

KMC11-A

DDCMP - MD L UNIT TSTS
CZKCEB0

AH-A911B-MC
FICHE 1 OF 1

NOV 1980
COPYRIGHT © 77-80
MADE IN USA



A large grid of data tables, likely test results, covering the majority of the page. Each cell in the grid contains a small table with multiple columns and rows of data, including various alphanumeric characters and symbols.



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

.REM \$

IDENTIFICATION

PRODUCT CODE: AC-A910B-MC
PRODUCT NAME: CZKCEBO DDCMP MD L UNIT 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

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.

DZKCE 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 DDCMP MODE. THE MODEM SIGNALS ARE ALSO CHECKED. DZKCE 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
3. DZKCE [REV] DDCMP LINE UNIT TESTS
4. DZKCF [REV] BITSTUFF LINE UNIT TESTS
5. DZKCA [REV] KMC11 CPU MICRO-DIAGNOSTICS

NOTE: AS UPDATES OCCUR, THE NAMES OF THESE DIAGNOSTICS MAY VARIE.

2. REQUIREMENTS

2.1 EQUIPMENT

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

90
91
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

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)

133
134
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

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
(ONLY IF M8201 AND LOOP BACK)
WHICH MODEM TYPE, TYPE 'D' FOR KMC11-DA (RS232C), 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.

187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214

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.

215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257

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.

258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305

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 ENABELED; AND THERE IS A HARD ERROR (CONSTANT); SW08 IS BEST. (SW14=1,0, SW10=0, SW09=0, SW08=1). FOR INTERMITTEMT 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 INTERATIONS.
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

306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353

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.

354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406

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 DZKCE 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:.

407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462

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) 1500/0000000001000000 MEANS THAT KMC11 NO.06 IS THE KMC11 NOW RUNNING.

KMCR00-KMCR17
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) 1470/0000000000011111 MEANS THAT KMC11 NO. 00,01,02,03,04 WILL BE TESTED. EXAMPLE: (KMACTV) 1470/0000000000010001 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 |

463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488

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 BOOT ADD)
- STAT3: BIT 2=0 KMC11-DA
BIT 2=1 KMC11-FA

489
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, IF NOT, THE ADDRESS IS UPDATED BY 10 AND THE SEARCH CONTINUES. A -1 INDICATES A KMC11 WITH NO CRAM, AND 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
586
587

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 56 TEST 57]

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

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

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.

\$


```

686 .TITLE CZKCE
687 ;*COPYRIGHT (C) 1976
688 ;*DIGITAL EQUIPMENT CORP.
689 ;*MAYNARD, MASS. 01754
690 ;*
691 ;*PROGRAM BY DINESH GORADIA
692 ;*
693 ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
694 ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
695 ;*
696
697
698
699
700
701 ;*CZKCE KMC11 DDCMP LINE UNIT TESTS
702 ;*COPYRIGHT 1976, DIGITAL EQUIPMENT CORP., MAYNARD, MASS. 01754
703 ;*-----
704
705 ;STARTING PROCEDURE
706 ;LOAD PROGRAM
707 ;LOAD ADDRESS 000200
708 ;SWR=0 AUTOSIZE KMC11
709 ;SW07=1 USE CURRENT KMC11 PARAMETERS
710 ;SW00=1 INPUT NEW KMC11 PARAMETERS
711 ;PRESS START
712 ;PROGRAM WILL TYPE "CZKCE KMC11 DDCMP LINE UNIT TESTS"
713 ;PROGRAM WILL TYPE STATUS MAP
714 ;PROGRAM WILL TYPE "R" TO INDICATE THAT TESTING HAS STARTED
715 ;AT THE END OF A PASS, PROGRAM WILL TYPE PASS COMPLETE MESSAGE
716 ;AND THEN RESUME TESTING
717 ;SUBSEQUENT RESTARTS WILL NOT TYPE PROGRAM TITLE
718
719 .SBTTL BASIC DEFINITIONS
720
721 ;*INITIAL ADDRESS OF THE STACK POINTER *** 1200 ***
722 001200 STACK= 1200
723 .EQUIV EMT,ERROR ;;BASIC DEFINITION OF ERROR CALL
724 .EQUIV IOT,SCOPE ;;BASIC DEFINITION OF SCOPE CALL
725
726 ;*MISCELLANEOUS DEFINITIONS
727 000011 HT= 11 ;;CODE FOR HORIZONTAL TAB
728 000012 LF= 12 ;;CODE FOR LINE FEED
729 000015 CR= 15 ;;CODE FOR CARRIAGE RETURN
730 000200 CRLF= 200 ;;CODE FOR CARRIAGE RETURN-LINE FEED
731 177776 PS= 177776 ;;PROCESSOR STATUS WORD
732 .EQUIV PS,PSW
733 177774 STKLMT= 177774 ;;STACK LIMIT REGISTER
734 177772 PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
735 177570 DSWR= 177570 ;;HARDWARE SWITCH REGISTER
736 177570 DDISP= 177570 ;;HARDWARE DISPLAY REGISTER
737
738 ;*GENERAL PURPOSE REGISTER DEFINITIONS
739 000000 R0= %0 ;;GENERAL REGISTER
740 000001 R1= %1 ;;GENERAL REGISTER
741 000002 R2= %2 ;;GENERAL REGISTER

```

| | | | | |
|-----|--------|--|----------|--------------------|
| 742 | 000003 | R3= | %3 | ::GENERAL REGISTER |
| 743 | 000004 | R4= | %4 | ::GENERAL REGISTER |
| 744 | 000005 | R5= | %5 | ::GENERAL REGISTER |
| 745 | 000006 | R6= | %6 | ::GENERAL REGISTER |
| 746 | 000007 | R7= | %7 | ::GENERAL REGISTER |
| 747 | 000006 | SP= | %6 | ::STACK POINTER |
| 748 | 000007 | PC= | %7 | ::PROGRAM COUNTER |
| 749 | | | | |
| 750 | | ;*PRIORITY LEVEL DEFINITIONS | | |
| 751 | 000000 | PR0= | 0 | ::PRIORITY LEVEL 0 |
| 752 | 000040 | PR1= | 40 | ::PRIORITY LEVEL 1 |
| 753 | 000100 | