;CHECK ADDRESS IN R1
1496 003160 111204      MOVVB   (R2),R4        ;IF NO TIMEOUT, GET NEXT ADDRESS
1497 003162 060401      ADD     R4,R1          ;IN R1
1498 003164 005201      INC     R1             ;
1499 003166 040401      BIC     R4,R1         ;
1500 003170 005703      TST     R3            ;ANY MORE DEVICES TO CHECK FOR?
1501 003172 001371      BNE     FLOAT         ;BR IF YES
1502 003174 012737 003244 000004      MOV      #ERR,@#4      ;OK ONLY KMC'S ARE LEFT, SET UP FOR TIMEOUT
1503 003207 005711      FY:    TST      (R1)      ;CHECK KMC ADDRESS
1504 003204 020137 002066      CMP     R1,KMCSR      ;DOES IT MATCH
1505 003210 001403      BEQ     OK            ;BR IF YES
1506 003212 062701 000010      ADD     #10,R1        ;GET NEXT KMC ADDRESS
1507 003216 000771      BR      FY            ;DO IT AGAIN
1508 003220 062700 000010      OK:    ADD     #10,R0      ;SKIP TO NEXT KMC CSR
1509 003224 062701 000010      ADD     #10,R1        ; GET NEXT KMC ADDRESS
1510 003230 011037 002066      MOV     (R0),KMCSR    ; GET NEXT KMC CSR
1511 003234 001447      BEQ     AUDONE        ; BRANCH IF ALL DONE.
1512 003236 000761      FY     BR            ; DO IT AGAIN.
1513 003240 122243      NODEV: CMPB     (R2)+,-(R3) ;ON TIMEOUT, INC R2, DEC R3
1514 003242 000002      RTI                                ;$LPADR
1515 003244 005737 001302      ERR:   TST      $TMP2    ;CHECK FLAG IF = 0 TYPE HEADER
1516 003250 001014      BNE     1$            ;SKIP HEADER
1517 003252 104401      TYPE                                ;TYPEOUT HEADER MESSAGE
1518 003254 011107      CONERR                                ;CONFIGURATION ERROR!!!
1519 003256 012737 003244 001460      MOV     #ERR,SAVPC    ;SAVE PC FOR TYPEOUT
1520 003264 104417      CNVRT                                ;TYPE OUT ERROR PC
1521 003266 003322      ERRPC                                ;
1522 003270 104401      TYPE                                ;TYPE REST OF HEADER
1523 003272 011154      CNERR                                ;
1524 003274 012737 177777 001302      MOV     #-1,$TMP2     ;SET FLAG SO IT ONLY GETS TYPED ONCE
1525 003302 010137 001264      1$:    MOV     R1,$REG1   ;SAVE R1 FOR TYPEOUT
1526 003306 104416      CNVRT                                ;
1527 003310 003330      CONTAB                                ;TYPE CSR VALUES
1528 003312 104401      3$:    TYPE                                ;
1529 003314 011175      KMC#M                                ;
1530 003316 022626      4$:    CMP     (SP)+,(SP)+ ;ADJUST STACK
1531 003320 000737      BR     OK            ;BR TO GET OUT
1532 003322 000001      ERRPC: 1
1533 003324 006 002      .BYTE 6,2
1534 003326 001460      SAVPC
1535 003330 000002      CONTAB: 2
1536 003332 006 004      .BYTE 6,4
1537 003334 001264      $REG1
1538 003336 006 002      .BYTE 6,2
1539 003340 002066      KMCSR
1540 003342 007      DEVTAB: .BYTE 7 ;DJ
1541 003343 017      .BYTE 17 ;DH
1542 003344 007      .BYTE 7 ;DQ
1543 003345 007      .BYTE 7 ;DU
1544 003346 007      .BYTE 7 ;DUP
1545 003347 007      .BYTE 7 ;LK
1546 003350 007      .BYTE 7 ;DMC
1547 003351 007      .BYTE 7 ;DZ
1548 003352 007      .BYTE 7 ;KMC

```

```

1549      003354      003354      .EVEN
1550      003354      012637      000006      AUDONE:
1551      003354      012637      000004      1$:  MOV      (SP)+,@#6      ;RESTORE LOC 6
1552      003360      012637      000004      MOV      (SP)+,@#4      ;RESTORE LOC 4
1553      003364      032737      000010      001446      BIT      #SW03,STRTSW      ;SELECT SPECIFIC DEVICES??
1554      003372      001422      BEQ      3$      ;BR IF NO.
1555      003374      104401      010017      TYPE     ,MNEW      ;TYPE THE MESSAGE.
1556      003400      005000      CLR      RO      ;ZERO DATA LIGHTS
1557      003402      000000      HALT     ;WAIT FOR USER TO TELL WHAT DEVICES TO RUN
1558      003404      027737      175630      001474      CMP      @SWR,SAVACT      ;IS THE NUMBER VALID?
1559      003412      101404      BLOS     2$      ;BR IF NUMBER IS OK.
1560      003414      104401      007672      TYPE     ,MERR3      ;TELL USER OF INVALID NUMBER.
1561      003420      000000      HALT     ;STOP EVERY THING.
1562      003422      000776      BR       -2      ;RESTART THE PROGRAM AGAIN.
1563      003424      017737      175610      001470      2$:  MOV      @SWR,KMACTV      ;GET NEW DEVICE PATTERN
1564      003432      013700      001470      MOV      KMACTV,RO      ;SHOW THE USER WHAT HE SELECTED.
1565      003436      000000      HALT     ;CONTINUE DYNAMIC SWITCHES.
1566      003440      012700      000300      3$:  MOV      #300,RO      ;PREPARE TO CLEAR THE FLOATING
1567      003444      012701      000302      MOV      #302,R1      ;VECTOR AREA. 300-776
1568      003450      010120      4$:  MOV      R1,(RO)+      ;START PUTTING 'PC+2 - HALT'
1569      003452      005021      CLR      (R1)+      ;IN VECTOR AREA.
1570      003454      022021      CMP      (RO)+,(R1)+      ;POP POINTERS
1571      003456      022700      001000      CMP      #1000,RO      ;ALL DONE??
1572      003462      001372      BNE      4$      ;BR IF NO.
1573
1574      ;TEST START AND RESTART
1575      ;-----
1576
1577      003464      012706      001200      .BEGIN: MOV      #STACK,SP      ;SET UP STACK
1578      003470      013746      000006      MOV      @#6,-(SP)      ;SAVE LOC 6
1579      003474      013746      000004      MOV      @#4,-(SP)      ;SAVE LOC 4
1580      003500      005000      CLR      RO      ;START AT 0
1581      003502      012737      003546      000004      MOV      #2$,@#4      ;SET UP FOR TIME OUT
1582      003510      005037      000006      CLR      @#6      ;TO AUTOSIZE MEMORY
1583      003514      005720      6$:  TST      (RO)+      ;CHECK ADDRESS IN RO
1584      003516      022700      157776      CMP      #157776,RO      ;IS IT AT LEAST 28K
1585      003522      001374      BNE      6$      ;BR IF NO
1586      003524      162700      007776      SUB      #7776,RO      ;SAVE 2K FOR MONITORS
1587      003530      010037      001466      7$:  MOV      RO,MEMLIM      ;STORE MEMORY LIMIT
1588      003534      012637      000004      MOV      (SP)+,@#4      ;RESTORE LOC 4
1589      003540      012637      000006      MOV      (SP)+,@#6      ;RESTORE LOC 6
1590      003544      000413      BR       10$      ;CONTINUE
1591      003546      022626      2$:  CMP      (SP)+,(SP)+      ;ADJUST STACK
1592      003550      162700      000004      SUB      #4,RO      ;GET LAST GOOD ADDRESS
1593      003554      162700      007776      SUB      #7776,RO      ;SAVE 2K FOR MONITORS
1594      003560      022700      030000      CMP      #30000,RO      ;IS IT 8K?
1595      003564      001361      BNE      7$      ;BR IF NO
1596      003566      012700      037400      MOV      #37400,RO      ;IF 8K DON'T SAVE 2K
1597      003572      000756      BR       7$      ;
1598      003574      012737      000340      177776      10$: MOV      #340,PS      ;LOCK OUT INTERRUPTS
1599      003602      032737      ^00004      001446      BIT      #BIT2,STRTSW      ;CHECK FOR LOCK ON TEST
1600      003610      001406      BEQ      1$      ;BR IF NO LOCK DESIRED.
1601      003612      104401      007716      TYPE     ,MLOCK      ;TYPE LOCK SELECTED.
1602      003616      012737      000240      004146      MOV      #NOP,TTST      ;SET UP TO LOCK
1603      003624      000403      BR       3$      ;CONTINUE ALONG.
1604      003626      013737      004360      004146      1$:  MOV      BRW,TTST      ;PREPARE NORMAL SCOPE ROUTINE
  
```

CZKCF MACY11 30A(1052) 08-JUL-80 08:27 PAGE 36  
CZKCF.P11 08-JUL-80 08:24

PROGRAM INITIALIZATION AND START UP.

SEQ 0035

|      |        |        |        |        |      |      |                |  |
|------|--------|--------|--------|--------|------|------|----------------|--|
| 1605 | 003634 | 012737 | 011606 | 001206 | 3\$: | MOV  | #CYCLE,\$LPADR | ; START AT "CYCLE" FIND WHICH DEVICE TO TEST |
| 1606 | 003642 | 032737 | 000002 | 001446 | 4\$: | BIT  | #SW01,STRTSW   | ; IS TEST NO. SELECTED?                      |
| 1607 | 003650 | 001002 |        |        |      | BNE  | 5\$            | ; BR IF YES                                  |
| 1608 | 003652 | 104401 | 007642 |        |      | TYPE | ,MR            | ; TYPE R                                     |
| 1609 | 003656 | 000177 | 175324 |        | 5\$: | JMP  | @\$LPADR       | ; START TESTING                              |

```

1610 ;END OF PASS
1611 ;TYPE NAME OF TEST
1612 ;UPDATE PASS COUNT
1613 ;CHECK FOR EXIT TO ACT-11
1614 ;RESTART TEST
1615
1616 .SBTTL END OF PASS ROUTINE
1617
1618 ::*****
1619 ;*INCREMENT THE PASS NUMBER ($PASS)
1620 ;*IF THERES A MONITOR GO TO IT
1621 ;*IF THERE ISN'T JUMP TO CYCLE
1622
1623 $EOP:
1624 003662 000005 RESET
1625 003664 005237 001324 INC $PASS ; INCREMENT THE PASS COUNT
1626 003670 105037 001203 CLR $ERFLG ; CLEAR ERROR FLAG
1627 003674 104401 007620 TYPE ,MEPASS ; TYPE END PASS.
1628 003700 104401 007745 TYPE ,MCSRX ; TYPE "CSR"
1629 003704 104417 004104 CNVRT ,XCSR ; SHOW IT.
1630 003710 104401 007753 TYPE ,MVECX ; TYPE VECTOR.
1631 003714 104417 004112 CNVRT ,XVEC ; SHOW IT.
1632 003720 104401 007761 TYPE ,MPASSX ; TYPE " PASSES "
1633 003724 104417 004120 CNVRT ,XPASS ; SHOW IT.
1634 003730 104401 007772 TYPE ,MERRX ; TYPE " ERRORS "
1635 003734 104417 004126 CNVRT ,XERR ; SHOW IT.
1636 003740 013700 001504 MOV MILK,RO ; SET POINTER TO PASSCNT.
1637 003744 013720 001324 MOV $PASS,(RO)+ ; SAVE THE PASS COUNT.
1638 003750 013720 001212 MOV $ERTTL,(RO)+ ; SAVE ERROR COUNT
1639 003754 013777 002060 176074 MOV @KMRVL,@KMRVEC ; RESTORE THE RECEIVER INTERRUPT VECTOR.
1640 003762 005077 176072 CLR @KMRVL ; RESTORE RECEIVER LEVEL
1641 003766 013777 002064 176066 MOV @KMTLVL,@KMTVEC ; RESTORE THE TRANSMIT INTERRUPT VECTOR.
1642 003774 005077 176064 CLR @KMTLVL ; RESTORE TRANSMITTER LEVEL
1643 004000 005337 001476 DEC SAVNUM ; ALL DEVICE TESTED?
1644 004004 001035 BNE $DOAGN ; BRANCH IF NO.
1645 004006 112737 000377 001511 MOVB #377,QV.FLG ; SET QUICK VERIFY FLAG.
1646 004014 013737 001472 001476 MOV @KMMNUM,SAVNUM ; RESTORE DEVICE COUNT.
1647 004022 005037 001216 CLR $ERRPC ; CLEAR LAST ERROR PC
1648 004026 005037 001310 CLR $TIMES ; ZERO THE NUMBER OF ITERATIONS
1649 004032 005237 001324 INC $PASS ; INCREMENT THE PASS NUMBER
1650 004036 042737 100000 001324 BIC #100000,$PASS ; DON'T ALLOW A NEG. NUMBER
1651 004044 005327 DEC (PC)+ ; LOOP?
1652 004046 000001 $EOPCT: .WORD 1
1653 004050 003013 BGT $DOAGN ;:YES
1654 004052 012737 MOV (PC)+,@(PC)+ ;:RESTORE COUNTER
1655 004054 000001 $ENDCT: .WORD 1
1656 004056 004046 $EOPCT
1657 004060 013700 000042 $GET42: MOV @#42,RO ;:GET MONITOR ADDRESS
1658 004064 001405 BEQ $DOAGN ;:BRANCH IF NO MONITOR
1659 004066 000005 RESET ;:CLEAR THE WORLD
1660 004070 004710 $ENDAD: JSR PC,(RO) ;:GO TO MONITOR
1661 004072 000240 NOP ;:SAVE ROOM
1662 004074 000240 NOP ;:FOR
1663 004076 000240 NOP ;:ACT11
1664 004100 $DOAGN:
1665 004100 000137 JMP @(PC)+ ;:RETURN

```

1666 004102 011606  
 1667 004104 000001  
 1668 004106 006 002  
 1669 004110 002066  
 1670 004112 000001  
 1671 004114 004 002  
 1672 004116 002056  
 1673 004120 000001  
 1674 004122 006 002  
 1675 004124 001324  
 1676 004126 000001  
 1677 004130 006 002  
 1678 004132 001212

\$RTNAD: .WORD CYCLE  
 XCSR: 1  
 .BYTE 6,2  
 KMCSR  
 XVEC: 1  
 .BYTE 4,2  
 KMRVEC  
 XPASS: 1  
 .BYTE 6,2  
 \$PASS  
 XERR: 1  
 .BYTE 6,2  
 \$ERTTL

;;SCOPE LOOP AND INTERATION HANDLER  
 -----

.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:U>)  
 ;\*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  
 ;\*CALL  
 ;\* SCOPE ;;SCOPE=IOT

1694  
 1695 004134  
 1696 004134 005037 001216  
 1697 004140 023716 014142  
 1698 004144 001413  
 1699 004146 000406  
 1700 004150 105777 175070  
 1701 004154 100067  
 1702 004156 017766 175064 177776  
 1703 004164 032777 040000 175046  
 1704 004172 001060  
 1705  
 1706 004174 000416  
 1707  
 1708 004176 013746 000004  
 1709 004202 012737 004222 000004  
 1710 004210 005737 177060  
 1711 004214 012637 000004  
 1712 004220 000436  
 1713 004222 022626  
 1714 004224 012637 000004  
 1715 004230 000441  
 1716 004232  
 1717 004232 105737 001203  
 1718 004236 001404  
 1719 004240 105037 001203  
 1720 004244 005037 001310  
 1721 004250 032777 004000 174762

\$SCOPE:  
 CLR \$ERRPC ; CLEAR LAST ERROR PC  
 CMP TST1+2,(SP) ; IS THIS TEST #1 ?  
 BEQ \$XTSTR ; IF SO DON'T LOOP.  
 TTST: BR 1\$ ;  
 TSTB @\$TKS ; KEYBOARD DONE ?  
 BPL \$OVER ; IF NO DONT WAIT.  
 MOV @\$TKB,-2(SP) ;  
 1\$: BIT #BIT14,\$SWR ;;LOOP ON PRESENT TEST?  
 BNE \$OVER ;;YES IF SW14=1  
 ;\*\*\*\*\*START OF CODE FOR THE XOR TESTER\*\*\*\*\*  
 \$XTSTR: BR 6\$ ;IF RUNNING ON THE "XOR" TESTER CHANGE  
 ;THIS INSTRUCTION TO A "NOP" (NOP=240)  
 MOV @\$ERRVEC,-(SP) ;SAVE THE CONTENTS OF THE ERROR VECTOR  
 MOV #5,\$@\$ERRVEC ;SET FOR TIMEOUT  
 TST #177060 ;TIME OUT ON XOR?  
 MOV (SP)+,\$@\$ERRVEC ;RESTORE THE ERROR VECTOR  
 BR \$\$VLAD ;GO TO THE NEXT TEST  
 5\$: CMP (SP)+,(SP)+ ;CLEAR THE STACK AFTER A TIME OUT  
 MOV (SP)+,\$@\$ERRVEC ;RESTORE THE ERROR VECTOR  
 BR \$OVER ;LOOP ON THE PRESENT TEST  
 6\$: ;\*\*\*\*\*END OF CODE FOR THE XOR TESTER\*\*\*\*\*  
 2\$: TSTB \$ERFLG ;HAS AN ERROR OCCURRED?  
 BEQ 3\$ ;BR IF NO  
 4\$: CLRB \$ERFLG ;ZERO THE ERROR FLAG  
 CLR \$TIMES ;CLEAR THE NUMBER OF ITERATIONS TO MAKE  
 3\$: BIT #BIT11,\$SWR ;INHIBIT ITERATIONS?

```

1722 004256 001011          BNE      1$          ;;BR IF YES
1723 004260 005737 001324  TST      $PASS      ;;IF FIRST PASS OF PROGRAM
1724 004264 001406          BEQ      1$          ;;      INHIBIT ITERATIONS
1725 004266 005237 001204  INC      $ICNT      ;;      INCREMENT ITERATION COUNT
1726 004272 023737 001310 001204  CMP      $TIMES,$ICNT ;;CHECK THE NUMBER OF ITERATIONS MADE
1727 004300 002015          BGE      $OVER      ;;BR IF MORE ITERATION REQUIRED
1728 004302 012737 000001 001204 1$: MOV      #1,$ICNT   ;;REINITIALIZE THE ITERATION COUNTER
1729 004310 013737 004362 001310 001310  MOV      $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
1730 004316 105237 001202  $SVLAD: INCB     $STNM   ;;COUNT TEST NUMBERS
1731 004322 113737 001202 001322  MOV      $STNM,$TESTN ;;SET TEST NUMBER IN APT MAILBOX
1732 004330 011637 001206          MOV      (SP),$LPADR ;;SAVE SCOPE LOOP ADDRESS
1733 004334 013777 001202 174700 $OVER: MOV      $STNM,@DISPLAY ;;DISPLAY TEST NUMBER
1734 004342 013716 001206          MOV      $LPADR,(SP) ;;FUDGE RETURN ADDRESS
1735 004346 005037 001444          CLR      LOCK      ; RESET LOCK ON DATA.
1736 004352 013701 002066          MOV      KMCSR,R1   ; R1 CONTAINS BASE KMC ADDRESS.
1737 004356 000002          RTI
1738 004360 000406          BRW: .WORD 406
1739 004362 000020          $MXCNT: 20          ;;MAX. NUMBER OF ITERATIONS
    
```

```

1740          ;CHECK FOR FREEZE ON CURRENT DATA
1741          ;-----
1742
1743
    
```

```

1744 004364 004737 011340          .SCOPI: JSR      PC,CKSWR          ;CHECK FOR SOFT SWR
1745 004370 032777 001000 174642  BIT      #SW09,@SWR          ;IS SW09=1(SET)?
1746 004376 001405          BEQ      1$          ;BR IF NOT SET.
1747 004400 005737 001444          TST      LOCK
1748 004404 001402          BEQ      1$
1749 004406 013716 001444          MOV      LOCK,(SP)          ;GOTO THE ADDRESS IN LOCK.
1750 004412 000002          1$: RTI          ;GO BACK.
    
```

```

1751          ;TELETYPE OUTPUT ROUTINE
1752          ;-----
1753
    
```

1754 .SBTTL TYPE ROUTINE

```

1755          ;*****
1756          ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
1757          ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
1758          ;*NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
1759          ;*NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
1760          ;*NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
    
```

```

1761          ;*CALL:
1762          ;*1) USING A TRAP INSTRUCTION
1763          ;* TYPE ,MESADR          ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
1764          ;*OR
1765          ;* TYPE
1766          ;* MESADR
1767          ;*
    
```

```

1771          $TYPE: TSTB     $TPFLG          ;; IS THERE A TERMINAL?
1772 004414 105737 001257          BPL      1$          ;; BR IF YES
1773 004420 100002          HALT     ;HALT HERE IF NO TERMINAL
1774 004422 000000          BR      3$          ;; LEAVE
1775 004424 000430          1$: MOV      RO,-(SP)          ;; SAVE RO
1776 004426 010046          MOV      @2(SP),RO      ;; GET ADDRESS OF ASCIZ STRING
1777 004430 017600 000002
    
```

```

1778 004434 122737 000001 001336      CMPB  #APTENV,$ENV      ;;RUNNING IN APT MODE
1779 004442 001011                    BNE   62$              ;;NO,GO CHECK FOR APT CONSOLE
1780 004444 132737 000100 001337      BITB  #APTSPOOL,$ENVM  ;;SPOOL MESSAGE TO APT
1781 004452 001405                    BEQ   62$              ;;NO,GO CHECK FOR CONSOLE
1782 004454 010037 004464              MOV   RO,61$          ;;SETUP MESSAGE ADDRESS FOR APT
1783 004460 004737 004704              JSR   PC,$ATY3        ;;SPOOL MESSAGE TO APT
1784 004464 000000                    .WORD 0                ;;MESSAGE ADDRESS
1785 004466 132737 000040 001337      61$: BITB  #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
1786 004474 001003                    BNE   60$              ;;YES,SKIP TYPE OUT
1787 004476 112046                    2$:  MOVB (RO)+,-(SP)    ;;PUSH CHARACTER TO BE TYPED ONTO STACK
1788 004500 001005                    BNE   4$                ;;BR IF IT ISN'T THE TERMINATOR
1789 004502 005726                    TST  (SP)+             ;;IF TERMINATOR POP IT OFF THE STACK
1790 004504 012600                    60$: MOV   (SP)+,RO      ;;RESTORE RO
1791 004506 062716 000002                    3$:  ADD   #2,(SP)      ;;ADJUST RETURN PC
1792 004512 000002                    RTI                          ;;RETURN
1793 004514 122716 000011                    4$:  CMPB  #HT,(SP)      ;;BRANCH IF <HT>
1794 004520 001430                    BEQ   8$                ;;
1795 004522 122716 000200                    CMPB  #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
1796 004526 001006                    BNE   5$                ;;
1797 004530 005726                    TST  (SP)+             ;;POP <CR><LF> EQUIV
1798 004532 104401                    TYPE                          ;;TYPE A CR AND LF
1799 004534 001313                    $CRLF
1800 004536 105037 004672                    CLRB  $CHARCNT        ;;CLEAR CHARACTER COUNT
1801 004542 000755                    BR    2$                ;;GET NEXT CHARACTER
1802 004544 004737 004626                    5$:  JSR   PC,$TYPEC     ;;GO TYPE THIS CHARACTER
1803 004550 123726 001256                    6$:  CMPB  $FILLC,(SP)+  ;;IS IT TIME FOR FILLER CHARS.?
1804 004554 001350                    BNE   2$                ;;IF NO GO GET NEXT CHAR.
1805 004556 013746 001254                    MOV   $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
1806                                ;;AND THE NULL CHAR.
1807 004562 105366 000001                    7$:  DECB  1(SP)        ;;DOES A NULL NEED TO BE TYPED?
1808 004566 002770                    BLT   6$                ;;BR IF NO--GO POP THE NULL OFF OF STACK
1809 004570 004737 004626                    JSR   PC,$TYPEC     ;;GO TYPE A NULL
1810 004574 105337 004672                    DECB  $CHARCNT        ;;DO NOT COUNT AS A COUNT
1811 004600 000770                    BR    7$              ;;LOOP
1812
1813                                ;HORIZONTAL TAB PROCESSOR
1814
1815 004602 112716 000040                    8$:  MOVB  #' ,(SP)     ;;REPLACE TAB WITH SPACE
1816 004606 004737 004626                    9$:  JSR   PC,$TYPEC     ;;TYPE A SPACE
1817 004612 132737 000007 004672      BITB  #7,$CHARCNT     ;;BRANCH IF NOT AT
1818 004620 001372                    BNE   9$                ;;TAB STOP
1819 004622 005726                    TST  (SP)+             ;;POP SPACE OFF STACK
1820 004624 000724                    BR    2$                ;;GET NEXT CHARACTER
1821 004626 105777 174416                    $TYPEC: TSTB @STPS     ;;WAIT UNTIL PRINTER IS READY
1822 004632 100375                    BPL  $TYPEC            ;;
1823 004634 116677 000002 174410      MOVB  2(SP),@STPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
1824 004642 122766 000015 000002      CMPB  #CR,2(SP)      ;;IS CHARACTER A CARRIAGE RETURN?
1825 004650 001003                    BNE   1$                ;;BRANCH IF NO
1826 004652 105037 004672                    CLRB  $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
1827 004656 000406                    BR    $TYPEX           ;;EXIT
1828 004660 122766 000012 000002      1$:  CMPB  #LF,2(SP)   ;;IS CHARACTER A LINE FEED?
1829 004666 001402                    BEQ   $TYPEX           ;;BRANCH IF YES
1830 004670 105227                    INCB  (PC)+           ;;COUNT THE CHARACTER
1831 004672 000000                    $CHARCNT: .WORD 0     ;;CHARACTER COUNT STORAGE
1832 004674 000207                    $TYPEX: RTS          PC
1833

```

```

1834 .SBTTL APT COMMUNICATIONS ROUTINE
1835
1836 *****
1837 004676 112737 000001 005142 $ATY1: MOVB #1,$FFLG ;;TO REPORT FATAL ERROR
1838 004704 112737 000001 005140 $ATY3: MOVB #1,$MFLG ;;TO TYPE A MESSAGE
1839 004712 000403 BR $ATYC
1840 004714 112737 000001 005142 $ATY4: MOVB #1,$FFLG ;;TO ONLY REPORT FATAL ERROR
1841 004722 $ATYC:
1842 004722 010046 MOV RO,-(SP) ;;PUSH RO ON STACK
1843 004724 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
1844 004726 105737 005140 TSTB $MFLG ;;SHOULD TYPE A MESSAGE?
1845 004732 001450 BEQ 5$ ;;IF NOT: BR
1846 004734 122737 000001 001336 CMPB #APTENV,$ENV ;;OPERATING UNDER APT?
1847 004742 001031 BNE 3$ ;;IF NOT: BR
1848 004744 132737 000100 001337 BITB #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
1849 004752 001425 BEQ 3$ ;;IF NOT: BR
1850 004754 017600 000004 MOV @4(SP),RO ;;GET MESSAGE ADDR.
1851 004760 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
1852 004766 005737 001316 1$: TST $MSGTYPE ;;SEE IF DONE W/ LAST XMISSION?
1853 004772 001375 BNE 1$ ;;IF NOT: WAIT
1854 004774 010037 001332 MOV RO,$MSGAD ;;PUT ADDR IN MAILBOX
1855 005000 105720 2$: TSTB (RO)+ ;;FIND END OF MESSAGE
1856 005002 001376 BNE 2$
1857 005004 163700 001332 SUB $MSGAD,RO ;;SUB START OF MESSAGE
1858 005010 006200 ASR RO ;;GET MESSAGE LNTH IN WORDS
1859 005012 010037 001334 MOV RO,$MSGGLT ;;PUT LENGTH IN MAILBOX
1860 005016 012737 000004 001316 MOV #4,$MSGTYPE ;;TELL APT TO TAKE MSG.
1861 005024 000413 BR 5$
1862 005026 017637 000004 005052 3$: MOV @4(SP),4$ ;;PUT MSG ADDR IN JSR LINKAGE
1863 005034 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDRESS
1864 005042 013746 177776 MOV 177776,-(SP) ;;PUSH 177776 ON STACK
1865 005046 004737 004414 JSR PC,$TYPE ;;CALL TYPE MACRO
1866 005052 000000 4$: .WORD 0
1867 005054 5$:
1868 005054 105737 005142 10$: TSTB $FFLG ;;SHOULD REPORT FATAL ERROR?
1869 005060 001416 BEQ 12$ ;;IF NOT: BR
1870 005062 005737 001336 TST $ENV ;;RUNNING UNDER APT?
1871 005066 001413 BEQ 12$ ;;IF NOT: BR
1872 005070 005737 001316 11$: TST $MSGTYPE ;;FINISHED LAST MESSAGE?
1873 005074 001375 BNE 11$ ;;IF NOT: WAIT
1874 005076 017637 000004 001320 MOV @4(SP),$FATAL ;;GET ERROR #
1875 005104 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
1876 005112 005237 001316 INC $MSGTYPE ;;TELL APT TO TAKE ERROR
1877 005116 105037 005142 12$: CLRB $FFLG ;;CLEAR FATAL FLAG
1878 005122 105037 005141 CLRB $LFLG ;;CLEAR LOG FLAG
1879 005126 105037 005140 CLRB $MFLG ;;CLEAR MESSAGE FLAG
1880 005132 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
1881 005134 012600 MOV (SP)+,RO ;;POP STACK INTO RO
1882 005136 000207 RTS PC ;;RETURN
1883 005140 000 $MFLG: .BYTE 0 ;;MESSG. FLAG
1884 005141 000 $LFLG: .BYTE 0 ;;LOG FLAG
1885 005142 000 $FFLG: .BYTE 0 ;;FATAL FLAG
1886 005144 .EVEN
1887 000200 APTSIZE=200
1888 000001 APTENV=001
1889 000100 APTSPOOL=100

```



```

1890          000040
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909 005144 011646
1910 005146 016666 000004 000002
1911 005154 105777 174064
1912 005160 100375
1913 005162 117766 174060 000004
1914 005170 042766 177600 000004
1915 005176 026627 000004 000023
1916 005204 001013
1917 005206 105777 174032
1918 005212 100375
1919 005214 117746 174026
1920 005220 042716 177600
1921 005224 022627 000021
1922 005230 001366
1923 005232 000750
1924 005234 026627 000004 000140
1925 005242 002407
1926 005244 026627 000004 000175
1927 005252 003003
1928 005254 042766 000040 000004
1929 005262 000002
1930
1931
1932
1933
1934
1935
1936
1937 005264 010346
1938 005266 005046
1939 005270 012703 005520
1940 005274 022703 005527
1941 005300 101456
1942 005302 104402
1943 005304 112613
1944 005306 122713 000177
1945 005312 001022

```

```

APTCSUP=040
:-----
.SBTTL TTY INPUT ROUTINE
:*****
.ENABL LSB
.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
:
SRDCHR: MOV      (SP),-(SP)      ;; PUSH DOWN THE PC
        MOV      4(SP),2(SP)    ;; SAVE THE PS
1$:     TSTB     @STKS          ;; WAIT FOR
        BPL      1$            ;; A CHARACTER
        MOVB     @STKB,4(SP)    ;; READ THE TTY
        BIC      #'C<177>,4(SP) ;; GET RID OF JUNK IF ANY
        CMP      4(SP),#23     ;; IS IT A CONTROL-S?
        BNE      3$            ;; BRANCH IF NO
2$:     TSTB     @STKS          ;; WAIT FOR A CHARACTER
        BPL      2$            ;; LOOP UNTIL ITS THERE
        MOVB     @STKB,-(SP)    ;; GET CHARACTER
        BIC      #'C177,(SP)   ;; MAKE IT 7-BIT ASCII
        CMP      (SP)+,#21     ;; IS IT A CONTROL-Q?
        BNE      2$            ;; IF NOT DISCARD IT
        BR       1$            ;; YES, RESUME
3$:     CMP      4(SP),#140    ;; IS IT UPPER CASE?
        BLT      4$            ;; BRANCH IF YES
        CMP      4(SP),#175    ;; IS IT A SPECIAL CHAR?
        BGT      4$            ;; BRANCH IF YES
        BIC      #40,4(SP)     ;; MAKE IT UPPER CASE
4$:     RTI              ;; GO BACK TO USER
:*****
*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
*CALL:
*      RDLIN         ;; INPUT A STRING FROM THE TTY
*      RETURN HERE   ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
*                  ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
:
SRDLIN: MOV      R3,-(SP)      ;; SAVE R3
        CLR      -(SP)        ;; CLEAR THE RUBOUT KEY
1$:     MOV      #$TTYIN,R3    ;; GET ADDRESS
2$:     CMP      #$TTYIN+7,R3  ;; BUFFER FULL?
        BLOS     4$            ;; BR IF YES
        RDCHR    ;; GO READ ONE CHARACTER FROM THE TTY
        MOVB     (SP)+,(R3)    ;; GET CHARACTER
10$:    CMPB     #177,(R3)     ;; IS IT A RUBOUT
        BNE     5$            ;; BR IF NO
        BNE     5$

```

```

1946 005314 005716 TST (SP) ;; IS THIS THE FIRST RUBOUT?
1947 005316 001007 BNE 6$ ;; BR IF NO
1948 005320 112737 000134 005516 MOVB #' \,9$ ;; TYPE A BACK SLASH
1949 005326 104401 005516 TYPE ,9$
1950 005332 012716 177777 MOV #-1,(SP) ;; SET THE RUBOUT KEY
1951 005336 005303 6$: DEC R3 ;; BACKUP BY ONE
1952 005340 020327 005520 CMP R3,#$TTYIN ;; STACK EMPTY?
1953 005344 103434 BLO 4$ ;; BR IF YES
1954 005346 111337 005516 MOVB (R3),9$ ;; SETUP TO TYPEOUT THE DELETED CHAR.
1955 005352 104401 005516 TYPE ,9$ ;; GO TYPE
1956 005356 000746 BR 2$ ;; GO READ ANOTHER CHAR.
1957 005360 005716 5$: TST (SP) ;; RUBOUT KEY SET?
1958 005362 001406 BEQ 7$ ;; BR IF NO
1959 005364 112737 000134 005516 MOVB #' \,9$ ;; TYPE A BACK SLASH
1960 005372 104401 005516 TYPE ,9$
1961 005376 005016 CLR (SP) ;; CLEAR THE RUBOUT KEY
1962 005400 122713 000025 7$: CMPB #25,(R3) ;; IS CHARACTER A CTRL U?
1963 005404 001003 BNE 8$ ;; BR IF NO
1964 005406 104401 005527 TYPE , $CNTLU ;; TYPE A CONTROL 'U'
1965 005412 000726 BR 1$ ;; GO START OVER
1966 005414 122713 000022 8$: CMPB #22,(R3) ;; IS CHARACTER A "'R'?
1967 005420 001011 BNE 3$ ;; BRANCH IF NO
1968 005422 105013 CLRB (R3) ;; CLEAR THE CHARACTER
1969 005424 104401 001313 TYPE , $CRLF ;; TYPE A 'CR' & 'LF'
1970 005430 104401 005520 TYPE , $TTYIN ;; TYPE THE INPUT STRING
1971 005434 000717 BR 2$ ;; GO PICKUP ANOTHER CHACTER
1972 005436 104401 001312 4$: TYPE , $QUES ;; TYPE A '?'
1973 005442 000712 BR 1$ ;; CLEAR THE BUFFER AND LOOP
1974 005444 111337 005516 3$: MOVB (R3),9$ ;; ECHO THE CHARACTER
1975 005450 104401 005516 TYPE ,9$
1976 005454 122723 000015 CMPB #15,(R3)+ ;; CHECK FOR RETURN
1977 005460 001305 BNE 2$ ;; LOOP IF NOT RETURN
1978 005462 105063 177777 CLRB -1(R3) ;; CLEAR RETURN (THE 15)
1979 005466 104401 001314 TYPE , $LF ;; TYPE A LINE FEED
1980 005472 005726 TST (SP)+ ;; CLEAN RUBOUT KEY FROM THE STACK
1981 005474 012603 MOV (SP)+,R3 ;; RESTORE R3
1982 005476 011646 MOV (SP),-(SP) ;; ADJUST THE STACK AND PUT ADDRESS OF THE
1983 005500 016666 000004 000002 MOV 4(SP),2(SP) ;; FIRST ASCII CHARACTER ON IT
1984 005506 012766 005520 000004 MOV #$TTYIN,4(SP)
1985 005514 000002 RTI ;; RETURN
1986 005516 000 9$: .BYTE 0 ;; STORAGE FOR ASCII CHAR. TO TYPE
1987 005517 000 .BYTE 0 ;; TERMINATOR
1988 005520 000007 $TTYIN: .BLKB 7 ;; RESERVE 7 BYTES FOR TTY INPUT
1989 005527 136 006525 000012 $CNTLU: .ASCIZ /*U/<15><12> ;; CONTROL 'U'
1990 005534 043536 005015 000 $CNTLG: .ASCIZ /*G/<15><12> ;; CONTROL 'G'
1991 005541 015 051412 051127 $MSWR: .ASCIZ <15><12>/SWR = /
1992 005546 036440 000040
1993 005552 020040 042516 020127 $MNEW: .ASCIZ / NEW = /
1994 005560 020075 000
1995 005564 .EVEN
1996 .SBTTL READ AN OCTAL NUMBER FROM THE TTY
1997
1998 ;*****
1999 ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
2000 ;*CHANGE IT TO BINARY.
2001 ;*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL

```

CZKCF MACY11 30A(1052) 08-JUL-80 08:27 PAGE 44  
 CZKCF.P11 08-JUL-80 08:24 READ AN OCTAL NUMBER FROM THE TTY

SEQ 0043

```

2002      ;*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED
2003      ;*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
2004      ;*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
2005      ;*CALL:
2006      ;*      RDOCT          ;;READ AN OCTAL NUMBER
2007      ;*      RETURN HERE   ;;LOW ORDER BITS ARE ON TOP OF THE STACK
2008      ;*
2009      ;*
2010      005564 011646          $RDOCT: MOV      (SP),-(SP)      ;;PROVIDE SPACE FOR THE
2011      005566 016666 000004 000002 MOV      4(SP),2(SP)      ;;INPUT NUMBER
2012      005574 010046          MOV      R0,-(SP)        ;;PUSH R0 ON STACK
2013      005576 010146          MOV      R1,-(SP)        ;;PUSH R1 ON STACK
2014      005600 010246          MOV      R2,-(SP)        ;;PUSH R2 ON STACK
2015      005602 104403          1$:  RDLIN          ;;READ AN ASCII LINE
2016      005604 001420          MOV      (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
2017      005606 010037 005712 MOV      R0,5$          ;;AND SAVE IT
2018      005612 005001          CLR      R1            ;;CLEAR DATA WORD
2019      005614 005002          CLR      R2
2020      005616 112046          2$:  MOVB     (R0)+,-(SP)      ;;PICKUP THIS CHARACTER
2021      005620 001420          BEQ     3$            ;;IF ZERO GET OUT
2022      005622 122716 000060 CMPB    #'0,(SP)        ;;MAKE SURE THIS CHARACTER
2023      005626 003026          BGT     4$            ;;IS AN OCTAL DIGIT
2024      005630 122716 000067 CMPB    #'7,(SP)
2025      005634 002423          BLT     4$
2026      005636 006301          ASL     R1            ;;*2
2027      005640 006102          ROL     R2
2028      005642 006301          ASL     R1            ;;*4
2029      005644 006102          ROL     R2
2030      005646 006301          ASL     R1            ;;*8
2031      005650 006102          ROL     R2
2032      005652 042716 177770 BIC     #'C7,(SP)      ;;STRIP THE ASCII JUNK
2033      005656 062601          ADD     (SP)+,R1      ;;ADD IN THIS DIGIT
2034      005660 000756          BR      2$           ;;LOOP
2035      005662 005726          3$:  TST     (SP)+      ;;CLEAN TERMINATOR FROM STACK
2036      005664 010166 000012 MOV     R1,12(SP)      ;;SAVE THE RESULT
2037      005670 010237 005722 MOV     R2,$HIOCT
2038      005674 012602          MOV     (SP)+,R2      ;;POP STACK INTO R2
2039      005676 012601          MOV     (SP)+,R1      ;;POP STACK INTO R1
2040      005700 012600          MOV     (SP)+,R0      ;;POP STACK INTO R0
2041      005702 000002          RTI
2042      005704 005726          4$:  TST     (SP)+      ;;CLEAN PARTIAL FROM STACK
2043      005706 105010          CLRB   (R0)          ;;SET A TERMINATOR
2044      005710 104401          TYPE
2045      005712 000000          5$:  .WORD   0          ;;TYPE UP THRU THE BAD CHAR.
2046      005714 104401 001312 TYPE    ,SQUES        ;;?" "CR" & "LF"
2047      005720 000730          BR      1$           ;;TRY AGAIN
2048      005722 000000          $HIOCT: .WORD  0      ;;HIGH ORDER BITS GO HERE
2049      ;
2050      ; INPUT OCTAL NUMBER ROUTINE
2051      ;-----
2052      ;
2053      005724 010546          $INPUT: MOV     R5,-(SP)      ; SAVE REGISTER R5.
2054      005726 016605 000002 MOV     2(SP),R5        ; GET FIRST PARAMETER ADDRESS.
2055      005732 012537 005770 MOV     (R5)+,WHAT      ; GET MESSAGE ADDRESS.
2056      005736 012537 006050 MOV     (R5)+,LOLIM     ; GET LOW LIMIT FOR THE #
2057      005742 012537 006052 MOV     (R5)+,HILIM     ; GET HIGH LIMIT FOR THE #.

```

```

2058 005746 012537 006054      MOV      (R5)+,WHERE      ; GET ADDRESS OF INBUFFER
2059 005752 112537 006056      MOVVB   (R5)+,LOBITS     ; GET LOWMASK BITS.
2060 005756 112537 006057      MOVVB   (R5)+,ADRCNT    ; GET # OF #'S TO BE GENERATED.
2061 005762 010566 000002      MOV      R5,2(SP)       ; SAVE THE RETURN ADDRESS.
2062 005766 104401                INLP1:  TYPE             ; TYPE THE MESSAGE.
2063 005770 000000                WHAT:   .WORD           0
2064 005772 104404                RDOCT   .WORD           0
2065 005774 021637 006052      CMP      (SP),HILIM     ; READ OCTAL # FROM KEYBOARD.
2066 006000 003003                BGT     2$              ; IS IT IN HIGH LIMIT?
2067 006002 021637 006050      CMP      (SP),LOLIM    ; BRANCH IF NO.
2068 006006 002005                BGE     3$              ; IS IT MORE THAN LOW LIMIT.
2069 006010 104401 001312      2$:     TYPE            ,SQUES ; BRANCH IF YES.
2070 006014 104401 001313      TYPE            ,SCLF   ; TYPE "? "
2071 006020 000762                BR      INLP1          ; TYPE <CR>,<LF>
2072 006022 013705 006054      3$:     MOV      WHERE,R5 ; GET BUFFER ADDRESS.
2073 006026 011625                4$:     MOV      (SP),(R5)+ ; SAVE THE # IN RIGHT PLACE.
2074 006030 062716 000002      ADD      #2,(SP)       ; NEXT SEQUENTIAL NUMBER.
2075 006034 105337 006057      DECB    ADRCNT        ; COUNT BY 1.
2076 006040 001372                BNE     4$             ; BRANCH IF NOT DONE.
2077 006042 005726                TST     (SP)+          ; POP THE STACK POINTER.
2078 006044 012605                MOV     (SP)+,R5      ; POP THE REG.5
2079 006046 000002                RTI
2080 006050 000000                LOLIM:  .WORD           0
2081 006052 000000                HILIM:  .WORD           0
2082 006054 000000                WHERE:  .WORD           0
2083 006056          000                LOBITS: .BYTE           0
2084 006057          000                ADRCNT: .BYTE           0
2085
2086                ; ADVANCE TO NEXT TEST HANDLER
2087                ;-----
2088
2089 006060 013716 001442      .ADVANCE: MOV      NEXT,(SP) ; CRUNCH STACK WITH ADDRESSOF SCOPE CALL
2090 006064 005037 001444      CLR      LOCK          ; RESET TIGHT LOOP ADDRESS
2091 006070 000002                RTI                    ; CHECK TO SEE IF OLD TEST GETS REPEATED
2092
2093                ;SAVE PC OF TEST THAT FAILED AND R0-R5
2094                ;-----
2095
2096 006072 016637 000004 001460 .SAV05: MOV      4(SP),SAVPC ;SAVE R7 (PC)
2097
2098                ;SAVE R0-R5
2099
2100 006100 010537 001274      SV05:  MOV      R5,$REG5 ;SAVE R5
2101 006104 010437 001272      MOV      R4,$REG4 ;SAVE R4
2102 006110 010337 001270      MOV      R3,$REG3 ;SAVE R3
2103 006114 010237 001266      MOV      R2,$REG2 ;SAVE R2
2104 006120 010137 001264      MOV      R1,$REG1 ;SAVE R1
2105 006124 010037 001262      MOV      R0,$REG0 ;SAVE R0
2106 006130 000002                RTI                    ;LEAVE.
2107
2108                ;RESTORE R0-R5
2109
2110 006132 013700 001262      .RES05: MOV      $REG0,R0 ;RESTORE R0
2111 006136 013701 001264      MOV      $REG1,R1 ;RESTORE R1
2112 006142 013702 001266      MOV      $REG2,R2 ;RESTORE R2
2113 006146 013703 001270      MOV      $REG3,R3 ;RESTORE R3

```

CZKCF MACY11 30A(1052) 08-JUL-80 08:27 PAGE 46  
 CZKCF.F11 08-JUL-80 08:24 READ AN OCTAL NUMBER FROM THE TTY

SE2 0045

```

2114 006152 013704 001272      MOV    $REG4,R4      ;RESTORE R4
2115 006156 013705 001274      MOV    $REG5,R5      ;RESTORE R5
2116 006162 000002                RTI                    ;LEAVE
2117
2118      ;      ;CONVERT OCTAL NUMBER TO ASCII AND OUTPUT TO TELEPRINTER
2119      ;      ;-----
2120
2121 006164 104401 001313      .CONVR: TYPE        ,SCLRF
2122 006170 010046      .CNVRT: MOV          R0,-(SP)
2123 006172 010146      MOV          R1,-(SP)
2124 006174 010346      MOV          R3,-(SP)
2125 006176 010446      MOV          R4,-(SP)
2126 006200 010546      MOV          R5,-(SP)
2127 006202 017601 000012      MOV          @12(SP),R1
2128 006206 062766 000002 000012      ADD          #2,12(SP)
2129 006214 012137 006406      MOV          (R1)+,WRDCNT
2130 006220 112137 006410      1$:  MOVB       (R1)+,CHRCNT
2131 006224 112137 006411      MOVB       (R1)+,SPACNT
2132 006230 013137 006412      MOV          @ (R1)+,BINWRD
2133 006234 122737 000003 006410      CMPB       #3,CHRCNT
2134 006242 001003      BNE        2$
2135 006244 042737 177400 006412      BIC        #177400,BINWRD
2136 006252 013704 006412      2$:  MOV          BINWRD,R4
2137 006256 113705 006410      MOVB       CHRCNT,R5
2138 006262 012700 011234      MOV          #TEMP,R0
2139 006266 010403      3$:  MOV          R4,R3
2140 006270 042703 177770      BIC        #177770,R3
2141 006274 062703 000060      ADD          #060,R3
2142 006300 110320      MOVB       R3,(R0)+
2143 006302 000241      CLC
2144 006304 006004      ROR        R4
2145 006306 000241      CLC
2146 006310 006004      ROR        R4
2147 006312 000241      CLC
2148 006314 006004      ROR        R4
2149 006316 005305      DEC        R5
2150 006320 001362      BNE        3$
2151 006322 012703 011276      MOV          #MDATA,R3
2152 006326 114023      4$:  MOVB       -(R0),(R3)+
2153 006330 105337 006410      DECB       CHRCNT
2154 006334 001374      BNE        4$
2155 006336 105737 006411      TSTB       SPACNT
2156 006342 001405      BEQ        6$
2157 006344 112723 000040      5$:  MOVB       #040,(R3)+
2158 006350 105337 006411      DECB       SPACNT
2159 006354 001373      BNE        5$
2160 006356 105013      6$:  CLRB       (R3)
2161 006360 104401 011276      TYPE      ,MDATA
2162 006364 005337 006406      DEC        WRDCNT
2163 006370 001313      BNE        1$
2164 006372 012605      MOV        (SP)+,R5
2165 006374 012604      MOV        (SP)+,R4
2166 006376 012603      MOV        (SP)+,R3
2167 006400 012601      MOV        (SP)+,R1
2168 006402 012600      MOV        (SP)+,R0
2169 006404 000002      RTI

```

2170 006406 000000  
2171 006410 000000  
2172 006411 000000  
2173 006412 000000

WRDCNT: 0  
CHRCNT: 0  
SPACNT=CHRCNT+1  
BINWRD: 0

2174  
2175  
2176  
2177  
2178  
2179  
2180

:TRAP DISPATCH SERVICE  
:ARGUMENT OF TRAP IS EXTRACTED  
:AND USED AS OFFSET TO OBTAIN POINTER  
:TO SELECTED SUBROUTINE

2181

.SBTTL TRAP DECODER

2182  
2183

::\*\*\*\*\*  
:\*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.

2184  
2185  
2186

2187

2188

2189

006414 010046  
006416 016600 000002

\$TRAP: MOV R0,-(SP) ;;SAVE R0  
MOV 2(SP),R0 ;;GET TRAP ADDRESS  
TST -(R0) ;;BACKUP BY 2  
MOVB (R0),R0 ;;GET RIGHT BYTE OF TRAP  
ASL R0 ;;POSITION FOR INDEXING  
MOV \$TRPAD(R0),R0 ;;INDEX TO TABLE  
RTS R0 ;;GO TO ROUTINE

2190  
2191  
2192

2193

2194

006422 005740  
006424 111000  
006426 006300 006450  
006430 016000  
006434 000200

2195  
2196  
2197

::THIS IS USE TO HANDLE THE "GETPRI" MACRO

2198  
2199

2200

006436 011646  
006440 016666 000004 000002

\$TRAP2: MOV (SP),-(SP) ;;MOVE THE PC DOWN  
MOV 4(SP),2(SP) ;;MOVE THE PSW DOWN  
RTI ;;RESTORE THE PSW

2201  
2202  
2203

.SBTTL TRAP TABLE

2204  
2205

::\*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED  
:\*BY THE "TRAP" INSTRUCTION.

2206  
2207  
2208

2209

: ROUTINE  
:-----

2210  
2211

006450 006436  
006452 004414

\$TRPAD: .WORD \$TRAP2  
\$TYPE ;;CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE

2212  
2213  
2214

2215

006454 005144  
006456 005264

\$RDCHR ;;CALL=RDCHR TRAP+2(104402) TTY TYPEIN CHARACTER ROUTINE  
\$RDLIN ;;CALL=RDLIN TRAP+3(104403) TTY TYPEIN STRING ROUTINE  
\$RDOCT ;;CALL=RDOCT TRAP+4(104404) READ AN OCTAL NUMBER FROM TTY  
.SCOPI ;;CALL=SCOPI TRAP+5(104405) CALL TO LOOP ON CURRENT DATA HANDLER  
.SAVOS ;;CALL=SAVOS TRAP+6(104406) CALL TO REGISTER SAVE ROUTINE  
.RESOS ;;CALL=RESOS TRAP+7(104407) CALL TO REGISTER RESTORE ROUTINE  
.MSTCLR ;;CALL=MSTCLR TRAP+10(104410) CALL TO ISSUE A MASTER CLEAR  
.DELAY ;;CALL=DELAY TRAP+11(104411) CALL TO DELAY  
.ROMCLK ;;CALL=ROMCLK TRAP+12(104412) CALL TO CLOCK ROM ONCE  
.DATACLK ;;CALL=DATACLK TRAP+13(104413) CALL TO CLOCK DATA  
.TIMER ;;CALL=TIMER TRAP+14(104414) CALL TO DELAY A CLOCK TICK

2216  
2217  
2218  
2219  
2220  
2221  
2222  
2223  
2224  
2225

006460 005564  
006462 004364  
006464 006072  
006466 006132  
006470 007362  
006472 007332  
006474 007400  
006476 007446  
006500 007512

```

2226 006502 005724 $INPUT ;;CALL=INPUT TRAP+15(104415) CALL TO OCTAL # INPUT ROUTINE
2227 006504 006164 .CONVRT ;;CALL=CONVRT TRAP+16(104416) CALL TO .....
2228 006506 006170 .CNVRT ;;CALL=CNVRT TRAP+17(104417) CALL TO .....
2229 006510 006060 .ADVANCE ;;CALL=ADVANCE TRAP+20(104420) CALL TO ADVANCE TO NEXT TEST

```

```

2230
2231
2232 :-----:
2233 :*****:
2234 :ERROR HANDLER:
2235 :-----:

```

```

2236 006512 004737 011340 $ERROR: JSR PC,CKSWR ;CHECK FOR SOFT SWR
2237 006516 032777 010000 172514 BIT #SW12,@SWR ;BELL ON ERROR?
2238 006524 001406 BEQ XBX ;BR IF NO BELL
2239 006526 105777 172516 TSTB @STPS ;TTY READY.
2240 006532 100003 BPL XBX ;DON'T WAIT IF TTY NOT READY.
2241 006534 112777 000207 172510 MOVB #207,@STPB ;PUSH A BELL AT THE TTY.
2242 006542 032777 020000 172470 XBX: BIT #SW13,@SWR ;DELETE ERROR PRINT OUT?
2243 006550 001107 BNE HALTS ;BR IF NO PRINT OUT WANTED.
2244 006552 021637 001216 CMP (SP),$ERRPC ;WAS THIS ERROR FOUND LAST TIME?
2245 006556 001404 BEQ 1$ ;BR IF YES
2246 006560 011637 001216 MOV (SP),$ERRPC ;RECORD BEING HERE
2247 006564 105037 001203 CLRB $ERFLG ;PREPARE HEADER
2248 006570 104406 1$: SAVO5 ;SAVE ALL PROC REGISTERS
2249 006572 011605 MOV (SP),R5 ;GET THE PC OF ERROR
2250 006574 162705 000002 SUB #2,R5 ;GET ADDRESS OF TRAP CALL
2251 006600 011504 MOV (R5),R4 ;GET ERROR INSTRUCTION
2252 006602 110437 001214 MOVB R4,$ITEMB ; COPY ERROR # FOR APT HANDLING
2253 006606 006304 ASL R4 ;MULT BY TWO
2254 006610 061504 ADD (R5),R4 ;DOUBLE IT
2255 006612 006304 ASL R4 ;MULT AGAIN
2256 006614 042704 177001 BIC #177001,R4 ;CLEAR JUNK
2257 006620 062704 001512 ADD #$ERRTB,R4 ;GET POINTER
2258 006624 012437 006740 MOV (R4)+,ERRMSG ;GET ERROR MESSAGE
2259 006630 012437 006752 MOV (R4)+,DATAHD ;GET DATA HEADRER
2260 006634 011437 006764 MOV (R4),DATABP ;GET DATA TABLE
2261 006640 105737 001203 TSTB $ERFLG ;TYPE HEADREER
2262 006644 001403 BEQ TYPMSG ;BR IF YES
2263 006646 005737 006764 TST DATABP ;DOES DATA TABLE EXIST?
2264 006652 001040 BNE TYPDAT ;BR IF YES.
2265 006654 104401 001313 TYPMSG: TYPE ,SCRLF
2266 006660 104401 001313 TYPE ,SCRLF
2267 006664 005737 001444 TST LOCK
2268 006670 001402 BEQ 1$
2269 006672 104401 010015 TYPE ,MASTEK
2270 006676 104401 010003 1$: TYPE ,MTSTN
2271 006702 104417 007120 CNVRT ,XTSTN ;SHOW IT
2272 006706 104401 010072 TYPE ,MERRPC ;TYPE PC.
2273 006712 104417 007112 CNVRT ,ERTABO ;SHOW IT
2274 006716 104401 001313 TYPE ,SCRLF ;GIVE A CR/LF
2275 006722 112737 177777 001203 MOVB #-1,$ERFLG ;NO MORE HEADER UNLESS NO DATA TABLE.
2276 006730 005737 006740 TST ERRMSG ;IS THERE AN ERROR MESSAGE?
2277 006734 001402 BEQ WRKO.FM ;BR IF NO.
2278 006736 104401 TYPE ;TYPE
2279 006740 000000 ERRMSG: 0 ; ERROR MESSAGE
2280 006742 WRKO.FM: ;
2281 006742 005737 006752 TST DATAHD ;DATA HEADER?

```

|      |        |        |        |        |               |                            |  |  |                                    |
|------|--------|--------|--------|--------|---------------|----------------------------|--|--|------------------------------------|
| 2282 | 006746 | 001402 |        |        | BEQ           | TYPDAT                     |  |  | :BR IF NO                          |
| 2283 | 006750 | 104401 |        |        | TYPE          |                            |  |  | :TYPE                              |
| 2284 | 006752 | 000000 |        |        | DATAHD: 0     |                            |  |  | : DATA HEADER                      |
| 2285 | 006754 | 005737 | 006764 |        | TYPDAT: TST   | DATABP                     |  |  | :DATA TABLE?                       |
| 2286 | 006760 | 001402 |        |        | BEQ           | RESREG                     |  |  | :BR IF NO.                         |
| 2287 | 006762 | 104416 |        |        | CONVRT        |                            |  |  | :SHOW                              |
| 2288 | 006764 | 0000G0 |        |        | DATABP: 0     |                            |  |  | : DATA TABLE                       |
| 2289 | 006766 | 104407 |        |        | RESREG: RES05 |                            |  |  | :RESTORE PROC REGISTERS            |
| 2290 | 006770 | 122737 | 000001 | 001336 | HALTS: CMPB   | #APTENV,\$ENV              |  |  | : IS APT RUNNING ?                 |
| 2291 | 006776 | 001007 |        |        | BNE           | 3\$                        |  |  | : SKIP APT CALL IF NOT.            |
| 2292 | 007000 | 113737 | 001214 | 007012 | MOVB          | \$ITEMB,6\$                |  |  | : COPY ERROR #.                    |
| 2293 | 007006 | 004737 | 004714 |        | JSR           | PC,\$ATY4                  |  |  | : CALL APT SERVICES.               |
| 2294 | 007012 | 000000 |        |        | 6\$: .WORD    | 0                          |  |  | : ERROR # GOES HERE.               |
| 2295 | 007014 | 000777 |        |        | 9\$: BR       | 9\$                        |  |  | : LOCK HERE.                       |
| 2296 | 007016 | 022737 | 004070 | 000042 | 3\$: CMP      | #SENDAD,@#42               |  |  | : IF ACT-11 AUTOMATIC MODE, HALT!! |
| 2297 | 007024 | 001403 |        |        | BEQ           | 1\$                        |  |  |                                    |
| 2298 | 007026 | 005777 | 172206 |        | TST           | @SWR                       |  |  | :HALT ON ERROR?                    |
| 2299 | 007032 | 100005 |        |        | BPL           | EXITER                     |  |  | :BR IF NO HALT ON ERROR            |
| 2300 | 007034 | 010046 |        |        | 1\$: PUSHRO   |                            |  |  | :SAVE RO                           |
| 2301 | 007036 | 016600 | 000002 |        | MOV           | 2(SP),RO                   |  |  | :SHOW ERROR PC IN DATA LIGHTS      |
| 2302 | 007042 | 000000 |        |        | HALT          |                            |  |  | :HALT                              |
| 2303 | 007044 | 012600 |        |        | POPPO         |                            |  |  | :GET RO                            |
| 2304 | 007046 | 005237 | 001212 |        | EXITER: INC   | \$ERTTL                    |  |  | :UPDATE ERROR COUNT                |
| 2305 | 007052 | 032777 | 000400 | 172160 | BIT           | #SW08,@SWR                 |  |  | :GOTO TOP OF TEST?                 |
| 2306 | 007060 | 001007 |        |        | BNE           | 1\$                        |  |  | :BR IF YES                         |
| 2307 | 007062 | 032777 | 002000 | 172150 | BIT           | #SW10,@SWR                 |  |  | :GOTO NEXT TEST?                   |
| 2308 | 007070 | 001407 |        |        | BEQ           | 2\$                        |  |  | :BR IF NO                          |
| 2309 | 007072 | 013737 | 001442 | 001206 | MOV           | NEXT,\$LPADR               |  |  | :SET FOR NEXT TEST                 |
| 2310 | 007100 | 012706 | 001200 |        | 1\$: MOV      | #STACK,SP                  |  |  | :RESET SP                          |
| 2311 | 007104 | 000177 | 172076 |        | JMP           | @\$LPADR                   |  |  | :GOTO SPECIFIED TEST               |
| 2312 | 007110 | 000002 |        |        | 2\$: RTI      |                            |  |  | :\$LPADR                           |
| 2313 | 007112 | 000001 |        |        | ERTABO: 1     |                            |  |  |                                    |
| 2314 | 007114 | 006    | 002    |        | .BYTE         | 6,2                        |  |  |                                    |
| 2315 | 007116 | 001460 |        |        | SAVPC         |                            |  |  |                                    |
| 2316 | 007120 | 000001 |        |        | XTSTN: 1      |                            |  |  |                                    |
| 2317 | 007122 | 003    | 002    |        | .BYTE         | 3,2                        |  |  |                                    |
| 2318 | 007124 | 001202 |        |        | \$STNM        |                            |  |  |                                    |
| 2319 |        |        |        |        |               |                            |  |  | :ENTER HERE ON POWER FAILURE       |
| 2320 |        |        |        |        |               |                            |  |  | :-----                             |
| 2321 |        |        |        |        |               |                            |  |  |                                    |
| 2322 |        |        |        |        | .SBTTL        | POWER DOWN AND UP ROUTINES |  |  |                                    |
| 2323 |        |        |        |        |               |                            |  |  |                                    |
| 2324 |        |        |        |        |               |                            |  |  | :*****                             |
| 2325 |        |        |        |        |               |                            |  |  | :POWER DOWN ROUTINE                |
| 2326 | 007126 | 012737 | 007316 | 000024 | \$PWRDN: MOV  | #\$ILLUP,@#PWRVEC          |  |  | ::SET FOR FAST UP                  |
| 2327 | 007134 | 012737 | 000340 | 000026 | MOV           | #340,@#PWRVEC+2            |  |  | ::PRIO:7                           |
| 2328 | 007142 | 010046 |        |        | MOV           | RO,-(SP)                   |  |  | ::PUSH RO ON STACK                 |
| 2329 | 007144 | 010146 |        |        | MOV           | R1,-(SP)                   |  |  | ::PUSH R1 ON STACK                 |
| 2330 | 007146 | 010246 |        |        | MOV           | R2,-(SP)                   |  |  | ::PUSH R2 ON STACK                 |
| 2331 | 007150 | 010346 |        |        | MOV           | R3,-(SP)                   |  |  | ::PUSH R3 ON STACK                 |
| 2332 | 007152 | 010446 |        |        | MOV           | R4,-(SP)                   |  |  | ::PUSH R4 ON STACK                 |
| 2333 | 007154 | 010546 |        |        | MOV           | R5,-(SP)                   |  |  | ::PUSH R5 ON STACK                 |
| 2334 | 007156 | 017746 | 172056 |        | MOV           | @SWR,-(SP)                 |  |  | ::PUSH @SWR ON STACK               |
| 2335 | 007162 | 010637 | 007322 |        | MOV           | SP,\$SAVR6                 |  |  | ::SAVE SP                          |
| 2336 | 007166 | 012737 | 007200 | 000024 | MOV           | #\$PWRUP,@#PWRVEC          |  |  | ::SET UP VECTOR                    |
| 2337 | 007174 | 000000 |        |        | HALT          |                            |  |  |                                    |



```

2338 007176 000776          BR      -2          ;;HANG UP
2339
2340          ;:*****
2341          ;:POWER UP ROUTINE
2342 007200 012737 007316 000024 $PWRUP: MOV    #SILLUP,@#PWRVEC ;;SET FOR FAST DOWN
2343 007206 013706 007322          MOV    $SAVR6,SP      ;;GET SP
2344 007212 005037 007322          CLR    $SAVR6        ;;WAIT LOOP FOR THE TTY
2345 007216 005237 007322          1$:  INC    $SAVR6        ;;WAIT FOR THE INC
2346 007222 001375          BNE    1$            ;;OF WORD
2347 007224 104401 007562          TYPE   ,MPFAIL
2348 007230 104417 007324          CNVRT  ,PFTAB
2349 007234 105037 001203          CLRB  $ERFLG        ;CLEAR ERROR FLAG.
2350 007240 005037 001216          CLR   $ERRPC        ;CLEAR LAST ERROR PC
2351 007244 013701 002066          MOV   KMCSR,R1      ;RESTORE DEVICE ADDRESS.
2352 007250 005011          CLR   (R1)          ;CLEAR THE CSR.
2353 007252 104410          MSTCLR
2354 007254 012677 171760          MOV   (SP)+,@SWR    ;;POP STACK INTO @SWR
2355 007260 012605          MOV   (SP)+,R5      ;;POP STACK INTO R5
2356 007262 012604          MOV   (SP)+,R4      ;;POP STACK INTO R4
2357 007264 012603          MOV   (SP)+,R3      ;;POP STACK INTO R3
2358 007266 012602          MOV   (SP)+,R2      ;;POP STACK INTO R2
2359 007270 012601          MOV   (SP)+,R1      ;;POP STACK INTO R1
2360 007272 012600          MOV   (SP)+,R0      ;;POP STACK INTO R0
2361 007274 012737 007126 000024          MOV   #SPWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
2362 007302 012737 000340 000026          MOV   #340,@#PWRVEC+2 ;;PRIO:7
2363 007310 104401          TYPE   ,MPFAIL
2364 007312 007562          $PWRMG: .WORD MPFAIL ;;POWER FAIL MESSAGE POINTER
2365 007314 000002          RTI
2366 007316 000000          $SILLUP: HALT
2367 007320 000776          BR      -2          ;;THE POWER UP SEQUENCE WAS STARTED
2368 007322 000000          $SAVR6: 0          ;;BEFORE THE POWER DOWN WAS COMPLETE
2369          ;;PUT THE SP HERE
2370 007324 000001          PFTAB: 1
2371 007326 003 002          .BYTE 3,2
2372 007330 001202          .STSTNM
2373
2374 007332          .DELAY:
2375 007332 012777 000020 172534          MOV   #20,@KMPO4
2376 007340 104412          ROMCLK
2377 007342 121111          121111          ;:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2378 007344          1$:
2379 007344 104412          ROMCLK          ;:POKE CLOCK DELAY BIT
2380 007346 121224          121224          ;:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2381 007350 032777 000020 172516          BIT   #BIT4,@KMPO4 ;:PORT4 IBUS*11
2382 007356 001772          BEQ   1$            ;:IS CLOCK BIT SET?
2383 007360 000002          RTI              ;:BR IF NO
2384
2385 007362          .MSTCLR:
2386 007362 152777 000100 172500          BISB  #BIT6,@KMCSRH ;:SET MASTER CLEAR
2387 007370 142777 000300 172472          BICB  #BIT6!BIT7,@KMCSRH ;:CLEAR MASTER CLEAR AND RUN
2388 007376 000002          RTI              ;:RETURN
2389
2390 007400          .ROMCLK:
2391 007400 152777 000002 172462          BISB  #BIT1,@KMCSRH ;:SET ROMI
2392 007406 013677 172464          MOV   @ (SP)+,@KMPO6 ;:LOAD INSTRUCTION IN SEL6
2393 007412 062746 000002          ADD   #2,-(SP)      ;:ADJUST STACK

```

POWER DOWN AND UP ROUTINES

```

2394 007416 032777 000100 171614 BIT #SW06,@SWR ;HALT IF SW06 =1
2395 007424 001401 BEQ 1$ ;BR IF SW06 =0
2396 007426 000000 HALT ;HALT BEFORE CLOCKING INSTRUCTION
2397 007430 152777 000003 172432 1$: BISB #BIT1!BIT0,@KMCSRH ;CLOCK INSTRUCTION
2398 007436 142777 000007 172424 BICB #BIT2.BIT1!BIT0,@KMCSRH ;CLEAR ROMO, ROMI, STEP
2399 007444 000002 RTI

2400
2401 007446 .DATACLK:
2402 007446 013637 011234 MOV @(SP)+,TEMP ;PUT TICK COUNT IN TEMP
2403 007452 062746 000002 ADD #2,-(SP) ;ADJUST STACK
2404 007456 152777 000020 172404 1$: BISB #BIT4,@KMCSRH ;SET STEP LU
2405 007464 027777 172376 172374 CMP @KMCSR,@KMCSR ;WASTE TIME
2406 007472 142777 000020 172370 BICB #BIT4,@KMCSRH ;CLEAR STEP LU
2407 007500 005337 011234 DEC TEMP ;DEC TICK COUNT
2408 007504 001364 BNE 1$ ;BR IF NOT DONE
2409 007506 000002 RTI ;RETURN
2410 007510 000001 3$: .BLKW 1

2411
2412 007512 .TIMER:
2413 007512 013637 011234 MOV @(SP)+,TEMP ;MOVE COUNT TO TEMP
2414 007516 062746 000002 ADD #2,-(SP) ;ADJUST STACK
2415 007522 1$:
2416 007522 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2417 007524 021364 021364 ;PORT4 IBUS* REG11
2418 007526 032777 000002 172340 BIT #2,@KMP04 ;IS PGM CLOCK BIT CLEAR?
2419 007534 001772 BEQ 1$ ;BR IF YES
2420 2$:
2421 007536 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2422 007540 021364 021364 ;PORT4 IBUS* REG11
2423 007542 032777 000002 172324 BIT #2,@KMP04 ;IS PGM CLOCK BIT SET?
2424 007550 001372 BNE 2$ ;BR IF YES
2425 007552 005337 011234 DEC TEMP ;DEC COUNT
2426 007556 001361 BNE 1$ ;BR IF NOT DONE
2427 007560 000002 RTI ;RETURN
2428
2429 007562 050200 051127 043040 MPFAIL: .ASCIZ <200>/PWR FAILED. RESTART AT TEST /
(2) 007620 042600 042116 050040 MEPASS: .ASCIZ <200>/END PASS CZKCF /
(2) 007642 051200 000 MR: .ASCIZ <200>/R/
(2) 007645 200 047516 042040 MERR2: .ASCIZ <200>/NO DEVICES PRESENT./
(2) 007672 044600 051516 043125 MERR3: .ASCIZ <200>/INSUFFICIENT DATA!/
(2) 007716 046200 041517 020113 MLOCK: .ASCIZ <200>/LOCK ON SELECTED TEST/
(2) 007745 103 051123 020072 MCSRX: .ASCIZ /CSR: /
(2) 007753 126 041505 020072 MVECX: .ASCIZ /VEC: /
(2) 007761 120 051501 042523 MPASSX: .ASCIZ /PASSES: /
(2) 007772 051105 047522 051522 MERRX: .ASCIZ /ERRORS: /
(2) 010003 124 051505 020124 MTSTN: .ASCIZ /TEST NO: /
(2) 010015 052 000 MASTEK: .ASCIZ /*/
(2) 010017 200 042523 020124 MNEW: .ASCIZ <200>/SET SWITCH REG TO KMC11'S DESIRED ACTIVE./
(2) 010072 041520 020072 000 MERRPC: .ASCIZ /PC: /
(2) 010077 200 020040 020040 XHEAD: .ASCII <200>/
(2) 010136 020200 020040 020040 .ASCII <200>/
(2) 010175 200 020040 041520 .ASCII <200>/ PC CSR STAT1 STAT2 STAT3/
(2) 010247 200 026455 026455 .ASCIZ <200>/-----/
(2) 010323 200 047510 020127 NUM: .ASCIZ <200>/HOW MANY KMC11'S TO BE TESTED?/
(2) 010363 200 051503 020122 CSR: .ASCIZ <200>/CSR ADDRESS?/
(2) 010401 200 042526 052103 VEC: .ASCIZ <200>/VECTOR ADDRESS?/

```

```

(2) 010422 041200 020122 051120  PRI0: .ASCIZ <200>/BR PRIORITY LEVEL? (4,5,6,7)?/
(2) 010461 200 044127 041511  MODU: .ASCIZ <200>/WHICH LINE UNIT? IF NONE TYPE 'N', IF M8201 TYPE '1', IF M8202 TYP
(2) 010573 200 053523 052111  LINE: .ASCIZ <200>/SWITCH PAC#1 (DDCMP LINE #)?/
(2) 010631 200 053523 052111  BM: .ASCIZ <200>/SWITCH PAC#2 (BM873 BOOT ADD)?/
(2) 010671 200 051511 052040  CONN: .ASCIZ <200>/IS THE LOOP BACK CONNECTOR ON?/
(2) 010731 200 047516 042040  NOACT: .ASCIZ <200>/NO DEVICES ARE SELECTED/
(2) 010762 053600 044510 044103  MV35: .ASCIZ <200>/WHICH MODEM TYPE, TYPE 'D' FOR KMC11-DA (RS232C),OR/
(2) 011046 052200 050131 020105  .ASCIZ <200>/TYPE 'F' FOR KMC11-FA (V.35) ? /
(2) 011107 200 045600 041515  CONERR: .ASCIZ <200><200>/KMC11 AT NONSTANDARD ADDRESS PC: /
(2) 011154 042600 050130 041505  CNERR: .ASCIZ <200>/EXPECTED FOUND/
(2) 011175 040 045450 041515  KCM: .ASCIZ / (KMC) /
(2) 011206 000005  .EVEN
(2) 011206 000005  XSTATQ: 5
2430 011210 006 003  .BYTE 6,3
2431 011212 001276  $TMP0
2432 011214 006 003  .BYTE 6,3
2433 011216 001300  $TMP1
2434 011220 006 003  .BYTE 6,3
2435 011222 001302  $TMP2
2436 011224 006 003  .BYTE 6,3
2437 011226 001304  $TMP3
2438 011230 006 002  .BYTE 6,2
2439 011232 001306  $TMP4
2440  .EVEN
2441
2442  :BUFFERS FOR INPUT-OUTPUT
2443
2444 011234 000000  TEMP: 0
2445 011276  .=. +40
2446 011276 000000  MDATA: 0
2447 011340 000000  .=. +40
2448
2449
2450  :ROUTINE USED TO CHANGE SOFTWARE SWITCH
2451  :REGISTER USING THE CONSOLE TERMINAL
2452  :-----
2453
2454 011340 022737 000176 001240  CKSWR: CMP #SWREG,SWR ;IS THE SOFT SWR BEING USED?
2455 011346 001075  BNE CKSWR5 ;BR IF NO
2456 011350 132737 000001 001336  BITB #1,$ENV ; IS IT RUNNING UNDER APT?
2457 011356 001071  BNE CKSWR5 ; EXIT IF YES.
2458 011360 022777 000007 167660  CMP #7,@$TKB ;WAS CTRL G TYPED? (7 BIT ASCII)
2459 011366 001404  BEQ 1$ ;BR IF YES
2460 011370 022777 000207 167650  CMP #207,@$TKB ;WAS CTRL G TYPED? (8 BIT ASCII)
2461 011376 001061  BNE CKSWR5 ;BR IF NO
2462 011400 010246 1$: MOV R2,-(SP) ;STORE R2
2463 011402 010346 MOV R3,-(SP) ;STORE R3
2464 011404 010446 MOV R4,-(SP) ;STORE R4
2465 011406 012737 177777 011544 MOV #-1,SWFLG ;SET SOFT TYPE OUT FLAG
2466 011414 005002 CKSWR1: CLR R2 ;CLEAR NEW SWR CONTENTS
2467 011416 012704 177777 MOV #-1,R4 ;SET FLAG TO ALL ONES
2468 011422 104401 005541 TYPE ,SMSWR ;TYPE 'SWR= '
2469 011426 104417 CKSWR2: CNVRT ;TYPE OUT PRESENT CONTENTS
2470 011430 011600 SOFTSW ;OF SOFT SWITCH REGISTER
2471 011432 104401 005552 CKSWR3: TYPE ,SMNEW ;TYPE 'NEW? '

```

```

2472 011436 004737 011546      CKSWR4: JSR    PC,INCHAR      ;GET RESPONSE
2473 011442 022703 000015      CMP    #15,R3              ;WAS IT A CR?
2474 011446 001424              BEQ    5$                  ;BR IF YES
2475 011450 022703 000012      CMP    #12,R3              ;WAS IT A LF?
2476 011454 001416              BEQ    4$                  ;BR IF YES
2477 011456 022703 000025      CMP    #25,R3              ;WAS IT CTRL U?
2478 011462 001754              BEQ    CKSWR1              ;BR IF YES(START OVER)
2479 011464 022703 000007      CMP    #7,R3               ;IF CNTL G GET NEXT CHAR
2480 011470 001762              BEQ    CKSWR4
2481 011472 005004              CLR    R4                  ;IT MUST BE A DIGIT SO CLR FLAG
2482 011474 042703 177770      BIC    #177770,R3          ;ONLY 0-7 ARE LEGAL SO MASK OFF BITS
2483 011500 006302              ASL    R2                  ;SHIFT R2 3 TIMES
2484 011502 006302              ASL    R2
2485 011504 006302              ASL    R2
2486 011506 050302              BIS    R3,R2              ;ADD LAST DIGIT
2487 011510 000752              BR     CKSWR4              ;GET NEXT CHARACTER
2488 011512 012766 002402 000006 4$:  MOV    #.START,6(SP)      ;LF WAS TYPED SO GO TO START
2489 011520 005704              5$:  TST    R4              ;IS FLAG CLEAR?
2490 011522 001002              BNE    6$                  ;IF NOT DON'T CHANGE SOFT SWR
2491 011524 010277 167510      MOV    R2,@SWR             ;IF YES THEN WRITE NEW CONTENTS TO SOFT SWR
2492 011530 005037 011544      6$:  CLR    SWFLG            ;CLEAR TYPEOUT FLAG
2493 011534 012604              MOV    (SP)+,R4           ;RESTORE R4
2494 011536 012603              MOV    (SP)+,R3           ;RESTORE R3
2495 011540 012602              MOV    (SP)+,R2           ;RESTORE R2
2496 011542 000207      CKSWR5: RTS    PC          ;RETURN
2497
2498 011544 000000      SWFLG: 0
2499
2500 011546 105777 167472      INCHAR: TSTB   @$TKS
2501 011552 100375              BPL    -4
2502 011554 017703 167466      MOV    @$TKB,R3
2503 011560 105777 167464      TSTB   @$TPS
2504 011564 100375              BPL    -4
2505 011566 010377 167460      MOV    R3,@$TPB
2506 011572 042703 000200      BIC    #BIT7,R3
2507 011576 000207      RTS    PC
2508
2509 011600 000001      SOFTSW: 1
2510 011602 006 002          .BYTE 6,2
2511 011604 000176          SWREG

```

```

2512
2513
2514 ;ROUTINE USED TO 'CYCLE' THROUGH UP TO 16 KMC11'S
2515 ;THIS ROUTINE SETS UP THE CONTROL ADDRESS FOR THE DIAGNOSTIC
2516 ;AND RUNS THE SPECIFIED KMC11'S. THIS ROUTINE *MUST*
2517 ;BE RUN FIRST BEFORE ENTERING THE DIAGNOSTIC FOR THE
2518 ;SETUP NECESSARY.
2519 ;
2520
2521 011606 005737 001470 CYCLE: TST KMACTV ;ARE ANY KMC11'S TO BE TESTED?
2522 011612 001004 BNE 1$ ;BR IF OK.
2523 011614 104401 010731 TYPE ,NOACT ;NO KMC11'S SELECTED!!
2524 011620 000000 HALT ;STOP THE SHOW.
2525 011622 000776 BR -2 ;DISQUALIFY CONT. SW.
2526 011624 000241 1$: CLC ;CLEAR PROC. CARRY BIT.
2527 011626 006137 001500 ROL RUN ;UPDATE POINTER
2528 011632 005537 001500 ADC RUN ;CATCH CARRY FROM RUN
2529 011636 062737 000004 001504 ADD #4,MILK ;UPDATE POINTER
2530 011644 062737 000010 001502 ADD #10,CREAM ;UPDATE ADDRESS POINTER.
2531 011652 022737 002300 001502 CMP #KM.MAP+200,CREAM
2532 011660 001006 BNE 2$ ;KEEP GOING; NOT ALL TESTED FOR.
2533 011662 012737 002100 001502 MOV #KM.MAP,CREAM ;RESET ADDRESS POINTER.
2534 011670 012737 002302 001504 MOV #CNT.MAP,MILK ;RESET PASS COUNT POINTER
2535 011676 033737 001500 001470 2$: BIT RUN,KMACTV ;IS THIS ONE ACTIVE?
2536 011704 001747 BEQ 1$ ;BR IF NO
2537 011706 013700 001502 MOV CREAM,R0 ;GET ADDRESS POINTER
2538 011712 013702 001504 MOV MILK,R2 ;GET PASS COUNT POINTER
2539 011716 012037 002066 MOV (R0)+,KMCSR ;LOAD SYSTEM CTRL. REG
2540 011722 011037 002056 MOV (R0),KMRVEC ;LOAD VECTOR
2541 011726 042737 177000 002056 BIC #177000,KMRVEC ;CLEAR UNWANTED BITS
2542 011734 012037 002050 MOV (R0)+,STAT1 ;LOAD STAT1
2543 011740 012037 002052 MOV (R0)+,STAT2 ;LOAD STAT2
2544 011744 012037 002054 MOV (R0)+,STAT3 ;LOAD STAT3
2545 011750 012237 001324 MOV (R2)+,$PASS ;LOAD PASS COUNT
2546 011754 012237 001212 MOV (R2)+,$ERTTL ;LOAD ERROR COUNT
2547 011760 012700 000002 MOV #2,R0 ;SAVE CORE THIS WAY!
2548 011764 013737 002066 002070 MOV KMCSR,KMCSRH
2549 011772 005237 002070 INC KMCSRH
2550 011776 013737 002070 002072 MOV KMCSRH,KMCTL
2551 012004 005237 002072 INC KMCTL
2552 012010 013737 002072 002074 MOV KMCTL,KMPO4
2553 012016 060037 002074 ADD R0,KMPO4
2554 012022 013737 002074 002076 MOV KMPO4,KMPO6
2555 012030 060037 002076 ADD R0,KMPO6
2556
2557 012034 013737 002056 002060 MOV KMRVEC,KMRLVL ;PTY LVL
2558 012042 060037 002060 ADD R0,KMRLVL ;
2559 012046 013737 002060 002062 MOV KMRLVL,KMTVEC ;TX VEC
2560 012054 060037 002062 ADD R0,KMTVEC ;
2561 012060 013737 002062 002064 MOV KMTVEC,KMTLVL ;TX LVL
2562 012066 060037 002064 ADD R0,KMTLVL ;
2563
2564 012072 032737 000002 001446 BIT #SW01,STRTSW ;IS TEST NO. SELECTED
2565 012100 001447 BEQ 7$ ;BR IF NO
2566 012102 4$:
2567 012102 005737 000042 TST @#42 ;RUNNING IN AUTO MODE?

```

```

2568 012106 001044      BNE      7$          ;BR IF YES
2569 012110 104401 001313  TYPE      ,SCLRF
2570 012114 104415      INPUT
2571 012116 010003      MTSTN
2572 012120 000001      1
2573 012122 001000      1000
2574 012124 001202      $STNM
2575 012126      000      .BYTE
2576 012127      001      .BYTE
2577 012130 012700 014140  MOV      #TST1,R0
2578 012134 022710 5$:      CMP      (PC)+,(R0)      ;CMP FIRST WORD TO 12737
2579 012136 012737      MOV      (PC)+,@(PC)+
2580 012140 001020      BNE      6$          ;BR IF NOT SAME
2581 012142 023760 001202 000002  CMP      $STNM,2(R0)      ;DOES $STNM MATCH?
2582 012150 001014      BNE      6$          ;BR IF NO
2583 012152 022760 001202 000004  CMP      #$STNM,4(R0)      ;IS LAST WORD OK?
2584 012160 001010      BNE      6$          ;BR IF NO
2585 012162 010037 001206  MOV      R0,$LPADR      ;IT IS A LEGAL TEST SO DO IT
2586 012166 104401 007642  TYPE      ,MR
2587 012172 042737 000002 001446  BIC      #SW01,STRTSW
2588 012200 000412      BR
2589 012202 005720 6$:      TST      (R0)+          ;POP R0
2590 012204 020027 027716  CMP      R0,#TLAST+10      ;AT END YET?
2591 012210 001351      BNE      5$          ;BR IF NO
2592 012212 104401 001312  TYPE      , $QUES      ;YES ILLEGAL TEST NO.
2593 012216 000731      BR      4$          ;TRY AGAIN
2594
2595 012220 012737 014140 001206 7$:      MOV      #TST1,$LPADR      ;PREPARE $LPADR ADDRESS
2596 012226 013701 002066 8$:      MOV      KMC1,R1          ;R1 = BASE KMC11 ADDRESS
2597 012232 000177 166750  JMP      @ $LPADR      ;GO START TESTING.
2598
2599
2600
2601      ;ROUTINE USED TO 'AUTO SIZE' THE KMC11
2602      ;CSR AND VECTOR.
2603      ;NOTE: THE CSR MAY BE ANY WHERE IN THE FLOATING
2604      ;      ADDRESS RANGE (160000:164000)
2605      ;      AND THE VECTOR MAY BE ANY WHERE IN THE
2606      ;      FLOATING VECTOR RANGE (300:770)
2607      ;
2608
2609 012236      AUTO.SIZE:
2610 012236 000005      RESET
2611 012240 012702 002100  CSRMAP: MOV      #KM.MAP,R2      ;INSURE A BUS INIT.
2612 012244 005022 1$:      CLR      (R2)+          ;LOAD MAP POINTER.
2613 012246 022702 002300      CMP      #KM.END,R2      ;ZERO ENTIRE MAP
2614 012252 001374      BNE      1$          ;ALL DONE?
2615 012254 005037 001472      CLR      KMMNUM          ;BR IF NO
2616 012260 012702 002100      MOV      #KM.MAP,R2      ;SET OCTAL NUMBER OF KMC11'S TO 0
2617 012264 005037 001470      CLR      KMACTV          ;R2 POINTS TO KMC MAP
2618 012270 032737 000001 001446  BIT      #SW00,STRTSW      ;CLEAR ACTIVE
2619 012276 001002      BNE      +6          ;QUESTIONS?
2620 012300 000137 012740      JMP      7$          ;BR IF YES
2621 012304 012737 000001 001306  MOV      #1,$TMP4          ;IF NO SKIP QUESTIONS
2622 012312 104415      INPUT      ;START WITH 1
2623 012314 010323      NUM
2624 012316 000001      1

```

```

2624 012320 000020          16.
2625 012322 001302          $TMP2
2626 012324      000          .BYTE 0
2627 012325      001          .BYTE 1
2628 012326 013737 001302 001472 12$: MOV $TMP2,KMNUM ;KMNUM = HOW MANY
2629 012334 104401 001313          TYPE .$CRLF
2630 012340 104416          CONVRT ;TYPE WHICH KMC IS BEING DONE
2631 012342 013372          WHICH ;$TMP4 IS WHICH KMC
2632 012344 005237 001306          INC $TMP4
2633 012350 104415          INPUT
2634 012352 010363          CSR
2635 012354 160000          160000
2636 012356 164000          164000
2637 012360 001304          $TMP3
2638 012362      000          .BYTE 0
2639 012363      001          .BYTE 1
2640 012364 013722 001304          MOV $TMP3,(R2)+ ;STORE CSR IN MAP
2641 012370 104415          INPUT
2642 012372 010401          VEC
2643 012374 000000          0
2644 012376 000776          776
2645 012400 001304          $TMP3
2646 012402      000          .BYTE 0
2647 012403      001          .BYTE 1
2648 012404 013712 001304          MOV $TMP3,(R2) ;STORE VECTOR IN MAP
2649 012410 104401          TYPE 10$:
2650 012412 010422          PRIO ;ASK WHAT BR LEVEL
2651 012414 004737 013664          JSR PC,INTTY ;GET RESPONSE
2652 012420 022703 000024          CMP #24,R3
2653 012424 101014          BHI 50$ ;BR IF LESS THAN 4
2654 012426 022703 000027          CMP #27,R3
2655 012432 103411          BLO 50$ ;BR IF GREATER THAN 7
2656 012434 012704 000011          MOV #11,R4 ;R4 = NUMBER OF SHIFTS
2657 012440 006303          ASL R3 ;SHIFT R3 LEFT
2658 012442 005304          DEC R4 ;DEC SHIFT COUNT
2659 012444 001375          BNE -4 ;BR IF NOT DONE
2660 012446 042703 170777          BIC #170777,R3 ;BIC UNWANTED BITS
2661 012452 050312          BIS R3,(R2) ;PUT BR LEVEL IN STATUS MAP
2662 012454 000403          BR 8$ ;CONTINUE
2663 012456 104401          TYPE 50$:
2664 012460 001312          $QUES ;RESPONSE IS OUT OF LIMITS
2665 012462 000752          BR 10$ ;TRY AGAIN
2666 012464          8$:
2667 012464          9$:
2668 012464          16$:
2669
2670 012464 104401          TYPE
2671 012466 010461          MODU
2672 012470 004737 013664          JSR PC,INTTY ;ASK WHICH LINE UNIT
2673 012474 022703 000021          CMP #21,R3 ;GET REPLY
2674 012500 001417          BEQ 30$ ;'1'
2675 012502 022703 000022          CMP #22,R3 ;'2'
2676 012506 001412          BEQ 31$ ;'N'
2677 012510 022703 000116          CMP #116,R3
2678 012514 001403          BEQ 32$
2679 012516 104401          TYPE

```

```

2680 012520 001312          $QUES          ;IF NOT A 1,2 OR N TYPE "?"
2681 012522 000760          BR 16$          ;TRY AGAIN
2682 012524 052722 010000 32$: BIS #BIT12,(R2)+ ;SET BIT 12 IN STAT2 IF NO LU
2683 012530 022222          CMP (R2)+,(R2)+ ;POP OVER STAT2 AND STAT3
2684 012532 000475          BR 33$
2685 012534 052712 020000 31$: BIS #BIT13,(R2) ;SET BIT 13 IN STAT2 IF M8202
2686 012540 104401          30$: TYPE
2687 012542 010671          CONN          ;ASK IF LOOP-BACK IS ON
2688 012544 004737 013664 JSR PC,INTTY   ;GET REPLY
2689 012550 022703 000131  CMP #131,R3   ;Y
2690 012554 001406          BEQ 17$
2691 012556 022703 000116  CMP #116,R3   ;N
2692 012562 001436          BEQ 18$
2693 012564 104401          TYPE
2694 012566 001312          $QUES          ;IF NOT Y OR N TYPE "?"
2695 012570 000763          BR 30$          ;TRY AGAIN
2696 012572 052722 040000 17$: BIS #BIT14,(R2)+ ;TURNAROUND IS CONNECTED
2697 012576 032762 020000 177776 BIT #BIT13,-2(R2) ; M8202?
2698 012604 001027          BNE 19$          ; BR IF YES.
2699 012606          440$:
2700
2701 012606 104401          TYPE          ; ASK QUESTION
2702 012610 010762          MV35          ; ABOUT MODEM TTYPE
2703 012612 004737 013664 JSR PC,INTTY   ; GET ANSWER.
2704 012616 122703 000104  CMPB #D,R3    ; IS IT DMC11-DA?
2705 012622 001004          BNE 442$       ; NO.
2706
2707 012624 042762 000004 000002 BIC #BIT2,2(R2) ; YES INDICATE IT IN STAT3
2708 012632 000411          BR 441$
2709
2710 012634 122703 000106 442$: CMPB #F,R3    ;IS IT A DMC11-FA (V.35)?
2711 012640 001403          BEQ 443$       ; YES-TAKE CARE OF IT.
2712
2713 012642 104401          TYPE          ; NO ASK OPERATER WHATS GOING ON.
2714 012644 001312          $QUES
2715 012646 000757          BR 440$       ; REASK QUESTION
2716
2717 012650 052762 000004 000002 443$: BIS #BIT2,2(R2) ; YES V.35, RECORD IN STAT3
2718
2719          441$:          ; END DECISION POINT.
2720 012656 000402          BR 19$
2721 012660 042722 040000 18$: BIC #BIT14,(R2)+ ;NO TURNAROUND
2722 012664          19$:
2723 012664 104415          INPUT
2724 012666 010573          LINE
2725 012670 000000          0
2726 012672 000377          377
2727 012674 001304          $TMP3
2728 012676          000          .BYTE 0
2729 012677          001          .BYTE 1
2730 012700 113722 001304  MOVB $TMP3,(R2)+ ;STORE SWITCH PAC IN MAP
2731 012704 104415          INPUT
2732 012706 010631          BM
2733 012710 000000          0
2734 012712 000377          377
2735 012714 001304          $TMP3

```



```

2736 012716 000 .BYTE 0
2737 012717 001 .BYTE 1
2738 012720 113722 001304 MOVB $TMP3,(R2)+ ;STORE SWITCH PAC IN MAP
2739 012724 005722 TST (R2)+ ;POP OVER STAT3
2740 012726 005337 001302 33$: DEC $TMP2 ;DEC KMC COUNT
2741 012732 001200 BNE 12$ ;BR IF MORE TO DO
2742 012734 000137 013272 JMP 13$ ;CONTINUE
2743 012740 012701 160000 7$: MOV #160000,R1 ;SET FOR FIRST ADDRESS TO BE TESTED
2744 012744 012737 013364 000004 2$: MOV #6$,@#4 ;SET FOR NON-EXISTANT DEVICE TIME OUT
2745 012752 005011 CLR (R1) ;CLEAR SEL0
2746 012754 005711 TST (R1) ;IF KMC11 KMCSR S/B 0
2747 012756 001135 BNE 3$ ;IF NO DEV ; TRAP TO 4. IF NO BIT 8 THEN NO KMC11
2748 012760 005061 000006 CLR 6(R1) ;CLEAR SEL6
2749 012764 005761 000006 TST 6(R1) ;IF KMC11 THEN KMRIC S/B -0!
2750 012770 001130 BNE 3$ ;BR IF NOT KMC11
2751 012772 012711 002000 MOV #BIT10,(R1) ;SET ROMO
2752 012776 005061 000004 CLR 4(R1) ;CLEAR SEL4
2753 013002 012761 125252 000006 MOV #125252,6(R1) ;WRITE THIS TO SEL6
2754 013010 052711 020000 BIS #BIT13,(R1) ;WRITE IT!
2755 013014 022761 125252 000004 CMP #125252,4(R1) ;WAS IT WRITTEN?
2756 013022 001113 BNE 3$ ;IF NO IT IS NOT CRAM
2757 ;AT THIS POINT !T IS ASSUMED THAT R1 HOLDS A KMC11 CSR ADDRESS.
2758 013024 21$:
2759 013024 010122 22$: MOV R1,(R2)+ ;STORE CSR IN CORE TABLE.
2760 013026 012711 001000 15$: MOV #BIT9,(R1) ;CLEAR LINE UNIT LOOP
2761 013032 005061 000004 CLR 4(R1) ;CLEAR PORT4
2762 013036 012761 122113 000006 MOV #122113,6(R1) ;LOAD INSTRUCTION (CLR DTR)
2763 013044 052711 000400 BIS #BIT8,(R1) ;CLOCK INSTRUCTION
2764 013050 012761 021264 000006 MOV #021264,6(R1) ;LOAD INSTRUCTION
2765 013056 052711 000400 BIS #BIT8,(R1) ;CLOCK INSTRUCTION
2766 013062 122761 000377 000004 CMPB #377,4(R1) ;IS IT ALL ONES?
2767 013070 001003 .+10 ;BR IF NO
2768 013072 052712 010000 BIS #BIT12,(R2) ;IF YES, NO LINE UNIT, SET STATUS BIT
2769 013076 000436 BR 20$
2770 013100 032761 000002 000004 BIT #BIT1,4(R1) ;IS SWITCH A ONE?
2771 013106 001403 .+10 ;BR IF M9201
2772 013110 052712 060000 BIS #BIT13!BIT14,(R2) ;M9202 ASSUME CONNECTOR
2773 013114 000427 BR 20$ ;CONNECTOR ON)
2774 013116 032761 000010 000004 BIT #BIT3,4(R1) ;IS MRDY SET
2775 013124 001023 BNE 20$ ;BR IF M9201 NO CONNECTOR (ON LINE)
2776 013126 012761 000100 000004 MOV #BIT6,4(R1) ;LOAD PORT4
2777 013134 012761 122113 000006 MOV #122113,6(R1) ;LOAD INSTRUCTION
2778 013142 052711 000400 BIS #BIT8,(R1) ;CLOCK INSTRUCTION(SET DTR)
2779 013146 012761 021264 000006 MOV #021264,6(R1) ;LOAD INSTRUCTION
2780 013154 052711 000400 BIS #BIT8,(R1) ;CLOCK INSTRUCTION(READ MODEM REG)
2781 013160 032761 000010 000004 BIT #BIT3,4(R1) ;IS MRDY SET NOW?
2782 013166 001402 BEQ 20$ ;BR IF NO CONNECTOR
2783 013170 052712 040000 BIS #BIT14,(R2) ;SET STATUS BIT FOR CONNECTOR
2784 013174 005722 20$: TST (R2)+ ;POP POINTER
2785 013176 012761 021324 000006 MOV #021324,6(R1) ;PUT INSTRUCTION IN PORT6
2786 013204 012711 001400 MOV #BIT9!BIT8,(R1) ;PORT4_LU 15
2787 013210 156122 000004 BISB 4(R1),(R2)+ ;STORE DDCMP LINE # IN TABLE
2788 013214 012761 021344 000006 MOV #021344,6(R1) ;PORT6_INSTRUCTION
2789 013222 012711 001400 MOV #BIT8!BIT9,(R1) ;CLOCK_INSTR.
2790 013226 156122 000004 BISB 4(R1),(R2)+ ;STORE BMB73 ADD IN TABLE
2791 013232 005722 TST (R2)+ ;POP OVER STAT3
    
```

|      |        |        |        |        |             |                 |  |   |
|------|--------|--------|--------|--------|-------------|-----------------|--|---|
| 2792 | 013234 | 005011 |        |        | CLR         | (R1)            |  | ;CLEAR ROMI                                     |
| 2793 | 013236 | 005237 | 001472 |        | INC         | KMNUM           |  | ;UPDATE DEVICE COUNTER                          |
| 2794 | 013242 | 022737 | 000020 | 001472 | CMP         | #20,KMNUM       |  | ;ARE MAX. NO. OF DEV FOUND?                     |
| 2795 | 013250 | 001410 |        |        | BEQ         | 13\$            |  | ;YES DON'T LOOK FOR ANY MORE.                   |
| 2796 | 013252 | 005011 |        |        | 3\$: CLR    | (R1)            |  | ;CLEAR BIT 10                                   |
| 2797 | 013254 | 005061 | 000006 |        | CLR         | 6(R1)           |  | ;CLEAR SEL 6                                    |
| 2798 | 013260 | 062701 | 000010 |        | 14\$: ADD   | #10,R1          |  | ;UPDATE CSR POINTER ADDRESS                     |
| 2799 | 013264 | 022701 | 164000 |        | CMP         | #164000,R1      |  |   |
| 2800 | 013270 | 001230 |        |        | BNE         | 2\$             |  | ;BR IF MORE ADDRESS TO CHECK.                   |
| 2801 | 013272 | 005037 | 001470 |        | 13\$: CLR   | KMACTV          |  |   |
| 2802 | 013276 | 005737 | 001472 |        | TST         | KMNUM           |  | ;WERE ANY KMC11'S FOUND AT ALL?                 |
| 2803 | 013302 | 001423 |        |        | BEQ         | 5\$             |  | ;ERROR AUTO SIZER FOUND NO KMC11'S IN THIS SYS. |
| 2804 | 013304 | 013701 | 001472 |        | MOV         | KMNUM,R1        |  |   |
| 2805 | 013310 | 010137 | 001476 |        | MOV         | R1,SAVNUM       |  | ;SAVE NUMBER OF DEVICES                         |
| 2806 | 013314 | 000241 |        |        | 4\$: CLC    |                 |  |   |
| 2807 | 013316 | 006137 | 001470 |        | ROL         | KMACTV          |  | ;GENERATE ACTIVE REGISTER OF DEVICES.           |
| 2808 | 013322 | 005237 | 001470 |        | INC         | KMACTV          |  | ;SET THE BIT                                    |
| 2809 | 013326 | 005301 |        |        | DEC         | R1              |  |   |
| 2810 | 013330 | 001371 |        |        | BNE         | 4\$             |  | ;BR IF MORE TO GENERATE                         |
| 2811 | 013332 | 012737 | 000006 | 000004 | MOV         | #6,#4           |  | ;RESTORE TRAP VECTOR                            |
| 2812 | 013340 | 013737 | 001470 | 001474 | MOV         | KMACTV,SAVACT   |  | ;SAVE ACTIVE REGISTER                           |
| 2813 | 013346 | 000137 | 013400 |        | JMP         | VECMAP          |  | ;GO FIND THE VECTOR NOW.                        |
| 2814 | 013352 | 104401 | 007645 |        | 5\$: TYPE   | ,MERR2          |  | ;NOTIFY OPR THAT NO KMC11'S FOUND.              |
| 2815 | 013356 | 005000 |        |        | CLR         | RO              |  | ;MAKE DATA LIGHTS ZERO                          |
| 2816 | 013360 | 000000 |        |        | HALT        |                 |  | ;STOP THE SHOW                                  |
| 2817 | 013362 | 000776 |        |        | BR          | .-2             |  | ;DISABLE CONT. SW.                              |
| 2818 | 013364 | 012716 | 013260 |        | 6\$: MOV    | #14\$, (SP)     |  | ;ENTERED BY NON-EXISTANT TIME-OUT.              |
| 2819 | 013370 | 000002 |        |        | RTI         |                 |  | ;RETURN TO MAINSTREAM                           |
| 2820 |        |        |        |        |             |                 |  |   |
| 2821 | 013372 | 000001 |        |        | WHICH: 1    |                 |  |   |
| 2822 | 013374 | 002    | 002    |        | .BYTE       | 2,2             |  |   |
| 2823 | 013376 | 001306 |        |        | \$TMP4      |                 |  |   |
| 2824 |        |        |        |        |             |                 |  |   |
| 2825 | 013400 | 032737 | 000001 | 001446 | VECMAP: BIT | #SW00,STRTSW    |  |   |
| 2826 | 013406 | 001114 |        |        | BNE         | 5\$             |  |   |
| 2827 | 013410 | 012737 | 000340 | 000022 | MOV         | #340,#22        |  | ;SET IOT TRAP PRIO TO 7                         |
| 2828 | 013416 | 012737 | 013572 | 000020 | MOV         | #4\$,#20        |  | ;SET IOT TRAP VECTOR                            |
| 2829 | 013424 | 012702 | 002100 |        | MOV         | #KM.MAP,R2      |  | ;SET SOFTWARE POINTER                           |
| 2830 | 013430 | 012700 | 000300 |        | MOV         | #300,RO         |  | ;FLOATING VECTORS START HERE.                   |
| 2831 | 013434 | 012701 | 000302 |        | MOV         | #302,R1         |  | ;PC OF IOT INSTR.                               |
| 2832 | 013440 | 010120 |        |        | 1\$: MOV    | R1,(R0)+        |  | ;START FILLING VECTOR AREA                      |
| 2833 | 013442 | 012721 | 000004 |        | MOV         | #4,(R1)+        |  | ;WITH .+2; IOT                                  |
| 2834 | 013446 | 022021 |        |        | CMP         | (R0)+,(R1)+     |  | ;ADD 2 TO RO +R1                                |
| 2835 | 013450 | 020127 | 001000 |        | CMP         | R1,#1000        |  |   |
| 2836 | 013454 | 101771 |        |        | BLOS        | 1\$             |  | ;BR IF MORE TO FILL                             |
| 2837 | 013456 | 013737 | 001470 | 001276 | MOV         | KMACTV,\$TMP0   |  | ;STORE TEMPORALLY                               |
| 2838 | 013464 | 006037 | 001276 |        | 2\$: ROR    | \$TMP0          |  | ;BRING OUT A BIT                                |
| 2839 | 013470 | 103063 |        |        | BCC         | 5\$             |  | ;BR IF ALL DONE                                 |
| 2840 | 013472 | 012704 | 000012 |        | MOV         | #12,R4          |  | ;R4 IS INDEX REGISTER                           |
| 2841 | 013476 | 016437 | 013650 | 177776 | MOV         | BRLVL(R4),PS    |  | ;SET PS TO 7                                    |
| 2842 | 013504 | 011201 |        |        | MOV         | (R2),R1         |  |   |
| 2843 | 013506 | 012761 | 000200 | 000004 | MOV         | #200,4(R1)      |  |   |
| 2844 | 013514 | 012711 | 001000 |        | MOV         | #BIT9,(R1)      |  | ;SET ROMI                                       |
| 2845 | 013520 | 012761 | 121111 | 000006 | MOV         | #121111,6(R1)   |  | ;PUT INSTRUCTION IN PORT6                       |
| 2846 | 013526 | 012711 | 001400 |        | MOV         | #BIT9:BIT8,(R1) |  | ;FORCE AN INTERRUPT                             |
| 2847 | 013532 | 105200 |        |        | 7\$: INCB   | RO              |  | ;STALL  |

```

2848 013534 001376      BNE      -2      ;FOR TIME TO INTERRUPT
2849 013536 162704 000002  SUB      #2,R4    ;GET NEXT LOWEST PS LEVEL
2850 013542 001404      BEQ      6$      ;BR IF R4 = 0
2851 013544 016437 013650 177776  MOV      BRLVL(R4),PS ;MOVE NEXT LOWER LEVEL IN PS
2852 013552 000767      BR       7$      ;BR TO DELAY
2853 013554 052762 005300 000002 6$:  BIS      #5300,2(R2) ;NO INTERRUPT ASSUME 300 AT LEVEL 5 AND FIX KMC11 LATER
2854 013562 005011      3$:  CLR      (R1)    ;CLEAR FOMI
2855 013564 062702 000010      ADD      #10,R2   ;POP SOFTWARE POINTER
2856 013570 000735      BR       2$      ;KEEP GOING
2857 013572 051662 000002      4$:  BIS      (SP),2(R2) ;GET VECTOR ADDRESS
2858 013576 042762 000007 000002  BIC      #7,2(R2)  ;CLEAR JUNK
2859 013604 016405 013652      MOV      BRLVL+2(R4),R5 ;GET BR LEVEL OF KMC11
2860 013610 006305      ASL      R5      ;SHIFT LEVEL 4 PLACES
2861 013612 006305      ASL      R5      ;TO THE LEFT FOR THE
2862 013614 006305      ASL      R5      ;STATUS TABLE
2863 013616 006305      ASL      R5
2864 013620 042705 170777      BIC      #170777,R5 ;CLEAR UNWANTED BITS
2865 013624 050562 000002      BIS      R5,2(R2) ;PUT BR LEVEL IN STATUS TABLE
2866 013630 022626      CMP      (SP)+,(SP)+ ;POP ICT JUNK OFF STACK
2867 013632 012716 013562      MOV      #3$, (SP) ;SET FOR RETURN
2868 013636 000002      RTI
2869 013640 012737 004134 000020 5$:  MOV      #$$SCOPE,@#20 ; RESTORE SCOPE VECTOR
2870 013646 000207      RTS      PC      ;ALL DONE WITH "AUTO SIZING"
2871
2872 013650 000000      BRLVL:  PRO      ;LEVEL 0
2873 013652 000000      PRO      ;LEVEL 0
2874 013654 000200      PR4      ;LEVEL 4
2875 013656 000240      PR5      ;LEVEL 5
2876 013660 000300      PR6      ;LEVEL 6
2877 013662 000340      PR7      ;LEVEL 7
2878
2879
2880 013664 105777 165354  INTTY:  TSTB     @STKS ;WAIT FOR DONE
2881 013670 100375      BPL     -4
2882 013672 017703 165350      MOV     @STKB,R3 ;PUT CHAR IN R3
2883 013676 105777 165346      TSTB   @STPS ;WAIT UNTIL PRINTER IS READY
2884 013702 100375      BPL     -4
2885 013704 010377 165342      MOV     R3,@STPB ;ECHO CHAR
2886 013710 042703 000240      BIC     #BIT7!BITS,R3 ;MASK OFF LOWER CASE
2887 013714 000207      RTS     PC      ;RETURN
2888
2889 013716      APT.SIZE:
2890 013716 000005      RESET
2891 013720 010046      MOV     R0,-(SP) ;: PUSH R0 ON STACK
2892 013722 010146      MOV     R1,-(SP) ;: PUSH R1 ON STACK
2893 013724 010246      MOV     R2,-(SP) ;: PUSH R2 ON STACK
2894 013726 010346      MOV     R3,-(SP) ;: PUSH R3 ON STACK
2895 013730 005037 014132      CLR     VECTR    ; CLEAR THE LOCAL VARIABLE
2896 013734 005037 014136      CLR     PRIORITY ; CLEAN UP LOCAL VARIABLE
2897 013740 013700 001376      MOV     $CDW1,R0 ; GET THE DEVICE COUNT
2898 013744 010037 001476      MOV     R0,SAVNUM ; SAVE THE NO. OF DEVICES
2899 013750 012701 001346      MOV     #SMAMS1,R1 ; GET EXTRA INFO, BITS POINTER
2900 013754 013737 0G1372 014134      MOV     $BASE,BASE ; GET BASE CSR ADDRESS
2901 013762 113737 001366 014132      MOVB   $VECT1,VECTR ; GET THE VECTOR
2902 013770 113737 001367 014136      MOVB   $VECT1+1,PRIORITY ; GET THE PRIORITY
2903 013776 013737 001374 001470      MOV     $DEVN,KMACTV ; SAVE THE KMC'S SELECTED ACTIVE
    
```

```

2904 014004 013737 001470 001474 MOV KMACTV,SAVACT ; SAVE THE ACTIVE REGISTER
2905 014012 012702 001402 MOV #SDDW0,R2 ; GET ADDRESS OF FIRST DEVICE DESCRIPTOR WORD
2906 014016 012703 002100 MOV #KM.MAP,R3 ; GET POINTER TO DEVICE MAP
2907 014022 005023 3$: CLR (R3)+ ; CLEAR DEVICE MAP
2908 014024 022703 002300 CMP #KM.END,R3 ; IS WHOLE DEV.MAP CLEARED?
2909 014030 003374 BGT 3$ ; NO, THEN GO ON.
2910 014032 012703 002100 MOV #KM.MAP,R3 ; RESTORE DEV.MAP POINTER.
2911 014036 013723 014134 1$: MOV BASE,(R3)+ ; LOAD CSR ADDRESS
2912 014042 112163 000001 MOV#B (R1)+,1(R3) ; GET EXTRA INFO. BITS
2913 014046 006213 ASR (R3) ; SET IT IN RIGHT POSITION.
2914 014050 006213 ASR (R3) ; SET IT IN RIGHT POSITION.
2915 014052 053713 014136 BIS PRIRTY,(R3) ; GET PRIORITY IN STAT1
2916 014056 006313 ASL (R3) ; SET THEM IN RIGHT POSITION
2917 014060 006313 ASL (R3) ;
2918 014062 006313 ASL (R3) ;
2919 014064 006313 ASL (R3) ;
2920 014066 053723 014132 BIS VECTR,(R3)+ ; GET THE VECTOR IN STAT1.
2921 014072 012223 MOV (R2)+,(R3)+ ; GET THE STAT2 FROM DDWXX
2922 014074 005723 TST (R3)+ ; SK.P OVER STAT3
2923 014076 005300 DEC R0 ; COUNT BY 1
2924 014100 001407 BEQ 2$ ; ALL DONE?
2925 014102 062737 000010 014134 ADD #10,BASE ; INCREMENT BASE CSR ADDRESS BY 10
2926 014110 062737 000010 014132 ADD #10,VECTR ; INCREMENT VECTOR ADDRESS BY 10
2927 014116 000747 BR 1$ ; SET THE NEXT MAP ENTRY
2928 014120 2$:
2929 014120 012603 MOV (SP)+,R3 ;:POP STACK INTO R3
2930 014122 012602 MOV (SP)+,R2 ;:POP STACK INTO R2
2931 014124 012601 MOV (SP)+,R1 ;:POP STACK INTO R1
2932 014126 012600 MOV (SP)+,R0 ;:POP STACK INTO R0
2933 014130 000207 RTS PC ; RETURN
2934 014132 000000 VECTR: .WORD 0
2935 014134 000000 BASE: .WORD 0
2936 014136 000000 PRIRTY: .WORD 0
2937
2938
2939
2940 ***** TEST 1 *****
2941 ;*OUT CONTROL REGISTER READ/ONLY TEST
2942 ;*DO A MASTER CLEAR, VERIFY THAT ALL READ/ONLY
2943 ;*BITS ARE IN THE CORRECT STATE
2944 ;:*****
2945
2946 ; TEST 1
2947 ;-----
2948 ;:*****
2949 014140 000004 TST1: SCOPE
2950 014142 012737 000001 001202 MOV #1,$TSTNM ; LOAD THE NO. OF THIS TEST
2951 014150 012737 014214 001442 MOV #TST2,NEXT ; POINT TO THE START OF NEXT TEST.
2952 ;R1 CONTAINS BASE KMC11 ADDRESS
2953 014156 005077 165704 CLR @KMCSR ;CLEAR SELO
2954 014162 012702 000011 MOV #11,R2 ;SAVE R2 FOR TYPEOUT
2955 014166 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2956 014170 021224 021004!<20*11> ;PORT4 LINE UNIT REG 11
2957 014172 016104 000004 MOV 4(R1),R4 ;PUT 'FOUND' IN R4
2958 014176 042704 000054 BIC #54,R4 ;CLEAR UNKNOWN BITS
2959 014202 012705 000020 MOV #20,R5 ;PUT 'EXPECTED' IN R5

```

2960 014206 120504  
 2961 014210 001401  
 2962 014212 104002  
 2963 014214  
 2964  
 2965  
 2966  
 2967  
 2968  
 2969  
 2970  
 2971  
 2972  
 2973  
 2974  
 2975 014214 000004  
 2976 014216 012737 000002 001202  
 2977 014224 012737 014262 001442  
 2978  
 2979 014232 012702 000012  
 2980 014236 104412  
 2981 014240 021244  
 2982 014242 016104 000004  
 2983 014246 042704 000017  
 2984 014252 005005  
 2985 014254 120504  
 2986 014256 001401  
 2987 014260 104002  
 2988 014262  
 2989  
 2990  
 2991  
 2992  
 2993  
 2994  
 2995  
 2996  
 2997  
 2998  
 2999  
 3000 014262 000004  
 3001 014264 012737 000003 001202  
 3002 014272 012737 014334 001442  
 3003  
 3004 014300 104410  
 3005 014302 012702 000013  
 3006 014306 104412  
 3007 014310 021264  
 3008 014312 016104 000004  
 3009 014316 042704 000213  
 3010 014322 012705 000100  
 3011 014326 120504  
 3012 014330 001401  
 3013 014332 104002  
 3014 014334  
 3015

```

CMPB R5,R4 ;IS OUT READY SET?
BEQ 1$ ;BR IF YES
ERROR 2 ;ERROR IN LU 11
1$:

***** TEST 2 *****
; *IN CONTROL REGISTER READ/ONLY TEST
; *DO A MASTER CLEAR, VERIFY THAT ALL READ/ONLY
; *BITS ARE IN THE CORRECT STATE
; *****

; TEST 2
; -----
; *****
TST2: SCOPE ; LOAD THE NO. OF THIS TEST
MOV #2,$TSTNM ; POINT TO THE START OF NEXT TEST.
MOV #TST3,NEXT ; R1 CONTAINS BASE KMC11 ADDRESS
; R1 CONTAINS BASE KMC11 ADDRESS
MOV #12,R2 ;SAVE R2 FOR TYPEOUT
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
021004!<20*12> ;PORT4 LINE UNIT REG 12
MOV 4(R1),R4 ;PUT 'FOUND' IN R4
BIC #17,R4 ;CLEAR UNKNOWN BITS
CLR R5 ;PUT 'EXPECTED' IN R5
CMPB R5,R4 ;ARE ALL BITS CLEARED?
BEQ 1$ ;BR IF YES
ERROR 2 ;ERROR IN LU 12
1$:

***** TEST 3 *****
; *MODEM CONTROL REGISTER READ/ONLY TEST
; *DO A MASTER CLEAR, VERIFY THAT ALL READ/ONLY
; *BITS ARE IN THE CORRECT STATE
; *****

; TEST 3
; -----
; *****
TST3: SCOPE ; LOAD THE NO. OF THIS TEST
MOV #3,$TSTNM ; POINT TO THE START OF NEXT TEST.
MOV #TST4,NEXT ; R1 CONTAINS BASE KMC11 ADDRESS
; R1 CONTAINS BASE KMC11 ADDRESS
MSTCLR ;MASTER CLEAR KMC11
MOV #13,R2 ;SAVE R2 FOR TYPEOUT
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
021004!<20*13> ;PORT4 LINE UNIT REG 13
MOV 4(R1),R4 ;PUT 'FOUND' IN R4
BIC #213,R4 ;CLEAR UNKNOWN BITS
MOV #100,R5 ;PUT 'EXPECTED' IN R5
CMPB R5,R4 ;ARE RING, DTR, AND MODEM READY SET?
BEQ 1$ ;BR IF YES
ERROR 2 ;ERROR IN LU 13
1$:

```

3016  
3017  
3018  
3019  
3020  
3021  
3022  
3023  
3024  
3025  
3026  
3027  
3028  
3029  
3030  
3031  
3032  
3033  
3034  
3035  
3036  
3037  
3038  
3039  
3040  
3041  
3042  
3043  
3044  
3045  
3046  
3047  
3048  
3049  
3050  
3051  
3052  
3053  
3054  
3055  
3056  
3057  
3058  
3059  
3060  
3061  
3062  
3063  
3064  
3065  
3066  
3067  
3068  
3069  
3070  
3071

014334 000004  
014336 012737 000004 001202  
014344 012737 014426 001442  
  
014352 104410  
014354 012702 000017  
014360 104412  
014362 021364  
014364 016104 000004  
014370 042704 000206  
014374 012705 000051  
014400 032737 020000 002050  
014406 001404  
014410 042704 000040  
014414 042705 000040  
014420 120504  
014422 001401  
014424 104002  
014426

```
***** TEST 4 *****  
*MAINTENANCE REGISTER READ/ONLY TEST  
*DO A MASTER CLEAR, VERIFY THAT ALL READ/ONLY  
*BITS ARE IN THE CORRECT STATE  
*****  
: TEST 4  
-----  
:*****  
TST4: SCOPE ; LOAD THE NO. OF THIS TEST  
MOV #4,$TSTNM ; POINT TO THE START OF NEXT TEST.  
MOV #TST5,NEXT ; R1 CONTAINS BASE KMC11 ADDRESS  
; MASTER CLEAR KMC11  
MSTCLR ; SAVE R2 FOR TYPEOUT  
MOV #17,R2 ; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304  
ROMCLK ; PORT4 LINE UNIT REG 17  
021004!<20*17> ; PUT 'FOUND' IN R4  
MOV 4(R1),R4 ; CLEAR UNKNOWN BITS  
BIC #206,R4 ; PUT 'EXPECTED' IN R5  
MOV #51,R5 ; IS LU AN MB202 OR MB201?  
BIT #BIT13,STAT1 ; BR IF MB201  
BEQ .+12 ; MASK OFF SI BIT IF MB202  
BIC #40,R4 ; SI BIT IS UNKNOWN ON AN MB202  
BIC #BIT5,R5 ; ARE SI AND ICIR SET?  
CMPB R5,R4 ; BR IF YES  
BEQ 1$ ; ERROR IN LU 17  
ERROR 2  
1$:
```

```
***** TEST 5 *****  
*LINE UNIT REGISTER WRITE/READ TEST  
*SET BITS IN LU REGISTER 12, VERIFY IT IS SET  
*CLEAR BITS IN LU REGISTER 12, VERIFY IT IS CLEAR  
*****
```

: TEST 5  
-----

```
:*****  
TST5: SCOPE ; LOAD THE NO. OF THIS TEST  
MOV #5,$TSTNM ; POINT TO THE START OF NEXT TEST.  
MOV #TST6,NEXT ; ADDRESS FOR LOCK ON DATA.  
MOV #1$,LOCK ; R1 CONTAINS BASE KMC11 ADDRESS  
; MASTER CLEAR KMC11  
MSTCLR ; SAVE REGISTER ADDRESS FOR TYPEOUT  
MOV #12,R2 ; LOAD PORT4  
MOV #40,4(R1) ; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304  
ROMCLK ; SET BITS IN LU-12  
122112 ; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304  
ROMCLK ; READ LU-12  
021245 ; PUT 'EXPECTED' IN R5  
MOV #40,R5 ; PUT 'FOUND' IN R4  
MOV 5(R1),R4 ; CLEAR UNWANTED BITS  
BIC #357,R4 ; IS BITS SET?  
CMPB R5,R4
```

```

3072 014514 001401 BEQ 2$ :BR IF YES
3073 014516 104003 ERROR 3 :ERROR, BIT 5 IS NOT SET
3074 014520 104405 2$: SCOPI :SCOPE SUBTEST (SW09=1)
3075 014522 012737 014530 001444 MOV #3$,LOCK :NEW SCOPI
3076 014530 005061 000004 3$: CLR 4(R1) :LOAD PORT4
3077 014534 104412 ROMCLK :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3078 014536 122112 122112 :CLEAR BIT 5 IN LU-12
3079 014540 104412 ROMCLK :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3080 014542 021245 021245 :READ LU-12
3081 014544 005005 CLR R5 :PUT 'EXPECTED' IN R5
3082 014546 116104 000005 MOVB 5(R1),R4 :PUT 'FOUND' IN R4
3083 014552 042704 000337 BIC #337,R4 :CLEAR UNWANTED BITS
3084 014556 120504 CMPB R5,R4 :IS BIT5 CLEAR?
3085 014560 001401 BEQ 4$ :BR IF YES
3086 014562 104003 ERROR 3 :ERROR, BIT5 IS NOT CLEAR
3087 014564 104405 4$: SCOPI :SCOPE SUBTEST (SW09=1)
3088
3089
3090 :***** TEST 6 *****
3091 :*LINE UNIT REGISTER WRITE/READ TEST
3092 :*SET BIT1 IN LU REGISTER 17, VERIFY IT IS SET
3093 :*CLEAR BIT1 IN LU REGISTER 17, VERIFY IT IS CLEAR
3094 :*****
3095
3096 : TEST 6
3097 :-----
3098 :*****
3099 TST6: SCOPE :*****
3100 014566 000004 MOV #6,$TSTNM : LOAD THE NO. OF THIS TEST
3101 014570 012737 000006 001202 MOV #TST7,NEXT : POINT TO THE START OF NEXT TEST.
3102 014576 012737 014726 001442 MOV #1$,LOCK : ADDRESS FOR LOCK ON DATA.
3103 014604 012737 014620 001444 :R1 CONTAINS BASE KMC11 ADDRESS
3104 014612 104410 MSTCLR :MASTER CLEAR KMC11
3105 014614 012702 000017 MOV #17,R2 :SAVE REGISTER ADDRESS FOR TYPEOUT
3106 014620 012761 000001 000004 1$: MOV #1,4(R1) :LOAD PORT4
3107 014626 104412 ROMCLK :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3108 014630 122117 122117 :SET BIT1 IN LU-17
3109 014632 104412 ROMCLK :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3110 014634 021365 021365 :READ LU-17
3111 014636 012705 000001 MOV #1,R5 :PUT 'EXPECTED' IN R5
3112 014642 116104 000005 MOVB 5(R1),R4 :PUT 'FOUND' IN R4
3113 014646 042704 000376 BIC #376,R4 :CLEAR UNWANTED BITS
3114 014652 120504 CMPB R5,R4 :IS BIT1 SET?
3115 014654 001401 BEQ 2$ :BR IF YES
3116 014656 104003 ERROR 3 :ERROR, BIT 1 IS NOT SET
3117 014660 104405 2$: SCOPI :SCOPE SUBTEST (SW09=1)
3118 014662 012737 014670 001444 MOV #3$,LOCK :NEW SCOPI
3119 014670 005061 000004 3$: CLR 4(R1) :LOAD PORT4
3120 014674 104412 ROMCLK :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3121 014676 122117 122117 :CLEAR BIT 1 IN LU-17
3122 014700 104412 ROMCLK :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3123 014702 021365 021365 :READ LU-17
3124 014704 005005 CLR R5 :PUT 'EXPECTED' IN R5
3125 014706 116104 000005 MOVB 5(R1),R4 :PUT 'FOUND' IN R4
3126 014712 042704 000376 BIC #376,R4 :CLEAR UNWANTED BITS
3127 014716 120504 CMPB R5,R4 :IS BIT1 CLEAR?

```

```
3128 014720 001401 BEQ 4$ ;BR IF YES
3129 014722 104003 ERROR 3 ;ERROR, BIT1 IS NOT CLEAR
3130 014724 104405 4$: SCOPE1 ;SCOPE SUBTEST (SW09=1)
3131
3132
3133 ;***** TEST 7 *****
3134 ;*LINE UNIT REGISTER WRITE/READ TEST
3135 ;*FLOAT A 1 THROUGH LINE UNIT REGISTER 13
3136 ;*FLOAT A 0 THROUGH LINE UNIT REGISTER 13
3137 ;*****
3138
3139 ; TEST 7
3140 ;-----
3141 ;*****
3142 014726 000004 TST7: SCOPE
3143 014730 012737 000007 001202 MOV #7,$TSTNM ; LOAD THE NO. OF THIS TEST
3144 014736 012737 015136 001442 MOV #TST10,NEXT ; POINT TO THE START OF NEXT TEST.
3145 014744 012737 014764 001444 MOV #64$,LOCK ; ADDRESS FOR LOCK ON DATA.
3146 ;R1 CONTAINS BASE KMC11 ADDRESS
3147 014752 104410 MSTCLR ;MASTER CLEAR KMC11
3148 014754 012702 000013 MOV #13,R2 ;SAVE REGISTER ADDRESS FOR TYPEOUT
3149 014760 012700 000001 MOV #1,R0 ;START WITH BIT 0
3150 014764 64$:
3151 014764 010061 000004 MOV R0,4(R1) ;PUT PATTERN INTO PORT4
3152 014770 042761 000257 000004 BIC #257,4(R1) ;CLEAR UNWANTED BITS
3153 014776 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3154 015000 122113 122100!13 ;MOV DATA TO IBUS REGISTER 13
3155 015002 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3156 015004 021265 21005!<13*20> ;READ FROM IBUS REGISTER 13
3157 015006 010005 MOV R0,R5 ;PUT EXPECTED IN R5
3158 015010 042705 000257 BIC #257,R5 ;CLEAR UNWANTED BITS
3159 015014 116104 000005 MOVB 5(R1),R4 ;PUT "FOUND" INTO R4
3160 015020 042704 000257 BIC #257,R4 ;CLEAR UNWANTED BITS
3161 015024 120504 CMPB R5,R4 ;DATA CORRECT?
3162 015026 001401 BEQ 65$ ;BR IF YES
3163 015030 104003 ERROR 3 ;ERROR
3164 015032 104405 65$: SCOPE1 ;SW09=1?
3165 015034 000241 CLC ;CLEAR CARRY
3166 015036 106100 ROLB R0 ;SHIFT BIT IN R0
3167 015040 001351 BNE 64$ ;IF R0=0 THEN DONE
3168 015042 012737 015056 001444 MOV #67$,LOCK ;NEW SCOPE1
3169 015050 012700 000001 MOV #1,R0 ;START WITH BIT 0
3170 015054 005100 69$: COM R0 ;CHANGE TO FLOATING ZERO
3171 015056 67$:
3172 015056 010061 000004 MOV R0,4(R1) ;PUT PATTERN INTO PORT4
3173 015062 042761 000257 000004 BIC #257,4(R1) ;CLEAR UNWANTED BITS
3174 015070 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3175 015072 122113 122100!13 ;MOV DATA TO IBUS REGISTER 13
3176 015074 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3177 015076 021265 21005!<13*20> ;READ FROM IBUS REGISTER 13
3178 015100 010005 MOV R0,R5 ;PUT EXPECTED IN R5
3179 015102 042705 000257 BIC #257,R5 ;CLEAR UNWANTED BITS
3180 015106 116104 000005 MOVB 5(R1),R4 ;PUT "FOUND" INTO R4
3181 015112 042704 000257 BIC #257,R4 ;CLEAR UNWANTED BITS
3182 015116 120504 CMPB R5,R4 ;DATA CORRECT?
3183 015120 001401 BEQ 68$ ;BR IF YES
```



LINE UNIT WRITE/READ TESTS

3184 015122 104003  
 3185 015124 104405  
 3186 015126 005100  
 3187 015130 000241  
 3188 015132 106100  
 3189 015134 001347  
 3190  
 3191  
 3192  
 3193  
 3194  
 3195  
 3196  
 3197  
 3198  
 3199  
 3200  
 3201 015136 000004  
 3202 015140 012737 000010 001202  
 3203 015146 012737 015312 001442  
 3204 015154 012737 015174 001444  
 3205  
 3206 015162 104410  
 3207 015164 012702 000014  
 3208 015170 012700 000001  
 3209 015174  
 3210 015174 010061 000004  
 3211 015200 104412  
 3212 015202 122114  
 3213 015204 104412  
 3214 015206 021305  
 3215 015210 010005  
 3216 015212 116104 000005  
 3217 015216 120504  
 3218 015220 001401  
 3219 015222 104003  
 3220 015224 104405  
 3221 015226 000241  
 3222 015230 106100  
 3223 015232 001360  
 3224 015234 012737 015250 001444  
 3225 015242 012700 000001  
 3226 015246 005100  
 3227 015250  
 3228 015250 010061 000004  
 3229 015254 104412  
 3230 015256 122114  
 3231 015260 104412  
 3232 015262 021305  
 3233 015264 010005  
 3234 015266 116104 000005  
 3235 015272 120504  
 3236 015274 001401  
 3237 015276 104003  
 3238 015300 104405  
 3239 015302 005100

```

68$: ERROR 3 ;ERROR
SCOPI ;SW09=1?
COM RO ;CHANGE TO FLOATING 1
CLC ;CLEAR CARRY
ROLB RO ;SHIFT BIT IN RO
BNE 69$ ;IF RO=0 THEN DONE

:***** TEST 10 *****
:*LINE UNIT REGISTER WRITE/READ TEST
:*FLOAT A 1 THROUGH LINE UNIT REGISTER 14
:*FLOAT A 0 THROUGH LINE UNIT REGISTER 14
:*****

; TEST 10
-----
:*****
TST10: SCOPE ;
MOV #10,$STNM ; LOAD THE NO. OF THIS TEST
MOV #TST11,NEXT ; POINT TO THE START OF NEXT TEST.
MOV #64$,LOCK ; ADDRESS FOR LOCK ON DATA.
;R1 CONTAINS BASE KMC11 ADDRESS
MSTCLR ;MASTER CLEAR KMC11
MOV #14,R2 ;SAVE REGISTER ADDRESS FOR TYPEOUT
MOV #1,RO ;START WITH BIT 0

64$: MOV RO,4(R1) ;PUT PATTERN INTO PORT4
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
122100!14 ;MOV DATA TO IBUS REGISTER 14
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
21005!<14*20> ;READ FROM IBUS REGISTER 14
MOV RO,R5 ;PUT EXPECTED IN R5
MOVB 5(R1),R4 ;PUT "FOUND" INTO R4
CMPB R5,R4 ;DATA CORRECT?
BEQ 65$ ;BR IF YES
ERROR 3 ;ERROR
65$: SCOPI ;SW09=1?
CLC ;CLEAR CARRY
ROLB RO ;SHIFT BIT IN RO
BNE 64$ ;IF RO=0 THEN DONE
MOV #67$,LOCK ;NEW SCOPI
MOV #1,RO ;START WITH BIT 0
69$: COM RO ;CHANGE TO FLOATING ZERO
67$: MOV RO,4(R1) ;PUT PATTERN INTO PORT4
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
122100!14 ;MOV DATA TO IBUS REGISTER 14
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
21005!<14*20> ;READ FROM IBUS REGISTER 14
MOV RO,R5 ;PUT EXPECTED IN R5
MOVB 5(R1),R4 ;PUT "FOUND" INTO R4
CMPB R5,R4 ;DATA CORRECT?
BEQ 68$ ;BR IF YES
ERROR 3 ;ERROR
68$: SCOPI ;SW09=1?
COM RO ;CHANGE TO FLOATING 1

```

|      |        |        |  |  |      |      |  |                    |
|------|--------|--------|--|--|------|------|--|--------------------|
| 3240 | 015304 | 000241 |  |  | CLC  |      |  | :CLEAR CARRY       |
| 3241 | 015306 | 106100 |  |  | ROLB | RO   |  | :SHIFT BIT IN RO   |
| 3242 | 015310 | 001356 |  |  | BNE  | 69\$ |  | :IF RO=0 THEN DONE |

```

3243
3244
3245 :***** TEST 11 *****
3246 :*SWITCH PAC TEST
3247 :*THIS TEST READS SWITCH PAC#1
3248 :*THIS SWITCH PAC CONTAINS THE DDCMP LINE #
3249 :*****

```

```

3250
3251 : TEST 11
3252 :-----
3253 :*****

```

|      |        |        |        |        |              |             |  |   |
|------|--------|--------|--------|--------|--------------|-------------|--|---|
| 3254 | 015312 | 000004 |        |        | TST11: SCOPE |             |  |   |
| 3255 | 015314 | 012737 | 000011 | 001202 | MOV          | #11,\$STNM  |  | : LOAD THE NO. OF THIS TEST               |
| 3256 | 015322 | 012737 | 015354 | 001442 | MOV          | #TST12,NEXT |  | : POINT TO THE START OF NEXT TEST.        |
| 3257 |        |        |        |        |              |             |  | :R1 CONTAINS BASE KMC11 ADDRESS           |
| 3258 | 015330 | 104410 |        |        | MSTCLR       |             |  | :MASTER CLEAR KMC11                       |
| 3259 | 015332 | 104412 |        |        | ROMCLK       |             |  | :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 |
| 3260 | 015334 | 021324 |        |        | 021324       |             |  | :PORT4 LU15                               |
| 3261 | 015336 | 016104 | 000004 |        | MOV          | 4(R1),R4    |  | :PUT 'FOUND' IN R4                        |
| 3262 | 015342 | 113705 | 002052 |        | MOVB         | STAT2,R5    |  | :PUT 'EXPECTED' IN R5                     |
| 3263 | 015346 | 120504 |        |        | CMPB         | R5,R4       |  | :SW OK?                                   |
| 3264 | 015350 | 001401 |        |        | BEQ          | 1\$         |  | :BR IF YES                                |
| 3265 | 015352 | 104031 |        |        | ERROR        | 31          |  | :ERROR, SWITCH PAC READ ERROR             |
| 3266 | 015354 |        |        |        | 1\$:         |             |  |   |

```

3267
3268 :***** TEST 12 *****
3269 :*SWITCH PAC TEST
3270 :*THIS TEST READS SWITCH PAC#2
3271 :*THIS SWITCH PAC CONTAINS THE BM873 BOOT ADD
3272 :*****
3273
3274
3275 : TEST 12
3276 :-----
3277 :*****

```

|      |        |        |        |        |              |             |  |   |
|------|--------|--------|--------|--------|--------------|-------------|--|---|
| 3278 | 015354 | 000004 |        |        | TST12: SCOPE |             |  |   |
| 3279 | 015356 | 012737 | 000012 | 001202 | MOV          | #12,\$STNM  |  | : LOAD THE NO. OF THIS TEST               |
| 3280 | 015364 | 012737 | 015416 | 001442 | MOV          | #TST13,NEXT |  | : POINT TO THE START OF NEXT TEST.        |
| 3281 |        |        |        |        |              |             |  | :R1 CONTAINS BASE KMC11 ADDRESS           |
| 3282 | 015372 | 104410 |        |        | MSTCLR       |             |  | :MASTER CLEAR KMC11                       |
| 3283 | 015374 | 104412 |        |        | ROMCLK       |             |  | :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 |
| 3284 | 015376 | 021344 |        |        | 021344       |             |  | :PORT4 LU16                               |
| 3285 | 015400 | 016104 | 000004 |        | MOV          | 4(R1),R4    |  | :PUT 'FOUND' IN R4                        |
| 3286 | 015404 | 113705 | 002053 |        | MOVB         | STAT2+1,R5  |  | :PUT 'EXPECTED' IN R5                     |
| 3287 | 015410 | 120504 |        |        | CMPB         | R5,R4       |  | :SW OK?                                   |
| 3288 | 015412 | 001401 |        |        | BEQ          | 1\$         |  | :BR IF YES                                |
| 3289 | 015414 | 104031 |        |        | ERROR        | 31          |  | :ERROR, SWITCH PAC READ ERROR             |
| 3290 | 015416 |        |        |        | 1\$:         |             |  |   |

```

3291
3292 :***** TEST 13 *****
3293 :*LINE UNIT CLOCK TEST
3294 :*THIS TEST VERIFYS THAT THE LU INTERNAL CLOCK
3295

```

```

3296                ;*(BIT 1 IN LU-17) IS WORKING
3297                ;:*****
3298
3299                ; TEST 13
3300                ;-----
3301                ;:*****
3302 TST13: SCOPE
3303 015416 000004      MOV      #13,$STSTM      ; LOAD THE NO. OF THIS TEST
3304 015420 012737 000013 001202      MOV      #TST14,NEXT    ; POINT TO THE START OF NEXT TEST.
3305 015426 012737 015516 001442      ;R1 CONTAINS BASE KMC11 ADDRESS
3306 015434 104410      MSTCLR
3307 015436 005037 011234      CLR      TEMP          ;MASTER CLEAR KMC11
3308 015442            1$:
3309 015442 104412      ROMCLK      ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3310 015444 021364      021364      ;PORT4 LU-17
3311 015446 032761 000002 000004      BIT      #2,4(R1)      ;IS CLOCK BIT SET?
3312 015454 001004      BNE      2$            ;BR IF YES
3313 015456 005237 011234      INC      TEMP          ;DELAY
3314 015462 001367      BNE      1$            ;DELAY FINISHED?
3315 015464 104004      ERROR    4            ;ERROR BIT IS STUCK CLEAR
3316 015466 005037 011234      CLR      TEMP          ;PREPARE FOR DELAY
3317 015472            2$:
3318 015472 104412      ROMCLK      ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3319 015474 021364      021364      ;PORT4 LU-17
3320 015476 032761 000002 000004      BIT      #2,4(R1)      ;IS CLOCK BIT CLEAR?
3321 015504 001404      BEQ      4$            ;BR IF YES
3322 015506 005237 011234      INC      TEMP          ;DELAY
3323 015512 001367      BNE      3$            ;BR IF DELAY NOT DONE
3324 015514 104004      ERROR    4            ;ERROR BIT IS STUCK SET
3325 015516            4$:
3326
3327
3328                ;***** TEST 14 *****
3329                ;*OUT DATA SILO TEST
3330                ;*SET SOM AND LOAD OUT DATA SILO
3331                ;*VERIFY THAT OCOR SET, INDICATING THAT THE
3332                ;*CHARACTER IS AT THE BOTTOM OF THE OUT SILO
3333                ;:*****
3334
3335                ; TEST 14
3336                ;-----
3337                ;:*****
3338 TST14: SCOPE
3339 015516 000004      MOV      #14,$STSTM      ; LOAD THE NO. OF THIS TEST
3340 015520 012737 000014 001202      MOV      #TST15,NEXT    ; POINT TO THE START OF NEXT TEST.
3341 015526 012737 015616 001442      ;R1 CONTAINS BASE KMC11 ADDRESS
3342 015534 104410      MSTCLR
3343 015536 012711 004000      MOV      #BIT11,(R1)    ;MASTER CLEAR KMC11
3344 015542 012761 000001 000004      MOV      #1,4(R1)      ;SET LINE UNIT LOOP
3345 015550 104412      ROMCLK      ;LOAD PORT4 WITH BIT0
3346 015552 122111      122111      ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3347 015554 104412      ROMCLK      ;SET SOM
3348 015556 122110      122110      ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3349 015560 104414 000002      TIMER, 2            ;LOAD OUT DATA SILO
3350 015564 012702 000017      MOV      #17,R2        ;WAIT FOR OCOR
3351 015570 104412      ROMCLK      ;SAVE ADDRESS FOR TYPEOUT
3352                ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304

```

```

3352 015572 021364          021364          ;PORT4 LU 17
3353 015574 016104 000004    MOV      4(R1),R4      ;PUT 'FOUND' IN R4
3354 015600 042704 000357    BIC      #357,R4      ;CLEAR UNWANTED BITS
3355 015604 012705 000020    MOV      #20,R5       ;PUT 'EXPECTED' IN R5
3356 015610 120504          CMPB     R5,R4         ;IS OCOR SET?
3357 015612 001401          BEQ      1$           ;BR IF YES
3358 015614 104005          ERROR    5
3359 015616
1$:
3360
3361
3362          ;***** TEST 15 *****
3363          ;*DDCMP TEST OF RTS AND OUT ACTIVE
3364          ;*SET SOM AND LOAD OUT DATA SILO
3365          ;*SINGLE STEP 2 DATA CLOCKS, VERIFY
3366          ;*THAT RTS AND ACTIVE ARE SET
3367          ;:*****
3368
3369          ; TEST 15
3370          ;-----
3371          ;:*****
3372 015616 000004          ;TST15: SCOPE
3373 015620 012737 000015 001202    MOV      #15,$STNM      ; LOAD THE NO. OF THIS TEST
3374 015626 012737 015754 001442    MOV      #TST16,NEXT    ; POINT TO THE START OF NEXT TEST.
3375          ;R1 CONTAINS BASE KMC11 ADDRESS
3376 015634 104410          MSTCLR
3377 015636 012711 004000          MOV      #BIT11,(R1)    ;MASTER CLEAR KMC11
3378 015642 012761 000001 000004    MOV      #1,4(R1)       ;SET LINE UNIT LOOP
3379 015650 104412          ROMCLK
122111          ;LOAD PORT4 WITH BIT0
3380 015652 122111          ROMCLK                ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
122111          ;SET SOM
3381 015654 104412          ROMCLK                ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
122110          ;LOAD OUT DATA SILO
3382 015656 122110          JSR      PC,OCOR        ;WAIT FOR OCOR
3383 015660 004737 030260          DATACL, 2            ;CLOCK DATA FOUR TIMES
3384 015664 104413 000002          MOV      #11,R2        ;SAVE ADDRESS FOR TYPEOUT
3385 015670 012702 000011          ROMCLK                ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
104412          ;PORT4 LU 11
3386 015674 104412          ROMCLK                ;PUT 'FOUND' IN R4
3387 015676 021224          MOV      4(R1),R4      ;CLEAR UNWANTED BITS
3388 015700 016104 000004          BIC      #257,R4       ;PUT 'EXPECTED' IN R5
3389 015704 042704 000257          MOV      #120,R5      ;IS ACTIVE SET?
3390 015710 012705 000120          CMPB     R5,R4         ;BR IF YES
3391 015714 120504          BEQ      1$
3392 015716 001401          ERROR    5
3393 015720 104005
1$:
3394 015722
3395 015722 012702 000013          MOV      #13,R2        ;SAVE ADDRESS FOR TYPEOUT
3396 015726 104412          ROMCLK                ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
021264          ;PORT4 LU 13
3397 015730 021264          MOV      4(R1),R4      ;PUT EXPECTED IN R4
3398 015732 016104 000004          BIC      #337,R4       ;CLEAR UNWANTED BITS
3399 015736 042704 000337          MOV      #BIT5,R5     ;PUT 'EXPECTED' IN R5, RTS SHOULD BE SET
3400 015742 012705 000040          CMPB     R5,R4         ;IS RTS OK?
3401 015746 120504          BEQ      2$           ;BR IF YES
3402 015750 001401          ERROR    5            ;RTS ERROR
3403 015752 104005
2$:
3404 015754
3405
3406          ;***** TEST 16 *****
3407

```

3408  
3409  
3410  
3411  
3412  
3413  
3414  
3415  
3416  
3417  
3418  
3419  
3420  
3421  
3422  
3423  
3424  
3425  
3426  
3427  
3428  
3429  
3430  
3431  
3432  
3433  
3434  
3435  
3436  
3437  
3438  
3439  
3440  
3441  
3442  
3443  
3444  
3445  
3446  
3447  
3448  
3449  
3450  
3451  
3452  
3453  
3454  
3455  
3456  
3457  
3458  
3459  
3460  
3461  
3462  
3463

:\*TEST OF OUT CLEAR  
:\*SET SOM AND LOAD OUT DATA SILO  
:\*SINGLE STEP DATA CLOCK, SET OUT CLEAR  
:\*VERIFY THAT OCOR,RTS, AND ACTIVE ARE CLEARED  
:\*\*\*\*\*

: TEST 16  
:-----

```

:*****
TST16: SCOPE
MOV #16,$TSTNM ; LOAD THE NO. OF THIS TEST
MOV #TST17,NEXT ; POINT TO THE START OF NEXT TEST.
;R1 CONTAINS BASE KMC11 ADDRESS
MSTCLR ;MASTER CLEAR KMC11
MOV #BIT11,(R1) ;SET LINE UNIT LOOP
MOV #1,4(R1) ;LOAD PORT4 WITH BIT0
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
122111 ;SET SOM
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
122110 ;LOAD OUT DATA SILO
JSR PC,OCOR ;WAIT FOR OCOR
DATACLK, 2 ;CLOCK DATA FOUR TIMES
MOV #BIT7,4(R1) ;SET BIT7 IN PORT4
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
122111 ;SET OUT CLEAR
DATACLK, 1 ;GIVE A TICK TO CLEAR RTS
MOV #17,R2 ;SAVE ADDRESS FOR TYPEOUT
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
021364 ;PORT4 LU 17
MOV 4(R1),R4 ;PUT 'FOUND' IN R4
BIC #357,R4 ;CLEAR UNWANTED BITS
CLR R5 ;PUT 'EXPECTED' IN R5
CMPB R5,R4 ;IS OCOR CLEARED?
BEQ 1$ ;BR IF YES
ERROR 5

1$: MOV #13,R2 ;SAVE ADDRESS FOR TYPEOUT
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
021264 ;PORT4 LU 13
MOV 4(R1),R4 ;PUT EXPECTED IN R4
BIC #337,R4 ;CLEAR UNWANTED BITS
CLR R5 ;PUT 'EXPECTED' IN R5, RTS SHOULD BE CLEARED
CMPB R5,R4 ;IS RTS OK?
BEQ 2$ ;BR IF YES
ERROR 5 ;RTS ERROR

2$: MOV #11,R2 ;SAVE ADDRESS FOR TYPEOUT
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
021224 ;PORT4 LU11
MOV 4(R1),R4 ;PUT 'FOUND' IN R4
MOV #BIT4,R5 ;ONLY OUT READY SHOULD BE SET
CMPB R5,R4 ;IS ACTIVE CLEAR?
BEQ 3$ ;BR IF YES
ERROR 5 ;ERROR ACTIVE NOT CLEARED

3$:

```

3464  
3465  
3466  
3467  
3468  
3469  
3470  
3471  
3472  
3473  
3474  
3475  
3476  
3477  
3478  
3479  
3480  
3481  
3482  
3483  
3484  
3485  
3486  
3487  
3488  
3489  
3490  
3491  
3492  
3493  
3494  
3495  
3496  
3497  
3498  
3499  
3500  
3501  
3502  
3503  
3504  
3505  
3506  
3507  
3508  
3509  
3510  
3511  
3512  
3513  
3514  
3515  
3516  
3517  
3518  
3519

\*\*\*\*\* TEST 17 \*\*\*\*\*  
; \*DDCMP TRANSMITTER TEST  
; \*SINGLE CLOCK THE CHARACTER 0  
; \*VERIFY EACH BIT POSITION AS IT  
; \*PASSES THE BIT WINDOW (SI BIT)  
; \*ON AN ERROR, R3 CONTAINS BIT POSITION OF FAILURE  
; \*\*\*\*\*

; TEST 17  
; -----

\*\*\*\*\*  
TST17: SCOPE ;  
MOV #17,\$STNM ; LOAD THE NO. OF THIS TEST  
MOV #TST20,NEXT ; POINT TO THE START OF NEXT TEST.  
; R1 CONTAINS BASE KMC11 ADDRESS  
MSTCLR ; MASTER CLEAR KMC11  
MOV #BIT11,(R1) ; SET LINE UNIT LOOP  
JSR PC,OUTRDY ; WAIT FOR OUT-READY  
MOV #1,4(R1) ; SET BIT0 IN PORT4  
ROMCLK ; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304  
122111 ; SET SOM!  
MOV #0,R5 ; LOAD CHARACTER IN R5 FOR TYPEOUT  
JSR PC,OUTRDY ; WAIT FOR OUT-READY  
MOV R5,4(R1) ; LOAD PORT4 WITH CHARACTER  
ROMCLK ; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304  
122110 ; LOAD OUT DATA  
JSR PC,OCOR ; WAIT FOR OCOR TO SET  
CLR R3 ; CLEAR BIT COUNTER  
MOV R5,R2 ; LOAD CHARACTER IN R2  
DATACLK, 2 ; 2 TICKS TO SET UP TRANSMITTER  
DATACLK, 1 ; SHIFT NEXT BIT IN THE WINDOW (SI BIT)  
RORB R2 ; SHIFT NEXT SOFTWARE BIT IN TO CARRY  
BCC 2\$ ; BR IF CARRY CLEAR  
JSR PC,GETSI ; GET THE WINDOW  
BCS 3\$ ; BR IF BIT IS A MARK  
ERROR 6 ; ERROR BIT WAS A SPACE  
BR 3\$ ; CONTINUE WITH TEST  
2\$: JSR PC,GETSI ; GET THE WINDOW  
BCC 3\$ ; BR IF BIT IS A SPACE  
ERROR 6 ; ERROR BIT WAS A MARK  
3\$: INC R3 ; NEXT BIT  
CMP #10,R3 ; DONE YET?  
BNE 1\$ ; BR IF NO  
DATACLK, 14 ; CLOCK TRANSMITTER 14 MORE TICKS  
ROMCLK ; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304  
021264 ; PORT4 LU-13  
BIT #BIT5,4(R1) ; RTS SHOULD BE CLEAR NOW  
BEQ 4\$ ; BR IF YES  
ERROR 34 ; ERROR, RTS NOT CLEAR  
4\$:

\*\*\*\*\* TEST 20 \*\*\*\*\*  
; \*DDCMP TRANSMITTER TEST

```

3520 ;*SINGLE CLOCK THE CHARACTER 125
3521 ;*VERIFY EACH BIT POSITION AS IT
3522 ;*PASSES THE BIT WINDOW (SI BIT)
3523 ;*ON AN ERROR, R3 CONTAINS BIT POSITION OF FAILURE
3524 ;:*****
3525
3526 ; TEST 20
3527 ;-----
3528 ;:*****
3529 016334 000004 TST20: SCOPE
3530 016336 012737 000020 001202 MOV #20,$TSTNM ; LOAD THE NO. OF THIS TEST
3531 016344 012737 016516 001442 MOV #TST21,NEXT ; POINT TO THE START OF NEXT TEST.
3532 ;:*****
3533 016352 104410 MSTCLR ;R1 CONTAINS BASE KMC11 ADDRESS
3534 016354 012711 004000 MOV #BIT11,(R1) ;MASTER CLEAR KMC11
3535 016360 004737 030412 JSR PC,OUTRDY ;SET LINE UNIT LOOP
3536 016364 012761 000001 000004 MOV #1,4(R1) ;WAIT FOR OUT-READY
3537 016372 104412 ROMCLK ;SET BIT0 IN PORT4
3538 016374 122111 122111 ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3539 016376 012705 000125 MOV #125,R5 ;LOAD CHARACTER IN R5 FOR TYPEOUT
3540 016402 004737 030412 JSR PC,OUTRDY ;WAIT FOR OUT-READY
3541 016406 010561 000004 MOV R5,4(R1) ;LOAD PORT4 WITH CHARACTER
3542 016412 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3543 016414 122110 122110 ;LOAD OUT DATA
3544 016416 004737 030260 JSR PC,OCOP ;WAIT FOR OCOR TO SET
3545 016422 005003 CLR R3 ;CLEAR BIT COUNTER
3546 016424 010502 MOV R5,R2 ;LOAD CHARACTER IN R2
3547 016426 104413 000002 DATACLK, 2 ;2 TICKS TO SET UP TRANSMITTER
3548 016432 104413 000001 1$: DATACLK, 1 ;SHIFT NEXT BIT IN THE WINDOW (SI BIT)
3549 016436 106002 RORB R2 ;SHIFT NEXT SOFTWARE BIT IN TO CARRY
3550 016440 103005 BCC 2$ ;BR IF CARRY CLEAR
3551 016442 004737 030226 JSR PC,GETSI ;GET THE WINDOW
3552 016446 103406 BCS 3$ ;BR IF BIT IS A MARK
3553 016450 104006 ERROR 6 ;ERROR BIT WAS A SPACE
3554 016452 000404 BR 3$ ;CONTINUE WITH TEST
3555 016454 004737 030226 2$: JSR PC,GETSI ;GET THE WINDOW
3556 016460 103001 BCC 3$ ;BR IF BIT IS A SPACE
3557 016462 104006 ERROR 6 ;ERROR BIT WAS A MARK
3558 016464 3$:
3559 016464 005203 INC R3 ;NEXT BIT
3560 016466 022703 000010 CMP #10,R3 ;DONE YET?
3561 016472 001357 BNE 1$ ;BR IF NO
3562 016474 104413 000014 DATACLK, 14 ;CLOCK TRANSMITTER 14 MORE TICKS
3563 016500 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3564 016502 021264 021264 ;PORT4 LU-13
3565 016504 032761 000040 000004 BIT #BITS,4(R1) ;RTS SHOULD BE CLEAR NOW
3566 016512 001401 BEQ 4$ ;BR IF YES
3567 016514 104034 ERROR 34 ;ERROR, RTS NOT CLEAR
3568 4$:
3569
3570 ;:*****
3571 ;***** TEST 21 *****
3572 ;*DDCMP TRANSMITTER TEST
3573 ;*SINGLE CLOCK THE CHARACTER 252
3574 ;*VERIFY EACH BIT POSITION AS IT
3575 ;*PASSES THE BIT WINDOW (SI BIT)

```

```

3576 ;*ON AN ERROR, R3 CONTAINS BIT POSITION OF FAILURE
3577 ;:*****
3578 ;
3579 ; TEST 21
3580 ;-----
3581 ;:*****
3582 016516 000004 TST21: SCOPE
3583 016520 012737 000021 001202 MOV #21,$STNM ; LOAD THE NO. OF THIS TEST
3584 016526 012737 016700 001442 MOV #TST22,NEXT ; POINT TO THE START OF NEXT TEST.
3585 ;R1 CONTAINS BASE KMC11 ADDRESS
3586 016534 104410 MSTCLR ;MASTER CLEAR KMC11
3587 016536 012711 004000 MOV #BIT11,(R1) ;SET LINE UNIT LOOP
3588 016542 004737 030412 JSR PC,OUTRDY ;WAIT FOR OUT-READY
3589 016546 012761 000001 000004 MOV #1,4(R1) ;SET BIT0 IN PORT4
3590 016554 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3591 016556 122111 ;SET SOM!
3592 016560 012705 000252 MOV #252,R5 ;LOAD CHARACTER IN R5 FOR TYPEOUT
3593 016564 004737 030412 JSR PC,OUTRDY ;WAIT FOR OUT-READY
3594 016570 010561 000004 MOV R5,4(R1) ;LOAD PORT4 WITH CHARACTER
3595 016574 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3596 016576 122110 ;LOAD OUT DATA
3597 016600 004737 030260 JSR PC,OCOR ;WAIT FOR OCOR TO SET
3598 016604 005003 CLR R3 ;CLEAR BIT COUNTER
3599 016606 010502 MOV R5,R2 ;LOAD CHARACTER IN R2
3600 016610 104413 000002 DATACLK, 2 ;2 TICKS TO SET UP TRANSMITTER
3601 016614 104413 000001 1$: DATACLK, 1 ;SHIFT NEXT BIT IN THE WINDOW (SI BIT)
3602 016620 106002 RORB R2 ;SHIFT NEXT SOFTWARE BIT IN TO CARRY
3603 016622 103005 BCC 2$ ;BR IF CARRY CLEAR
3604 016624 004737 030226 JSR PC,GETSI ;GET THE WINDOW
3605 016630 103406 BCS 3$ ;BR IF BIT IS A MARK
3606 016632 104006 ERROR 6 ;ERROR BIT WAS A SPACE
3607 016634 000404 BR 3$ ;CONTINUE WITH TEST
3608 016636 004737 030226 2$: JSR PC,GETSI ;GET THE WINDOW
3609 016642 103001 BCC 3$ ;BR IF BIT IS A SPACE
3610 016644 104006 ERROR 6 ;ERROR BIT WAS A MARK
3611 016646 3$:
3612 016646 005203 INC R3 ;NEXT BIT
3613 016650 022703 000010 CMP #10,R3 ;DONE YET?
3614 016654 001357 BNE 1$ ;BR IF NO
3615 016656 104413 000014 DATACLK, 14 ;CLOCK TRANSMITTER 14 MORE TICKS
3616 016662 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3617 016664 021264 ;PORT4 LU=13
3618 016666 032761 000040 000004 BIT #BITS,4(R1) ;RTS SHOULD BE CLEAR NOW
3619 016674 001401 BEQ 4$ ;BR IF YES
3620 016676 104034 ERROR 34 ;ERROR, RTS NOT CLEAR
3621 016700 4$:
3622
3623
3624 ;***** TEST 22 *****
3625 ;*DDCMP TRANSMITTER TEST
3626 ;*SINGLE CLOCK THE CHARACTER 377
3627 ;*VERIFY EACH BIT POSITION AS IT
3628 ;*PASSES THE BIT WINDOW (SI BIT)
3629 ;*ON AN ERROR, R3 CONTAINS BIT POSITION OF FAILURE
3630 ;:*****
3631
    
```



```

3632 ; TEST 22
3633 ;-----
3634 :*****
3635 016700 000004 TST22: SCOPE
3636 016702 012737 000022 001202 MOV #22,$STNM ; LOAD THE NO. OF THIS TEST
3637 016710 012737 017062 001442 MOV #TST23,NEXT ; POINT TO THE START OF NEXT TEST.
3638 ;R1 CONTAINS BASE KMC11 ADDRESS
3639 016716 104410 MSTCLR ;MASTER CLEAR KMC*1
3640 016720 012711 004000 MOV #BIT11,(R1) ;SET LINE UNIT LOOP
3641 016724 004737 030412 JSR PC,OUTRDY ;WAIT FOR OUT-READY
3642 016730 012761 000001 000004 MOV #1,4(R1) ;SET BIT0 IN PORT4
3643 016736 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3644 016740 122111 122111 ;SET SOM!
3645 016742 012705 000377 MOV #377,R5 ;LOAD CHARACTER IN R5 FOR TYPEOUT
3646 016746 004737 030412 JSR PC,OUTRDY ;WAIT FOR OUT-READY
3647 016752 010561 000004 MOV R5,4(R1) ;LOAD PORT4 WITH CHARACTER
3648 016756 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3649 016760 122110 122110 ;LOAD OUT DATA
3650 016762 004737 030260 JSR PC,OCOR ;WAIT FOR OCOR TO SET
3651 016766 005003 CLR R3 ;CLEAR BIT COUNTER
3652 016770 010502 MOV R5,R2 ;LOAD CHARACTER IN R2
3653 016772 104413 000002 DATACL, 2 ;2 TICKS TO SET UP TRANSMITTER
3654 016776 104413 000001 1$: DATACL, 1 ;SHIFT NEXT BIT IN THE WINDOW (SI BIT)
3655 017002 106002 RORB R2 ;SHIFT NEXT SOFTWARE BIT IN TO CARRY
3656 017004 103005 BCC 2$ ;BR IF CARRY CLEAR
3657 017006 004737 030226 JSR PC,GETSI ;GET THE WINDOW
3658 017012 103406 BCS 3$ ;BR IF BIT IS A MARK
3659 017014 104006 ERROR 6 ;ERROR BIT WAS A SPACE
3660 017016 000404 BR 3$ ;CONTINUE WITH TEST
3661 017020 004737 030226 2$: JSR PC,GETSI ;GET THE WINDOW
3662 017024 103001 BCC 3$ ;BR IF BIT IS A SPACE
3663 017026 104006 ERROR 6 ;ERROR BIT WAS A MARK
3664 017030 3$:
3665 017030 005203 INC R3 ;NEXT BIT
3666 017032 022703 000010 CMP #10,R3 ;DONE YET?
3667 017036 001357 BNE 1$ ;BR IF NO
3668 017040 104413 000014 DATACL, 14 ;CLOCK TRANSMITTER 14 MORE TICKS
3669 017044 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3670 017046 021264 021264 ;PORT4 LU-13
3671 017050 032761 000040 000004 BIT #BIT5,4(R1) ;RTS SHOULD BE CLEAR NOW
3672 017056 001401 BEQ 4$ ;BR IF YES
3673 017060 104034 ERROR 34 ;ERROR, RTS NOT CLEAR
3674 017062 4$:
3675
3676
3677 :***** TEST 23 *****
3678 ;*DDCMP TRANSMITTER TEST
3679 ;*SINGLE CLOCK A BINARY COUNT PATTERN
3680 ;*VERIFY EACH BIT POSITION AS IT
3681 ;*PASSES THE BIT WINDOW (SI BIT)
3682 ;*ON AN ERROR, R3 CONTAINS BIT POSITION OF FAILURE
3683 ;*AND R5 CONTAINS THE CHARACTER THAT FAILED
3684 :*****
3685
3686
3687 : TEST 23
;-----

```

```

3688
3689 017062 000004          ;*****
3690 017064 012737 000023 001202 1ST23 SCGPE                ;LOAD THE NO. OF THIS TEST
3691 017072 012737 017270 001442      MOV #23,$TSTNM          ;POINT TO THE START OF NEXT TEST.
3692                                     MOV #TST24,NEXT
3693 017100 104410          ;R1 CONTAINS BASE KMC11 ADDRESS
3694 017102 012711 004000      MSTCLR                ;MASTER CLEAR KMC11
3695 017106 005003          MOV #BIT11,(R1)       ;SET LINE UNIT LOOP
3696 017110 005004          CLR R3                ;R3 CONTAINS BIT COUNT
3697 017112 005005          CLR R4                ;R4 CONTAINS CHAR TO BE LOADED I. SILO
3698 017114 004737 030412      CLR R5                ;R5 CONTAINS CHARACTER CURRENTLY BEING SHIFTED OUT
3699 017120 012761 000001 000004      JSR PC,OUTRDY        ;WAIT FOR OUT-READY
3700 017126 104412          MOV #1,4(R1)         ;SET BIT0 IN PORT4
3701 017130 122111          ROMCLK                ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3702 017132 004737 030412      122111                ;SET SOM!
3703 017136 010461 000004      JSR PC,OUTRDY        ;WAIT FOR OUT-READY
3704 017142 104412          MOV R4,4(R1)         ;LOAD PORT4 WITH CHARACTER
3705 017144 122110          ROMCLK                ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3706 017146 005204          122110                ;LOAD OUT DATA
3707 017150 004737 030412      INC R4                ;INCREMENT TO NEXT CHARACTER
3708 017154 010461 000004      JSR PC,OUTRDY        ;WAIT FOR OUT-READY
3709 017160 104412          MOV R4,4(R1)         ;LOAD PORT4 WITH CHARACTER
3710 017162 122110          ROMCLK                ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3711 017164 004737 030260      122110                ;LOAD OUT DATA
3712 017170 104413 000002      JSR PC,OCOR           ;WAIT FOR OCOR TO SET
3713 017174 005003          DATACLK, 2          ;2 TICKS TO SET UP TRANSMITTER
3714 017176 010502          CLR R3                ;CLEAR BIT COUNTER
3715 017200 104413 000001      MOV R5,R2             ;LOAD CHARACTER IN R2
3716 017204 106002          1$: DATACLK, 1      ;SHIFT NEXT BIT IN THE WINDOW (SI BIT)
3717 017206 103005          RORB R2               ;SHIFT NEXT SOFTWARE BIT IN TO CARRY
3718 017210 004737 030226      BCC 2$                ;BR IF CARRY CLEAR
3719 017214 103406          JSR PC,GETSI         ;GET THE WINDOW
3720 017216 104006          BCS 3$                ;BR IF BIT IS A MARK
3721 017220 000404          ERROR 6               ;ERROR BIT WAS A SPACE
3722 017222 004737 030226      BR 3$                 ;CONTINUE WITH TEST
3723 017226 103001          2$: JSR PC,GETSI       ;GET THE WINDOW
3724 017230 104006          BCC 3$                ;BR IF BIT IS A SPACE
3725 017232          ERROR 6               ;ERROR BIT WAS A MARK
3726 017232 005203          3$: INC R3              ;NEXT BIT
3727 017234 022703 000010      CMP #10,R3           ;DONE YET?
3728 017240 001357          BNE 1$                ;BR IF NO
3729 017242 005204          INC R4                ;NEXT CHARACTER
3730 017244 004737 030412      JSR PC,OUTRDY        ;WAIT FOR OUT-READY
3731 017250 010461 000004      MOV R4,4(R1)         ;LOAD PORT4 WITH CHARACTER
3732 017254 104412          ROMCLK                ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3733 017256 122110          122110                ;LOAD OUT DATA
3734 017260 005205          INC R5                ;NEXT CHARACTER
3735 017262 022705 000400      CMP #400,R5          ;DONE YET?
3736 017266 001342          BNE 4$                ;BR IF NO
3737 017270          5$:
3738
3739
3740          ;***** TEST 24 *****
3741          ;*DDCMP STRIP SYNC TEST
3742          ;*SET LU LOOP, SINGLE STEP 5 SYNCs.
3743          ;*VERIFY THAT IN ACTIVE DOES NOT SET

```

```

3744 ;:*****
3745 ;
3746 ; TEST 24
3747 ;-----
3748 ;:*****
3749 017270 000004 TST24: SCOPE
3750 017272 012737 000024 001202 MOV #24,$STSTM ; LOAD THE NO. OF THIS TEST
3751 017300 012737 017356 001442 MOV #TST25,NEXT ; POINT TO THE START OF NEXT TEST.
3752 ;R1 CONTAINS BASE KMC11 ADDRESS
3753 017306 104410 MSTCLR ;MASTER CLEAR KMC11
3754 017310 012711 004000 MOV #BIT11,(R1) ;SET LU LOOP
3755 017314 012702 000012 MOV #12,R2 ;SAVE LU REG FOR TYPEOUT
3756 017320 004737 030276 JSR PC,SYNC ;SINGLE CLOCK 5 SYNC CHARACTERS
3757 017324 000005 5
3758 017326 104413 000054 DATACLK, 54
3759 017332 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3760 017334 021244 021244 ;PORT4 LU12
3761 017336 016104 000004 MOV 4(R1),R4 ;PUT 'FOUND' IN R4
3762 017342 042704 000277 BIC #277,R4 ;CLEAR UNWANTED BITS
3763 017346 005005 CLR R5 ;PUT 'EXPECTED' IN R5
3764 017350 120504 CMPB R5,R4 ;IS ACTIVE CLEAR?
3765 017352 001401 BEQ 1$ ;BR IF YES
3766 017354 104040 ERROR 40 ;ERROR ACTIVE IS NOT CLEAR
3767 017356
3768
3769
3770
3771 ;:***** TEST 25 *****
3772 ;*DDCMP IN ACTIVE TEST
3773 ;*SET LU LOOP, SINGLE STEP 5 SYNC AND A NON-SYNC (301)
3774 ;*VERIFY THAT IN ACTIVE IS SET
3775 ;:*****
3776 ;
3777 ; TEST 25
3778 ;-----
3779 017356 000004 TST25: SCOPE
3780 017360 012737 000025 001202 MOV #25,$STSTM ; LOAD THE NO. OF THIS TEST
3781 017366 012737 017446 001442 MOV #TST26,NEXT ; POINT TO THE START OF NEXT TEST.
3782 ;R1 CONTAINS BASE KMC11 ADDRESS
3783 017374 104410 MSTCLR ;MASTER CLEAR KMC11
3784 017376 012711 004000 MOV #BIT11,(R1) ;SET LU LOOP
3785 017402 012702 000012 MOV #12,R2 ;SAVE LU REG FOR TYPEOUT
3786 017406 004737 030276 JSR PC,SYNC ;SINGLE CLOCK 5 SYNC CHARACTERS
3787 017412 000005 5
3788 017414 104413 000064 DATACLK, 64
3789 017420 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3790 017422 021244 021244 ;PORT4 LU12
3791 017424 016104 000004 MOV 4(R1),R4 ;PUT 'FOUND' IN R4
3792 017430 042704 000277 BIC #277,R4 ;CLEAR UNWANTED BITS
3793 017434 012705 000100 MOV #BIT6,R5 ;PUT 'EXPECTED' IN R5
3794 017440 120504 CMPB R5,R4 ;IS ACTIVE SET?
3795 017442 001401 BEQ 1$ ;BR IF YES
3796 017444 104040 ERROR 40 ;ERROR ACTIVE IS NOT SET
3797 017446
3798
3799
    
```

```

3800 :***** TEST 26 *****
3801 :*DDCMP IN ACTIVE TEST
3802 :*SET LU LOOP, SINGLE STEP 1 SYNC AND A NON-SYNC (301)
3803 :*VERIFY THAT IN ACTIVE DOES NOT SET
3804 :*****
3805

```

```

3806 : TEST 26
3807 :-----
3808 :*****

```

```

3809 017446 000004 TST26: SCOPE
3810 017450 012737 000026 001202 MOV #26,$STNM ; LOAD THE NO. OF THIS TEST
3811 017456 012737 017534 001442 MOV #TST27,NEXT ; POINT TO THE START OF NEXT TEST.
3812 ;R1 CONTAINS BASE KMC11 ADDRESS
3813 017464 104410 MSTCLR ;MASTER CLEAR KMC11
3814 017466 012711 004000 MOV #BIT11,(R1) ;SET LU LOOP
3815 017472 012702 000012 MOV #12,R2 ;SAVE LU REG FOR TYPEOUT
3816 017476 004737 030276 JSR PC,SYNC ;SINGLE CLOCK 1 SYNC CHARACTERS
3817 017502 000001 1
3818 017504 104413 000024 DATACLK, 24
3819 017510 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3820 017512 021244 021244 ;PORT4 LU12
3821 017514 016104 000004 MOV 4(R1),R4 ;PUT 'FOUND' IN R4
3822 017520 042704 000277 BIC #277,R4 ;CLEAR UNWANTED BITS
3823 017524 005005 CLR R5 ;PUT 'EXPECTED' IN R5
3824 017526 120504 CMPB R5,R4 ;IS ACTIVE CLEAR?
3825 017530 001401 BEQ 1$ ;BR IF YES
3826 017532 104040 ERROR 40 ;ERROR ACTIVE IS NOT CLEAR
3827 017534 1$:

```

```

3828
3829
3830 :***** TEST 27 *****
3831 :*DDCMP IN ACTIVE TEST
3832 :*SET LU LOOP, SINGLE STEP 2 SYNCs AND A NON-SYNC (301)
3833 :*VERIFY THAT IN ACTIVE IS SET
3834 :*****
3835

```

```

3836 : TEST 27
3837 :-----
3838 :*****

```

```

3839 017534 000004 TST27: SCOPE
3840 017536 012737 000027 001202 MOV #27,$STNM ; LOAD THE NO. OF THIS TEST
3841 017544 012737 017624 001442 MOV #TST30,NEXT ; POINT TO THE START OF NEXT TEST.
3842 ;R1 CONTAINS BASE KMC11 ADDRESS
3843 017552 104410 MSTCLR ;MASTER CLEAR KMC11
3844 017554 012711 004000 MOV #BIT11,(R1) ;SET LU LOOP
3845 017560 012702 000012 MOV #12,R2 ;SAVE LU REG FOR TYPEOUT
3846 017564 004737 030276 JSR PC,SYNC ;SINGLE CLOCK 2 SYNC CHARACTERS
3847 017570 000002 2
3848 017572 104413 000034 DATACLK, 34
3849 017576 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3850 017600 021244 021244 ;PORT4 LU12
3851 017602 016104 000004 MOV 4(R1),R4 ;PUT 'FOUND' IN R4
3852 017606 042704 000277 BIC #277,R4 ;CLEAR UNWANTED BITS
3853 017612 012705 000100 MOV #BIT6,R5 ;PUT 'EXPECTED' IN R5
3854 017616 120504 CMPB R5,R4 ;IS ACTIVE SET?
3855 017620 001401 BEQ 1$ ;BR IF YES

```

3856 017622 104040  
3857 017624

1\$: ERROR 40 ;ERROR ACTIVE IS NOT SET

3858  
3859  
3860  
3861  
3862  
3863  
3864  
3865  
3866  
3867  
3868  
3869

\*\*\*\*\* TEST 30 \*\*\*\*\*  
: \*IN CLEAR TEST  
: \*SYNC UP RECEIVER AND TRANSMIT A CHARACTER  
: \*WAIT FOR IN RDY, THEN SET IN CLEAR  
: \*VERIFY THAT IN ACTIVE AND IN RDY ARE CLEARED  
: \*\*\*\*\*  
: TEST 30  
: -----

3870 017624 000004  
3871 017626 012737 000030 001202  
3872 017634 012737 017776 001442

1\$T30: SCOPE ; LOAD THE NO. OF THIS TEST  
MOV #30,\$TSTNM ; POINT TO THE START OF NEXT TEST.  
MOV #TST31,NEXT ;R1 CONTAINS BASE KMC11 ADDRESS

3874 017642 104410  
3875 017644 012702 000012  
3876 017650 012711 004000  
3877 017654 004737 030444  
3878 017660 000301  
3879 017662 104413 000053  
3880 017666 104414 000002  
3881 017672 104412  
3882 017674 021244  
3883 017676 016104 000004  
3884 017702 042704 000357  
3885 017706 012705 000020  
3886 017712 120504  
3887 017714 001401  
3888 017716 104040

MSTCLR ;MASTER CLEAR KMC11  
MOV #12,R2 ;SAVE REG ADDRESS IN R2 FOR TYPEOUT  
MOV #BIT11,(R1) ;SET LINE UNIT LOOP  
JSR PC,CHAR ;LOAD SILO WITH 3 SYNCs  
301 ;AND A MON-SYNC (301)  
DATACLK, 53 ;SINGLE CLOCK THE DATA  
TIMER, 2 ;WAIT FOR INRDY  
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC-5304  
021244 ;PORT4 LU 12  
MOV 4(R1),R4 ;PUT 'FOUND' IN R4  
BIC #357,R4 ;CLEAR UNWANTED BITS  
MOV #BIT4,R5 ;PUT 'EXPECTED' IN R5  
CMPB R5,R4 ;IS INRDY SET?  
BEQ 1\$  
ERROR 40 ;ERROR, INRDY IS NOT SET

3889 017720  
3890 017720 012761 000200 000004  
3891 017726 104412  
3892 017730 122112  
3893 017732 104412  
3894 017734 021244  
3895 017736 016104 000004  
3896 017742 042704 000277  
3897 017746 005005  
3898 017750 120504  
3899 017752 001401  
3900 017754 104040

1\$: MOV #BIT7,4(R1) ;LOAD PORT4  
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304  
122112 ;SET IN CLEAR  
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC 5304  
021244 ;PORT4 LU 12  
MOV 4(R1),R4 ;PUT 'FOUND' IN R4  
BIC #277,R4 ;CLEAR UNWANTED BITS  
CLR R5 ;PUT 'EXPECTED' IN R5  
CMPB R5,R4 ;IS IN ACTIVE CLEAR?  
BEQ 2\$  
ERROR 40 ;ERROR, IN ACTIVE IS NOT CLEAR

3901 017756  
3902 017756 016104 000004  
3903 017762 042704 000357  
3904 017766 005005  
3905 017770 120504  
3906 017772 001401  
3907 017774 104040  
3908 017776

2\$: MOV 4(R1),R4 ;PUT 'FOUND' IN R4  
BIC #357,R4 ;CLEAR UNWANTED BITS  
CLR R5 ;PUT 'EXPECTED' IN R5  
CMPB R5,R4 ;IS INRDY CLEARED?  
BEQ 3\$  
ERROR 40 ;ERROR, INRDY IS NOT CLEARED

3909  
3910  
3911

\*\*\*\*\* TEST 31 \*\*\*\*\*

```

3912          : *DDCMP BASIC RECEICER TEST
3913          : *SYNC UP RECEIVER AND SINGLE CLOCK THE CHARACTER 0
3914          : *VERIFY THAT IN RDY IS SET, AND THAT THE CHARACTER WAS RECEIVED
3915          : :*****
3916          :
3917          : TEST 31
3918          : -----
3919          : :*****
3920 017776 000004          TST31: SCOPE
3921 020000 012737 000031 001202          MOV #31,$STNM          ; LOAD THE NO. OF THIS TEST
3922 020006 012737 020112 001442          MOV #TST32,NEXT      ; POINT TO THE START OF NEXT TEST.
3923          : R1 CONTAINS BASE KMC11 ADDRESS
3924 020014 104410          MSTCLR          ; MASTER CLEAR KMC11
3925 020016 012702 000012          MOV #12,R2          ; SAVE REG ADDRESS IN R2 FOR TYPEOUT
3926 020022 012711 004000          MOV #BIT11,(R1)     ; SET LINE UNIT LOOP
3927 020026 004737 030444          JSR PC,CHAR         ; LOAD SILO WITH 3 SYNCs
3928 020032 000000          0                  ; AND THE CHARACTER 0
3929 020034 104413 000053          DATACLK, 53       ; SINGLE CLOCK THE DATA
3930 020040 104414 000002          TIMER, 2           ; WAIT FOR INRDY
3931 020044 104412          ROMCLK             ; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3932 020046 021244          021244            ; PORT4 LU 12
3933 020050 016104 000004          MOV 4(R1),R4        ; PUT 'FOUND' IN R4
3934 020054 042704 000357          BIC #357,R4         ; CLEAR UNWANTED BITS
3935 020060 012705 000020          MOV #BIT4,R5        ; PUT 'EXPECTED' IN R5
3936 020064 120504          CMPB R5,R4         ; IS INRDY SET?
3937 020066 001401          BEQ 1$             ;
3938 020070 104040          ERROR 40           ; ERROR, INRDY IS NOT SET
3939 020072          :
3940 020072 104412          ROMCLK             ; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3941 020074 021204          021204            ; PORT4 IN DATA
3942 020076 016104 000004          MOV 4(R1),R4        ; PUT 'FOUND' IN R4
3943 020102 005005          CLR R5              ; PUT 'EXPECTED' IN R5
3944 020104 120504          CMPB R5,R4         ; WAS A 0 RECEIVED?
3945 020106 001401          BEQ 2$             ;
3946 020110 104010          ERROR 10           ; ERROR, RECEIVED DATA IS WRONG
3947 020112          :
3948          :
3949          :
3950          : ***** TEST 32 *****
3951          : *DDCMP BASIC RECEICER TEST;
3952          : *SYNC UP RECEIVER AND SINGLE CLOCK THE CHARACTER 125
3953          : *VERIFY THAT IN RDY IS SET, AND THAT THE CHARACTER WAS RECEIVED
3954          : :*****
3955          :
3956          : TEST 32
3957          : -----
3958          : :*****
3959 020112 000004          TST32: SCOPE
3960 020114 012737 000032 001202          MOV #32,$STNM          ; LOAD THE NO. OF THIS TEST
3961 020122 012737 020230 001442          MOV #TST33,NEXT      ; POINT TO THE START OF NEXT TEST.
3962          : R1 CONTAINS BASE KMC11 ADDRESS
3963 020130 104410          MSTCLR          ; MASTER CLEAR KMC11
3964 020132 012702 000012          MOV #12,R2          ; SAVE REG ADDRESS IN R2 FOR TYPEOUT
3965 020136 012711 004000          MOV #BIT11,(R1)     ; SET LINE UNIT LOOP
3966 020142 004737 030444          JSR PC,CHAR         ; LOAD SILO WITH 3 SYNCs
3967 020146 000125          125                ; AND THE CHARACTER 125
    
```

```
3968 020150 104413 000053          DATACLK,      53          ;SINGLE CLOCK THE DATA
3969 020154 104414 000002          TIMER,      2          ;WAIT FOR INRDY
3970 020160 104412          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3971 020162 021244          021244          ;PORT4 LU 12
3972 020164 016104 000004          MOV      4(R1),R4      ;PUT 'FOUND' IN R4
3973 020170 042704 000357          BIC      #357,R4      ;CLEAR UNWANTED BITS
3974 020174 012705 000020          MOV      #BIT4,R5      ;PUT 'EXPECTED' IN R5
3975 020200 120504          CMPB     R5,R4          ;IS INRDY SET?
3976 020202 001401          BEQ      1$            ;ERROR, INRDY IS NOT SET
3977 020204 104040          ERROR    40
3978 020206          1$:
3979 020206 104412          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3980 020210 021204          021204          ;PORT4 IN DATA
3981 020212 016104 000004          MOV      4(R1),R4      ;PUT 'FOUND' IN R4
3982 020216 012705 000125          MOV      #125,R5      ;PUT 'EXPECTED' IN R5
3983 020222 120504          CMPB     R5,R4          ;WAS A 125 RECEIVED?
3984 020224 001401          BEQ      2$            ;ERROR, RECEIVED DATA IS WRONG
3985 020226 104010          ERROR    10
3986 020230          2$:
3987
3988
3989
3990          ;***** TEST 33 *****
3991          ;*DDCMP BASIC REICEICER TEST
3992          ;*SYNC UP RECEIVER AND SINGLE CLOCK THE CHARACTER 252
3993          ;*VERIFY THAT IN RDY IS SET, AND THAT THE CHARACTER WAS RECEIVED
3994          ;*****
3995          ; TEST 33
3996          ;-----
3997          ;*****
```

```
3998 020230 000004          TST33: SCOPE
3999 020232 012737 000033 001202          MOV      #33,$TSTNM      ; LOAD THE NO. OF THIS TEST
4000 020240 012737 020346 001442          MOV      #TST34,NEXT     ; POINT TO THE START OF NEXT TEST.
4001          ;R1 CONTAINS BASE KMC11 ADDRESS
4002 020246 104410          MSTCLR          ;MASTER CLEAR KMC11
4003 020250 012702 000012          MOV      #12,R2          ;SAVE REG ADDRESS IN R2 FOR TYPEOUT
4004 020254 012711 004000          MOV      #BIT11,(R1)     ;SET LINE UNIT LOOP
4005 020260 004737 030444          JSR      PC,CHAR          ;LOAD SILO WITH 3 SYNCs
4006 020264 000252          252              ;AND THE CHARACTER 252
4007 020266 104413 000053          DATACLK,      53          ;SINGLE CLOCK THE DATA
4008 020272 104414 000002          TIMER,      2          ;WAIT FOR INRDY
4009 020276 104412          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4010 020300 021244          021244          ;PORT4 LU 12
4011 020302 016104 000004          MOV      4(R1),R4      ;PUT 'FOUND' IN R4
4012 020306 042704 000357          BIC      #357,R4      ;CLEAR UNWANTED BITS
4013 020312 012705 000020          MOV      #BIT4,R5      ;PUT 'EXPECTED' IN R5
4014 020316 120504          CMPB     R5,R4          ;IS INRDY SET?
4015 020320 001401          BEQ      1$            ;ERROR, INRDY IS NOT SET
4016 020322 104040          ERROR    40
4017 020324          1$:
4018 020324 104412          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4019 020326 021204          021204          ;PORT4 IN DATA
4020 020330 016104 000004          MOV      4(R1),R4      ;PUT 'FOUND' IN R4
4021 020334 012705 000252          MOV      #252,R5      ;PUT 'EXPECTED' IN R5
4022 020340 120504          CMPB     R5,R4          ;WAS A 252 RECEIVED?
4023 020342 001401          BEQ      2$
```

```
4024 020344 104010          ERROR 10          ;ERROR, RECEIVED DATA IS WRONG
4025 020346          2$:
4026
4027
4028          :***** TEST 34 *****
4029          :*DDCMP BASIC RECEICER TEST
4030          :*SYNC UP RECEIVER AND SINGLE CLOCK THE CHARACTER 377
4031          :*VERIFY THAT IN RDY IS SET, AND THAT THE CHARACTER WAS RECEIVED
4032          :*****
4033
4034          ; TEST 34
4035          :-----
4036          :*****
4037 020346 000004          TST34: SCOPE
4038 020350 012737 000034 001202          MOV #34,$TSTNM          ; LOAD THE NO. OF THIS TEST
4039 020356 012737 020464 001442          MOV #TST35,NEXT          ; POINT TO THE START OF NEXT TEST.
4040          ;R1 CONTAINS BASE KMC11 ADDRESS
4041 020364 104410          MSTCLR          ;MASTER CLEAR KMC11
4042 020366 012702 000012          MOV #12,R2          ;SAVE REG ADDRESS IN R2 FOR TYPEOUT
4043 020372 012711 004000          MOV #BIT11,(R1)      ;SET LINE UNIT LOOP
4044 020376 004737 030444          JSR PC,CHAR          ;LOAD SILO WITH 3 SYNCs
4045 020402 000377          377          ;AND THE CHARACTER 377
4046 020404 104413 000053          DATACLK, 53          ;SINGLE CLOCK THE DATA
4047 020410 104414 000002          TIMER, 2          ;WAIT FOR INRDY
4048 020414 104412          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4049 020416 021244          021244          ;PORT4 LU 12
4050 020420 016104 000004          MOV 4(R1),R4          ;PUT 'FOUND' IN R4
4051 020424 042704 000357          BIC #357,R4          ;CLEAR UNWANTED BITS
4052 020430 012705 000020          MOV #BIT4,R5          ;PUT 'EXPECTED' IN R5
4053 020434 120504          CMPB R5,R4          ;IS INRDY SET?
4054 020436 001401          BEQ 1$
4055 020440 104040          ERROR 40          ;ERROR, INRDY IS NOT SET
4056 020442          1$:
4057 020442 104412          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4058 020444 021204          021204          ;PORT4 IN DATA
4059 020446 016104 000004          MOV 4(R1),R4          ;PUT 'FOUND' IN R4
4060 020452 012705 000377          MOV #377,R5          ;PUT 'EXPECTED' IN R5
4061 020456 120504          CMPB R5,R4          ;WAS A 377 RECEIVED?
4062 020460 001401          BEQ 2$
4063 020462 104010          ERROR 10          ;ERROR, RECEIVED DATA IS WRONG
4064 020464          2$:
4065
4066
4067          :***** TEST 35 *****
4068          :*DDCMP DATA TEST
4069          :*THIS TEST SINGLE STEPS A BINARY COUNT PATTERN
4070          :*CHECKING EACH CHARACTER AS IT IS RECEIVED
4071          :*****
4072
4073          ; TEST 35
4074          :-----
4075          :*****
4076 020464 000004          TST35: SCOPE
4077 020466 012737 000035 001202          MOV #35,$TSTNM          ; LOAD THE NO. OF THIS TEST
4078 020474 012737 020614 001442          MOV #TST36,NEXT          ; POINT TO THE START OF NEXT TEST.
4079          ;R1 CONTAINS BASE KMC11 ADDRESS
```



|      |        |        |        |               |             |   |
|------|--------|--------|--------|---------------|-------------|---|
| 4080 | 020502 | 104410 |        | MSTCLR        |             | :MASTER CLEAR KMC11                       |
| 4081 | 020504 | 0J5037 | 031062 | CLR           | SCHAR       | :START BINARY COUNT AT ZERO               |
| 4082 | 020510 | 005037 | 031064 | CLR           | STUFLG      | :CLEAR BITSTUFF FLAG                      |
| 4083 | 020514 | 005002 |        | CLR           | R2          | :R2 IS "EXPECTED" DATA                    |
| 4084 | 020516 | 012703 | 000073 | MOV           | #73,R3      | :R3 IS CHARACTER COUNT                    |
| 4085 | 020522 | 012711 | 004000 | MOV           | #BIT11,(R1) | :SET LINE UNIT LOOP                       |
| 4086 | 020526 | 004737 | 030622 | JSR           | PC,SILOLD   | :LOAD SILO WITH COUNT PATTERN             |
| 4087 | 020532 | 104413 | 000043 | DATACLK,      | 43          | :SYNC RECEIVER AND GET IT ACTIVE          |
| 4088 | 020536 | 104413 | 000730 | 1\$: DATACLK, | 730         | :CLOCK IN 73 CHARACTERS                   |
| 4089 | 020542 | 004737 | 031066 | 4\$: JSR      | PC,INRDY    | :WAIT FOR INRDY                           |
| 4090 | 020546 | 104412 |        | ROMCLK        |             | :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 |
| 4091 | 020550 | 021204 |        | 021204        |             | :PORT4 IN DATA                            |
| 4092 | 020552 | 016104 | 000004 | MOV           | 4(R1),R4    | :PUT "FOUND" IN R4                        |
| 4093 | 020556 | 010205 |        | MOV           | R2,R5       | :PUT "EXPECTED" IN R5                     |
| 4094 | 020560 | 120504 |        | CMPB          | R5,R4       | :IS DATA CORRECT?                         |
| 4095 | 020562 | 001401 |        | BEQ           | 2\$         | :BR IF YES                                |
| 4096 | 020564 | 104010 |        | ERROR         | 10          | :DATA ERROR                               |
| 4097 | 020566 | 005202 |        | 2\$: INC      | R2          | :NEXT CHARACTER                           |
| 4098 | 020570 | 022702 | 000400 | CMP           | #400,R2     | :ALL DONE?                                |
| 4099 | 020574 | 001407 |        | BEQ           | 3\$         | :BR IF YES                                |
| 4100 | 020576 | 005303 |        | DEC           | R3          | :DECREMENT CHARACTER COUNT                |
| 4101 | 020600 | 001360 |        | BNE           | 4\$         | :BR IF SILO NOT EMPTY                     |
| 4102 | 020602 | 004737 | 030622 | JSR           | PC,SILOLD   | :LOAD SILO WITH MORE OF COUNT PATTERN     |
| 4103 | 020606 | 012703 | 000073 | MOV           | #73,R3      | :RELOAD CHARACTER COUNT                   |
| 4104 | 020612 | 000751 |        | BR            | 1\$         | :CONTINUE                                 |
| 4105 | 020614 |        |        | 3\$:          |             |   |

4106  
4107  
4108 :\*\*\*\*\* TEST 36 \*\*\*\*\*  
4109 :\*DDCMP DATA TEST  
4110 :\*THIS TEST SINGLE STEPS A BINARY COUNT PATTERN  
4111 :\*CHECKING EACH CHARACTER AS IT IS RECEIVED  
4112 :\*THIS TEST IS EXACTLY THE SAME AS THE LAST TEST.  
4113 :\*EXCEPT LINE UNIT LOOP IS SET IN LU REGISTER 12  
4114 :\*\*\*\*\*  
4115  
4116  
4117 : TEST 36  
4118 :-----  
4119 :\*\*\*\*\*

|      |        |        |        |               |             |   |
|------|--------|--------|--------|---------------|-------------|---|
| 4119 | 020614 | 000004 |        | TST36: SCOPE  |             |   |
| 4120 | 020616 | 012737 | 000036 | MOV           | #36,\$STNM  | : LOAD THE NO. OF THIS TEST               |
| 4121 | 020624 | 012737 | 020754 | MOV           | #TST37,NEXT | : POINT TO THE START OF NEXT TEST.        |
| 4122 |        |        |        |               |             | :R1 CONTAINS BASE KMC11 ADDRESS           |
| 4123 | 020632 | 104410 |        | MSTCLR        |             | :MASTER CLEAR KMC11                       |
| 4124 | 020634 | 005037 | 031062 | CLR           | SCHAR       | :START BINARY COUNT AT ZERO               |
| 4125 | 020640 | 005037 | 031064 | CLR           | STUFLG      | :CLEAR BITSTUFF FLAG                      |
| 4126 | 020644 | 005002 |        | CLR           | R2          | :R2 IS "EXPECTED" DATA                    |
| 4127 | 020646 | 012703 | 000073 | MOV           | #73,R3      | :R3 IS CHARACTER COUNT                    |
| 4128 | 020652 | 005011 |        | CLR           | (R1)        | :CLEAR LU LOOP IN MAINT REG               |
| 4129 | 020654 | 012761 | 000040 | MOV           | #BITS,4(R1) | :LOAD PORT4                               |
| 4130 | 020662 | 104412 |        | ROMCLK        |             | :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 |
| 4131 | 020664 | 122112 |        | 122112        |             | :SET LU LOOP IN LU REG 12                 |
| 4132 | 020666 | 004737 | 030622 | JSR           | PC,SILOLD   | :LOAD SILO WITH COUNT PATTERN             |
| 4133 | 020672 | 104413 | 000043 | DATACLK,      | 43          | :SYNC RECEIVER AND GET IT ACTIVE          |
| 4134 | 020676 | 104413 | 000730 | 1\$: DATACLK, | 730         | :CLOCK IN 73 CHARACTERS                   |
| 4135 | 020702 | 004737 | 031066 | 4\$: JSR      | PC,INRDY    | :WAIT FOR INRDY                           |

```

4136 020706 104412 ROMCLK :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4137 020710 021204 :PORT4 IN DATA
4138 020712 016104 000004 MOV 4(R1),R4 :PUT 'FOUND' IN R4
4139 020716 010205 MOV R2,R5 :PUT 'EXPECTED' IN R5
4140 020720 120504 CMPB R5,R4 :IS DATA CORRECT?
4141 020722 001401 BEQ 2$ :BR IF YES
4142 020724 104010 ERROR 10 :DATA ERROR
4143 020726 005202 2$: INC R2 :NEXT CHARACTER
4144 020730 022702 000400 CMP #400,R2 :ALL DONE?
4145 020734 001407 BEQ 3$ :BR IF YES
4146 020736 005303 DEC R3 :DECREMENT CHARACTER COUNT
4147 020740 001360 BNE 4$ :BR IF SILO NOT EMPTY
4148 020742 004737 030622 JSR PC,SILOLD :LOAD SILO WITH MORE OF COUNT PATTERN
4149 020746 012703 000073 MOV #73,R3 :RELOAD CHARACTER COUNT
4150 020752 000751 BR 1$ :CONTINUE
4151 020754 3$:

```

```

4152
4153
4154 :***** TEST 37 *****
4155 :*TRANSMITTER MARK TEST
4156 :*SINGLE CLOCK 3 SYNCs AND A 301 AND 20 EXTRA
4157 :*CLOCK TICKS, VERIFY THAT A 301, A 377 AND A 377
4158 :*WERE RECEIVED INDICATING THAT THE TRANSMITTER WENT
4159 :*TO A MARK STATE FOR 16 BITS WHEN OUT SILO WAS EMPTY
4160 :*****

```

```

4161 : TEST 37
4162 :-----
4163

```

```

4164 :*****
4165 020754 000004 TST37: SCOPE
4166 020756 012737 000037 001202 MOV #37,$STSTNM : LOAD THE NO. OF THIS TEST
4167 020764 012737 021114 001442 MOV #TST40,NEXT : POINT TO THE START OF NEXT TEST.
4168 :R1 CONTAINS BASE KMC11 ADDRESS
4169 020772 104410 MSTCLR :MASTER CLEAR KMC11
4170 020774 012711 004000 MOV #BIT11,(R1) :SET LINE UNIT LOOP
4171 021000 004737 030444 JSR PC,CHAR :LOAD SILO WITH 3 SYNCs
4172 021004 000301 301 :AND A 301
4173 021006 104413 000073 DATACLK, 73 :CLOCK THE 301 IN AND 20 EXTRA TICKS
4174 021012 004737 031066 JSR PC,INRDY :WAIT FOR INRDY
4175 021016 104412 ROMCLK :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4176 021020 021204 021204 :PORT4 IN DATA
4177 021022 016104 000004 MOV 4(R1),R4 :PUT 'FOUND' IN R4
4178 021026 012705 000301 MOV #301,R5 :PUT 'EXPECTED' IN R5
4179 021032 120504 CMPB R5,R4 :WAS A 301 RECEIVED?
4180 021034 001401 BEQ 1$
4181 021036 104010 ERROR 10 :ERROR FIRST CHARACTER INCORRECT
4182 021040 004737 031066 1$: JSR PC,INRDY :WAIT FOR INRDY
4183 021044 104412 ROMCLK :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4184 021046 021204 021204 :PORT4 IN DATA
4185 021050 016104 000004 MOV 4(R1),R4 :PUT 'FOUND' IN R4
4186 021054 012705 000377 MOV #377,R5 :PUT 'EXPECTED' IN R5
4187 021060 120504 CMPB R5,R4 :WAS A 377 RECEIVED?
4188 021062 001401 BEQ 2$
4189 021064 104010 ERROR 10 :ERROR, 377 WAS NOT RECEIVED
4190 021066 004737 031066 2$: JSR PC,INRDY :WAIT FOR INRDY
4191 021072 104412 ROMCLK :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304

```

```

4192 021074 021204          021204          ;PORT4 IN DATA
4193 021076 016104 000004  MOV      4(R1),R4      ;PUT 'FOUND' IN R4
4194 021102 012705 000377  MOV      #377,R5      ;PUT 'EXPECTED' IN R5
4195 021106 120504          CMPB     R5,R4        ;WAS A 377 RECEIVED?
4196 021110 001401          BEQ      3$           ;
4197 021112 104010          ERROR    10           ;ERROR, 177 WAS NOT RECEIVED
4198 021114
4199
4200
4201          ;***** TEST 40 *****
4202          ;*CABLE TURNAROUND TEST
4203          ;*CLEAR LINE UNIT LOOP, SET DTR
4204          ;*VERIFY THAT RING AND MODEM READY ARE SET
4205          ;*CLEAR DTR, VERIFY THAT RING AND MRDY ARE CLEARED
4206          ;*****
4207
4208          ; TEST 40
4209          ;-----
4210          ;*****
4211 021114 000004          TST40: SCOPE
4212 021116 012737 000040 001202  MOV      #40,$TSTNM    ; LOAD THE NO. OF THIS TEST
4213 021124 012737 021342 001442  MOV      #TST41,NEXT   ; POINT TO THE START OF NEXT TEST.
4214          ;R1 CONTAINS BASE KMC11 ADDRESS
4215 021132 104410          MSTCLR   ;MASTER CLEAR KMC11
4216 021134 032737 020000 002050  BIT      #BIT13,STAT1  ;IS LINE UNIT M8202?
4217 021142 001004          BNE     .+12          ;BR IF YES (DO TEST EVEN IF NO LOOP-BACK CONN)
4218 021144 032737 040000 002050  BIT      #BIT14,STAT1  ;IS TURNAROUND CONNECTOR ON?
4219 021152 001473          BEQ     2$           ;SKIP TEST IF NO
4220 021154 005011          CLR     (R1)         ;CLEAR LINE UNIT LOOP
4221 021156 012761 000100 000004  MOV      #100,4(R1)    ;LOAD PORT4
4222 021164 104412          ROMCLK  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4223 021166 122113          TIMER,  2           ;SET DTR
4224 021170 104414 000002          ;WAIT
4225 021174 104412          ROMCLK  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4226 021176 021264          ;PORT4 LU13
4227 021200 016104 000004  MOV      4(R1),R4      ; PUT FOUND IN R4      ++++NEW
4228 021204 042704 000023  BIC     #23,R4        ; CLEAR JUNK
4229 021210 012705 000310  MOV      #310,R5      ; GET EXPECTED.
4230 021214 032737 020000 002050  BIT      #BIT13,STAT1  ; IS LINE UNIT M8202?
4231 021222 001402          BEQ     .+6          ;
4232 021224 042705 000200          BIC     #BIT7,R5      ; NO RING ON M8202
4233
4234 021230 032737 000004 002054  BIT      #BIT2,STAT3   ; IS THIS V.35 MODEM?
4235 021236 001402          BEQ     3$           ; NO THEN GO AHEAD.
4236
4237 021240 042705 000200          BIC     #BIT7,R5      ; YES-NO RING ON V.35 MODEM
4238 021244
4239 021244 020504          3$:  CMP     R5,R4        ; ARE RING AND MODEM READY SET?
4240 021246 001401          BEQ     1$           ; WARNING! IF V.35 AND AUTO STARTED,
4241          ; YOU WILL GET THIS ERROR. YOU MUST
4242          ; MANUALL NASWER THE QUESTIONS FOR V.35.
4243 021250 104011          1$:  ERROR    11        ;ERROR, RING OR MRDY NOT SET
4244 021252 005061 000004  CLR     4(R1)        ;CLEAR PORT4
4245 021256 104412          ROMCLK  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4246 021260 122113          TIMER,  2           ;CLEAR DTR
4247 021262 104414 000002
    
```

```

4248 021266 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4249 021270 021264 ;PORT4 LU13
4250 021272 016104 000004 MOV 4(R1),R4 ;PUR FOUND IN R4 ***NEW
4251 021276 042704 000023 BIC #23,R4 ;STRIP JUNK
4252 021302 005005 CLR R5 ;SET EXPECTED.
4253 021304 032737 020000 002050 BIT #BIT13,STAT1 ;IS THIS A M8202?
4254 021312 001402 BEQ .+6
4255 021314 052705 000010 BIS #BIT3,R5 ;YES THEN EXPECT MRDY SET.
4256 021320 032737 000004 002054 BIT #BIT2,STAT3 ;IS THIS V.35?
4257 021326 001402 BEQ 4$
4258 021330 042704 000200 BIC #BIT7,R4
4259 021334 4$:
4260 021334 120405 CMPB R4,R5 ;ARE RING AND MRDY CLEAR?
4261 021336 001401 BEQ 2$
4262 ;WARNING! YOU MAY GET THIS ERROR IF V.35
4263 ;AND AUTOSTART. YOU MUST MANNUALLY ANSWER
4264 ;ALL QUESTIONS IF V.35.
4265 021340 104011 ERROR 11 ;ERROR, RING OR MRDY NOT CLEAR
4266 021342 2$:
4267
4268
4269 :***** TEST 41 *****
4270 :*CABLE TURNAROUND TEST
4271 :*CLEAR LINE UNIT LOOP, LOAD OUT DATA SILO
4272 :*VERIFY THAT ALL MODEM SIGNALS ARE SET
4273 :*****
4274
4275 : TEST 41
4276 :-----
4277 :*****
4278 021342 000004 TST41: SCOPE
4279 021344 012737 000041 001202 MOV #41,$TSTNM ;LOAD THE NO. OF THIS TEST
4280 021352 012737 021536 001442 MOV #TST42,NEXT ;POINT TO THE START OF NEXT TEST.
4281 ;R1 CONTAINS BASE KMC11 ADDRESS
4282 021360 104410 MSTCLR ;MASTER CLEAR KMC11
4283 021362 032737 020000 002050 BIT #BIT13,STAT1 ;IS LINE UNIT M8202?
4284 021370 001004 BNE .+12 ;BR IF YES (DO TEST EVEN IF NO LOOP-BACK CONN)
4285 021372 032737 040000 002050 BIT #BIT14,STAT1 ;IS TURNAROUND CONNECTOR ON?
4286 021400 001456 BEQ 1$ ;SKIP TEST IF NO
4287 021402 012711 004000 MOV #BIT11,(R1) ;SET LINE UNIT LOOP
4288 021406 012761 000100 000004 MOV #100, 4(R1) ;LOAD PORT4
4289 021414 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4290 021416 122113 ;CLEAR ALL MODEM SIGNALS,EXCEPT DTR
4291 021420 104414 000002 TIMER, 2 ;WAIT
4292 021424 012761 000001 000004 MOV #1,4(R1) ;LOAD PORT4
4293 021432 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4294 021434 122111 122111 ;SET SOM
4295 021436 004537 031530 JSR R5,MESLD ;FILL OUT DATA SILO
4296 021442 032012 MESDAT ;WITH 64 CHARACTERS
4297 021444 000100 64.
4298 021446 012700 000050 MOV #50,R0 ;PREPARE FOR DELAY
4299 021452 005011 CLR (R1) ;CLEAR LINE UNIT LOOP
4300 2$:
4301 021454 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4302 021456 021264 ;PORT4 LU13
4303 021460 016104 000004 MOV 4(R1),R4 ;PUT 'FOUND' IN R4

```

```

4304 021464 042704 000023      BIC   #23,R4      ;CLEAR UNWANTED BITS
4305 021470 012705 000354      MOV   #354,R5     ;PUT 'EXPECTED' IN R5
4306 021474 032737 000004 002054  BIT   #BIT2,STAT3
4307 021502 001402              BEQ   4$
4308 021504 042705 000200      RIC   #BIT7,R5    ; NO RING ON V.35
4309 021510              4$:
4310 021510 032737 020000 002050  BIT   #BIT13,STAT1 ;IS LINE UNIT M8202?
4311 021516 001402              BEQ   .+6         ;BR IF NO
4312 021520 042705 000200      BIC   #BIT7,R5    ;NO RING ON M8202
4313 021524 120504              CMPB  R5,R4       ;COMPARE EXPECTED AND FOUND
4314 021526 001403              BEQ   1$         ;BR IF OK
4315 021530 005300              DEC   R0          ;DEC DELAY COUNT
4316 021532 001350              BNE   2$         ;BR IF NOT ZERO
4317 021534 104011              ERROR 11         ;ERROR, ALL SIGNALS ARE NOT SET
4318                                ; WARNING YOU MUST MANUALLY ANSWER QUESTIONS
4319                                ; IF YOU HAVE V.35.
4320 021536              1$:
4321
4322
4323                                ;***** TEST 42 *****
4324                                ;*TEST OF CRC OPERATION
4325                                ;*USING THE CRC16 POLYNOMIAL, SINGLE CLOCK THE CHARACTER
4326                                ;*0, VERIFY THE LSB OF THE BCC ON EACH SHIFT
4327                                ;*TEST TRANSMITTER FIRST THEN THE RECEIVER BCC
4328                                ;*****
4329
4330                                ; TEST 42
4331                                ;-----
4332                                ;*****
4333 021536 000004      TST42: SCOPE
4334 021540 012737 000042 001202  MOV   #42,$STSTM  ; LOAD THE NO. OF THIS TEST
4335 021546 012737 022052 001442  MOV   #TST43,NEXT ; POINT TO THE START OF NEXT TEST.
4336 021554 012737 021570 001444  MOV   #64$,LOCK   ; ADDRESS FOR LOCK ON DATA.
4337                                ;R1 CONTAINS BASE KMC11 ADDRESS
4338 021562 104410      MSTCLR  ;MASTER CLEAR KMC11
4339 021564 012711 004000 64$: MOV   #BIT11,(R1) ;SET LU LOOP
4340 021570 004737 031572  JSR   PC,CLRIO   ;CLEAR BCC REGISTERS
4341 021574 005000      CLR   R0         ;START SHIFT COUNTER AT ZERO
4342 021576 012737 120001 031226  MOV   #CRC16,XPOLY ;LOAD POLYNOMIAL FOR SOFTWARE BCC
4343 021604 012737 000000 021644  MOV   #0,66$     ;LOAD CHAR FOR SOFTWARE BCC
4344 021612 005037 021646      CLR   67$       ;CLEAR OLD SOFTWARE BCC
4345 021616 004737 031232  JSR   PC,BCCLD   ;LOAD OUT SILO WITH 2 SYNC'S
4346 021622 000000      0           ;AND THE CHARACTER 0
4347 021624 104413 000021 65$: DATACLK, 21  ;GET TRANSMITTER ACTIVE
4348 021630 104413 000001      DATACLK, 1    ;SHIFT BCC ONCE
4349 021634 005200      INC   R0        ;BUMP SHIFT COUNT
4350 021636 004537 031122  JSR   R5,SIMBCC  ;CALCULATE SOFTWARE BCC LSB
4351 021642 000001      1           ;ONE SHIFT
4352 021644 000000 66$: 0           ;DATA CHARACTER
4353 021646 000000 67$: 0           ;OLD BCC
4354 021650 103405      BCS   68$       ;BR IF SOFT BCC LSB IS SET
4355 021652 004737 031344  JSR   PC,GETQ0   ;GET HARDWARE TRANSMITTER BCC LSB
4356 021656 103006      BCC   69$       ;BR IF HARD BCC LSB IS CLEAR
4357 021660 104012      ERROR 12       ;ERROR, BCC LSB IS SET
4358 021662 000404      BR    69$      ;CONTINUE
4359 021664 004737 031344 68$: JSR   PC,GETQ0   ;GET HARDWARE TRANSMITTER BCC LSB

```

|      |        |        |        |        |          |              |          |  |
|------|--------|--------|--------|--------|----------|--------------|----------|--|
| 4360 | 021670 | 103401 |        |        | BCS      | 69%          |          | ;BR IF HARD BCC LSB IS SET                               |
| 4361 | 021672 | 104016 |        |        | ERROR    | 16           |          | ;ERROR, HARD BCC LSB IS CLEAR                            |
| 4362 | 021674 |        |        |        | 69%:     |              |          |  |
| 4363 | 021674 | 006037 | 021644 |        | ROR      | 66%          |          | ;SHIFT SOFT DATA   |
| 4364 | 021700 | 013737 | 031230 | 021646 | MOV      | CALBCC,67%   |          | ;LOAD OLD SOFT BCC                                       |
| 4365 | 021706 | 022700 | 000010 |        | CMP      | #10,RO       |          | ;DONE YET?   |
| 4366 | 021712 | 001346 |        |        | BNE      | 65%          |          | ;BR IF NOT DONE  |
| 4367 | 021714 | 104405 |        |        | SCOPE1   |              |          | ;SCOPE SUBTEST (SW09=1)                                  |
| 4368 | 021716 | 012737 | 021724 | 001444 | MOV      | #71%,LOCK    |          | ;NEW SCOPE1  |
| 4369 | 021724 | 004737 | 031572 |        | 71%:     | JSR          | PC,CLRIO | ;CLEAR BCC REGISTERS                                     |
| 4370 | 021730 | 005000 |        |        | CLR      | RO           |          | ;START SHIFT COUNTER AT ZERO                             |
| 4371 | 021732 | 012737 | 120001 | 031226 | MOV      | #CRC16,XPOLY |          | ;LOAD POLYNOMIAL FOR SOFTWARE BCC                        |
| 4372 | 021740 | 012737 | 000000 | 022000 | MOV      | #0,73%       |          | ;LOAD CHAR FOR SOFTWARE BCC                              |
| 4373 | 021746 | 005037 | 022002 |        | CLR      | 74%          |          | ;CLEAR OLD SOFTWARE BCC                                  |
| 4374 | 021752 | 004737 | 031232 |        | JSR      | PC,BCCLD     |          | ;LOAD OUT SILO WITH 2 SYNC                               |
| 4375 | 021756 | 000000 |        |        | 0        |              |          | ;AND THE CHARACTER 0                                     |
| 4376 | 021760 | 104413 | 000032 |        | DATACLK, | 32           |          | ;GET RECEIVER ACTIVE                                     |
| 4377 | 021764 | 104413 | 000001 |        | 72%:     | DATACLK,     | 1        | ;SHIFT BCC ONCE  |
| 4378 | 021770 | 005200 |        |        | INC      | RO           |          | ;BUMP SHIFT COUNT  |
| 4379 | 021772 | 004537 | 031122 |        | JSR      | R5,SIMBCC    |          | ;CALCULATE SOFTWARE BCC LSB                              |
| 4380 | 021776 | 000001 |        |        | 1        |              |          | ;ONE SHIFT   |
| 4381 | 022000 | 000000 |        |        | 73%:     | 0            |          | ;DATA CHARACTER  |
| 4382 | 022002 | 000000 |        |        | 74%:     | 0            |          | ;OLD BCC   |
| 4383 | 022004 | 103405 |        |        | BCS      | 75%          |          | ;BR IF SOFT BCC LSB IS SET                               |
| 4384 | 022006 | 004737 | 031356 |        | JSR      | PC,GETQI     |          | ;GET HARDWARE RECEIVER BCC LSB                           |
| 4385 | 022012 | 103006 |        |        | BCC      | 76%          |          | ;BR IF HARD BCC LSB IS CLEAR                             |
| 4386 | 022014 | 104013 |        |        | ERROR    | 13           |          | ;ERROR, BCC LSB IS SET                                   |
| 4387 | 022016 | 000404 |        |        | BR       | 76%          |          | ;CONTINUE  |
| 4388 | 022020 | 004737 | 031356 |        | 75%:     | JSR          | PC,GETQI | ;GET HARDWARE RECEIVER BCC LSB                           |
| 4389 | 022024 | 103401 |        |        | BCS      | 76%          |          | ;BR IF HARD BCC LSB IS SET                               |
| 4390 | 022026 | 104017 |        |        | ERROR    | 17           |          | ;ERROR, BCC LSB IS CLEAR                                 |
| 4391 | 022030 |        |        |        | 76%:     |              |          |  |
| 4392 | 022030 | 006037 | 022000 |        | ROR      | 73%          |          | ;SHIFT SOFT DATA   |
| 4393 | 022034 | 013737 | 031230 | 022002 | MOV      | CALBCC,74%   |          | ;LOAD OLD SOFT BCC                                       |
| 4394 | 022042 | 022700 | 000010 |        | CMP      | #10,RO       |          | ;DONE YET?   |
| 4395 | 022046 | 001346 |        |        | BNE      | 72%          |          | ;BR IF NOT DONE  |
| 4396 | 022050 | 104405 |        |        | SCOPE1   |              |          | ;SCOPE SUBTEST (SW09=1)                                  |
| 4397 | 022052 |        |        |        | 77%:     |              |          |  |
| 4398 |        |        |        |        |          |              |          |  |
| 4399 |        |        |        |        |          |              |          |  |
| 4400 |        |        |        |        |          |              |          | ;***** TEST 43 *****                                     |
| 4401 |        |        |        |        |          |              |          | ;*TEST OF CRC OPERATION                                  |
| 4402 |        |        |        |        |          |              |          | ;*USING THE CRC16 POLYNOMIAL, SINGLE CLOCK THE CHARACTER |
| 4403 |        |        |        |        |          |              |          | ;*377, VERIFY THE LSB OF THE BCC ON EACH SHIFT           |
| 4404 |        |        |        |        |          |              |          | ;*TEST TRANSMITTER FIRST THEN THE RECEIVER BCC           |
| 4405 |        |        |        |        |          |              |          | ;*****   |
| 4406 |        |        |        |        |          |              |          |  |
| 4407 |        |        |        |        |          |              |          | ; TEST 43  |
| 4408 |        |        |        |        |          |              |          | -----  |
| 4409 |        |        |        |        |          |              |          | ;*****   |
| 4410 | 022052 | 000004 |        |        | TST43:   | SCOPE        |          |  |
| 4411 | 022054 | 012737 | 000043 | 001202 | MOV      | #43,\$TSTNM  |          | ; LOAD THE NO. OF THIS TEST                              |
| 4412 | 022062 | 012737 | 022366 | 001442 | MOV      | #TST44,NEXT  |          | ; POINT TO THE START OF NEXT TEST.                       |
| 4413 | 022070 | 012737 | 022104 | 001444 | MOV      | #64\$,LOCK   |          | ; ADDRESS FOR LOCK ON DATA.                              |
| 4414 |        |        |        |        |          |              |          | ;R1 CONTAINS BASE KMC11 ADDRESS                          |
| 4415 | 022076 | 104410 |        |        | MSTCLR   |              |          | ;MASTER CLEAR KMC11                                      |

|      |        |        |        |        |       |          |              |                                   |
|------|--------|--------|--------|--------|-------|----------|--------------|-----------------------------------|
| 4416 | 022100 | 012711 | 004000 |        |       | MOV      | #BIT11,(R1)  | ;SET LU LOOP                      |
| 4417 | 022104 | 004737 | 031572 |        | 64\$: | JSR      | PC,CLR!      | ;CLEAR BCC REGISTERS              |
| 4418 | 022110 | 005000 |        |        |       | CLR      | R0           | ;START SHIFT COUNTER AT ZERO      |
| 4419 | 022112 | 012737 | 120001 | 031226 |       | MOV      | #CRC16,XPOLY | ;LOAD POLYNOMIAL FOR SOFTWARE BCC |
| 4420 | 022120 | 012737 | 000377 | 022160 |       | MOV      | #377,66\$;   | ;LOAD CHAR FOR SOFTWARE BCC       |
| 4421 | 022126 | 005037 | 022162 |        |       | CLR      | 67\$         | ;CLEAR OLD SOFTWARE BCC           |
| 4422 | 022132 | 004737 | 031232 |        |       | JSR      | PC,BCCLD     | ;LOAD OUT SILO WITH 2 SYNC        |
| 4423 | 022136 | 000377 |        |        |       | 377      |              | ;AND THE CHARACTER 377            |
| 4424 | 022140 | 104413 | 000021 |        |       | DATACLK, | 21           | ;GET TRANSMITTER ACTIVE           |
| 4425 | 022144 | 104413 | 000001 |        | 65\$: | DATACLK, | 1            | ;SHIFT BCC ONCE                   |
| 4426 | 022150 | 005200 |        |        |       | INC      | R0           | ;BUMP SHIFT COUNT                 |
| 4427 | 022152 | 004537 | 031122 |        |       | JSR      | R5,SIMBCC    | ;CALCULATE SOFTWARE BCC LSB       |
| 4428 | 022156 | 000001 |        |        |       | 1        |              | ;ONE SHIFT                        |
| 4429 | 022160 | 000000 |        |        | 66\$: | 0        |              | ;DATA CHARACTER                   |
| 4430 | 022162 | 000000 |        |        | 67\$: | 0        |              | ;OLD BCC                          |
| 4431 | 022164 | 103405 |        |        |       | BCS      | 68\$         | ;BR IF SOFT BCC LSB IS SET        |
| 4432 | 022166 | 004737 | 031344 |        |       | JSR      | PC,GETQ0     | ;GET HARDWARE TRANSMITTER BCC LSB |
| 4433 | 022172 | 103006 |        |        |       | BCC      | 69\$         | ;BR IF HARD BCC LSB IS CLEAR      |
| 4434 | 022174 | 104012 |        |        |       | ERROR    | 12           | ;ERROR, BCC LSB IS SET            |
| 4435 | 022176 | 000404 |        |        |       | BR       | 69\$         | ;CONTINUE                         |
| 4436 | 022200 | 004737 | 031344 |        | 68\$: | JSR      | PC,GETQ0     | ;GET HARDWARE TRANSMITTER BCC LSB |
| 4437 | 022204 | 103401 |        |        |       | BCS      | 69\$         | ;BR IF HARD BCC LSB IS SET        |
| 4438 | 022206 | 104016 |        |        |       | ERROR    | 16           | ;ERROR, HARD BCC LSB IS CLEAR     |
| 4439 | 022210 |        |        |        | 69\$: |          |              |                                   |
| 4440 | 022210 | 006037 | 022160 |        |       | ROR      | 66\$         | ;SHIFT SOFT DATA                  |
| 4441 | 022214 | 013737 | 031230 | 022162 |       | MOV      | CALBCC,67\$  | ;LOAD OLD SOFT BCC                |
| 4442 | 022222 | 022700 | 000010 |        |       | CMP      | #10,R0       | ;DONE YET?                        |
| 4443 | 022226 | 001346 |        |        |       | BNE      | 65\$         | ;BR IF NOT DONE                   |
| 4444 | 022230 | 104405 |        |        |       | SCOPE1   |              | ;SCOPE SUBTEST (SW09=1)           |
| 4445 | 022232 | 012737 | 022240 | 001444 |       | MOV      | #71\$,LOCK   | ;NEW SCOPE1                       |
| 4446 | 022240 | 004737 | 031572 |        | 71\$: | JSR      | PC,CLR!0     | ;CLEAR BCC REGISTERS              |
| 4447 | 022244 | 005000 |        |        |       | CLR      | R0           | ;START SHIFT COUNTER AT ZERO      |
| 4448 | 022246 | 012737 | 120001 | 031226 |       | MOV      | #CRC16,XPOLY | ;LOAD POLYNOMIAL FOR SOFTWARE BCC |
| 4449 | 022254 | 012737 | 000377 | 022314 |       | MOV      | #377,73\$;   | ;LOAD CHAR FOR SOFTWARE BCC       |
| 4450 | 022262 | 005037 | 022316 |        |       | CLR      | 74\$         | ;CLEAR OLD SOFTWARE BCC           |
| 4451 | 022266 | 004737 | 031232 |        |       | JSR      | PC,BCCLD     | ;LOAD OUT SILO WITH 2 SYNC        |
| 4452 | 022272 | 000377 |        |        |       | 377      |              | ;AND THE CHARACTER 377            |
| 4453 | 022274 | 104413 | 000032 |        |       | DATACLK, | 32           | ;GET RECEIVER ACTIVE              |
| 4454 | 022300 | 104413 | 000001 |        | 72\$: | DATACLK, | 1            | ;SHIFT BCC ONCE                   |
| 4455 | 022304 | 005200 |        |        |       | INC      | R0           | ;BUMP SHIFT COUNT                 |
| 4456 | 022306 | 004537 | 031122 |        |       | JSR      | R5,SIMBCC    | ;CALCULATE SOFTWARE BCC LSB       |
| 4457 | 022312 | 000001 |        |        |       | 1        |              | ;ONE SHIFT                        |
| 4458 | 022314 | 000000 |        |        | 73\$: | 0        |              | ;DATA CHARACTER                   |
| 4459 | 022316 | 000000 |        |        | 74\$: | 0        |              | ;OLD BCC                          |
| 4460 | 022320 | 103405 |        |        |       | BCS      | 75\$         | ;BR IF SOFT BCC LSB IS SET        |
| 4461 | 022322 | 004737 | 031356 |        |       | JSR      | PC,GETQ1     | ;GET HARDWARE RECEIVER BCC LSB    |
| 4462 | 022326 | 103006 |        |        |       | BCC      | 76\$         | ;BR IF HARD BCC LSB IS CLEAR      |
| 4463 | 022330 | 104013 |        |        |       | ERROR    | 13           | ;ERROR, BCC LSB IS SET            |
| 4464 | 022332 | 000404 |        |        |       | BR       | 76\$         | ;CONTINUE                         |
| 4465 | 022334 | 004737 | 031356 |        | 75\$: | JSR      | PC,GETQ1     | ;GET HARDWARE RECEIVER BCC LSB    |
| 4466 | 022340 | 103401 |        |        |       | BCS      | 76\$         | ;BR IF HARD BCC LSB IS SET        |
| 4467 | 022342 | 104017 |        |        |       | ERROR    | 17           | ;ERROR, BCC LSB IS CLEAR          |
| 4468 | 022344 |        |        |        | 76\$: |          |              |                                   |
| 4469 | 022344 | 006037 | 022314 |        |       | ROR      | 73\$         | ;SHIFT SOFT DATA                  |
| 4470 | 022350 | 013737 | 031230 | 022316 |       | MOV      | CALBCC,74\$  | ;LOAD OLD SOFT BCC                |
| 4471 | 022356 | 022700 | 000010 |        |       | CMP      | #10,R0       | ;DONE YET?                        |

4472 022362 001346 BNE 72\$ ;BR IF NOT DONE  
 4473 022364 104405 SCOP1 ;SCOPE SUBTEST (SW09=1)  
 4474 022366 77\$:

\*\*\*\*\* TEST 44 \*\*\*\*\*  
 :\*TEST OF CRC OPERATION  
 :\*USING THE CRC16 POLYNOMIAL, SINGLE CLOCK THE CHARACTER  
 :\*125, VERIFY THE LSB OF THE BCC ON EACH SHIFT  
 :\*TEST TRANSMITTER FIRST THEN THE RECEIVER BCC  
 :\*\*\*\*\*

: TEST 44

```

4487 022366 000004 TST44: SCOPE
4488 022370 012737 000044 001202 MOV #44,$TSTNM ; LOAD THE NO. OF THIS TEST
4489 022376 012737 022702 001442 MOV #TST45,NEXT ; POINT TO THE START OF NEXT TEST.
4490 022404 012737 022420 001444 MOV #64$,LOCK ; ADDRESS FOR LOCK ON DATA.
4491 ;R1 CONTAINS BASE KMC11 ADDRESS
4492 022412 104410 MSTCLR ;MASTER CLEAR KMC11
4493 022414 012711 004000 MOV #BIT11,(R1) ;SET LU LOOP
4494 022420 004737 031572 64$: JSR PC,CLRIO ;CLEAR BCC REGISTERS
4495 022424 005000 CLR RO ;START SHIFT COUNTER AT ZERO
4496 022426 012737 120001 031226 MOV #CRC16,XPOLY ;LOAD POLYNOMIAL FOR SOFTWARE BCC
4497 022434 012737 000125 022474 MOV #125,66$; ;LOAD CHAR FOR SOFTWARE BCC
4498 022442 005037 022476 CLR 67$ ;CLEAR OLD SOFTWARE BCC
4499 022446 004737 031232 JSR PC,BCCLD ;LOAD OUT SILO WITH 2 SYNCs
4500 022452 000125 125 ;AND THE CHARACTER 125
4501 022454 104413 000021 DATACLK, 21 ;GET TRANSMITTER ACTIVE
4502 022460 104413 000001 65$: DATACLK, 1 ;SHIFT BCC ONCE
4503 022464 005200 INC RO ;BUMP SHIFT COUNT
4504 022466 004537 031122 JSR R5,SIMBCC ;CALCULATE SOFTWARE BCC LSB
4505 022472 000001 1 ;ONE SHIFT
4506 022474 000000 66$: 0 ;DATA CHARACTER
4507 022476 000000 67$: 0 ;OLD BCC
4508 022500 103405 BCS 68$ ;BR IF SOFT BCC LSB IS SET
4509 022502 004737 031344 JSR PC,GETQ0 ;GET HARDWARE TRANSMITTER BCC LSB
4510 022506 103006 BCC 69$ ;BR IF HARD BCC LSB IS CLEAR
4511 022510 104012 ERROR 12 ;ERROR, BCC LSB IS SET
4512 022512 000404 BR 69$ ;CONTINUE
4513 022514 004737 031344 68$: JSR PC,GETQ0 ;GET HARDWARE TRANSMITTER BCC LSB
4514 022520 103401 BCS 69$ ;BR IF HARD BCC LSB IS SET
4515 022522 104016 ERROR 16 ;ERROR, HARD BCC LSB IS CLEAR
4516 022524 69$:
4517 022524 006037 022474 ROR 66$ ;SHIFT SOFT DATA
4518 022530 013737 031230 022476 MOV CALBCC,67$ ;LOAD OLD SOFT BCC
4519 022536 022700 000010 CMP #10,RO ;DONE YET?
4520 022542 001346 BNE 65$ ;BR IF NOT DONE
4521 022544 104405 SCOP1 ;SCOPE SUBTEST (SW09=1)
4522 022546 012737 022554 001444 MOV #71$,LOCK ;NEW SCOPE1
4523 022554 004737 031572 71$: JSR PC,CLRIO ;CLEAR BCC REGISTERS
4524 022560 005000 CLR RO ;START SHIFT COUNTER AT ZERO
4525 022562 012737 120001 031226 MOV #CRC16,XPOLY ;LOAD POLYNOMIAL FOR SOFTWARE BCC
4526 022570 012737 000125 022630 MOV #125,73$; ;LOAD CHAR FOR SOFTWARE BCC
4527 022576 005037 022632 CLR 74$ ;CLEAR OLD SOFTWARE BCC
  
```



|      |        |        |        |        |        |             |    |                                |
|------|--------|--------|--------|--------|--------|-------------|----|--------------------------------|
| 4528 | 022602 | 004737 | 031232 |        | JSR    | PC,BCCLD    |    | ;LOAD OUT SILO WITH 2 SYNCs    |
| 4529 | 022606 | 000125 |        |        |        | 125         |    | ;AND THE CHARACTER 125         |
| 4530 | 022610 | 104413 | 000032 |        |        | DATACLK,    | 32 | ;GET RECEIVER ACTIVE           |
| 4531 | 022614 | 104413 | 000001 | 72\$:  |        | DATACLK,    | 1  | ;SHIFT BCC ONCE                |
| 4532 | 022620 | 005200 |        |        | INC    | RO          |    | ;BUMP SHIFT COUNT              |
| 4533 | 022622 | 004537 | 031122 |        | JSR    | R5,SIMBCC   |    | ;CALCULATE SOFTWARE BCC LSB    |
| 4534 | 022626 | 000001 |        |        |        | 1           |    | ;ONE SHIFT                     |
| 4535 | 022630 | 000000 |        | 73\$:  |        | 0           |    | ;DATA CHARACTER                |
| 4536 | 022632 | 000000 |        | 74\$:  |        | 0           |    | ;OLD BCC                       |
| 4537 | 022634 | 103405 |        |        | BCS    | 75\$        |    | ;BR IF SOFT BCC LSB IS SET     |
| 4538 | 022636 | 004737 | 031356 |        | JSR    | PC,GETQI    |    | ;GET HARDWARE RECEIVER BCC LSB |
| 4539 | 022642 | 103006 |        |        | BCC    | 76\$        |    | ;BR IF HARD BCC LSB IS CLEAR   |
| 4540 | 022644 | 104013 |        |        | ERROR  | 13          |    | ;ERROR, BCC LSB IS SET         |
| 4541 | 022646 | 000404 |        |        | BR     | 76\$        |    | ;CONTINUE                      |
| 4542 | 022650 | 004737 | 031356 | 75\$:  | JSR    | PC,GETQI    |    | ;GET HARDWARE RECEIVER BCC LSB |
| 4543 | 022654 | 103401 |        |        | BCS    | 76\$        |    | ;BR IF HARD BCC LSB IS SET     |
| 4544 | 022656 | 104017 |        |        | ERROR  | 17          |    | ;ERROR, BCC LSB IS CLEAR       |
| 4545 | 022660 |        |        | 76\$:  |        |             |    |                                |
| 4546 | 022660 | 006037 | 022630 |        | ROR    | 73\$        |    | ;SHIFT SOFT DATA               |
| 4547 | 022664 | 013737 | 031230 | 022632 | MOV    | CALBCC,74\$ |    | ;LOAD OLD SOFT BCC             |
| 4548 | 022672 | 022700 | 000010 |        | CMP    | #10,RO      |    | ;DONE YET?                     |
| 4549 | 022676 | 001346 |        |        | BNE    | 72\$        |    | ;BR IF NOT DONE                |
| 4550 | 022700 | 104405 |        |        | SCOPE1 |             |    | ;SCOPE SUBTEST (SW09=1)        |
| 4551 | 022702 |        |        | 77\$:  |        |             |    |                                |

\*\*\*\*\* TEST 45 \*\*\*\*\*  
 ;\*TEST OF CRC OPERATION  
 ;\*USING THE CRC16 POLYNOMIAL, SINGLE CLOCK THE CHARACTER  
 ;\*252, VERIFY THE LSB OF THE BCC ON EACH SHIFT  
 ;\*TEST TRANSMITTER FIRST THEN THE RECEIVER BCC  
 ;\*\*\*\*\*

; TEST 45

|      |        |        |        |        |        |          |              |                                    |
|------|--------|--------|--------|--------|--------|----------|--------------|------------------------------------|
| 4564 | 022702 | 000004 |        |        | TST45: | SCOPE    |              |                                    |
| 4565 | 022704 | 012737 | 000045 | 001202 |        | MOV      | #45,\$TSTNM  | ; LOAD THE NO. OF THIS TEST        |
| 4566 | 022712 | 012737 | 023216 | 001442 |        | MOV      | #TST46,NEXT  | ; POINT TO THE START OF NEXT TEST. |
| 4567 | 022720 | 012737 | 022734 | 001444 |        | MOV      | #64\$,LOCK   | ; ADDRESS FOR LOCK ON DATA.        |
| 4568 |        |        |        |        |        |          |              | ;R1 CONTAINS BASE KMC11 ADDRESS    |
| 4569 | 022726 | 104410 |        |        |        | MSTCLR   |              | ;MASTER CLEAR KMC11                |
| 4570 | 022730 | 012711 | 004000 |        |        | MOV      | #BIT11,(R1)  | ;SET LU LOOP                       |
| 4571 | 022734 | 004737 | 031572 |        | 64\$:  | JSR      | PC,CLRIO     | ;CLEAR BCC REGISTERS               |
| 4572 | 022740 | 005000 |        |        |        | CLR      | RO           | ;START SHIFT COUNTER AT ZERO       |
| 4573 | 022742 | 012737 | 120001 | 031226 |        | MOV      | #CRC16,XPOLY | ;LOAD POLYNOMIAL FOR SOFTWARE BCC  |
| 4574 | 022750 | 012737 | 000252 | 023010 |        | MOV      | #252,66\$;   | ;LOAD CHAR FOR SOFTWARE BCC        |
| 4575 | 022756 | 005037 | 023012 |        |        | CLR      | 67\$         | ;CLEAR OLD SOFTWARE BCC            |
| 4576 | 022762 | 004737 | 031232 |        |        | JSR      | PC,BCCLD     | ;LOAD OUT SILO WITH 2 SYNCs        |
| 4577 | 022766 | 000252 |        |        |        | 252      |              | ;AND THE CHARACTER 252             |
| 4578 | 022770 | 104413 | 000021 |        |        | DATACLK, | 21           | ;GET TRANSMITTER ACTIVE            |
| 4579 | 022774 | 104413 | 000001 |        | 65\$:  | DATACLK, | 1            | ;SHIFT BCC ONCE                    |
| 4580 | 023000 | 005200 |        |        |        | INC      | RO           | ;BUMP SHIFT COUNT                  |
| 4581 | 023002 | 004537 | 031122 |        |        | JSR      | R5,SIMBCC    | ;CALCULATE SOFTWARE BCC LSB        |
| 4582 | 023006 | 000001 |        |        |        | 1        |              | ;ONE SHIFT                         |
| 4583 | 023010 | 000000 |        |        | 66\$:  | 0        |              | ;DATA CHARACTER                    |

BASIC RECEIVER TESTS

SEQ 0090

|      |        |        |        |        |      |          |              |  |  |                                   |
|------|--------|--------|--------|--------|------|----------|--------------|--|--|-----------------------------------|
| 4584 | 023012 | 000000 |        |        | 67%: | 0        |              |  |  | ;OLD BCC                          |
| 4585 | 023014 | 103405 |        |        |      | BCS      | 68%          |  |  | ;BR IF SOFT BCC LSB IS SET        |
| 4586 | 023016 | 004737 | 031344 |        |      | JSR      | PC,GETQ0     |  |  | ;GET HARDWARE TRANSMITTER BCC LSB |
| 4587 | 023022 | 103006 |        |        |      | BCC      | 69%          |  |  | ;BR IF HARD BCC LSB IS CLEAR      |
| 4588 | 023024 | 104012 |        |        |      | ERROR    | 12           |  |  | ;ERROR, BCC LSB IS SET            |
| 4589 | 023026 | 000404 |        |        |      | BR       | 69%          |  |  | ;CONTINUE                         |
| 4590 | 023030 | 004737 | 031344 |        | 68%: | JSR      | PC,GETQ0     |  |  | ;GET HARDWARE TRANSMITTER BCC LSB |
| 4591 | 023034 | 103401 |        |        |      | BCS      | 69%          |  |  | ;BR IF HARD BCC LSB IS SET        |
| 4592 | 023036 | 104016 |        |        |      | ERROR    | 16           |  |  | ;ERROR, HARD BCC LSB IS CLEAR     |
| 4593 | 023040 |        |        |        | 69%: |          |              |  |  |                                   |
| 4594 | 023040 | 006037 | 023010 |        |      | ROR      | 66%          |  |  | ;SHIFT SOFT DATA                  |
| 4595 | 023044 | 013737 | 031230 | 023012 |      | MOV      | CALBCC,67%   |  |  | ;LOAD OLD SOFT BCC                |
| 4596 | 023052 | 022700 | 000010 |        |      | CMP      | #10,RO       |  |  | ;DONE YET?                        |
| 4597 | 023056 | 001346 |        |        |      | BNE      | 65%          |  |  | ;BR IF NOT DONE                   |
| 4598 | 023060 | 104405 |        |        |      | SCOPI    |              |  |  | ;SCOPE SUBTEST (SW09=1)           |
| 4599 | 023062 | 012737 | 023070 | 001444 |      | MOV      | #71%,LOCK    |  |  | ;NEW SCOPE1                       |
| 4600 | 023070 | 004737 | 031572 |        | 71%: | JSR      | PC,CLRIO     |  |  | ;CLEAR BCC REGISTERS              |
| 4601 | 023074 | 005000 |        |        |      | CLR      | RO           |  |  | ;START SHIFT COUNTER AT ZERO      |
| 4602 | 023076 | 012737 | 120001 | 031226 |      | MOV      | #CRC16,XPOLY |  |  | ;LOAD POLYNOMIAL FOR SOFTWARE BCC |
| 4603 | 023104 | 012737 | 000252 | 023144 |      | MOV      | #252,73%;    |  |  | ;LOAD CHAR FOR SOFTWARE BCC       |
| 4604 | 023112 | 005037 | 023146 |        |      | CLR      | 74%          |  |  | ;CLEAR OLD SOFTWARE BCC           |
| 4605 | 023116 | 004737 | 031232 |        |      | JSR      | PC,BCCLD     |  |  | ;LOAD OUT SILO WITH 2 SYNCS       |
| 4606 | 023122 | 000252 |        |        |      | 252      |              |  |  | ;AND THE CHARACTER 252            |
| 4607 | 023124 | 104413 | 000032 |        |      | DATACLK, | 32           |  |  | ;GET RECEIVER ACTIVE              |
| 4608 | 023130 | 104413 | 000001 |        | 72%: | D:TACLK, | 1            |  |  | ;SHIFT BCC ONCE                   |
| 4609 | 023134 | 005200 |        |        |      | INC      | RO           |  |  | ;BUMP SHIFT COUNT                 |
| 4610 | 023136 | 004537 | 031122 |        |      | JSR      | R5,SIMBCC    |  |  | ;CALCULATE SOFTWARE BCC LSB       |
| 4611 | 023142 | 000001 |        |        |      | 1        |              |  |  | ;ONE SHIFT                        |
| 4612 | 023144 | 000000 |        |        | 73%: | 0        |              |  |  | ;DATA CHARACTER                   |
| 4613 | 023146 | 000000 |        |        | 74%: | 0        |              |  |  | ;OLD BCC                          |
| 4614 | 023150 | 103405 |        |        |      | BCS      | 75%          |  |  | ;BR IF SOFT BCC LSB IS SET        |
| 4615 | 023152 | 004737 | 031356 |        |      | JSR      | PC,GETQ1     |  |  | ;GET HARDWARE RECEIVER BCC LSB    |
| 4616 | 023156 | 103006 |        |        |      | BCC      | 76%          |  |  | ;BR IF HARD BCC LSB IS CLEAR      |
| 4617 | 023160 | 104013 |        |        |      | ERROR    | 13           |  |  | ;ERROR, BCC LSB IS SET            |
| 4618 | 023162 | 000404 |        |        |      | BR       | 76%          |  |  | ;CONTINUE                         |
| 4619 | 023164 | 004737 | 031356 |        | 75%: | JSR      | PC,GETQ1     |  |  | ;GET HARDWARE RECEIVER BCC LSB    |
| 4620 | 023170 | 103401 |        |        |      | BCS      | 76%          |  |  | ;BR IF HARD BCC LSB IS SET        |
| 4621 | 023172 | 104017 |        |        |      | ERROR    | 17           |  |  | ;ERROR, BCC LSB IS CLEAR          |
| 4622 | 023174 |        |        |        | 76%: |          |              |  |  |                                   |
| 4623 | 023174 | 006037 | 023144 |        |      | ROR      | 73%          |  |  | ;SHIFT SOFT DATA                  |
| 4624 | 023200 | 013737 | 031230 | 023146 |      | MOV      | CALBCC,74%   |  |  | ;LOAD OLD SOFT BCC                |
| 4625 | 023206 | 022700 | 000010 |        |      | CMP      | #10,RO       |  |  | ;DONE YET?                        |
| 4626 | 023212 | 001346 |        |        |      | BNE      | 72%          |  |  | ;BR IF NOT DONE                   |
| 4627 | 023214 | 104405 |        |        |      | SCOPI    |              |  |  | ;SCOPE SUBTEST (SW09=1)           |
| 4628 | 023216 |        |        |        | 77%: |          |              |  |  |                                   |
| 4629 |        |        |        |        |      |          |              |  |  |                                   |
| 4630 |        |        |        |        |      |          |              |  |  |                                   |
| 4631 |        |        |        |        |      |          |              |  |  |                                   |
| 4632 |        |        |        |        |      |          |              |  |  |                                   |
| 4633 |        |        |        |        |      |          |              |  |  |                                   |
| 4634 |        |        |        |        |      |          |              |  |  |                                   |
| 4635 |        |        |        |        |      |          |              |  |  |                                   |
| 4636 |        |        |        |        |      |          |              |  |  |                                   |
| 4637 |        |        |        |        |      |          |              |  |  |                                   |
| 4638 |        |        |        |        |      |          |              |  |  |                                   |
| 4639 |        |        |        |        |      |          |              |  |  |                                   |

```
***** TEST 46 *****
; *TRANSMITTER CRC TEST
; *USING THE CRC16 POLYNOMIAL, SINGLE CLOCK A BINARY
; *COUNT PATTERN, VERIFY THE LSB OF THE TRANSMITTER BCC ON EACH SHIFT
*****
```

```
; TEST 46
;-----
*****
```

|      |        |        |        |        |        |            |              |  |   |
|------|--------|--------|--------|--------|--------|------------|--------------|--|---|
| 4640 | 023216 | 000004 |        |        | TST46: | SCOPE      |              |  |   |
| 4641 | 023220 | 012737 | 000046 | 001202 |        | MOV        | #46,\$STNM   |  | ; LOAD THE NO. OF THIS TEST                   |
| 4642 | 023226 | 012737 | 023454 | 001442 |        | MOV        | #TST47,NEXT  |  | ; POINT TO THE START OF NEXT TEST.            |
| 4643 |        |        |        |        |        |            |              |  | ;R1 CONTAINS BASE KMC11 ADDRESS               |
| 4644 | 023234 | 104410 |        |        |        | MSTCLR     |              |  | ;MASTER CLEAR KMC11                           |
| 4645 | 023236 | 012711 | 004000 |        |        | MOV        | #BIT11,(R1)  |  | ;SET LINE UNIT LOOP                           |
| 4646 | 023242 | 005003 |        |        |        | CLR        | R3           |  | ;ZERO BIT COUNT                               |
| 4647 | 023244 | 005004 |        |        |        | CLR        | R4           |  | ;R4 CONTAINS CHAR TO BE LOADED IN SILO        |
| 4648 | 023246 | 005005 |        |        |        | CLR        | R5           |  | ;R5 CONTAINS CHAR CURRENTLY BEING SHIFTED OUT |
| 4649 | 023250 | 005037 | 023352 |        |        | CLR        | 4\$          |  | ;CLEAR SOFT BCC                               |
| 4650 | 023254 | 012737 | 120001 | 031226 |        | MOV        | #CRC16,XPOLY |  | ;LOAD POLYNOMIAL                              |
| 4651 | 023262 | 004737 | 031374 |        |        | JSR        | PC,SYNLD     |  | ;LOAD SILO WITH 2 SYNCs, SOM SET              |
| 4652 | 023266 | 010461 | 000004 |        |        | MOV        | R4,4(R1)     |  | ;PORT4 CHAR                                   |
| 4653 | 023272 | 104412 |        |        |        | ROMCLK     |              |  | ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304     |
| 4654 | 023274 | 122110 |        |        |        | 122110     |              |  | ;LOAD OUT DATA                                |
| 4655 | 023276 | 005204 |        |        |        | INC        | R4           |  | ;INCREMENT TO NEXT CHARACTER                  |
| 4656 | 023300 | 010461 | 000004 |        |        | MOV        | R4,4(R1)     |  | ;PORT4 CHAR                                   |
| 4657 | 023304 | 104412 |        |        |        | ROMCLK     |              |  | ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304     |
| 4658 | 023306 | 122110 |        |        |        | 122110     |              |  | ;LOAD OUT DATA                                |
| 4659 | 023310 | 005204 |        |        |        | INC        | R4           |  | ;INCREMENT TO NEXT CHARACTER                  |
| 4660 | 023312 | 010461 | 000004 |        |        | MOV        | R4,4(R1)     |  | ;PORT4 CHAR                                   |
| 4661 | 023316 | 104412 |        |        |        | ROMCLK     |              |  | ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304     |
| 4662 | 023320 | 122110 |        |        |        | 122110     |              |  | ;LOAD OUT DATA                                |
| 4663 | 023322 | 004737 | 030260 |        |        | JSR        | PC,OCOR      |  | ;WAIT FOR OCOR                                |
| 4664 | 023326 | 104413 | 000021 |        |        | DATACLK,21 |              |  | ;CLOCK DATA                                   |
| 4665 | 023332 | 010537 | 023350 |        | 1\$:   | MOV        | R5,3\$       |  | ;LOAD CHAR FOR SOFT CRC                       |
| 4666 | 023336 | 104413 | 000001 |        | 2\$:   | DATACLK,1  |              |  | ;SHIFT BCC ONCE                               |
| 4667 | 023342 | 004537 | 031122 |        |        | JSR        | R5,SIMBCC    |  | ;CALCULATE SOFT BCC                           |
| 4668 | 023346 | 000001 |        |        |        | 1          |              |  | ;SOFT SHIFT COUNT                             |
| 4669 | 023350 | 000000 |        |        | 3\$:   | 0          |              |  | ;SOFT CHARACTER                               |
| 4670 | 023352 | 000000 |        |        | 4\$:   | 0          |              |  | ;OLD SOFT BCC                                 |
| 4671 | 023354 | 103405 |        |        |        | BCC        | 5\$          |  | ;BR IF SOFT BCC LSB IS SET                    |
| 4672 | 023356 | 004737 | 031344 |        |        | JSR        | PC,GETQO     |  | ;GET HARDWARE TRANSMITTER BCC LSB             |
| 4673 | 023362 | 103006 |        |        |        | BCC        | 6\$          |  | ;BR IF OK (CLEARED)                           |
| 4674 | 023364 | 104020 |        |        |        | ERROR      | 20           |  | ;ERROR, BCC LSB WAS SET                       |
| 4675 | 023366 | 000404 |        |        |        | BR         | 6\$          |  | ;CONTINUE WITH TEST                           |
| 4676 | 023370 | 004737 | 031344 |        | 5\$:   | JSR        | PC,GETQO     |  | ;GET HARDWARE TRANSMITTER BCC LSB             |
| 4677 | 023374 | 103401 |        |        |        | BCC        | 6\$          |  | ;BR IF OK (SET)                               |
| 4678 | 023376 | 104021 |        |        |        | ERROR      | 21           |  | ;ERROR, BCC LSB WAS CLEAR                     |
| 4679 |        |        |        |        |        |            |              |  |   |
| 4680 | 023400 |        |        |        | 6\$:   |            |              |  | ;SHIFT SOFT DATA                              |
| 4681 | 023400 | 006037 | 023350 |        |        | ROR        | 3\$          |  | ;LOAD OLD SOFT BCC                            |
| 4682 | 023404 | 013737 | 031230 | 023352 |        | MOV        | CALBCC,4\$   |  | ;INCREMENT BIT COUNTER                        |
| 4683 | 023412 | 005203 |        |        |        | INC        | R3           |  | ;DONE A FULL CHARACTER YET?                   |
| 4684 | 023414 | 022703 | 000010 |        |        | CMP        | #10,R3       |  | ;BR IF NO                                     |
| 4685 | 023420 | 001346 |        |        |        | BNE        | 2\$          |  | ;RESTART BIT COUNTER                          |
| 4686 | 023422 | 005003 |        |        |        | CLR        | R3           |  | ;INCREMENT DATA FOR SILO                      |
| 4687 | 023424 | 005204 |        |        |        | INC        | R4           |  | ;DONE BINARY COUNT YET?                       |
| 4688 | 023426 | 022704 | 000400 |        |        | CMP        | #400,R4      |  | ;BR IF YES                                    |
| 4689 | 023432 | 003404 |        |        |        | BLE        | 9\$          |  | ;PORT4 DATA                                   |
| 4690 | 023434 | 010461 | 000004 |        |        | MOV        | R4,4(R1)     |  | ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304     |
| 4691 | 023440 | 104412 |        |        |        | ROMCLK     |              |  | ;LOAD OUT DATA                                |
| 4692 | 023442 | 122110 |        |        |        | 122110     |              |  | ;INCREMENT DATA                               |
| 4693 | 023444 | 005205 |        |        | 9\$:   | INC        | R5           |  | ;DONE BINARY PATTERN YET?                     |
| 4694 | 023446 | 022705 | 000400 |        |        | CMP        | #400,R5      |  | ;BR IF NO                                     |
| 4695 | 023448 | 001327 |        |        |        | BNE        | 1\$          |  |   |

4696 023454 7\$:  
4697  
4698  
4699  
4700  
4701  
4702  
4703  
4704  
4705  
4706  
4707  
4708 023454 000004  
4709 023456 012737 000047 001202  
4710 023464 012737 023712 001442  
4711  
4712 023472 104410  
4713 023474 012711 004000  
4714 023500 005003  
4715 023502 005004  
4716 023504 005005  
4717 023506 005037 023610  
4718 023512 012737 120001 031226  
4719 023520 004737 031374  
4720 023524 010461 000004  
4721 023530 104412  
4722 023532 122110  
4723 023534 005204  
4724 023536 010461 000004  
4725 023542 104412  
4726 023544 122110  
4727 023546 005204  
4728 023550 010461 000004  
4729 023554 104412  
4730 023556 122110  
4731 023560 004737 030260  
4732 023564 104413 000032  
4733 023570 010537 023606 1\$:  
4734 023574 104413 000001 2\$:  
4735 023600 004537 031122  
4736 023604 000001  
4737 023606 000000 3\$:  
4738 023610 000000 4\$:  
4739 023612 103405  
4740 023614 004737 031356  
4741 023620 103006  
4742 023622 104022  
4743 023624 000404  
4744 023626 004737 031356 5\$:  
4745 023632 103401  
4746 023634 104023  
4747  
4748 023636 6\$:  
4749 023636 006037 023606  
4750 023642 013737 031230 023610  
4751 023650 005203

```

***** TEST 47 *****
*RECEIVER CRC TEST
*USING THE CRC16 POLYNOMIAL, SINGLE CLOCK A BINARY
*COUNT PATTERN, VERIFY THE LSB OF THE RECEIVER BCC ON EACH SHIFT
*****

```

```

: TEST 47
:-----
:*****

```

```

TST47: SCOPE
MOV #47,$TSTNM ; LOAD THE NO. OF THIS TEST
MOV #TST50,NEXT ; POINT TO THE START OF NEXT TEST.
;R1 CONTAINS BASE KMC11 ADDRESS
MSTCLR ; MASTER CLEAR KMC11
MOV #BIT11,(R1) ; SET LINE UNIT LOOP
CLR R3 ; ZERO BIT COUNT
CLR R4 ; R4 CONTAINS CHAR TO BE LOADED IN SILO
CLR R5 ; R5 CONTAINS CHAR CURRENTLY BEING SHIFTED OUT
CLR 4$ ; CLEAR SOFT BCC
MOV #CRC16,XPOLY ; LOAD POLYNOMIAL
JSR PC,SYNLD ; LOAD SILO WITH 2 SYNCs, SOM SET
MOV R4,4(R1) ; PORT4 CHAR
ROMCLK ; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
122110 ; LOAD OUT DATA
INC R4 ; INCREMENT TO NEXT CHARACTER
MOV R4,4(R1) ; PORT4 CHAR
ROMCLK ; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
122110 ; LOAD OUT DATA
INC R4 ; INCREMENT TO NEXT CHARACTER
MOV R4,4(R1) ; PORT4 CHAR
ROMCLK ; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
122110 ; LOAD OUT DATA
JSR PC,OCOR ; WAIT FOR OCOR
DATACLK,32 ; CLOCK DATA
MOV R5,3$ ; LOAD CHAR FOR SOFT CRC
DATACLK,1 ; SHIFT BCC ONCE
JSR R5,SIMBCC ; CALCULATE SOFT BCC
1 ; SOFT SHIFT COUNT
0 ; SOFT CHARACTER
0 ; OLD SOFT BCC
BCS 5$ ; BR IF SOFT BCC LSB IS SET
JSR PC,GETQ1 ; GET HARDWARE RECEIVER BCC LSB
BCC 6$ ; BR IF OK (CLEARED)
ERROR 22 ; ERROR, BCC LSB WAS SET
BR 6$ ; CONTINUE WITH TEST
JSR PC,GETQ1 ; GET HARDWARE RECEIVER BCC LSB
BCS 6$ ; BR IF OK (SET)
ERROR 23 ; ERROR, BCC LSB WAS CLEAR

ROR 3$ ; SHIFT SOFT DATA
MOV CALBCC,4$ ; LOAD OLD SOFT BCC
INC R3 ; INCREMENT BIT COUNTER

```

|      |        |        |        |          |          |   |
|------|--------|--------|--------|----------|----------|---|
| 4752 | 023652 | 022703 | 000010 | CMP      | #10,R3   | :DONE A FULL CHARACTER YET?               |
| 4753 | 023656 | 001346 |        | BNE      | 2\$      | :BR IF NO                                 |
| 4754 | 023660 | 005003 |        | CLR      | R3       | :RESTART BIT COUNTER                      |
| 4755 | 023662 | 005204 |        | INC      | R4       | :INCREMENT DATA FOR SILO                  |
| 4756 | 023664 | 022704 | 000400 | CMP      | #400,R4  | :DONE BINARY COUNT YET?                   |
| 4757 | 023670 | 003404 |        | BLE      | 9\$      | :BR IF YES                                |
| 4758 | 023672 | 010461 | 000004 | MOV      | R4,4(R1) | :PORT4 DATA                               |
| 4759 | 023676 | 104412 |        | ROMCLK   |          | :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 |
| 4760 | 023700 | 122110 |        | 122110   |          | :LOAD OUT DATA                            |
| 4761 | 023702 | 005205 |        | 9\$: INC | R5       | :INCREMENT DATA                           |
| 4762 | 023704 | 022705 | 000400 | CMP      | #400,R5  | :DONE BINARY PATTERN YET?                 |
| 4763 | 023710 | 001327 |        | BNE      | 1\$      | :BR IF NO                                 |
| 4764 | 023712 |        |        | 7\$:     |          |   |

```

:***** TEST 50 *****
:*TRANSMITTER DDCMP CRC TEST
:*THIS TEST TRANSMITS A FOUR CHARACTER MESSAGE WITH CRC
:*BOTH DATA AND THE BCC ARE VERIFIED IN THE BIT
:*WINDOW. THE FOUR CHARACTERS ARE 0,125,252,377
:*THE TRANSMITTER IS CHECKED FOR GOING TO A MARK STATE AFTER THE BCC
:*****

```

: TEST 50

```

:-----
:*****
TST50: SCOPE
MOV #50,$TSTNM ; LOAD THE NO. OF THIS TEST
MOV #TST51,NEXT ; POINT TO THE START OF NEXT TEST.
MSTCLR ;R1 CONTAINS BASE KMC11 ADDRESS
;MASTER CLEAR KMC11

;LOAD OUT DATA SILO

MOV #BIT11,(R1) ;SET LINE UNIT LOOP
MOV #MESDAT,R4 ;LOAD PCINTER TO DATA
CLR 10$ ;CLEAR SOFT BCC
MOV #4,R0 ;LOAD CHARACTER COUNT
JSR PC,SYNLD ;LOAD 2 SYNCS IN OUT SILO
JSR PC,OUTRDY ;WAIT FOR OUTRDY
JSR R5,MESLD ;LOAD SILO WITH 4 CHAR MESS
MESDAT ;ADDRESS OF MESSAGE
4 ;NUMBER OF CHARACTERS
JSR PC,EOM ;LOAD GARBAGE CHARACTER, WITH EOM SET
JSR PC,OCOR ;WAIT FOR OCOR
CLR R3 ;CLEAR BIT COUNTER
DATACLK,22 ;CLOCK DATA
12$: MOVB (R4)+,R5 ;LOAD R5 WITH CHAR
MOV R5,R2 ;LOAD R2 WITH CHAR

;CHECK FIRST FOUR CHARACTER MESSAGE
;IN THE BIT WINDOW (0,125,252,377)

MOV #CRC16,XPOLY ;LOAD POLYNOMIAL
MOV R5,67$ ;LOAD SOFT CHAR FOR BCC
JSR R5,SIMBCC ;CALCULATE SOFT BCC

```

|      |        |        |        |        |       |                 |  |
|------|--------|--------|--------|--------|-------|-----------------|--|
| 4808 | 024032 | 000010 |        |        | 10    |                 | :SHIFT COUNT                                       |
| 4809 | 024034 | 000000 |        |        | 67\$: | 0               | :CHARACTER   |
| 4810 | 024036 | 000000 |        |        | 10\$: | 0               | :OLD BCC   |
| 4811 | 024040 | 013737 | 031230 | 024036 |       | MOV CALBCC,10\$ | :LOAD SOFT BCC FOR NEXT SHIFT                      |
| 4812 | 024046 | 104413 | 000001 |        | 64\$: | DATACLK, 1      | :SHIFT DATA IN TO BIT WINDOW                       |
| 4813 | 024052 | 106002 |        |        |       | RORB R2         | :SHIFT SOFT DATA                                   |
| 4814 | 024054 | 103005 |        |        |       | BCC 65\$        | :BR IF A SPACE                                     |
| 4815 | 024056 | 004737 | 030226 |        |       | JSR PC,GETSI    | :LOOK AT BIT WINDOW                                |
| 4816 | 024062 | 103406 |        |        |       | BCS 66\$        | :BR IF OK (MARK)                                   |
| 4817 | 024064 | 104006 |        |        |       | ERROR 6         | :ERROR, BIT WINDOW WAS A SPACE                     |
| 4818 | 024066 | 000404 |        |        |       | BR 66\$         | :CONTINUE  |
| 4819 | 024070 | 004737 | 030226 |        | 65\$: | JSR PC,GETSI    | :LOOK AT BIT WINDOW                                |
| 4820 | 024074 | 103001 |        |        |       | BCC 66\$        | :BR IF OK (SPACE)                                  |
| 4821 | 024076 | 104006 |        |        |       | ERROR 6         | :ERROR, BIT WINDOW WAS A MARK                      |
| 4822 | 024100 |        |        |        | 66\$: |                 |  |
| 4823 | 024100 | 005203 |        |        |       | INC R3          | :BUMP BIT COUNTER                                  |
| 4824 | 024102 | 022703 | 000010 |        |       | CMP #10,R3      | :DONE FULL 8 BITS YET                              |
| 4825 | 024106 | 001357 |        |        |       | BNE 64\$        | :BR IF NO  |
| 4826 | 024110 | 005003 |        |        |       | CLR R3          | :CLEAR BIT COUNTER                                 |
| 4827 | 024112 | 005300 |        |        |       | DEC R0          | :DEC CHARACTER COUNT                               |
| 4828 | 024114 | 001335 |        |        |       | BNE 12\$        | :BR IF NOT DONE YET                                |
| 4829 |        |        |        |        |       |                 |  |
| 4830 |        |        |        |        |       |                 | :CHECK BCC FOR PRECEDING MESSAGE IN THE BIT WINDOW |
| 4831 |        |        |        |        |       |                 |  |
| 4832 | 024116 | 013700 | 031230 |        |       | MOV CALBCC,R0   | :PUT BCC IN R0                                     |
| 4833 | 024122 | 104413 | 000001 |        | 68\$: | DATACLK, 1      | :SHIFT HARDWARE BCC                                |
| 4834 | 024126 | 006000 |        |        |       | ROR R0          | :SHIFT SOFT BCC                                    |
| 4835 | 024130 | 103005 |        |        |       | BCC 69\$        | :BR IF CARRY CLEAR                                 |
| 4836 | 024132 | 004737 | 030226 |        |       | JSR PC,GETSI    | :LOOK AT BIT WINDOW                                |
| 4837 | 024136 | 103406 |        |        |       | BCS 70\$        | :BR IF OK (MARK)                                   |
| 4838 | 024140 | 104014 |        |        |       | ERROR 14        | :ERROR, CRC WRONG (SPACE)                          |
| 4839 | 024142 | 000404 |        |        |       | BR 70\$         | :CONTINUE  |
| 4840 | 024144 | 004737 | 030226 |        | 69\$: | JSR PC,GETSI    | :LOOK AT BIT WINDOW                                |
| 4841 | 024150 | 103001 |        |        |       | BCC 70\$        | :BR IF OK (SPACE)                                  |
| 4842 | 024152 | 104014 |        |        |       | ERROR 14        | :ERROR, CRC WRONG (MARK)                           |
| 4843 | 024154 |        |        |        | 70\$: |                 |  |
| 4844 | 024154 | 005203 |        |        |       | INC R3          | :BUMP BIT COUNTER                                  |
| 4845 | 024156 | 022703 | 000020 |        |       | CMP #20,R3      | :FINISHED BCC YET?                                 |
| 4846 | 024162 | 001357 |        |        |       | BNE 68\$        | :BR IF NO  |
| 4847 | 024164 | 005003 |        |        |       | CLR R3          | :CLEAR BIT COUNTER                                 |
| 4848 |        |        |        |        |       |                 |  |
| 4849 |        |        |        |        |       |                 | :CHECK TO SEE IF TRANSMITTER IS MARKING            |
| 4850 |        |        |        |        |       |                 |  |
| 4851 | 024166 | 104413 | 000001 |        | 2\$:  | DATACLK, 1      | :CLOCK TRANSMITTER                                 |
| 4852 | 024172 | 004737 | 030226 |        |       | JSR PC,GETSI    | :LOOK AT WINDOW                                    |
| 4853 | 024176 | 103401 |        |        |       | BCS 3\$         | :IT SHOULD BE MARKING                              |
| 4854 | 024200 | 104024 |        |        |       | ERROR 24        | :ERROR, BIT WAS A SPACE                            |
| 4855 | 024202 | 005203 |        |        | 3\$:  | INC R3          | :BUMP BIT COUNTER                                  |
| 4856 | 024204 | 022703 | 000007 |        |       | CMP #7,R3       | :DONE YET  |
| 4857 | 024210 | 001366 |        |        |       | BNE 2\$         | :BR IF NO  |
| 4858 | 024212 | 104413 | 000010 |        |       | DATACLK, 10     | :GIVE ENOUGH TICKS TO CLEAR OUT ACTIVE             |
| 4859 | 024216 | 005003 |        |        |       | CLR R3          | :CLEAR BIT COUNTER                                 |
| 4860 | 024220 | 104413 | 000001 |        | 4\$:  | DATACLK, 1      | :SHIFT OUT NEXT BIT                                |
| 4861 | 024224 | 004737 | 030226 |        |       | JSR PC,GETSI    | :LOOK AT BIT WINDOW                                |
| 4862 | 024230 | 103401 |        |        |       | BCS +4          | :BR IF IT IS A MARK                                |
| 4863 | 024232 | 104024 |        |        |       | ERROR 24        | :ERROR, TRANSMITTER IS NOT MARKING                 |

4864 024234 005203  
4865 024236 022703 000020  
4866 024242 001366  
4867 024244

INC R3 ;INC BIT COUNT  
CMP #20,R3 ;DONE YET?  
BNE 4\$ ;BR IF NO

5\$:

4868  
4869  
4870  
4871  
4872  
4873  
4874  
4875  
4876  
4877  
4878  
4879

\*\*\*\*\* TEST 51 \*\*\*\*\*  
:RECEIVER DDCMP CRC TEST  
:THIS TEST CLOCKS A FOUR CHARACTER MESSAGE WITH BCC  
:AND VERIFYS CORRECT DATA RECEPTION AND BCC MATCH  
:THE FOUR CHARACTER MESSAGE IS 0,125,252,377  
:\*\*\*\*\*

: TEST 51  
:-----

4880 024244 000004  
4881 024246 012737 000051 001202  
4882 024254 012737 024446 001442  
4883  
4884 024262 104410  
4885 024264 012711 004000  
4886 024270 012702 032012  
4887 024274 012700 000004  
4888 024300 004737 031374  
4889 024304 004737 030412  
4890 024310 004537 031530  
4891 024314 032012  
4892 024316 000004  
4893 024320 004737 031504  
4894 024324 004737 030260  
4895 024330 104413 000114  
4896 024334 004737 031066  
4897 024340 104412  
4898 024342 021204  
4899 024344 016104 000004  
4900 024350 112205  
4901 024352 120504  
4902 024354 001401  
4903 024356 104010  
4904 024360 005300  
4905 024362 001364

TST51: SCOPE ; LOAD THE NO. OF THIS TEST  
MOV #51,\$TSTNM ; POINT TO THE START OF NEXT TEST.  
MOV #TST52,NEXT ; R1 CONTAINS BASE KMC11 ADDRESS  
; MASTER CLEAR KMC11  
MSTCLR ; SET LINE UNIT LOOP  
MOV #BIT11,(R1) ; LOAD POINTER TO DATA  
MOV #MESDAT,R2 ; LOAD CHARACTER COUNT  
MOV #4,R0 ; LOAD 2 SYNCs IN OUT SILO  
JSR PC,SYNLD ; WAIT FOR OUTRDY  
JSR PC,OUTRDY ; LOAD SILO WITH 4 CHAR MESS  
JSR R5,MESLD ; ADDRESS OF MESSAGE  
MESDAT ; NUMBER OF CHARACTERS  
4 ; LOAD GARBAGE CHARACTER, WITH EOM SET  
JSR PC,EOM ; WAIT FOR OCOR  
JSR PC,OCOR ; CLOCK DATA  
DATACLK,114 ; WAIT FOR INRDY  
3\$: JSR PC,INRDY ; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304  
ROMCLK ; GET IN DATA  
021204 ; PUT "FOUND" IN R4  
MOV 4(R1),R4 ; PUT "EXPECTED" IN R5  
MOVB (R2)+,R5 ; COMPARE RECEIVED DATA  
CMPB R5,R4 ; BR IF OK  
1\$: BEQ 1\$ ; DATA ERROR  
ERROR 10 ; DEC CHARACTER COUNT  
DEC R0 ; BR IF NOT DONE YET  
BNE 3\$

;CHECK TO SEE THAT IN BCC MATCH IS SET

4906  
4907  
4908  
4909 024364 004737 031066  
4910 024370 104412  
4911 024372 021204  
4912 024374 116137 000004 001302  
4913 024402 042737 177400 001302  
4914 024410 004737 031066  
4915 024414 104412  
4916 024416 021244  
4917 024420 016104 000004  
4918 024424 042704 000376  
4919 024430 012705 000001

JSR PC,INRDY ; WAIT FOR INRDY  
ROMCLK ; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304  
021204 ; GET FIRST HALF OF CRC  
MOVB 4(R1),\$TMP2 ; PUT IN \$TMP2  
BIC #177400,\$TMP2 ; CLEAR HI BYTE  
JSR PC,INRDY ; WAIT FOR INRDY  
ROMCLK ; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304  
021244  
MOV 4(R1),R4 ; PUT "FOUND" IN R4  
BIC #376,R4 ; CLEAR UNWANTED BITS  
MOV #1,R5 ; PUT "EXPECTED" IN R5

|      |        |        |      |        |       |   |
|------|--------|--------|------|--------|-------|---|
| 4920 | 024434 | 120504 |      | CMPB   | R5,R4 | ;IS IN BCC MATCH SET?                     |
| 4921 | 024436 | 001401 |      | BEQ    | 25%   |   |
| 4922 | 024440 | 104015 |      | ERROR  | 15    | ;IN BCC MATCH ERROR                       |
| 4923 | 024442 |        | 25%: |        |       |   |
| 4924 | 024442 | 104412 |      | ROMCLK |       | ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 |
| 4925 | 024444 | 021204 |      | 021204 |       | ;GET LAST HALF                            |
| 4926 | 024446 |        | 2%:  |        |       |   |

```

***** TEST 52 *****
:DDCMP EOM FUNCTION TEST
:THIS TEST LOADS OUT SILO WITH: 2 SYNCs,4 CHAR MESSAGE,EOM
:*4 CHARACTER MESS,EOM. THE DATA STREAM IS CHECKED TO BE
:*4 CHAR,BCC,4 CHAR,BCC,MARKS. THIS TEST VERIFYS THAT
:*THE CHARCTERS LOADED WITH EOM SET ARE LOST
:*ALL DATA AND BCC'S ARE CHECKED IN THE BIT WINDOW
:*THE FOUR CHARACTER MESSAGE IS 0,125,252,377
:*RECEIVED DATA IS VERIFIED, AND IN BCC MATCH IS CHECKED
:*****

```

: TEST 52

|      |        |        |        |          |              |                                       |
|------|--------|--------|--------|----------|--------------|---------------------------------------|
| 4943 | 024446 | 000004 |        | TST52:   | SCOPE        |                                       |
| 4944 | 024450 | 012737 | 000052 | MOV      | #52,\$TSTNM  | ; LOAD THE NO. OF THIS TEST           |
| 4945 | 024456 | 012737 | 025546 | MOV      | #TST53,NEXT  | ; POINT TO THE START OF NEXT TEST.    |
| 4946 |        |        |        |          |              | ;R1 CONTAINS BASE KMC11 ADDRESS       |
| 4947 | 024464 | 104410 |        | MSTCLR   |              | ;MASTER CLEAR KMC11                   |
| 4948 |        |        |        |          |              |                                       |
| 4949 |        |        |        |          |              |                                       |
| 4950 |        |        |        |          |              |                                       |
| 4951 | 024466 | 012711 | 004000 | MOV      | #BIT11,(R1)  | ;SET LINE UNIT LOOP                   |
| 4952 | 024472 | 012704 | 032012 | MOV      | #MESDAT,R4   | ;LOAD POINTER TO DATA                 |
| 4953 | 024476 | 005037 | 024606 | CLR      | 10%          | ;CLEAR SOFT BCC                       |
| 4954 | 024502 | 012700 | 000004 | MOV      | #4,R0        | ;LOAD CHARACTER COUNT                 |
| 4955 | 024506 | 004737 | 031374 | JSR      | PC,SYNLD     | ;LOAD 2 SYNCs IN OUT SILO             |
| 4956 | 024512 | 004737 | 030412 | JSR      | PC,OUTRDY    | ;WAIT FOR OUTRDY                      |
| 4957 | 024516 | 004537 | 031530 | JSR      | R5,MESLD     | ;LOAD SILO WITH 4 CHAR MESS           |
| 4958 | 024522 | 032012 |        | MESDAT   |              | ;ADDRESS OF MESSAGE                   |
| 4959 | 024524 | 000004 |        | 4        |              | ;NUMBER OF CHARACTERS                 |
| 4960 | 024526 | 004737 | 031504 | JSR      | PC,EOM       | ;LOAD GARBAGE CHARACTER, WITH EOM SET |
| 4961 | 024532 | 004537 | 031530 | JSR      | R5,MESLD     | ;LOAD FOUR MORE CHARACTERS            |
| 4962 | 024536 | 032012 |        | MESDAT   |              | ;ADDRESS OF MESSAGE                   |
| 4963 | 024540 | 000004 |        | 4        |              | ;NUMBER OF CHACTERS                   |
| 4964 | 024542 | 004737 | 031504 | JSR      | PC,EOM       | ;SET EOM                              |
| 4965 | 024546 | 004737 | 030260 | JSR      | PC,OCOR      | ;WAIT FOR OCOR                        |
| 4966 | 024552 | 005003 |        | CLR      | R3           | ;CLEAR BIT COUNTER                    |
| 4967 | 024554 | 104413 | 000022 | DATACLK, | 22           | ;CLOCK DATA                           |
| 4968 | 024560 | 112405 | 12%:   | MOVB     | (R4)+,R5     | ;LOAD R5 WITH CHAR                    |
| 4969 | 024562 | 010502 |        | MOV      | R5,R2        | ;LOAD R2 WITH CHAR                    |
| 4970 |        |        |        |          |              |                                       |
| 4971 |        |        |        |          |              |                                       |
| 4972 |        |        |        |          |              |                                       |
| 4973 |        |        |        |          |              |                                       |
| 4974 | 024564 | 012737 | 120001 | MOV      | #CRC16,XPOLY | ;LOAD POLYNOMIAL                      |
| 4975 | 024572 | 010537 | 024604 | MOV      | R5,67%       | ;LOAD SOFT CHAR FOR BCC               |

```

;CHECK FIRST FOUR CHARACTER MESSAGE
;IN THE BIT WINDOW (0,125,252,377)

```



```

4976 024576 004537 031122      JSP      R5,SIMBCC      ;CALCULATE SOFT BCC
4977 024602 000010              10      ;SHIFT COUNT
4978 024604 000000      67$: 0      ;CHARACTER
4979 024606 000000      10$: 0      ;OLD BCC
4980 024610 013737 031230 024606  MOV      CALBCC,10$    ;LOAD SOFT BCC FOR NEXT SHIFT
4981 024616 104413 000001      64$: DATACLK, 1      ;SHIFT DATA IN TO BIT WINDOW
4982 024622 106002      RORB     R2      ;SHIFT SOFT DATA
4983 024624 103005      BCC      65$     ;BR IF A SPACE
4984 024626 004737 030226      JSR      PC,GETSI    ;LOOK AT BIT WINDOW
4985 024632 103406      BCS      66$     ;BR IF OK (MARK)
4986 024634 104006      ERROR   6      ;ERROR, BIT WINDOW WAS A SPACE
4987 024636 000404      BR       66$     ;CONTINUE
4988 024640 004737 030226      65$: JSR      PC,GETSI    ;LOOK AT BIT WINDOW
4989 024644 103001      BCC      66$     ;BR IF OK (SPACE)
4990 024646 104006      ERROR   6      ;ERROR, BIT WINDOW WAS A MARK
4991 024650      66$:
4992 024650 005203      INC      R3      ;BUMP BIT COUNTER
4993 024652 022703 000010      CMP      #10,R3    ;DONE FULL 8 BITS YET
4994 024656 001357      BNE      64$     ;BR IF NO
4995 024660 005003      CLR      R3      ;CLEAR BIT COUNTER
4996 024662 005300      DEC      R0      ;DEC CHARACTER COUNT
4997 024664 001335      BNE      12$     ;BR IF NOT DONE YET
4998
4999
5000
5001 024666 013700 031230      MOV      CALBCC,R0    ;PUT BCC IN R0
5002 024672 104413 000001      68$: DATACLK,1      ;SHIFT HARDWARE BCC
5003 024676 006000      ROR      R0      ;SHIFT SOFT BCC
5004 024700 103005      BCC      69$     ;BR IF CARRY CLEAR
5005 024702 004737 030226      JSR      PC,GETSI    ;LOOK AT BIT WINDOW
5006 024706 103406      BCS      70$     ;BR IF OK (MARK)
5007 024710 104014      ERROR   14      ;ERROR, CRC WRONG (SPACE)
5008 024712 000404      BR       70$     ;CONTINUE
5009 024714 004737 030226      69$: JSR      PC,GETSI    ;LOOK AT BIT WINDOW
5010 024720 103001      BCC      70$     ;BR IF OK (SPACE)
5011 024722 104014      ERROR   14      ;ERROR, CRC WRONG (MARK)
5012 024724      70$:
5013 024724 005203      INC      R3      ;BUMP BIT COUNTER
5014 024726 022703 000020      CMP      #20,R3    ;FINISHED BCC YET?
5015 024732 001357      BNE      68$     ;BR IF NO
5016 024734 005003      CLR      R3      ;CLEAR BIT COUNTER
5017 024736 012700 000004      MOV      #4,R0     ;RESET CHARACTER COUNTER
5018 024742 012704 032012      MOV      #MESDAT,R4 ;LOAD MESSAGE POINTER
5019 024746 005037 025000      CLR      11$     ;CLR SOFT BCC
5020 024752 112405      13$: MOV     (R4)+,R5    ;LOAD CHAR IN R5
5021 024754 010502      MOV      R5,R2    ;LOAD CHAR IN R2
5022
5023
5024
5025 024756 012737 120001 031226      MOV      #CRC16,XPOLY ;LOAD POLYNOMIAL
5026 024764 010537 024776      MOV      R5,76$    ;LOAD SOFT CHAR FOR BCC
5027 024770 004537 031122      JSR      R5,SIMBCC  ;CALCULATE SOFT BCC
5028 024774 000010              10      ;SHIFT COUNT
5029 024776 000000      76$: 0      ;CHARACTER
5030 025000 000000      11$: 0      ;OLD BCC
5031 025002 013737 031230 025000      MOV      CALBCC,11$ ;LOAD SOFT BCC FOR NEXT SHIFT

```

```

5032 025010 104413 000001      73$:  DATACLK,      1      ;SHIFT DATA IN TO BIT WINDOW
5033 025014 106002              RORB      R2      ;SHIFT SOFT DATA
5034 025016 103005              BCC       74$     ;BR IF A SPACE
5035 025020 004737 030226      JSR       PC,GETSI ;LOOK AT BIT WINDOW
5036 025024 103406              RCS       75$     ;BR IF OK (MARK)
5037 025026 104006              ERROR    6        ;ERROR, BIT WINDOW WAS A SPACE
5038 025030 000404              BR       75$     ;CONTINUE
5039 025032 004737 030226      74$:  JSR       PC,GETSI ;LOOK AT BIT WINDOW
5040 025036 103001              BCC      75$     ;BR IF OK (SPACE)
5041 025040 104006              ERROR    6        ;ERROR, BIT WINDOW WAS A MARK
5042 025042                    75$:
5043 025042 005203              INC      R3      ;BUMP BIT COUNTER
5044 025044 022703 000010      CMP      #10,R3  ;DONE FULL 8 BITS YET
5045 025050 001357              BNE      73$     ;BR IF NO
5046 025052 005003              CLR      R3      ;CLEAR BIT COUNTER
5047 025054 005300              DEC      R0      ;DEC CHARACTER COUNT
5048 025056 001335              BNE      13$     ;BR IF NOT DONE YET
5049
5050                      ;CHECK BCC FOR PRECEDING MESSAGE IN THE BIT WINDOW
5051
5052 025060 013700 031230      77$:  MOV      CALBCC,R0 ;PUT BCC IN R0
5053 025064 104413 000001      DATACLK,1      ;SHIFT HARDWARE BCC
5054 025070 006000              ROR      R0      ;SHIFT SOFT BCC
5055 025072 103005              BCC      78$     ;BR IF CARRY CLEAR
5056 025074 004737 030226      JSR      PC,GETSI ;LOOK AT BIT WINDOW
5057 025100 103406              BCS      79$     ;BR IF OK (MARK)
5058 025102 104014              ERROR    14      ;ERROR, CRC WRONG (SPACE)
5059 025104 000404              BR       79$     ;CONTINUE
5060 025106 004737 030226      78$:  JSR      PC,GETSI ;LOOK AT BIT WINDOW
5061 025112 103001              BCC      79$     ;BR IF OK (SPACE)
5062 025114 104014              ERROR    14      ;ERROR, CRC WRONG (MARK)
5063 025116                    79$:
5064 025116 005203              INC      R3      ;BUMP BIT COUNTER
5065 025120 022703 000020      CMP      #20,R3  ;FINISHED BCC YET?
5066 025124 001357              BNE      77$     ;BR IF NO
5067 025126 005003              CLR      R3      ;CLEAR BIT COUNTER
5068
5069                      ;CHECK TO SEE IF TRANSMITTER IS MARKING
5070
5071 025130 104413 000001      2$:   DATACLK,      1      ;CLOCK TRANSMITTER
5072 025134 004737 030226      JSR      PC,GETSI ;LOOK AT WINDOW
5073 025140 103401              BCS      3$     ;IT SHOULD BE MARKING
5074 025142 104024              ERROR    24      ;ERROR, BIT WAS A SPACE
5075 025144 005203              3$:   INC      R3      ;BUMP BIT COUNTER
5076 025146 022703 000007      CMP      #7,R3   ;DONE YET
5077 025152 001366              BNE      2$     ;BR IF NO
5078 025154 104413 000010      DATACLK,      10     ;GIVE ENOUGH TICKS TO CLEAR OUT ACTIVE
5079 025160 005003              CLR      R3      ;CLEAR BIT COUNTER
5080 025162 104413 000001      4$:   DATACLK,      1      ;SHIFT OUT NEXT BIT
5081 025166 004737 030226      JSR      PC,GETSI ;LOOK AT BIT WINDOW
5082 025172 103401              BCS      +4      ;BR IF IT IS A MARK
5083 025174 104024              ERROR    24      ;ERROR, TRANSMITTER IS NOT MARKING
5084 025176 005203              INC      R3      ;INC BIT COUNT
5085 025200 022703 000020      CMP      #20,R3  ;DONE YET?
5086 025204 001366              BNE      4$     ;BR IF NO
5087

```



```
5144 025422 104010          ERROR 10          ;RECEIVE DATA ERROR
5145 025424 005303          44$: DEC R3          ;DEC CHARACTER COUNT
5146 025426 001364          BNE 43$          ;BR IF NOT DONE YET
5147
5148                          ;CHECK TO SEE THAT IN BCC MATCH IS SET
5149                          ;AND THAT THE BCC WAS RECEIVED CORRECTLY
5150
5151 025430 004737 031066     JSR PC,INRDY      ;WAIT FOR INRDY
5152 025434 104412          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5153 025436 021204          021204         ;GET FIRST HALF OF CRC
5154 025440 116137 000004 001302  MOVB 4(R1),$TMP2 ;PUT IN $TMP2
5155 025446 042737 177400 001302  BIC #177400,$TMP2 ;CLEAR HI BYTE
5156 025454 004737 031066     JSR PC,INRDY      ;WAIT FOR INRDY
5157 025460 104412          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5158 025462 021244          021244
5159 025464 016104 000004     MOV 4(R1),R4      ;PUT 'FOUND' IN R4
5160 025470 042704 000376     BIC #376,R4      ;CLEAR UNWANTED BITS
5161 025474 012705 000001     MOV #1,R5        ;PUT 'EXPECTED' IN R5
5162 025500 120504          CMPB R5,R4       ;IS IN BCC MATCH SET?
5163 025502 001401          BEQ 51$         ;IN BCC MATCH ERROR
5164 025504 104015          ERROR 15
5165 025506          51$: ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5166 025506 104412          021204         ;GET LAST HALF
5167 025510 021204          MOVB 4(R1),$TMP1+1 ;PUT IN $TMP1
5168 025512 116137 000004 001301  BIC #377,$TMP1   ;CLEAR LO BYTE
5169 025520 042737 000377 001300  BIS $TMP1,$TMP2  ;16 BIT BCC NOW IN $TMP2
5170 025526 053737 001300 001302  CMP CALBCC,$TMP2 ;IS IT CORRECT?
5171 025534 023737 031230 001302  BEQ 5$           ;BR IF OK
5172 025542 001401          ERROR 27
5173 025544 104027          5$:
5174 025546
```

```
5175
5176
5177 ***** TEST 53 *****
5178 :*DDCMP EOM FUNCTION TEST
5179 :*THIS TEST LOADS OUT SILO WITH: 2 SYNC'S, 4 CHAR MESSAGE, EOM
5180 :*SOM, 4 CHAR MESS, EOM. THE DATA STREAM IS CHECKED TO BE
5181 :*4 CHAR, BCC, 4 CHAR, BCC, MARKS. THIS TEST VERIFYS THAT
5182 :*THE CHARCTERS LOADED WITH EOM SET ARE LOST
5183 :*ALSO THAT THE CHAR LOADED WITH SOM IS NOT IN THE BCC
5184 :*ALL DATA AND BCC'S ARE CHECKED IN THE BIT WINDOW
5185 :*THE FOUR CHARACTER MESSAGE IS 0,125,252,377
5186 :*RECEIVED DATA IS VERIFIED, AND IN BCC MATCH IS CHECKED
5187 :*****
5188
```

```
5189
5190 TEST 53
5191 -----
5192 025546 000004          TST53: SCOPE
5193 025550 012737 000053 001202     MOV #53,$STSNH   ; LOAD THE NO. OF THIS TEST
5194 025556 012737 026746 001442     MOV #TST54,NEXT ; POINT TO THE START OF NEXT TEST.
5195
5196 025564 104410          MSTCLR          ;R1 CONTAINS BASE KMC11 ADDRESS
5197
5198                          ;MASTER CLEAR KMC11
5199                          ;LOAD OUT DATA SILO
```

CZKCF MACY11 30A(1052) 08-JUL-80 08:27 PAGE 102  
 CZKCF.P11 08-JUL-80 08:24 BASIC RECEIVER TESTS

SEQ 0101

|      |        |        |        |        |          |              |  |
|------|--------|--------|--------|--------|----------|--------------|--|
| 5200 | 025566 | 012711 | 004000 |        | MOV      | #BIT11,(R1)  | :SET LINE UNIT LOOP                                |
| 5201 | 025572 | 012704 | 032012 |        | MOV      | #MESDAT,R4   | :LOAD POINTER TO DATA                              |
| 5202 | 025576 | 005037 | 025712 |        | CLR      | 10\$         | :CLEAR SOFT BCC                                    |
| 5203 | 025602 | 012700 | 000004 |        | MOV      | #4,R0        | :LOAD CHARACTER COUNT                              |
| 5204 | 025606 | 004737 | 031374 |        | JSR      | PC,SYNLD     | :LOAD 2 SYNC'S IN OUT SILO                         |
| 5205 | 025612 | 004737 | 030412 |        | JSR      | PC,OUTRDY    | :WAIT FOR OUTRDY                                   |
| 5206 | 025616 | 004537 | 031530 |        | JSR      | R5,MESLD     | :LOAD SILO WITH 4 CHAR MESS                        |
| 5207 | 025622 | 032012 |        |        | MESDAT   |              | :ADDRESS OF MESSAGE                                |
| 5208 | 025624 | 000004 |        |        | 4        |              | :NUMBER OF CHARACTERS                              |
| 5209 | 025626 | 004737 | 031504 |        | JSR      | PC,EOM       | :LOAD GARBAGE CHARACTER, WITH EOM SET              |
| 5210 | 025632 | 004737 | 031454 |        | JSR      | PC,SOM       | :LOAD GARBAGE CHAR WITH SOM SET                    |
| 5211 | 025636 | 004537 | 031530 |        | JSR      | R5,MESLD     | :LOAD FOUR MORE CHARACTERS                         |
| 5212 | 025642 | 032012 |        |        | MESDAT   |              | :ADDRESS OF MESSAGE                                |
| 5213 | 025644 | 000004 |        |        | 4        |              | :NUMBER OF CHACTERS                                |
| 5214 | 025646 | 004737 | 031504 |        | JSR      | PC,EOM       | :SET EOM   |
| 5215 | 025652 | 004737 | 030260 |        | JSR      | PC,OCOR      | :WAIT FOR OCOR                                     |
| 5216 | 025656 | 005003 |        |        | CLR      | R3           | :CLEAR BIT COUNTER                                 |
| 5217 | 025660 | 104413 | 000022 |        | DATACLK, | 22           | :CLOCK DATA  |
| 5218 | 025664 | 112405 |        | 12\$:  | MOVB     | (R4)+,R5     | :LOAD R5 WITH CHAR                                 |
| 5219 | 025666 | 010502 |        |        | MOV      | R5,R2        | :LOAD R2 WITH CHAR                                 |
| 5220 |        |        |        |        |          |              |  |
| 5221 |        |        |        |        |          |              |  |
| 5222 |        |        |        |        |          |              | :CHECK FIRST FOUR CHARACTER MESSAGE                |
| 5223 |        |        |        |        |          |              | :IN THE BIT WINDOW (0,125,252,377)                 |
| 5224 | 025670 | 012737 | 120001 | 031226 | MOV      | #CRC16,XPOLY | :LOAD POLYNOMIAL                                   |
| 5225 | 025676 | 010537 | 025710 |        | MOV      | R5,67\$      | :LOAD SOFT CHAR FOR BCC                            |
| 5226 | 025702 | 004537 | 031122 |        | JSR      | R5,SIMBCC    | :CALCULATE SOFT BCC                                |
| 5227 | 025706 | 000010 |        |        | 10       |              | :SHIFT COUNT                                       |
| 5228 | 025710 | 000000 |        | 67\$:  | 0        |              | :CHARACTER   |
| 5229 | 025712 | 000000 |        | 10\$:  | 0        |              | :OLD BCC   |
| 5230 | 025714 | 013737 | 031230 | 025712 | MOV      | CALBCC,10\$  | :LOAD SOFT BCC FOR NEXT SHIFT                      |
| 5231 | 025722 | 104413 | 000001 |        | 64\$:    | DATACLK, 1   | :SHIFT DATA IN TO BIT WINDOW                       |
| 5232 | 025726 | 106002 |        |        | RORB     | R2           | :SHIFT SOFT DATA                                   |
| 5233 | 025730 | 103005 |        |        | BCC      | 65\$         | :BR IF A SPACE                                     |
| 5234 | 025732 | 004737 | 030226 |        | JSR      | PC,GETSI     | :LOOK AT BIT WINDOW                                |
| 5235 | 025736 | 103406 |        |        | BCS      | 66\$         | :BR IF OK (MARK)                                   |
| 5236 | 025740 | 104006 |        |        | ERROR    | 6            | :ERROR, BIT WINDOW WAS A SPACE                     |
| 5237 | 025742 | 000404 |        |        | BR       | 66\$         | :CONTINUE  |
| 5238 | 025744 | 004737 | 030226 |        | 65\$:    | JSR          | PC,GETSI   |
| 5239 | 025750 | 103001 |        |        | BCC      | 66\$         | :BR IF OK (SPACE)                                  |
| 5240 | 025752 | 104006 |        |        | ERROR    | 6            | :ERROR, BIT WINDOW WAS A MARK                      |
| 5241 | 025754 |        |        | 66\$:  |          |              |  |
| 5242 | 025754 | 005203 |        |        | INC      | R3           | :BUMP BIT COUNTER                                  |
| 5243 | 025756 | 022703 | 000010 |        | CMP      | #10,R3       | :DONE FULL 8 BITS YET                              |
| 5244 | 025762 | 001357 |        |        | BNE      | 64\$         | :BR IF NO  |
| 5245 | 025764 | 005003 |        |        | CLR      | R3           | :CLEAR BIT COUNTER                                 |
| 5246 | 025766 | 005300 |        |        | DEC      | R0           | :DEC CHARACTER COUNT                               |
| 5247 | 025770 | 001335 |        |        | BNE      | 12\$         | :BR IF NOT DONE YET                                |
| 5248 |        |        |        |        |          |              |  |
| 5249 |        |        |        |        |          |              | :CHECK BCC FOR PRECEDING MESSAGE IN THE BIT WINDOW |
| 5250 |        |        |        |        |          |              |  |
| 5251 | 025772 | 013700 | 031230 |        | MOV      | CALBCC,R0    | :PUT BCC IN R0                                     |
| 5252 | 025776 | 104413 | 000001 |        | 68\$:    | DATACLK,1    | :SHIFT HARDWARE BCC                                |
| 5253 | 026002 | 006000 |        |        | ROR      | R0           | :SHIFT SOFT BCC                                    |
| 5254 | 026004 | 103005 |        |        | BCC      | 69\$         | :BR IF CARRY CLEAR                                 |
| 5255 | 026006 | 004737 | 030226 |        | JSR      | PC,GETSI     | :LOOK AT BIT WINDOW                                |

```

5256 026012 103406          BCS 70$          ;BR IF OK (MARK)
5257 026014 104014          ERROR 14         ;ERROR, CRC WRONG (SPACE)
5258 026016 000404          BR 70$          ;CONTINUE
5259 026020 004737 030226 69$: JSR PC,GETSI   ;LOOK AT BIT WINDOW
5260 026024 103001          BCC 70$         ;BR IF OK (SPACE)
5261 026026 104014          ERROR 14         ;ERROR, CRC WRONG (MARK)
5262 026030          70$:
5263 026030 005203          INC R3          ;BUMP BIT COUNTER
5264 026032 022703 000020  CMP #20,R3     ;FINISHED BCC YET?
5265 026036 001357          BNE 68$         ;BR IF NO
5266 026040 005003          CLR R3          ;CLEAR BIT COUNTER
5267
5268          ;CHECK CHARACTER LOADED WITH SOM (000), IN THE BIT WINDOW
5269
5270 026042 005005          CLR R5          ;CHARACTER LOADED WITH SOM
5271 026044 010502          MOV R5,R2      ;LOAD R2 WITH CHAR
5272 026046 104413 000001 32$: DATACLK, 1   ;CLOCK TRANSMITTER
5273 026052 106002          RORB R2        ;SHIFT SOFT DATA
5274 026054 103005          BCC 30$        ;BR IF SPACE
5275 026056 004737 030226  JSR PC,GETSI   ;LOOK AT BIT WINDOW
5276 026062 103406          BCS 31$        ;BR IF OK (MARK)
5277 026064 104006          ERROR 6        ;ERROR,BIT WINDOW WAS A SPACE
5278 026066 000404          BR 31$        ;CONTINUE
5279 026070 004737 030226 30$: JSR PC,GETSI   ;LOOK AT BIT WINDOW
5280 026074 103001          BCC 31$        ;BR IF OK (SPACE)
5281 026076 104006          ERROR 6        ;ERROR,BIT WINDOW WAS A MARK
5282 026100 005203          31$: INC R3      ;BUMP BIT COUNTER
5283 026102 022703 000010  CMP #10,R3     ;DONE CHARACTER YET?
5284 026106 001357          BNE 32$        ;BR IF NO
5285 026110 005003          CLR R3        ;RESET BIT COUNTER
5286 026112 012700 000004  MOV #4,R0      ;RESET CHARACTER COUNTER
5287 026116 012704 032012  MOV #MESDAT,R4 ;LOAD MESSAGE POINTER
5288 026122 005037 026154  CLR 11$       ;CLR SOFT BCC
5289 026126 112405          13$: MOVB (R4)+,R5  ;LOAD CHAR IN R5
5290 026130 010502          MOV R5,R2     ;LOAD CHAR IN R2
5291
5292          ;CHECK SECOND MESSAGE IN THE BIT WINDOW (0,125,252,377)
5293
5294 026132 012737 120001 031226 MOV #CRC16,XPOLY ;LOAD POLYNOMIAL
5295 026140 010537 026152  MOV R5,76$     ;LOAD SOFT CHAR FOR BCC
5296 026144 004537 031122  JSR R5,SIMBCC  ;CALCULATE SOFT BCC
5297 026150 000010          10          ;SHIFT COUNT
5298 026152 000000          76$: 0      ;CHARACTER
5299 026154 000000          11$: 0      ;OLD BCC
5300 026156 013737 031230 026154 MOV CALBCC,11$ ;LOAD SOFT BCC FOR NEXT SHIFT
5301 026164 104413 000001 73$: DATACLK, 1   ;SHIFT DATA IN TO BIT WINDOW
5302 026170 106002          RORB R2      ;SHIFT SOFT DATA
5303 026172 103005          BCC 74$      ;BR IF A SPACE
5304 026174 004737 030226  JSR PC,GETSI   ;LOOK AT BIT WINDOW
5305 026200 103406          BCS 75$      ;BR IF OK (MARK)
5306 026202 104006          ERROR 6     ;ERROR, BIT WINDOW WAS A SPACE
5307 026204 000404          BR 75$      ;CONTINUE
5308 026206 004737 030226 74$: JSR PC,GETSI   ;LOOK AT BIT WINDOW
5309 026212 103001          BCC 75$     ;BR IF OK (SPACE)
5310 026214 104006          ERROR 6     ;ERROR, BIT WINDOW WAS A MARK
5311 026216          75$:

```

CZKCF MACY11 30A(1052) 08-JUL-80 08:27 PAGE 104  
 CZKCF.P11 08-JUL-80 08:24 BASIC RECEIVER TESTS

SEQ 0103

```

5312 026216 005203          INC      R3          ;BUMP BIT COUNTER
5313 026220 022703 000010  CMP      #10,R3     ;DONE FULL 8 BITS YET
5314 026224 001357          BNE      73$        ;BR IF NO
5315 026226 005003          CLR      R3         ;CLEAR BIT COUNTER
5316 026230 005300          DEC      R0         ;DEC CHARACTER COUNT
5317 026232 001335          BNE      13$        ;BR IF NOT DONE YET
5318
5319                          ;CHECK BCC FOR PRECEDING MESSAGE IN THE BIT WINDOW
5320
5321 026234 013700 031230  MOV      CALBCC,R0   ;PUT BCC IN R0
5322 026240 104413 000001 77$:  DATACL,1          ;SHIFT HARDWARE BCC
5323 026244 006000          ROR      R0         ;SHIFT SOFT BCC
5324 026246 103005          BCC      78$        ;BR IF CARRY CLEAR
5325 026250 004737 030226  JSR      PC,GETSI    ;LOOK AT BIT WINDOW
5326 026254 103406          BCS      79$        ;BR IF OK (MARK)
5327 026256 104014          ERROR   14         ;ERROR, CRC WRONG (SPACE)
5328 026260 000404          BR       79$        ;CONTINUE
5329 026262 004737 030226 78$:  JSR      PC,GETSI    ;LOOK AT BIT WINDOW
5330 026266 103001          BCC      79$        ;BR IF OK (SPACE)
5331 026270 104014          ERRGR   14         ;ERROR, CRC WRONG (MARK)
5332 026272
5333 026272 005203          INC      R3          ;BUMP BIT COUNTER
5334 026274 022703 000020  CMP      #20,R3     ;FINISHED BCC YET?
5335 026300 001357          BNE      77$        ;BR IF NO
5336 026302 005003          CLR      R3         ;CLEAR BIT COUNTER
5337
5338                          ;CHECK TO SEE IF TRANSMITTER IS MARKING
5339
5340 026304 104413 000001 2$:  DATACL,1          ;CLOCK TRANSMITTER
5341 026310 004737 030226  JSR      PC,GETSI    ;LOOK AT WINDOW
5342 026314 103401          BCS      3$        ;IT SHOULD BE MARKING
5343 026316 104024          ERROR   24         ;ERROR, BIT WAS A SPACE
5344 026320 005203          INC      R3          ;BUMP BIT COUNTER
5345 026322 022703 000007 3$:  CMP      #7,R3     ;DONE YET
5346 026326 001366          BNE      2$        ;BR IF NO
5347 026330 104413 000010  DATACL,10         ;GIVE ENOUGH TICKS TO CLEAR OUT ACTIVE
5348 026334 005003          CLR      R3         ;CLEAR BIT COUNTER
5349 026336 104413 000001 4$:  DATACL,1          ;SHIFT OUT NEXT BIT
5350 026342 004737 030226  JSR      PC,GETSI    ;LOOK AT BIT WINDOW
5351 026346 103401          BCS      +4        ;BR IF IT IS A MARK
5352 026350 104024          ERROR   24         ;ERROR, TRANSMITTER IS NOT MARKING
5353 026352 005203          INC      R3         ;INC BIT COUNT
5354 026354 022703 000020  CMP      #20,R3     ;DONE YET?
5355 026360 001366          BNE      4$        ;BR IF NO
5356
5357                          ;CHECK TO SEE THAT FIRST FOUR CHARACTER MESSAGE
5358                          ;WAS RECEIVED CORRECTLY (0,125,252,377)
5359
5360 026362 104413 000001  DATACL,1          ;GET LAST BIT IN RECEIVER
5361 026366 012703 000004  MOV      #4,R3      ;R3=CHARACTER COUNT
5362 026372 012702 032012  MOV      #MESDAT,R2 ;LOAD MESSAGE POINTER IN R2
5363 026376 004737 031066 40$: JSR      PC,INRDY    ;WAIT FOR INRDY
5364 026402 104412          ROMCLK 021204      ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5365 026404 021204
5366 026406 016104 000004  MOV      4(R1),R4   ;PUT 'FOUND' IN R4
5367 026412 112205          MOVB    (R2)+,R5   ;PUT 'EXPECTED' IN R5

```

CZKCF MACY11 3GA(1052) 08-JUL-80 08:27 PAGE 105  
 CZKCF.P11 08-JUL-80 08:24 BASIC RECEIVER TESTS

SEQ 0104

```

5368 026414 120504          CMPB   R5,R4          ;IS RECEIVED DATA CORRECT?
5369 026416 001401          BEQ    41$           ;BR IF YES
5370 026420 104010          ERROR  10           ;RECEIVE DATA ERROR
5371 026422 005303          41$: DEC    R3       ;DEC CHARACTER COUNT
5372 026424 001364          BNE    40$           ;BR IF NOT DONE YET
5373
5374          ;CHECK TO SEE THAT IN BCC MATCH IS SET
5375          ;AND THAT THE BCC WAS RECEIVED CORRECTLY
5376
5377 026426 004737 031066      JSR    PC,INRDY      ;WAIT FOR INRDY
5378 026432 104412          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5379 026434 021204          021204         ;GET FIRST HALF OF CRC
5380 026436 116137 000004 001302  MOVB   4(R1),$TMP2   ;PUT IN $TMP2
5381 026444 042737 177400 001302  BIC    #177400,$TMP2 ;CLEAR HI BYTE
5382 026452 004737 031066      JSR    PC,INRDY      ;WAIT FOR INRDY
5383 026456 104412          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5384 026460 021244          021244
5385 026462 016104 000004          MOV    4(R1),R4     ;PUT 'FOUND' IN R4
5386 026466 042704 000376          BIC    #376,R4      ;CLEAR UNWANTED BITS
5387 026472 012705 000001          MOV    #1,R5        ;PUT 'EXPECTED' IN R5
5388 026476 120504          CMPB   R5,R4        ;IS IN BCC MATCH SET?
5389 026500 001401          BEQ    50$           ;IN BCC MATCH ERROR
5390 026502 104015          ERROR  15
5391 026504          50$:
5392 026504 104412          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5393 026506 021204          021204         ;GET LAST HALF
5394 026510 116137 000004 001301  MOVB   4(R1),$TMP1+1 ;PUT IN $TMP1
5395 026516 042737 000377 001300  BIC    #377,$TMP1   ;CLEAR LO BYTE
5396 026524 053737 001300 001302  BIS    $TMP1,$TMP2  ;16 BIT BCC NOW IN $TMP2
5397 026532 023737 031230 001302  CMP    CALBCC,$TMP2 ;IS IT CORRECT?
5398 026540 001401          BEQ    45$           ;BR IF OK
5399 026542 104027          ERROR  27
5400
5401          ;CHECK THAT CHARACTER LOADED WITH SOM WAS RECEIVED (000)
5402
5403 026544 004737 031066      45$: JSR    PC,INRDY      ;WAIT FOR INRDY
5404 026550 104412          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5405 026552 021204          021204         ;GET RECEIVE DATA
5406 026554 016104 000004          MOV    4(R1),R4     ;PUT 'FOUND' IN R4
5407 026560 005005          CLR    R5           ;PUT 'EXPECTED' IN R5
5408 026562 120504          CMPB   R5,R4        ;IS RECEIVED DATA CORRECT?
5409 026564 001401          BEQ    42$           ;BR IF YES
5410 026566 104010          ERROR  10           ;RECEIVE DATA ERROR
5411
5412          ;CHECK TO SEE THAT SECOND FOUR CHARACTER MESSAGE
5413          ;WAS RECEIVED CORRECTLY (0,125,252,377)
5414
5415 026570 012703 000004          42$: MOV    #4,R3       ;R3=CHARACTER COUNT
5416 026574 012702 032012          MOV    #MESDAT,R2   ;LOAD MESSAGE POINTER IN R2
5417 026600 004737 031066      43$: JSR    PC,INRDY      ;WAIT FOR INRDY
5418 026604 104412          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5419 026606 021204          021204
5420 026610 016104 000004          MOV    4(R1),R4     ;PUT 'FOUND' IN R4
5421 026614 112205          MOVB  (R2)+,R5      ;PUT 'EXPECTED' IN R5
5422 026616 120504          CMPB   R5,R4        ;IS RECEIVED DATA CORRECT?
5423 026620 001401          BEQ    44$           ;BR IF YES

```



```
5424 026622 104010          ERROR 10          ;RECEIVE DATA ERROR
5425 026624 005303          44$: DEC R3          ;DEC CHARACTER COUNT
5426 026626 001364          BNE 43$          ;BR IF NOT DONE YET
5427
5428          ;CHECK TO SEE THAT IN BCC MATCH IS SET
5429          ;AND THAT THE BCC WAS RECEIVED CORRECTLY
5430
5431 026630 004737 031066    JSR PC,INRDY      ;WAIT FOR INRDY
5432 026634 104412          ROMCLK           ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5433 026636 021204          021204          ;GET FIRST HALF OF CRC
5434 026640 116137 000004 001302    MOVB 4(R1),$TMP2 ;PUT IN $TMP2
5435 026646 042737 177400 001302    BIC #177400,$TMP2 ;CLEAR HI BYTE
5436 026654 004737 031066    JSR PC,INRDY      ;WAIT FOR INRDY
5437 026660 104412          ROMCLK           ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5438 026662 021244          021244          ;
5439 026664 016104 000004    MOV 4(R1),R4      ;PUT 'FOUND' IN R4
5440 026670 042704 000376    BIC #376,R4       ;CLEAR UNWANTED BITS
5441 026674 012705 000001    MOV #1,R5         ;PUT 'EXPECTED' IN R5
5442 026700 120504          CMPB R5,R4        ;IS IN BCC MATCH SET?
5443 026702 001401          BEQ 51$          ;
5444 026704 104015          ERROR 15          ;IN BCC MATCH ERROR
5445 026706          51$:
5446 026706 104412          ROMCLK           ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5447 026710 021204          021204          ;GET LAST HALF
5448 026712 116137 000004 001301    MOVB 4(R1),$TMP1+1 ;PUT IN $TMP1
5449 026720 042737 000377 001300    BIC #377,$TMP1    ;CLEAR LO BYTE
5450 026726 053737 001300 001302    BIS $TMP1,$TMP2   ;16 BIT BCC NOW IN $TMP2
5451 026734 023737 031230 001302    CMP CALBCC,$TMP2 ;IS IT CORRECT?
5452 026742 001401          BEQ 5$           ;BR IF OK
5453 026744 104027          ERROR 27
5454 026746          5$:
5455
5456
5457          ;***** TEST 54 *****
5458          ;*EMPTY SILO TEST
5459          ;*LOAD SILO WITH 2 SYNCs, 4 CHAR MESSAGE, SINGLE CLOCK
5460          ;*UNTIL THE SILO IS EMPTY, LOAD 4 MORE CHARACTERS IN THE
5461          ;*SILO. GIVE MORE TICKS, AND VERIFY THAT ONLY THE FIRST
5462          ;*4 CHARACTER MESSAGE WAS RECEIVED AND THAT RTS IS CLEAR
5463          ;*****
5464
5465          ; TEST 54
5466          ;-----
5467          ;*****
5468 026746 000004          TST54: SCOPE
5469 026750 012737 000054 001202    MOV #54,$STNM     ; LOAD THE NO. OF THIS TEST
5470 026756 012737 027200 001442    MOV #TST55,NEXT  ; POINT TO THE START OF NEXT TEST.
5471
5472 026764 104410          MSTCLR          ;R1 CONTAINS BASE KMC11 ADDRESS
5473 026766 012711 004000          MOV #BIT11,(R1) ;MASTER CLEAR KMC11
5474 026772 012702 032012          MOV #MESDAT,R2  ;SET LINE UNIT LOOP
5475 026776 012700 000004          MOV #4,R0       ;R2 POINTS TO MESSAGE
5476 027002 004737 031374          JSR PC,SYNLD    ;R0 = CHAR COUNT
5477 027006 004737 030412          JSR PC,OUTRDY  ;LOAD SILO WITH TWO SYNCs
5478 027012 004537 031530          JSR R5,MESLD   ;WAIT FOR OUTRDY
5479 027016 032012          MESDAT         ;LOAD MESSAGE IN SILO
                    ;START OF MESSAGE
```

```
5480 027020 000004 4 ; CHARACTER COUNT
5481 027022 004737 030260 JSR PC,OCOR ; WAIT FOR OCOR
5482 027026 104413 000065 DATACLK, 65 ; CLOCK DATA (EMPTY SILO)
5483 027032 004537 031530 JSR R5,MESLD ; PUT MORE CHARACTERS IN SILO
5484 027036 032012 MESDAT
5485 027040 000004 4
5486 027042 004737 030260 JSR PC,OCOR
5487 027046 104413 000005 DATACLK, 5 ; CLOCK UNTIL RTS IS CLEARED
5488 027052 104412 ROMCLK ; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5489 027054 021264 021264 ; GET RTS
5490 027056 032761 000040 000004 BIT #BIT5,4(R1) ; IS IT CLEAR?
5491 027064 001401 BEQ 5$ ; BR IF YES
5492 027066 104034 ERROR 34 ; ERROR, RTS NOT CLEAR
5493 027070 104413 000041 5$: DATACLK, 41 ; CLOCK XMITTER SOME MORE
5494 027074 004737 031066 1$: JSR PC,INRDY ; OK LETS CHECK WHAT WAS RECEIVED
5495 027100 104412 ROMCLK ; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5496 027102 021204 021204 ; GET RECEIVE DATA
5497 027104 016104 000004 MOV 4(R1),R4 ; PUT IT IN R4
5498 027110 112205 MOVB (R2)+,R5 ; R5 = 'EXPECTED'
5499 027112 120504 CMPB R5,R4 ; IS DATA CORRECT?
5500 027114 001401 BEQ 2$ ; BR IF OK
5501 027116 104010 ERROR 10 ; DATA ERROR
5502 027120 005300 2$: DEC R0 ; DEC CHAR COUNT
5503 027122 001364 BNE 1$ ; BR IF NOT DONE YET
5504 027124 004737 031066 3$: JSR PC,INRDY ; WAIT FOR INRDY
5505 027130 104412 ROMCLK ; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5506 027132 021204 021204 ; GET RECEIVE DATA
5507 027134 016104 000004 MOV 4(R1),R4 ; PUT IT IN 'FOUND'
5508 027140 012705 000377 MOV #377,R5 ; R5 = 'EXPECTED'
5509 027144 120504 CMPB R5,R4 ; SHOULD SEE 377
5510 027146 001401 BEQ 4$ ; BR IF OK
5511 027150 104010 ERROR 10 ; ERROR, TRANSMITTER DID NOT ABORT
5512 027152 004737 031066 4$: JSR PC,INRDY ; WAIT FOR INRDY
5513 027156 104412 ROMCLK ; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5514 027160 021204 021204 ; GET RECEIVE DATA
5515 027162 016104 000004 MOV 4(R1),R4 ; PUT IT IN 'FOUND'
5516 027166 012705 000377 MOV #377,R5 ; R5 = 'EXPECTED'
5517 027172 120504 CMPB R5,R4 ; SHOULD SEE 377
5518 027174 001401 BEQ 10$ ; BR IF OK
5519 027176 104010 ERROR 10 ; ERROR, TRANSMITTER DID NOT ABORT
5520 027200 10$:
```

```
5521
5522
5523 ;***** TEST 55 *****
5524 ;*HALF DUPLEX TEST
5525 ;*SET LINE UNIT LOOP AND HALF DUPLEX, SEND SYNCs AND A
5526 ;*MESSAGE. VERIFY THAT IN-ACTIVE AND IN-READY ARE CLEAR
5527 ;*****
```

```
5528
5529 ; TEST 55
5530 ;-----
5531 ;*****
5532 027200 000004 TST55: SCOPE
5533 027202 012737 000055 001202 MOV #55,$STNM ; LOAD THE NO. OF THIS TEST
5534 027210 012737 027316 001442 MOV #TST56,NEXT ; POINT TO THE START OF NEXT TEST.
5535 ;R1 CONTAINS BASE KMC11 ADDRESS
```

```

5536 027216 104410          MSTCLR          ;MASTER CLEAR KMC11
5537 027220 012702 000012    MOV #12,R2      ;SAVE R2 FOR TYPEOUT
5538 027224 012711 004000    MOV #BIT11,(R1) ;SET LINE UNIT LOOP
5539 027230 012761 000020 000004 MOV #BIT4,4(R1) ;LOAD PORT4
5540 027236 104412          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5541 027240 122113          122113         ;SET H/D BIT
5542 027242 004737 031374    JSR PC,SYNLD   ;LOAD 2 SYNCs
5543 027246 004737 030412    JSR PC,OUTRDY ;WAIT FOR OUTRDY
5544 027252 004537 031530    JSR R5,MESLD  ;LOAD 4 CHAR MESSAGE
5545 027256 032012          MESDAT         ;ADDRESS OF MESSAGE
5546 027260 000004          4              ;CHARACTER COUNT
5547 027262 004737 030260    JSR PC,OCOR   ;WAIT FOR OCOR
5548 027266 104413 000073    DATACLK, 73  ;SEND MESSAGE
5549 027272 104412          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5550 027274 021244          021244         ;READ LU-12
5551 027276 016104 000004    MOV 4(R1),R4  ;PUT "FOUND" IN R4
5552 027302 042704 000257    BIC #257,R4   ;CLEAR UNWANTED BITS
5553 027306 005005          CLR R5        ;R5 = "EXPECTED"
5554 027310 120504          CMPB R5,R4   ;IN-ACTIVE AND IN-RDY SHOULD BE CLEAR
5555 027312 001401          BEQ 1$      ;BR IF OK
5556 027314 104035          ERROR 35    ;ERROR BOTH ARE NOT CLEAR
5557 027316

```

1\$:

```

5558
5559
5560 ***** TEST 56 *****
5561 ;*DDCMP CABLE DATA TEST
5562 ;*THIS TEST LOADS OUT SILO WITH THE FOLLOWING:
5563 ;*4 SYNCs,16 CHAR,EOM,16 CHAR,EOM,16 CHAR,EOM
5564 ;*THE 16 CHARACTERS INCLUDE A FLOATING ONE AND ZERO
5565 ;*THE DATA IS TRANSMITTED OVER THE CABLE USING THE INTERNAL CLOCK
5566 ;*RECEIVED DATA IS VERIFIED AS IS IN BCC MATCH
5567 ;*LOOP-BACK CONNECTOR MUST BE ON TO RUN THIS TEST
5568 ;*****
5569

```

```

5570 ; TEST 56
5571 -----
5572 ;*****

```

```

5573 027316 000004          TST56: SCOPE
5574 027320 012737 000056 001202 MOV #56,$STNM ; LOAD THE NO. OF THIS TEST
5575 027326 012737 027706 001442 MOV #TST57,NEXT ; POINT TO THE START OF NEXT TEST.
5576
5577 027334 104410          MSTCLR          ;MASTER CLEAR KMC11
5578 027336 032737 040000 002050 BIT #BIT14,STAT1 ;SKIP TEST IF NO
5579 027344 001557          BEQ 3$        ;LOOPBACK CONNECTOR ON
5580 027346 012711 004000    MOV #BIT11,(R1) ;SET LINE UNIT LOOP
5581 027352 004737 031374    JSR PC,SYNLD   ;LOAD 2 SYNCs
5582 027356 004737 031374    JSR PC,SYNLD   ;LOAD 2 MORE SYNCs
5583 027362 012737 120001 031226 MOV #CRC16,XPOLY ;LOAD POLYNOMIAL FOR SOFT CRC CALC
5584 027370 005037 027420    CLR 6$        ;CLEAR OLD BCC
5585 027374 012703 000020    MOV #16,R3    ;CHARACTER COUNT
5586 027400 012702 032016    MOV #FLDAT,R2 ;R2= POINTER
5587 027404 112237 027416    MOVB (R2)+,5$ ;LOAD CHAR FOR SOFT BCC CALC.
5588 027410 004537 031122    JSR R5,SIMBCC ;CALC SOFT BCC
5589 027414 000010          10           ;SHIFT COUNT
5590 027416 000000          5$: 0        ;CHARACTER
5591 027420 000000          6$: 0        ;OLD BCC

```

|      |        |        |        |        |        |                |   |
|------|--------|--------|--------|--------|--------|----------------|---|
| 5592 | 027422 | 013737 | 031230 | 027420 | MOV    | CALBCC,6\$     | :LOAD OLD BCC                             |
| 5593 | 027430 | 005303 |        |        | DEC    | R3             | :DEC COUNT                                |
| 5594 | 027432 | 001364 |        |        | BNE    | 7\$            | :BR IF NOT DONE YET                       |
| 5595 | 027434 | 004537 | 031530 |        | JSR    | R5,MESLD       | :LOAD SILO                                |
| 5596 | 027440 | 032016 |        |        | FLTDAT |                | :MESSAGE ADDRESS                          |
| 5597 | 027442 | 000020 |        |        | 16.    |                | :CHARACTER COUNT                          |
| 5598 | 027444 | 004737 | 031504 |        | JSR    | PC,EOM         | :LOAD AN EOM                              |
| 5599 | 027450 | 004537 | 031530 |        | JSR    | R5,MESLD       | :LOAD SILO                                |
| 5600 | 027454 | 032016 |        |        | FLTDAT |                | :MESSAGE ADDRESS                          |
| 5601 | 027456 | 000020 |        |        | 16.    |                | :CHARACTER COUNT                          |
| 5602 | 027460 | 004737 | 031504 |        | JSR    | PC,EOM         | :LOAD AN EOM                              |
| 5603 | 027464 | 004537 | 031530 |        | JSR    | R5,MESLD       | :LOAD SILO                                |
| 5604 | 027470 | 032016 |        |        | FLTDAT |                | :MESSAGE ADDRESS                          |
| 5605 | 027472 | 000020 |        |        | 16.    |                | :CHARACTER COUNT                          |
| 5606 | 027474 | 004737 | 031504 |        | JSR    | PC,EOM         | :LOAD AN EOM                              |
| 5607 | 027500 | 004737 | 030260 |        | JSR    | PC,OCOR        | :WAIT FOR OCOR                            |
| 5608 | 027504 | 005011 |        |        | CLR    | (R1)           | :CLEAR LINE UNIT LOOP                     |
| 5609 | 027506 | 012700 | 000003 |        | MOV    | #3,R0          | :R0 = MESSAGE COUNT                       |
| 5610 | 027512 | 012703 | 000020 |        | MOV    | #16.,R3        | :R3= CHARACTER COUNT                      |
| 5611 | 027516 | 012702 | 032016 |        | MOV    | #FLTDAT,R2     | :LOAD MESSAGE POINTER IN R2               |
| 5612 | 027522 | 004737 | 031066 | 1\$:   | JSR    | PC,INRDY       | :WAIT FOR INRDY                           |
| 5613 | 027526 | 104412 |        |        | ROMCLK |                | :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 |
| 5614 | 027530 | 021204 |        |        | 021204 |                | :GET DATA FROM IN SILO                    |
| 5615 | 027532 | 016104 | 000004 |        | MOV    | 4(R1),R4       | :PUT CHARACTER IN 'FGUND'                 |
| 5616 | 027536 | 112205 |        |        | MOVB   | (R2)+,R5       | :PUT 'EXPECTED' IN R5                     |
| 5617 | 027540 | 120504 |        |        | CMPB   | R5,R4          | :IS RECEIVED DATA CORRECT                 |
| 5618 | 027542 | 001401 |        |        | BEQ    | 2\$            | :BR IF OK                                 |
| 5619 | 027544 | 104025 |        |        | ERROR  | 25             | :DATA ERROR                               |
| 5620 | 027546 |        |        | 2\$:   |        |                |   |
| 5621 | 027546 | 005303 |        |        | DEC    | R3             | :DEC CHARACTER COUNT                      |
| 5622 | 027550 | 001364 |        |        | BNE    | 1\$            | :BR IF NOT DONE THIS MESSAGE              |
| 5623 | 027552 | 012703 | 000020 |        | MOV    | #16.,R3        | :RESET CHARACTER COUNT                    |
| 5624 |        |        |        |        |        |                |   |
| 5625 |        |        |        |        |        |                | :CHECK TO SEE THAT IN BCC MATCH IS SET    |
| 5626 |        |        |        |        |        |                | :AND THAT THE BCC WAS RECEIVED CORRECTLY  |
| 5627 |        |        |        |        |        |                |   |
| 5628 | 027556 | 004737 | 031966 |        | JSR    | PC,INRDY       | :WAIT FOR INRDY                           |
| 5629 | 027562 | 104412 |        |        | ROMCLK |                | :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 |
| 5630 | 027564 | 021204 |        |        | 021204 |                | :GET FIRST HALF OF CRC                    |
| 5631 | 027566 | 116137 | 000004 | 001302 | MOVB   | 4(R1),\$TMP2   | :PUT IN \$TMP2                            |
| 5632 | 027574 | 042737 | 177400 | 001302 | BIC    | #177400,\$TMP2 | :CLEAR HI BYTE                            |
| 5633 | 027602 | 004737 | 031066 |        | JSR    | PC,INRDY       | :WAIT FOR INRDY                           |
| 5634 | 027606 | 104412 |        |        | ROMCLK |                | :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 |
| 5635 | 027610 | 021244 |        |        | 021244 |                |   |
| 5636 | 027612 | 016104 | 000004 |        | MOV    | 4(R1),R4       | :PUT 'FGUND' IN R4                        |
| 5637 | 027616 | 042704 | 000376 |        | BIC    | #376,R4        | :CLEAR UNWANTED BITS                      |
| 5638 | 027622 | 012705 | 000001 |        | MOV    | #1,R5          | :PUT 'EXPECTED' IN R5                     |
| 5639 | 027626 | 120504 |        |        | CMPB   | R5,R4          | :IS IN BCC MATCH SET?                     |
| 5640 | 027630 | 001401 |        |        | BEQ    | 25\$           |   |
| 5641 | 027632 | 104015 |        |        | ERROR  | 15             | :IN BCC MATCH ERROR                       |
| 5642 | 027634 |        |        | 25\$:  |        |                |   |
| 5643 | 027634 | 104412 |        |        | ROMCLK |                | :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 |
| 5644 | 027636 | 021204 |        |        | 021204 |                | :GET LAST HALF                            |
| 5645 | 027640 | 116137 | 000004 | 001301 | MOVB   | 4(R1),\$TMP1+1 | :PUT IN \$TMP1                            |
| 5646 | 027646 | 042737 | 000377 | 001300 | BIC    | #377,\$TMP1    | :CLEAR LO BYTE                            |
| 5647 | 027654 | 053737 | 001300 | 001302 | BIS    | \$TMP1,\$TMP2  | :16 BIT BCC NOW IN \$TMP2                 |

```

5648 027662 023737 031230 001302      CMP    CALBCC,$TMP2      ;IS IT CORRECT?
5649 027670 001401                      BEQ    4$                ;BR IF OK
5650 027672 104027                      ERROR  27
5651 027674 012702 032016      4$:  MOV    #FLTDAT,R2      ;RESET MESSAGE POINTER
5652 027700 005300                      DEC    R0                ;DECREMENT COUNTER
5653 027702 001307                      BNE   1$                ;BR IF NOT DONE
5654 027704 104420      3$:  ADVANCE          ; ADVANCE TO NEXT TEST

```

```

5655
5656
5657      :***** TEST 57 *****
5658      :*DDCMP CABLE DATA TEST
5659      :*THIS TEST LOADS OUT SILO WITH THE FOLLOWING:
5660      :*4 SYNCs,59 DATA CHARACTERS,EOM WITH GARBAGE CHARACTER
5661      :*THE DATA IS TRANSMITTED OVER THE CABLE USING THE INTERNAL CLOCK
5662      :*RECEIVED DATA IS VERIFIED AS IS IN BCC MATCH
5663      :*LOOP-BACK CONNECTOR MUST BE ON TO RUN THIS TEST
5664      :*****

```

```

5665
5666      : TEST 57
5667      :-----

```

```

5668      :*****
5669 027706 000004      TST57: SCOPE
5670 027710 012737 000057 001202      MOV    #57,$STSTNM      ; LOAD THE NO. OF THIS TEST
5671 027716 012737 003662 001442      MOV    #SEOP,NEXT      ; POINT TO THE END OF PASS HANDLER.
5672
5673 027724 104410      MSTCLR          ;R1 CONTAINS BASE KMC11 ADDRESS
5674 027726 032737 040000 002050      BIT    #BIT14,STAT1    ;MASTER CLEAR KMC11
5675 027734 001533                      BEQ    3$                ;SKIP TEST IF NO
5676 027736 012711 004000                      MOV    #BIT11,(R1)     ;LOOPBACK CONNECTOR ON
5677 027742 004737 031374                      JSR    PC,SYNLD        ;SET LINE UNIT LOOP
5678 027746 004737 031374                      JSR    PC,SYNLD        ;LOAD 2 SYNCs
5679 027752 012737 120001 031226      MOV    #CRC16,XPOLY    ;LOAD 2 MORE SYNCs
5680 027760 005037 030010                      CLR    6$              ;LOAD POLYNOMIAL FOR SOFT CRC CALC
5681 027764 012703 000073                      MOV    #59.,R3        ;CLEAR OLD BCC
5682 027770 012702 032012                      MOV    #MESDAT,R2     ;CHARACTER COUNT
5683 027774 112237 030006      7$:  MOVB   (R2)+,5$      ;R2= POINTER
5684 030000 004537 031122                      JSR    R5,SIMBCC      ;LOAD CHAR FOR SOFT BCC CALC.
5685 030004 000010                      10
5686 030006 000000                      5$:  0                    ;CALC SOFT BCC
5687 030010 000000                      6$:  0                    ;SHIFT COUNT
5688 030012 013737 031230 030010      MOV    CALBCC,6$      ;CHARACTER
5689 030020 005303                      DEC    R3                ;OLD BCC
5690 030022 001364                      BNE   7$                ;LOAD OLD BCC
5691 030024 004537 031530                      JSR    R5,MESLD       ;DEC COUNT
5692 030030 032012                      MESDAT 59.             ;BR IF NOT DONE YET
5693 030032 000073                      JSR    PC,EOM          ;LOAD SILO
5694 030034 004737 031504                      JSR    PC,OCOR        ;MESSAGE ADDRESS
5695 030040 004737 030260                      CLR    (R1)           ;CHARACTER COUNT
5696 030044 005011                      MOV    #59.,R0        ;LOAD AN EOM
5697 030046 012700 000073                      MOV    #MESDAT,R2     ;WAIT FOR OCOR
5698 030052 012702 032012                      JSR    PC,INRDY       ;CLEAR LINE UNIT LOOP
5699 030056 004737 031066      1$:  JSR    PC,INRDY       ;R0= CHARACTER COUNT
5700 030062 104412                      ROMCLK 021204         ;LOAD MESSAGE POINTER IN R2
5701 030064 021204                      MOV    4(R1),R4        ;WAIT FOR INRDY
5702 030066 016104 000004                      MOVB  (R2)+,R5        ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5703 030072 112205                      ;GET DATA FROM IN SILO
                    ;PUT CHARACTER IN 'FOUND'
                    ;PUT 'EXPECTED' IN R5

```



```

5760 030260 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5761 030262 021364 021364 ;PORT4 LU 17
5762 030264 032777 000020 151602 BIT #BIT4,@KMP04 ;IS OCOR SET?
5763 030272 001772 BEQ OCOR ;BR IF NO
5764 030274 000207 RTS PC ;OK OCOR IS SET, GO BACK
5765
5766

```

```

5767 030276 SYNC:
5768 ;THIS SUBROUTINE LOADS THE SILO WITH THE NUMBER OF SYNC
5769 ;CHARACTERS PASSED TO IT IN THE WORD AFTER THE JSR CALL
5770 ;AND A NON-SYNC CHARACTER (301)
5771

```

```

5772 030276 013637 001276 MOV @ (SP)+,$TMP0 ;GET COUNT
5773 030302 062746 000002 ADD #2,-(SP) ;ADJUST STACK
5774 030306 012761 000026 000004 MOV #26,4(R1) ;LOAD PORT4
5775 030314 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5776 030316 122114 122114 ;LOAD SYNC REGISTER
5777 030320 004737 030412 1$: JSR PC,OUTRDY ;WAIT FOR OUTRDY
5778 030324 012761 000001 000004 MOV #1,4(R1) ;LOAD PORT4
5779 030332 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5780 030334 122111 122111 ;SET SOM
5781 030336 012761 000026 000004 MOV #26,4(R1) ;LOAD PORT4
5782 030344 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5783 030346 122110 122110 ;LOAD OUT DATA
5784 030350 005337 001276 DEC $TMP0 ;ALL DONE?
5785 030354 001361 BNE 1$ ;BR IF NOT
5786 030356 004737 030412 JSR PC,OUTRDY ;WAIT FOR OUTRDY
5787 030362 005061 000004 CLR 4(R1) ;LOAD PORT4
5788 030366 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5789 030370 122111 122111 ;SET SOM
5790 030372 012761 000301 000004 MOV #301,4(R1) ;LOAD PORT4
5791 030400 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5792 030402 122110 122110 ;LOAD OUT DATA
5793 030404 004737 030260 JSR PC,OCOR ;WAIT FOR OCOR
5794 030410 000207 RTS PC
5795
5796

```

```

5797 030412 OUTRDY:
5798 ;THIS SUBROUTINE SPINS ON OUT READY
5799

```

```

5800 030412 005037 001306 CLR $TMP4 ;CLEAR TIMER
5801 030416 1$: ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5802 030416 104412 021224 ;PORT4 LU11
5803 030420 021224 021224 ;IS OUT RDY SET?
5804 030422 032777 000020 151444 BIT #BIT4,@KMP04 ;BR IF YES
5805 030430 001004 BNE 2$ ;KEEP CHECKING IF NOT DONE
5806 030432 005237 001306 INC $TMP4 ;ERROR, OUT READY NOT SET
5807 030436 001367 BNE 1$
5808 030440 104036 ERROR 36
5809 030442 000207 RTS PC
5810
5811

```

```

5812 030444 CHAR:
5813 ;THIS SUBROUTINE LOADS THE SILO WITH 3 SYNCs
5814 ;AND THE CHARACTER PASSED TO IT.
5815

```

```

5816 030444 013637 001300      MOV    @ (SP)+, $TMP1    ;GET CHARACTER
5817 030450 062746 000002      ADD    #2, -(SP)        ;ADJUST STACK
5818 030454 012737 000003 001276  MOV    #3, $TMP0        ;SET FOR 3 SYNCs
5819 030462 012761 000026 000004  MOV    #26, 4(R1)       ;LOAD PORT4
5820 030470 104412              ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5821 030472 122114              122114 ;LOAD SYNC REGISTER
5822 030474 004737 030412 1$: JSR    PC, OUTRDY       ;WAIT FOR OUTRDY
5823 030500 012761 000001 000004  MOV    #1, 4(R1)        ;LOAD PORT4
5824 030506 104412              ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5825 030510 122111              122111 ;SET SOM
5826 030512 012761 000026 000004  MOV    #26, 4(R1)       ;LOAD PORT4
5827 030520 104412              ROMCLK ;NEXT WORD IS INSTRUCTION ROMCLK PC=5304
5828 030522 122110              122110 ;LOAD OUT DATA
5829 030524 005337 001276      DEC    $TMP0            ;ALL DONE?
5830 030530 001361              BNE    1$              ;BR IF NOT
5831 030532 004737 030412      JSR    PC, OUTRDY       ;WAIT FOR OUTRDY
5832 030536 013761 001300 000004  MOV    $TMP1, 4(R1)     ;LOAD PORT4
5833 030544 104412              ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5834 030546 122110              122110 ;LOAD OUT DATA
5835 030550 004737 030260      JSR    PC, OCOR         ;WAIT FOR OCOR
5836 030554 000207              RTS    PC
5837
5838
5839 030556                      CHARSD:
5840                                ;THIS SUBROUTINE LOADS THE SILO WITH THE CHARACTER PASSED TO IT.
5841
5842 030556 013637 001300      MOV    @ (SP)+, $TMP1    ;GET CHARACTER
5843 030562 062746 000002      ADD    #2, -(SP)        ;ADJUST STACK
5844 030566 004737 030412      JSR    PC, OUTRDY       ;WAIT FOR OUTRDY
5845 030572 013761 001300 000004  MOV    $TMP1, 4(R1)     ;LOAD PORT4
5846 030600 104412              ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5847 030602 122110              122110 ;LOAD OUT DATA
5848 030604 004737 030412      JSR    PC, OUTRDY       ;WAIT FOR OUTRDY
5849 030610 104412              ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5850 030612 122110              122110 ;LOAD GARBAGE CHAR
5851 030614 004737 030260      JSR    PC, OCOR         ;WAIT FOR OCOR
5852 030620 000207              RTS    PC
5853
5854
5855 030622                      S!LOLD:
5856                                ;THIS SUBROUTINE FILLS THE OUT SILO
5857                                ; WITH A BINARY COUNT PATTERN
5858
5859 030622 012737 000073 001300  MOV    #73, $TMP1       ;LOAD COUNT
5860 030630 005737 031062      TST    $CHAR            ;FIRST TIME HERE?
5861 030634 100470              BMI    4$              ;BR IF BITSTUFF
5862 030636 001032              BNE    2$              ;BR IF NO
5863 030640 062737 000002 001300  ADD    #2, $TMP1        ;ADD 2 TO CHARACTER COUNT
5864 030646 012737 000003 001276  MOV    #3, $TMP0        ;SET FOR 3 SYNCs
5865 030654 012761 000026 000004  MOV    #26, 4(R1)       ;LOAD PORT4
5866 030662 104412              ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5867 030664 122114              122114 ;LOAD SYNC REGISTER
5868 030666 004737 030412 1$: JSR    PC, OUTRDY       ;WAIT FOR OUTRDY
5869 030672 012761 000001 000004  MOV    #1, 4(R1)        ;LOAD PORT4
5870 030700 104412              ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5871 030702 122111              122111 ;SET SOM
    
```



```

5872 030704 012761 000026 000004      MOV      #26,4(R1)      ;LOAD PORT4
5873 030712 104412                    ROMCLK                    ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5874 030714 122110                    122110                    ;LOAD OUT DATA
5875 030716 005337 001276      DEC      $TMP0          ;ALL DONE?
5876 030722 001361                    BNE     1$              ;BR IF NOT
5877 030724 004737 030412      JSR     PC,OUTRDY      ;WAIT FOR OUTRDY
5878 030730 013761 031062 000004      MOV      $CHAR,4(R1)   ;LOAD PORT4
5879 030736 104412                    ROMCLK                    ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5880 030740 122110                    122110                    ;LOAD OUT DATA
5881 030742 005737 031064      TST     $TUFLG         ;BITSTUFF???
5882 030746 001407                    BEQ     6$              ;BR IF NO
5883 030750 013737 031062 030762      MOV      $CHAR,5$     ;IT IS SLD SO CHECK BITSTUFFING
5884 030756 004537 031612      JSR     R5,$TFFCL     ;ADD ANY BIT STUFF CLOCK TICKS
5885 030762 000000                    5$: 0                    ;CHARACTER
5886 030764 000010                    10                          ;CHIFT COUNT
5887 030766 005237 031062      6$:  INC      $CHAR     ;NEXT CHARACTER
5888 030772 022737 000400 031062      CMP     #400,$CHAR    ;ALL DONE?
5889 031000 001403                    BEQ     3$              ;DECREMENT COUNT
5890 031002 005337 001300      DEC     $TMP1          ;BR IF NOT DONE
5891 031006 001346                    BNE     2$              ;WAIT FOR OCOR
5892 031010 004737 030260      3$:  JSR     PC,OCOR
5893 031014 000207                    RTS     PC
5894 031016 005037 031062      4$:  CLR     $CHAR     ;START PATTERN AT ZERO
5895 031022 012737 177777 031064      MOV     #-1,$TUFLG    ;SET BITSTUFF FLAG
5896 031030 005037 032010      CLR     $BITCON       ;CLEAR STUFF COUNT
5897 031034 062737 000002 001300      ADD     #2,$TMP1      ;ADD 2 TO CHARACTER COUNT
5898 031042 012761 000001 000004      MOV     #1,4(R1)     ;SET BIT0 IN PORT4
5899 031050 104412                    ROMCLK                    ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5900 031052 122111                    122111                    ;SET SOM!
5901 031054 104412                    ROMCLK                    ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5902 031056 122110                    122110                    ;LOAD GARBAGE CHAR
5903 031060 000721                    BR      2$              ;GO LOAD SILO
5904 031062 000000      $CHAR: 0
5905 031064 000000      $TUFLG: 0
5906
5907
5908 031066      INRDY:
5909      ;THIS SUBROUTINE SPINS ON INRDY
5910      ;IF INRDY FAILS TO SET THE DELAY TIMES OUT AND AN
5911      ;ERROR IS REPORTED. FOR BETTER SCOPE LOOPS THIS
5912      ;DELAY CAN BE MADE SHORTER BY ALTERING THE NUMBER
5913      ;INITIALLY LOADED INTO $TMP0, THE SMALLER THE NUMBER
5914      ;THE SHORTER THE DELAY. 0 IS THE LONGEST DELAY.
5915
5916 031066 012737 000000 001276      1$:  MOV     #0,$TMP0    ;SET UP DELAY COUNTER
5917 031074                    ROMCLK                    ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5918 031074 104412                    021244                    ;PORT4 LU12
5919 031076 021244                    BIT     #BIT4,$KMP04   ;IS INRDY SET?
5920 031100 032777 000020 150766      BNE     2$              ;BR IF YES
5921 031106 001004                    INC     $TMP0          ;INC DELAY
5922 031110 005237 001276      BNE     1$              ;TRY AGAIN
5923 031114 001367                    ERROR  37              ;ERROR,NO INRDY
5924 031116 104037                    2$:  RTS     PC        ;RETURN
5925 031120 000207
5926
5927

```

```

5928 031122      SIMBCC:
5929              ;THIS SUBROUTINE CALCULATES THE CRC USING POLYNOMIAL GIVEN
5930              ;IN XPOLY. THE CORRECT CRC IS $LPADRED IN CALBCC, AND THE
5931              ;STATE OF THE LSB OF THE BCC IS $LPADRED IN THE C BIT.
5932
5933 031122 010046      MOV      R0,-(SP)          ;SAVE R0 ON STACK
5934 031124 012537 001276      MOV      (R5)+,$TMP0      ;$TMP0 = SHIFT COUNT
5935 031130 012537 001300      MOV      (R5)+,$TMP1      ;$TMP1 = CHARACTER
5936 031134 012537 031230      MOV      (R5)+,CALBCC     ;CALBCC = OLD BCC
5937 031140 013700 031230      1$:      MOV      CALBCC,R0        ;PUT OLD BCC IN R0
5938 031144 000241      CLC
5939 031146 006037 031230      ROR      CALBCC          ;SHIFT OLD BCC
5940 031152 006037 001300      ROR      $TMP1          ;SHIFT CHARACTER
5941 031156 005500      ADC      R0              ;ADD CHAR CARRY TO OLD BCC
5942 031160 006000      ROR      R0              ;PUT BIT0 TO CARRY BIT
5943 031162 103011      BCC     2$              ;CARRY IS FEEDBACK BIT
5944 031164 013700 031226      MOV      XPOLY,R0        ;IF FEEDBACK = 1
5945 031170 043700 031230      BIC     CALBCC,R0        ;EXCLUSIVLY OR XPOLY TO CALBCC
5946 031174 043737 031226 031230      BIC     XPOLY,CALBCC
5947 031202 050037 031230      BIS     R0,CALBCC
5948 031206 005337 001276      2$:      DEC     $TMP0          ;DEC SHIFT COUNT
5949 031212 001352      BNE     1$              ;BR IF NOT DONE
5950 031214 013700 031230      MOV     CALBCC,R0        ;PUT RESULT IN R0
5951 031220 006000      ROR     R0              ;SHIFT BIT0 TO CARRY
5952 031222 012600      MOV     (SP)+,R0        ;RESTORE R0
5953 031224 000205      RTS     R5              ;$LPADR
5954 031226 000000      XPOLY: 0
5955 031230 000000      CALBCC: 0
5956              LRC8=200
5957              CRC16=120001
5958              CRC.CCITT=102010
5959
5960
5961 031232      BCCLD:
5962              ;THIS SUBROUTINE LOADS THE OUT SILO WITH 2 SYNCs
5963              ;WITH SOM SET, AND ONE CHARACTER PASSED TO IT
5964              ;WITH THE SOM BIT CLEAR (ENABLE CRC)
5965
5966 031232 013637 001300      MOV     @($P)+,$TMP1     ;GET CHARACTER
5967 031236 062746 000002      ADD     #2,-(SP)        ;ADJUST STACK
5968 031242 012737 000002 001276      MOV     #2,$TMP0        ;SET FOR 2 SYNCs
5969 031250 012761 000026 000004      MOV     #26,4(R1)      ;LOAD PORT4
5970 031256 104412      ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5971 031260 122114      122114          ;LOAD SYNC REGISTER
5972 031262 004737 030412 1$:      JSR     PC,OUTRDY      ;WAIT FOR OUTRDY
5973 031266 012761 000001 000004      MOV     #1,4(R1)      ;LOAD PORT4
5974 031274 104412      ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5975 031276 122111      122111          ;SET SOM
5976 031300 012761 000026 000004      MOV     #26,4(R1)      ;LOAD PORT4
5977 031306 104412      ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5978 031310 122110      122110          ;LOAD OUT DATA
5979 031312 005337 001276      DEC     $TMP0          ;ALL DONE?
5980 031316 001361      BNE     1$              ;BR IF NOT
5981 031320 004737 030412      JSR     PC,OUTRDY      ;WAIT FOR OUTRDY
5982 031324 013761 001300 000004      MOV     $TMP1,4(R1)    ;LOAD PORT4
5983 031332 104412      ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304

```

```

5984 031334 122110          122110          ;LOAD OUT DATA
5985 031336 004737 030260    JSR      PC,OCOR      ;WAIT FOR OCOR
5986 031342 000207          RTS      PC
5987
5988
5989 031344          GETOO:          ;THIS SUBROUTINE READS THE STATE OF THE TRANSMIT
5990          ;BCC LSB AND PUTS IT IN THE CARRY BIT
5991
5992
5993 031344 104412          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5994 031346 021364          021364          ;PORT4 LU-17
5995 031350 106177 150520    ROLB     @KMP04      ;PUT Q0 IN CARRY
5996 031354 000207          RTS      PC          ;RETURN
5997
5998
5999 031356          GETOI:          ;THIS SUBROUTINE READS THE STATE OF THE RECEIVE
6000          ;BCC LSB AND PUTS IT IN THE CARRY BIT
6001
6002
6003 031356 104412          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
6004 031360 021364          021364          ;PORT4 LU-17
6005 031362 106177 150506    ROLB     @KMP04      ;PUT Q0 IN CARRY
6006 031366 106177 150502    ROLB     @KMP04      ;PUT Q1 IN CARRY
6007 031372 000207          RTS      PC          ;RETURN
6008
6009
6010 031374          SYNLD:         ;THIS SUBROUTINE LOADS OUT SILO WITH
6011          ;2 SYNC CHARACTERS WITH SOM SET
6012
6013
6014 031374 012737 000002 001276  MOV     #2,$TMP0      ;LOAD COUNTER FOR 2 SYNCs
6015 031402 012761 000026 000004  MOV     #26,4(R1)     ;PORT4 26
6016 031410 104412          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
6017 031412 122114          122114          ;LOAD SYNC REG
6018 031414 004737 030412 1$:    JSR     PC,OUTRDY     ;WAIT FOR OUTRDY
6019 031420 012761 000001 000004  MOV     #1,4(R1)     ;LOAD PORT4
6020 031426 104412          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
6021 031430 122111          122111          ;SET SOM
6022 031432 012761 000026 000004  MOV     #26,4(R1)     ;PORT 26
6023 031440 104412          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
6024 031442 122110          122110          ;LOAD OUT DATA WITH SYNC
6025 031444 005337 001276    DEC     $TMP0        ;DECREMENT COUNTER
6026 031450 001361          BNE     1$          ;BR IF NOT DONE
6027 031452 000207          RTS      PC          ;RETURN
6028
6029
6030 031454          SOM:           ;THIS SUBROUTINE LOADS SOM AND OUT DATA WITH A
6031          ;GARBAGE CHARACTER (0)
6032
6033
6034 031454 004737 030412  JSR     PC,OUTRDY     ;WAIT FOR OUTRDY
6035 031460 012761 000001 000004  MOV     #1,4(R1)     ;PORT4 1
6036 031466 104412          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
6037 031470 122111          122111          ;SET SOM
6038 031472 005061 000004    CLR     4(R1)        ;CLEAR DATA CHAR
6039 031476 104412          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304

```

CZKCF MACY11 30A(1052) 08-JUL-80 08:27 PAGE 117  
 CZKCF.P11 08-JUL-80 08:24 SUBROUTINES

SEQ 0116

```

6040 031500 122110          122110          ;LOAD GARBAGE CHARACTER
6041 031502 000207          RTS          PC          ;RETURN
6042
6043
6044 031504          EOM:
6045          ;THIS SUBROUTINE LOADS EOM AND OUT DATA WITH A
6046          ;GARBAGE CHARACTER (2) TO ENABLE TRANSMISSION OF BCC
6047
6048 031504 004737 030412      JSR          PC,OUTRDY      ;WAIT FOR OUTRDY
6049 031510 012761 000002 000004  MOV          #2,4(R1)      ;PORT4 2
6050 031516 104412          ROMCLK         ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
6051 031520 122111          122111          ;SET EOM
6052 031522 104412          ROMCLK         ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
6053 031524 122110          122110          ;LOAD GARBAGE CHARACTER
6054 031526 000207          RTS          PC          ;RETURN
6055
6056
6057 031530          MESLD:
6058          ;THIS SUBROUTINE LOADS SILO WITH MESSAGE
6059          ;THE FIRST ARUGUMENT IS THE ADDRESS OF THE MESSAGE
6060          ;THE SECOND ARGUMENT IS THE NUMBER OF CHARACTERS IN THE MESSAGE
6061
6062 031530 010046          MOV          RO,-(SP)      ;SAVE RO
6063 031532 012500          MOV          (R5)+,RO      ;RO=MESSAGE POINTER
6064 031534 012537 001276      MOV          (R5)+,$TMP0    ;$TMP0=CHARACTER COUNT
6065 031540 004737 030412      JSR          PC,OUTRDY      ;WAIT FOR OUT RDY
6066 031544 112061 000004      MOVB         (RO)+,4(R1)    ;LOAD PORT4 WITH CHARACTER
6067 031550 104412          ROMCLK         ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
6068 031552 122110          122110          ;LOAD OUT DATA SILO
6069 031554 005337 001276      DEC          $TMP0        ;DEC CHAR COUNT
6070 031560 001367          BNE         1$           ;BR IF NOT DONE
6071 031562 004737 030260      JSR          PC,OCOR       ;WAIT FOR OCOR
6072 031566 012600          MOV          (SP)+,RO      ;RESTORE RO
6073 031570 000205          RTS          R5          ;RETURN
6074
6075
6076 031572          CLRIO:
6077          ;THIS SUBROUTINE SETS IN CLR AND OUT CLR TO
6078          ;CLEAR THE TRANSMIT AND RECEIVE BCC REGISTERS
6079
6080 031572 012761 000200 000004  MOV          #BIT7,4(R1)    ;LOAD PORT4
6081 031600 104412          ROMCLK         ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
6082 031602 122112          122112          ;SET IN CLR!
6083 031604 104412          ROMCLK         ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
6084 031606 122111          122111          ;SET OUT CLR!
6085 031610 000207          RTS          PC          ;RETURN
6086
6087
6088 031612          STFFCL:
6089          ;THIS SUBROUTINE ADDS ANY NECESSARY BIT STUFF CLOCK TICKS
6090          ;FIRST ARGUMENT IS CHAR, SECOND ARGUMENT IS SHIFT COUNT.
6091
6092 031612 010046          MOV          RO,-(SP)      ;SAVE RO
6093 031614 012500          MOV          (R5)+,RO      ;PUT CHAR IN RO
6094 031616 012537 001302      MOV          (R5)+,$TMP2    ;PUT SHIFT COUNT IN $TMP2
6095 031622 106000          1$:  RORB         RO          ;LOOK AT NEXT BIT

```

```

6096 031624 103403          BCS      2$          ;BR IF A MARK
6097 031626 005037 032010  CLR      BITCON     ;IT WAS A SPACE, CLEAR 1'S COUNTER
6098 031632 000412          BR       3$          ;CONTINUE
6099 031634 005237 032010  2$:      INC      BITCON     ;INC CONSECUTIVE 1'S COUNTER
6100 031640 022737 000005 032010  CMP      #5,BITCON   ;IS IT 5 YET?
6101 031646 001004          BNE      3$          ;BR IF NO
6102 031650 005037 032010  CLR      BITCON     ;YES! SO START AGAIN
6103 031654 104413 000001  DATACL, 1          ;GIVE EXTRA TICK TO STUFF ZERO
6104 031660 005337 001302  3$:      DEC      $TMP2     ;DEC SHIFT COUNT
6105 031664 001356          BNE      1$          ;BR IF NOT DONE
6106 031666 012600          MOV      (SP)+,R0    ;RESTORE R0
6107 031670 000205          RTS       R5         ;RETURN
  
```

```

6108
6109
6110 031672          STFFCK:
6111          ;THIS SUBROUTINE CHECKS TO SEE IF TRANSMITTER
6112          ;IS STUFFING ZEROS WHEN IT SHOULD. FIRST ARGUMENT
6113          ;IS THE CHARACTER, SECOND ARGUMENT IS SHIFT COUNT.
  
```

```

6114
6115 031672 010046          MOV      R0,-(SP)    ;SAVE R0
6116 031674 012500          MOV      (R5)+,R0   ;PUT CHAR IN R0
6117 031676 012537 001302  MOV      (R5)+,$TMP2 ;PUT SHIFT COUNT IN $TMP2
6118 031702 106000          1$:      RORB     R0        ;SHIFT OUT NEXT BIT
6119 031704 103403          BCS      2$          ;BR IF IT IS A MARK
6120 031706 005037 032010  CLR      BITCON     ;IT WAS A SPACE, CLEAR 1'S COUNTER
6121 031712 000416          BR       3$          ;CONTINUE
6122 031714 005237 032010  2$:      INC      BITCON     ;INC CONSECUTIVE 1'S COUNTER
6123 031720 022737 000005 032010  CMP      #5,BITCON   ;5 IN A ROW YET?
6124 031726 001010          BNE      3$          ;BR IF NO
6125 031730 005037 032010  CLR      BITCON     ;YES, SO START OVER
6126 031734 104413 000001  DATACL, 1          ;EXTRA TICK TO STUFF ZERO
6127 031740 004737 030226  JSR      PC,GETSI   ;LOOK AT WINDOW
6128 031744 103001          BCC      3$          ;IS IT A ZERO, BR IF YES
6129 031746 104030          ERROR   30        ;NO, ERROR ZERO WAS NOT STUFFED
6130 031750 005337 001302  3$:      DEC      $TMP2     ;DEC SHIFT COUNT
6131 031754 001352          BNE      1$          ;BR IF NOT DONE
6132 031756 012600          MOV      (SP)+,R0    ;RESTORE R0
6133 031760 000205          RTS       R5         ;RETURN
  
```

```

6134
6135
6136 031762          CTSDLY:
6137          ;THIS SUBROUTINE WASTES TIME UNTIL CTS SETS,
6138          ;BUT HOPEFULLY NOT SO LONG THAT THE SILO RUNS OUT
  
```

```

6139
6140 031762 010046          MOV      R0,-(SP)    ;SAVE R0
6141 031764 012700 000032  MOV      #32,R0     ;LOAD R0 WITH COUNT
6142 031770 027777 147250 147246 1$:      CMP      @STKS,@STKS ;WASTE TIME
6143 031776 005300          DEC      R0        ;DECREMENT COUNTER
6144 032000 001373          BNE      1$          ;DO IT AGAIN IF NOT = 0
6145 032002 012600          MOV      (SP)+,R0    ;RESTORE R0
6146 032004 000207          RTS       PC        ;RETURN
  
```

```

6147
6148
6149 032006 000176          FLAG     *B<01111110> ;FLAG CHARACTER
6150 032010 000000          BITCON: 0
6151 032012 000          MESDAT: .BYTE 0,125,252,377
  
```

|      |        |        |        |        |               |  |  |  |  |
|------|--------|--------|--------|--------|---------------|--|--|--|--|
| 6152 | 032015 | 377    |        |        |               |  |  |  |  |
| 6153 | 032016 | 001    | 002    | 004    | FLTDAT: .BYTE | 1,2,4,10,20,40,100,200,376,375,373,367,357,337,277,177         |  |  |  |
| 6154 | 032021 | 010    | 020    | 040    |               |  |  |  |  |
| 6155 | 032024 | 100    | 200    | 376    |               |  |  |  |  |
| 6156 | 032027 | 375    | 373    | 367    |               |  |  |  |  |
| 6157 | 032032 | 357    | 337    | 277    |               |  |  |  |  |
| 6158 | 032035 | 177    |        |        |               |  |  |  |  |
| 6159 | 032036 | 100    | 140    | 160    | STUFDT: .BYTE | 100,140,160,170,3,300,174,176,177,1                            |  |  |  |
| 6160 | 032041 | 170    | 003    | 300    |               |  |  |  |  |
| 6161 | 032044 | 174    | 176    | 177    |               |  |  |  |  |
| 6162 | 032047 | 001    |        |        |               |  |  |  |  |
| 6163 | 032050 | 363    | 347    | 317    | .BYTE         | 363,347,317,200,0,377,377,377,200,37                           |  |  |  |
| 6164 | 032053 | 200    | 000    | 377    |               |  |  |  |  |
| 6165 | 032056 | 377    | 377    | 200    |               |  |  |  |  |
| 6166 | 032061 | 037    |        |        |               |  |  |  |  |
| 6167 |        |        |        |        | .EVEN         |  |  |  |  |
| 6168 | 032062 | 046200 | 047111 | 020105 | EM1:          | .ASCIZ <200>/LINE UNIT INITIALIZATION TEST/                    |  |  |  |
|      | 032120 | 046200 | 047111 | 020105 | EM2:          | .ASCIZ <200>^LINE UNIT REGISTER READ/ONLY TEST^                |  |  |  |
|      | 032163 | 200    | 044514 | 042516 | EM3:          | .ASCIZ <200>^LINE UNIT REGISTER WRITE/READ TEST^               |  |  |  |
|      | 032227 | 200    | 044514 | 042516 | EM4:          | .ASCIZ <200>/LINE UNIT INTERNAL CLOCK FAILURE/                 |  |  |  |
|      | 032271 | 200    | 051124 | 047101 | EM5:          | .ASCIZ <200>/TRANSMITTER DATA ERROR/                           |  |  |  |
|      | 032321 | 200    | 042522 | 042503 | EM6:          | .ASCIZ <200>/RECEIVER TEST/                                    |  |  |  |
|      | 032340 | 051200 | 041505 | 044505 | EM7:          | .ASCIZ <200>/RECEIVER DATA ERROR/                              |  |  |  |
|      | 032365 | 200    | 047515 | 042504 | EM10:         | .ASCIZ <200>/MODEM SIGNAL ERROR/                               |  |  |  |
|      | 032410 | 052200 | 044510 | 020123 |               | .ASCIZ <200>/THIS ERROR COULD BE CAUSED IF YOU HAVE V.35 AND/  |  |  |  |
|      | 032470 | 040600 | 052125 | 051517 |               | .ASCIZ <200>/AUTOSIZED. YOU MUST MANUALLY ANSWER QUESTIONS IF/ |  |  |  |
|      | 032551 | 200    | 047531 | 020125 |               | .ASCIZ <200>/YOU HAVE V.35/                                    |  |  |  |
|      | 032570 | 052200 | 040522 | 051516 | EM11:         | .ASCIZ <200>/TRANSMITTER CRC ERROR/                            |  |  |  |
|      | 032617 | 200    | 042522 | 042503 | EM12:         | .ASCIZ <200>/RECEIVER CRC ERROR/                               |  |  |  |
|      | 032643 | 200    | 047111 | 041040 | EM13:         | .ASCIZ <200>/IN BCC MATCH ERROR (LU REG 12)/                   |  |  |  |
|      | 032703 | 200    | 051124 | 047101 | EM14:         | .ASCIZ <200>/TRANSMITTER FAILED TO GO TO MARK STATE/           |  |  |  |
|      | 032753 | 200    | 040503 | 046102 | EM15:         | .ASCIZ <200>/CABLE DATA TEST/                                  |  |  |  |
|      | 032774 | 043200 | 040514 | 020107 | EM16:         | .ASCIZ <200>/FLAG ERROR/                                       |  |  |  |
|      | 033010 | 052200 | 040522 | 051516 | EM17:         | .ASCIZ <200>/TRANSMITTER FAILED TO STUFF A ZERO/               |  |  |  |
|      | 033054 | 051600 | 044527 | 041524 | EM20:         | .ASCIZ <200>/SWITCH PAC TEST/                                  |  |  |  |
|      | 033075 | 200    | 041101 | 051117 | EM21:         | .ASCIZ <200>/ABORT ERROR/                                      |  |  |  |
|      | 033112 | 052200 | 040522 | 051516 | EM22:         | .ASCIZ <200>/TRANSMITTER ERROR/                                |  |  |  |
|      | 033135 | 200    | 040510 | 043114 | EM23:         | .ASCIZ <200>/HALF DUPLEX TEST/                                 |  |  |  |
|      | 033157 | 200    | 052517 | 020124 | EM24:         | .ASCIZ <200>/OUT READY NOT SET/                                |  |  |  |
|      | 033202 | 044600 | 020116 | 042522 | EM25:         | .ASCIZ <200>/IN READY NOT SET/                                 |  |  |  |
|      | 033224 | 042600 | 050130 | 041505 | DH1:          | .ASCIZ <200>/EXPECTED FOUND/                                   |  |  |  |
|      | 033245 | 200    | 054105 | 042520 | DH2:          | .ASCIZ <200>/EXPECTED FOUND LU-REGISTER/                       |  |  |  |
|      | 033303 | 200    | 044103 | 051101 | DH3:          | .ASCIZ <200>/CHARACTER BIT THAT FAILED/                        |  |  |  |
|      | 033341 | 200    | 047503 | 051122 | DH4:          | .ASCIZ <200>/CORRECT CRC BIT THAT FAILED/                      |  |  |  |
|      | 033401 | 200    | 054105 | 042520 | DH5:          | .ASCIZ <200>/EXPECTED FOUND SHIFT/                             |  |  |  |
|      | 033433 | 200    | 054105 | 042520 | DH6:          | .ASCIZ <200>/EXPECTED FOUND CHARACTER SHIFT/                   |  |  |  |
|      | 033501 | 200    | 046102 | 041517 | DH7:          | .ASCIZ <200>/BLOCK END NOT SET/                                |  |  |  |
|      | 033524 | 051200 | 051524 | 042040 | DH10:         | .ASCIZ <200>/RTS DID NOT CLEAR/                                |  |  |  |
|      | 033550 | 000002 |        |        | .EVEN         |  |  |  |  |
|      | 033552 | 003    | 007    |        | DT1:          | 2  |  |  |  |
|      | 033554 | 001274 |        |        | .BYTE         | 3,7  |  |  |  |
|      | 033556 | 003    | 002    |        | \$REG5        |  |  |  |  |
|      | 033560 | 001272 |        |        | .BYTE         | 3,2  |  |  |  |
|      |        |        |        |        | \$REG4        |  |  |  |  |

|        |        |     |       |        |      |
|--------|--------|-----|-------|--------|------|
| 033562 | 000003 |     | DT2:  | 3      |      |
| 033564 | 003    | 007 |       | .BYTE  | 3,7  |
| 033566 | 001274 |     |       | \$REG5 |      |
| 033570 | 003    | 010 |       | .BYTE  | 3,10 |
| 033572 | 001272 |     |       | \$REG4 |      |
| 033574 | 003    | 002 |       | .BYTE  | 3,2  |
| 033576 | 001266 |     |       | \$REG2 |      |
| 033600 | 000002 |     | DT3:  | 2      |      |
| 033602 | 003    | 017 |       | .BYTE  | 3,17 |
| 033604 | 001274 |     |       | \$REG5 |      |
| 033606 | 002    | 002 |       | .BYTE  | 2,2  |
| 033610 | 001270 |     |       | \$REG3 |      |
| 033612 | 000002 |     | DT4:  | 2      |      |
| 033614 | 006    | 021 |       | .BYTE  | 6,21 |
| 033616 | 031230 |     |       | CALBCC |      |
| 033620 | 002    | 002 |       | .BYTE  | 2,2  |
| 033622 | 001270 |     |       | \$REG3 |      |
| 033624 | 000003 |     | DT5:  | 3      |      |
| 033626 | 001    | 011 |       | .BYTE  | 1,11 |
| 033630 | 001462 |     |       | ZERO   |      |
| 033632 | 001    | 011 |       | .BYTE  | 1,11 |
| 033634 | 001464 |     |       | ONE    |      |
| 033636 | 002    | 002 |       | .BYTE  | 2,2  |
| 033640 | 001262 |     |       | \$REG0 |      |
| 033642 | 000003 |     | DT6:  | 3      |      |
| 033644 | 001    | 011 |       | .BYTE  | 1,11 |
| 033646 | 001464 |     |       | ONE    |      |
| 033650 | 001    | 011 |       | .BYTE  | 1,11 |
| 033652 | 001462 |     |       | ZERO   |      |
| 033654 | 002    | 002 |       | .BYTE  | 2,2  |
| 033656 | 001262 |     |       | \$REG0 |      |
| 033660 | 000004 |     | DT7:  | 4      |      |
| 033662 | 001    | 011 |       | .BYTE  | 1,11 |
| 033664 | 001462 |     |       | ZERO   |      |
| 033666 | 001    | 011 |       | .BYTE  | 1,11 |
| 033670 | 001464 |     |       | ONE    |      |
| 033672 | 003    | 007 |       | .BYTE  | 3,7  |
| 033674 | 001274 |     |       | \$REG5 |      |
| 033676 | 002    | 001 |       | .BYTE  | 2,1  |
| 033700 | 001270 |     |       | \$REG3 |      |
| 033702 | 000004 |     | DT10: | 4      |      |
| 033704 | 001    | 011 |       | .BYTE  | 1,11 |
| 033706 | 001464 |     |       | ONE    |      |
| 033710 | 001    | 011 |       | .BYTE  | 1,11 |
| 033712 | 001462 |     |       | ZERO   |      |
| 033714 | 003    | 007 |       | .BYTE  | 3,7  |
| 033716 | 001274 |     |       | \$REG5 |      |
| 033720 | 002    | 001 |       | .BYTE  | 2,1  |
| 033722 | 001270 |     |       | \$REG3 |      |
| 033724 | 000002 |     | DT11: | 2      |      |
| 033726 | 003    | 007 |       | .BYTE  | 3,7  |
| 033730 | 032006 |     |       | FLAG   |      |
| 033732 | 002    | 002 |       | .BYTE  | 2,2  |
| 033734 | 001270 |     |       | \$REG3 |      |
| 033736 | 000002 |     | DT12: | 2      |      |
| 033740 | 006    | 004 |       | .BYTE  | 6,4  |

CZKCF MACY11 30A(1052) 08-JUL-80 08:27 PAGE 121  
CZKCF.P11 08-JUL-80 08:24 SUBROUTINES

D 10

SEQ 0120

|        |        |     |        |     |
|--------|--------|-----|--------|-----|
| 033742 | 031230 |     | CALBCC |     |
| 033744 | 006    | 002 | .BYTE  | 6.2 |
| 033746 | 001302 |     | \$TMP2 |     |

|        |        |  |         |  |
|--------|--------|--|---------|--|
| 033750 |        |  | CORMAX: |  |
|        | 000001 |  | .END    |  |



CZKCF MACY11 3GA(1052) 08-JUL-80 08:27 PAGE 123  
 CZKCF.P11 08-JUL-80 08:24 CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0121

|                 |       |       |       |      |
|-----------------|-------|-------|-------|------|
| ABASE = 000000  | 952   | 993   |       |      |
| ACDW1 = 000000  | 952   | 995   |       |      |
| ACDW2 = 000000  | 952   | 996   |       |      |
| ACPUOP= 000000  | 952   | 967   |       |      |
| ADCW0 = 000000  | 952   | 997   |       |      |
| ADDW1 = 000000  | 952   | 998   |       |      |
| ADDW10= 000000  | 952   | 1007  |       |      |
| ADDW11= 000000  | 952   | 1008  |       |      |
| ADDW12= 000000  | 952   | 1009  |       |      |
| ADDW13= 000000  | 952   | 1010  |       |      |
| ADDW14= 000000  | 952   | 1011  |       |      |
| ADDW15= 000000  | 952   | 1012  |       |      |
| ADDW2 = 000000  | 952   | 999   |       |      |
| ADDW3 = 000000  | 952   | 1000  |       |      |
| ADDW4 = 000000  | 952   | 1001  |       |      |
| ADDW5 = 000000  | 952   | 1002  |       |      |
| ADDW6 = 000000  | 952   | 1003  |       |      |
| ADDW7 = 000000  | 952   | 1004  |       |      |
| ADDW8 = 000000  | 952   | 1005  |       |      |
| ADDW9 = 000000  | 952   | 1006  |       |      |
| ADEVCT= 000000  | 952   | 958   |       |      |
| ADEVM = 000000  | 952   | 994   |       |      |
| ADRCNT 006057   | 2060* | 2075* | 2084# |      |
| ADVANC= 104420  | 2229# | 5654  | 5737  |      |
| AENV = 000002   | 1#    | 952   | 963   |      |
| AENVM = 000000  | 952   | 964   |       |      |
| AFATAL= 000000  | 952   | 955   |       |      |
| AMADR1= 000000  | 952   | 980   |       |      |
| AMADR2= 000000  | 952   | 984   |       |      |
| AMADR3= 000000  | 952   | 987   |       |      |
| AMADR4= 000000  | 952   | 990   |       |      |
| AMAMS1= 000000  | 952   | 974   |       |      |
| AMAMS2= 000000  | 952   | 982   |       |      |
| AMAMS3= 000000  | 952   | 985   |       |      |
| AMAMS4= 000000  | 952   | 988   |       |      |
| AMSGAD= 000000  | 952   | 960   |       |      |
| AMSGLG= 000000  | 952   | 961   |       |      |
| AMSGTY= 000000  | 952   | 954   |       |      |
| AMTYP1= 000000  | 952   | 975   |       |      |
| AMTYP2= 000000  | 952   | 983   |       |      |
| AMTYP3= 000000  | 952   | 986   |       |      |
| AMTYP4= 000000  | 952   | 989   |       |      |
| APASS = 000000  | 952   | 957   |       |      |
| APRIOR= 000000  | 952   |       |       |      |
| APTC SU= 000040 | 1785  | 1890# |       |      |
| APTENV= 000001  | 1778  | 1846  | 1888# | 2290 |
| APTSIZ= 000200  | 1887# |       |       |      |
| APTSPO= 000100  | 1780  | 1848  | 1889# |      |
| APT.SI 013716   | 1453  | 2889# |       |      |
| ASWREG= 000000  | 952   | 965   |       |      |
| ATESTN= 000000  | 952   | 956   |       |      |
| AUDONE 003354   | 1490  | 1511  | 1550# |      |
| AUNIT = 000000  | 952   | 959   |       |      |
| AUSTRT 003126   | 1489# |       |       |      |
| AUSWR = 000000  | 952   | 966   |       |      |
| AL *O.S 012236  | 1451  | 2608# |       |      |

























|           |          |       |       |       |       |       |       |       |       |       |       |       |       |       |
|-----------|----------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| \$MSGTY   | 001316   | 954#  | 1852  | 1860* | 1872  | 1876* |       |       |       |       |       |       |       |       |
| \$MSWR    | 005541   | 1991# | 2468  |       |       |       |       |       |       |       |       |       |       |       |
| \$MTYP1   | 001347   | 975#  |       |       |       |       |       |       |       |       |       |       |       |       |
| \$MTYP2   | 001353   | 983#  |       |       |       |       |       |       |       |       |       |       |       |       |
| \$MTYP3   | 001357   | 986#  |       |       |       |       |       |       |       |       |       |       |       |       |
| \$MTYP4   | 001363   | 989#  |       |       |       |       |       |       |       |       |       |       |       |       |
| \$MXCNT   | 004362   | 1729  | 1739# |       |       |       |       |       |       |       |       |       |       |       |
| \$N       | = 000057 | 1#    | 2938  | 2944  | 2946  | 2953# | 2964  | 2970  | 2972  | 2979# | 2989  | 2995  | 2997  | 3004  |
|           |          | 3005# | 3015  | 3021  | 3023  | 3030  | 3031# | 3045  | 3051  | 3053  | 3061  | 3062# | 3088  | 3094  |
|           |          | 3096  | 3104  | 3105# | 3131  | 3137  | 3139  | 3147  | 3148# | 3190  | 3196  | 3198  | 3206  | 3207# |
|           |          | 3243  | 3249  | 3251  | 3258  | 3259# | 3267  | 3273  | 3275  | 3282  | 3283# | 3291  | 3297  | 3299  |
|           |          | 3306  | 3307# | 3326  | 3333  | 3335  | 3342  | 3343# | 3360  | 3367  | 3369  | 3376  | 3377# | 3405  |
|           |          | 3412  | 3414  | 3421  | 3422# | 3463  | 3471  | 3473  | 3480  | 3481# | 3516  | 3524  | 3526  | 3533  |
|           |          | 3534# | 3569  | 3577  | 3579  | 3586  | 3587# | 3622  | 3630  | 3632  | 3639  | 3640# | 3675  | 3684  |
|           |          | 3686  | 3693  | 3694# | 3738  | 3744  | 3746  | 3753  | 3754# | 3768  | 3774  | 3776  | 3783  | 3784# |
|           |          | 3798  | 3804  | 3806  | 3813  | 3814# | 3828  | 3834  | 3836  | 3843  | 3844# | 3858  | 3865  | 3867  |
|           |          | 3874  | 3875# | 3909  | 3915  | 3917  | 3924  | 3925# | 3948  | 3954  | 3956  | 3963  | 3964# | 3987  |
|           |          | 3993  | 3995  | 4002  | 4003# | 4026  | 4032  | 4034  | 4041  | 4042# | 4065  | 4071  | 4073  | 4080  |
|           |          | 4081# | 4106  | 4114  | 4116  | 4123  | 4124# | 4152  | 4160  | 4162  | 4169  | 4170# | 4199  | 4206  |
|           |          | 4208  | 4215  | 4216# | 4267  | 4273  | 4275  | 4282  | 4283# | 4321  | 4328  | 4330  | 4338  | 4339# |
|           |          | 4398  | 4405  | 4407  | 4415  | 4416# | 4475  | 4482  | 4484  | 4492  | 4493# | 4552  | 4559  | 4561  |
|           |          | 4569  | 4570# | 4629  | 4635  | 4637  | 4644  | 4645# | 4697  | 4703  | 4705  | 4712  | 4713# | 4765  |
|           |          | 4773  | 4775  | 4782  | 4783# | 4868  | 4875  | 4877  | 4884  | 4885# | 4927  | 4938  | 4940  | 4947  |
|           |          | 4948# | 5175  | 5187  | 5189  | 5196  | 5197# | 5455  | 5463  | 5465  | 5472  | 5473# | 5521  | 5527  |
|           |          | 5529  | 5536  | 5537# | 5558  | 5568  | 5570  | 5577  | 5578# | 5655  | 5664  | 5666  | 5673  | 5674# |
|           |          | 5738# |       |       |       |       |       |       |       |       |       |       |       |       |
|           |          | 927#  | 1805  | 1834  |       |       |       |       |       |       |       |       |       |       |
| \$NULL    | 001254   | 2948# | 2974# | 2999# | 3025# | 3055# | 3098# | 3141# | 3200# | 3253# | 3277# | 3301# | 3337# | 3371# |
| \$NWTST = | 000000   | 3416# | 3475# | 3528# | 3581# | 3634# | 3688# | 3748# | 3778# | 3808# | 3838# | 3869# | 3919# | 3958# |
|           |          | 3997# | 4036# | 4075# | 4118# | 4164# | 4210# | 4277# | 4332# | 4409# | 4486# | 4563# | 4639# | 4707# |
|           |          | 4777# | 4879# | 4942# | 5191# | 5467# | 5531# | 5572# | 5668# |       |       |       |       |       |
| \$OVER    | 004334   | 1701  | 1704  | 1715  | 1727  | 1733# |       |       |       |       |       |       |       |       |
| \$PASS    | 001324   | 957#  | 1625* | 1637  | 1649* | 1650* | 1667  | 1675  | 1723  | 1740  | 2545* |       |       |       |
| \$PASTM   | 002042   | 1154# |       |       |       |       |       |       |       |       |       |       |       |       |
| \$PWRDN   | 007126   | 865   | 1391  | 2326# | 2361  |       |       |       |       |       |       |       |       |       |
| \$PWRMG   | 007312   | 2364# |       |       |       |       |       |       |       |       |       |       |       |       |
| \$PWRUP   | 007200   | 2336  | 2342# |       |       |       |       |       |       |       |       |       |       |       |
| \$QUES    | 001312   | 945#  | 1834  | 1972  | 1989  | 2046  | 2049  | 2069  | 2592  | 2664  | 2680  | 2694  | 2714  |       |
| \$RDCHR   | 005144   | 1909# | 2215  |       |       |       |       |       |       |       |       |       |       |       |
| \$RDDEC = | ***** U  | 2218  |       |       |       |       |       |       |       |       |       |       |       |       |
| \$RDLIN   | 005264   | 1937# | 2216  |       |       |       |       |       |       |       |       |       |       |       |
| \$RDOCT   | 005564   | 2010# | 2217  |       |       |       |       |       |       |       |       |       |       |       |
| \$RDSZ =  | 000007   | 1930# |       |       |       |       |       |       |       |       |       |       |       |       |
| \$REGAD   | 001260   | 931#  |       |       |       |       |       |       |       |       |       |       |       |       |
| \$REG0    | 001262   | 933#  | 2105* | 2110  | 6168  |       |       |       |       |       |       |       |       |       |
| \$REG1    | 001264   | 934#  | 1525* | 1537  | 2104* | 2111  |       |       |       |       |       |       |       |       |
| \$REG2    | 001266   | 935#  | 2103* | 2112  | 6168  |       |       |       |       |       |       |       |       |       |
| \$REG3    | 001270   | 936#  | 2102* | 2113  | 6168  |       |       |       |       |       |       |       |       |       |
| \$REG4    | 001272   | 937#  | 2101* | 2114  | 6168  |       |       |       |       |       |       |       |       |       |
| \$REG5    | 001274   | 938#  | 2100* | 2115  | 6168  |       |       |       |       |       |       |       |       |       |
| \$RTNAD   | 004102   | 1666# |       |       |       |       |       |       |       |       |       |       |       |       |
| \$R2A =   | ***** U  | 2218  |       |       |       |       |       |       |       |       |       |       |       |       |
| \$S       | = 000061 | 1#    | 2951  | 2953# | 2977  | 2979# | 3002  | 3005# | 3028  | 3031# | 3058  | 3062# | 3101  | 3105# |
|           |          | 3144  | 3148# | 3203  | 3207# | 3256  | 3259# | 3280  | 3283# | 3304  | 3307# | 3340  | 3343# | 3374  |
|           |          | 3377# | 3419  | 3422# | 3478  | 3481# | 3531  | 3534# | 3584  | 3587# | 3637  | 3640# | 3691  | 3694# |
|           |          | 3751  | 3754# | 3781  | 3784# | 3811  | 3814# | 3841  | 3844# | 3872  | 3875# | 3922  | 3925# | 3961  |

CROSS REFERENCE TABLE -- USER SYMBOLS

|                  |       |       |       |       |       |       |       |       |       |       |       |       |       |
|------------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
|                  | 3964# | 4000  | 4003# | 4039  | 4042# | 4078  | 4081# | 4121  | 4124# | 4167  | 4170# | 4213  | 4216# |
|                  | 4280  | 4283# | 4335  | 4339# | 4412  | 4416# | 4489  | 4493# | 4566  | 4570# | 4642  | 4645# | 4710  |
|                  | 4713# | 4780  | 4783# | 4882  | 4885# | 4945  | 4948# | 5194  | 5197# | 5470  | 5473# | 5534  | 5537# |
|                  | 5575  | 5578# | 5671  | 5674# |       |       |       |       |       |       |       |       |       |
| \$SAVRE= ***** U | 2218  |       |       |       |       |       |       |       |       |       |       |       |       |
| \$SAVR6 007322   | 2335* | 2343  | 2344* | 2345* | 2368# |       |       |       |       |       |       |       |       |
| \$SCOPE 004134   | 863   | 1695# | 2869  |       |       |       |       |       |       |       |       |       |       |
| \$SETUP= 0C0000  | 1648  | 1696  | 1898  | 1995  |       |       |       |       |       |       |       |       |       |
| \$SVLAD 004316   | 1712  | 1730# |       |       |       |       |       |       |       |       |       |       |       |
| \$SVPC = 000040  | 875#  | 880   |       |       |       |       |       |       |       |       |       |       |       |
| \$SWR = 164000   | 1#    | 697   | 944   | 945   | 1620  | 1648  | 1659  | 1665  | 1667  | 1689  | 1690  | 1691  | 1692  |
|                  | 1703  | 1715  | 1717  | 1718  | 1719  | 1720  | 1721  | 1733  | 1739  | 2365  | 2950  | 2976  | 3001  |
|                  | 3027  | 3057  | 3100  | 3143  | 3202  | 3255  | 3279  | 3303  | 3339  | 3373  | 3418  | 3477  | 3530  |
|                  | 3583  | 3636  | 3690  | 3750  | 3780  | 3810  | 3840  | 3871  | 3921  | 3960  | 3999  | 4038  | 4077  |
|                  | 4120  | 4166  | 4212  | 4279  | 4334  | 4411  | 4488  | 4565  | 4641  | 4709  | 4779  | 4881  | 4944  |
|                  | 5193  | 5469  | 5533  | 5574  | 5670  |       |       |       |       |       |       |       |       |
| \$SWREG 001340   | 965#  | 1409  |       |       |       |       |       |       |       |       |       |       |       |
| \$SWRMK= 000000  | 1692  |       |       |       |       |       |       |       |       |       |       |       |       |
| \$TESTN 001322   | 956#  | 1731* |       |       |       |       |       |       |       |       |       |       |       |
| \$TIMES 001310   | 944#  | 1648* | 1720* | 1726  | 1729* | 1739  |       |       |       |       |       |       |       |
| \$TKB 001246     | 924#  | 1702  | 1896  | 1913  | 1919  | 2458  | 2460  | 2502  | 2882  |       |       |       |       |
| \$TKS 001244     | 923#  | 1700  | 1896  | 1911  | 1917  | 2500  | 2880  | 6142  |       |       |       |       |       |
| \$TMP0 001276    | 939#  | 1464* | 2431  | 2837* | 2838* | 5772* | 5784* | 5818* | 5829* | 5864* | 5875* | 5916* | 5922* |
|                  | 5934* | 5948* | 5968* | 5979* | 6014* | 6025* | 6064* | 6069* |       |       |       |       |       |
| \$TMP1 001300    | 940#  | 1465* | 2433  | 5125* | 5126* | 5127  | 5168* | 5169* | 5170  | 5394* | 5395* | 5396  | 5448* |
|                  | 5449* | 5450  | 5645* | 5646* | 5647  | 5731* | 5732* | 5733  | 5816* | 5832  | 5842* | 5845  | 5859* |
|                  | 5863* | 5890* | 5897* | 5935* | 5940* | 5966* | 5982  |       |       |       |       |       |       |
| \$TMP2 001302    | 941#  | 1467* | 1488* | 1515  | 1524* | 2435  | 2625  | 2628  | 2740* | 4912* | 4913* | 5111* | 5112* |
|                  | 5127* | 5128  | 5154* | 5155* | 5170* | 5171  | 5380* | 5381* | 5396* | 5397  | 5434* | 5435* | 5450* |
|                  | 5451  | 5631* | 5632* | 5647* | 5648  | 5717* | 5718* | 5733* | 5734  | 6094* | 6104* | 6117* | 6130* |
|                  | 6168  |       |       |       |       |       |       |       |       |       |       |       |       |
| \$TMP3 001304    | 942#  | 1468* | 2437  | 2637  | 2640  | 2645  | 2648  | 2727  | 2730  | 2735  | 2738  |       |       |
| \$TMP4 001306    | 943#  | 1469* | 2439  | 2620* | 2632* | 2823  | 5800* | 5806* |       |       |       |       |       |
| \$TN = 000060    | 1#    | 697   | 2948  | 2950# | 2974  | 2976# | 2999  | 3001# | 3025  | 3027# | 3055  | 3057# | 3098  |
|                  | 3100# | 3141  | 3143# | 3200  | 3202# | 3253  | 3255# | 3277  | 3279# | 3301  | 3303# | 3337  | 3339# |
|                  | 3371  | 3373# | 3416  | 3418# | 3475  | 3477# | 3528  | 3530# | 3581  | 3583# | 3634  | 3636# | 3688  |
|                  | 3690# | 3748  | 3750# | 3778  | 3780# | 3808  | 3810# | 3838  | 3840# | 3869  | 3871# | 3919  | 3921# |
|                  | 3958  | 3960# | 3997  | 3999# | 4036  | 4038# | 4075  | 4077# | 4118  | 4120# | 4164  | 4166# | 4210  |
|                  | 4212# | 4277  | 4279# | 4332  | 4334# | 4409  | 4411# | 4486  | 4488# | 4563  | 4565# | 4639  | 4641# |
|                  | 4707  | 4709# | 4777  | 4779# | 4879  | 4881# | 4942  | 4944# | 5191  | 5193# | 5467  | 5469# | 5531  |
|                  | 5533# | 5572  | 5574# | 5668  | 5670# |       |       |       |       |       |       |       |       |
| \$TPB 001252     | 926#  | 1823* | 1834  | 2241* | 2505* | 2885* |       |       |       |       |       |       |       |
| \$TPFLG 001257   | 930#  | 1772  | 1834  |       |       |       |       |       |       |       |       |       |       |
| \$TPS 001250     | 925#  | 1821  | 1834  | 2239  | 2503  | 2883  |       |       |       |       |       |       |       |
| \$TRAP 006414    | 869   | 2189# |       |       |       |       |       |       |       |       |       |       |       |
| \$TRAP2 006436   | 2200# | 2211  |       |       |       |       |       |       |       |       |       |       |       |
| \$TRP = 000021   | 2204# | 2213# | 2215  | 2216# | 2217# | 2218# | 2219# | 2220# | 2221# | 2222# | 2223# | 2224# | 2225# |
|                  | 2226# | 2227# | 2228# | 2229# | 2230# |       |       |       |       |       |       |       |       |
| \$TRPAD 006450   | 2194  | 2211# |       |       |       |       |       |       |       |       |       |       |       |
| \$STM 002040     | 1153# |       |       |       |       |       |       |       |       |       |       |       |       |
| \$STMM 001202    | 903#  | 1404* | 1688  | 1730* | 1731  | 1733  | 1740  | 2318  | 2372  | 2574  | 2581  | 2583  | 2950* |
|                  | 2976* | 3001* | 3027* | 3057* | 3100* | 3143* | 3202* | 3255* | 3279* | 3303* | 3339* | 3373* | 3418* |
|                  | 3477* | 3530* | 3583* | 3636* | 3690* | 3750* | 3780* | 3810* | 3840* | 3871* | 3921* | 3960* | 3999* |
|                  | 4038* | 4077* | 4120* | 4166* | 4212* | 4279* | 4334* | 4411* | 4488* | 4565* | 4641* | 4709* | 4779* |
|                  | 4881* | 4944* | 5193* | 5469* | 5533* | 5574* | 5670* |       |       |       |       |       |       |
| \$TTYIN 005520   | 1939  | 1940  | 1952  | 1970  | 1984  | 1988# |       |       |       |       |       |       |       |









|          |       |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|----------|-------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| \$STUFF  | 687#  |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| \$SWPAC  | 687#  | 3243 | 3267 |      |      |      |      |      |      |      |      |      |      |      |      |
| \$TCHAR  | 687#  | 4786 | 4885 | 4951 | 5200 |      |      |      |      |      |      |      |      |      |      |
| \$TCRC   | 687#  | 4765 | 4927 | 5175 |      |      |      |      |      |      |      |      |      |      |      |
| \$TRAMW  | 687#  | 4805 | 4974 | 5025 | 5224 | 5294 |      |      |      |      |      |      |      |      |      |
| \$TRAN1  | 687#  | 3326 | 3360 | 3405 |      |      |      |      |      |      |      |      |      |      |      |
| \$TSTN   | 1#    | 2946 | 2972 | 2997 | 3023 | 3053 | 3096 | 3139 | 3196 | 3251 | 3275 | 3299 | 3335 | 3369 | 3414 |
|          | 3473  | 3526 | 3579 | 3632 | 3686 | 3746 | 3776 | 3806 | 3836 | 3867 | 3917 | 3956 | 3995 | 4034 | 4073 |
|          | 4116  | 4162 | 4208 | 4275 | 4330 | 4407 | 4484 | 4561 | 4637 | 4705 | 4775 | 4877 | 4940 | 5189 | 5465 |
|          | 5529  | 5570 | 5666 |      |      |      |      |      |      |      |      |      |      |      |      |
| \$UPADD  | 1#    | 2347 |      |      |      |      |      |      |      |      |      |      |      |      |      |
| \$VARIA  | 1#    | 889  |      |      |      |      |      |      |      |      |      |      |      |      |      |
| \$WINDO  | 687#  | 3463 | 3516 | 3569 | 3622 |      |      |      |      |      |      |      |      |      |      |
| \$XZ     | 1#    | 2938 | 2944 | 2964 | 2970 | 2989 | 2995 | 3015 | 3021 | 3045 | 3051 | 3088 | 3094 | 3131 | 3137 |
|          | 3190  | 3196 | 3243 | 3249 | 3267 | 3273 | 3291 | 3297 | 3326 | 3333 | 3360 | 3367 | 3405 | 3412 | 3463 |
|          | 3471  | 3516 | 3524 | 3569 | 3577 | 3622 | 3630 | 3675 | 3684 | 3738 | 3744 | 3768 | 3774 | 3798 | 3804 |
|          | 3828  | 3834 | 3858 | 3865 | 3909 | 3915 | 3948 | 3954 | 3987 | 3993 | 4026 | 4032 | 4065 | 4071 | 4106 |
|          | 4114  | 4152 | 4160 | 4199 | 4206 | 4267 | 4273 | 4321 | 4328 | 4398 | 4405 | 4475 | 4482 | 4552 | 4559 |
|          | 4629  | 4635 | 4697 | 4703 | 4765 | 4773 | 4868 | 4875 | 4927 | 4938 | 5175 | 5187 | 5455 | 5463 | 5521 |
|          | 5527  | 5558 | 5568 | 5655 | 5664 |      |      |      |      |      |      |      |      |      |      |
| \$ZEROS  | 687#  |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| \$\$CMRE | 894#  | 933  | 934  | 935  | 936  | 937  | 938  |      |      |      |      |      |      |      |      |
| \$\$CMTM | 894#  | 939  | 940  | 941  | 942  | 943  |      |      |      |      |      |      |      |      |      |
| \$\$ESCA | 830#  |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| \$\$NEWT | 830#  | 2948 | 2974 | 2999 | 3025 | 3055 | 3098 | 3141 | 3200 | 3253 | 3277 | 3301 | 3337 | 3371 | 3416 |
|          | 3475  | 3528 | 3581 | 3634 | 3688 | 3748 | 3778 | 3808 | 3838 | 3869 | 3919 | 3958 | 3997 | 4036 | 4075 |
|          | 4118  | 4164 | 4210 | 4277 | 4332 | 4409 | 4486 | 4563 | 4639 | 4707 | 4777 | 4879 | 4942 | 5191 | 5467 |
|          | 5531  | 5572 | 5668 |      |      |      |      |      |      |      |      |      |      |      |      |
| \$\$SCOP | 1#    | 1679 |      |      |      |      |      |      |      |      |      |      |      |      |      |
| \$\$SET  | 2204# | 2215 | 2216 | 2217 | 2218 | 2219 | 2220 | 2221 | 2222 | 2223 | 2224 | 2225 | 2226 | 2227 | 2228 |
|          | 2229  |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| \$\$SKIP | 830#  |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| .EQUAT   | 1#    | 720  |      |      |      |      |      |      |      |      |      |      |      |      |      |
| .HEADE   | 1#    | 687  |      |      |      |      |      |      |      |      |      |      |      |      |      |
| .SETUP   | 1#    |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| .\$ACT1  | 1#    | 871  |      |      |      |      |      |      |      |      |      |      |      |      |      |
| .\$APT8  | 1#    | 949# |      |      |      |      |      |      |      |      |      |      |      |      |      |
| .\$APTH  | 1#    | 1135 |      |      |      |      |      |      |      |      |      |      |      |      |      |
| .\$APTY  | 1#    | 1834 |      |      |      |      |      |      |      |      |      |      |      |      |      |
| .\$CATC  | 1#    |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| .\$CMTA  | 1#    | 894  |      |      |      |      |      |      |      |      |      |      |      |      |      |
| .\$EOP   | 1#    | 1616 |      |      |      |      |      |      |      |      |      |      |      |      |      |
| .\$ERRO  | 1#    |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| .\$ERRT  | 1#    |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| .\$POME  | 1#    | 2322 |      |      |      |      |      |      |      |      |      |      |      |      |      |
| .\$RDOC  | 1#    | 1996 |      |      |      |      |      |      |      |      |      |      |      |      |      |
| .\$READ  | 1#    | 1893 |      |      |      |      |      |      |      |      |      |      |      |      |      |
| .\$SCOP  | 1#    | 1683 |      |      |      |      |      |      |      |      |      |      |      |      |      |
| .\$STRAP | 1#    | 2181 |      |      |      |      |      |      |      |      |      |      |      |      |      |
| .\$TYPE  | 1#    | 1755 |      |      |      |      |      |      |      |      |      |      |      |      |      |
| .\$TYPO  | 1#    |      |      |      |      |      |      |      |      |      |      |      |      |      |      |

. ABS. 033750 000

CZKCF MACY11 30A(1052) 08-JUL-80 08:27 PAGE 142  
CZKCF.P11 08-JUL-80 08:24 CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0139

ERRORS DETECTED: 0

CZKCF,CZKCF/NL:TOC/SOL/CRF\_CZKCF.MAC,CZKCF.P11  
RUN-TIME: 29 25 1 SECONDS  
RUN-TIME RATIO: 73/56=1.2  
CORE USED: 53K (105 PAGES)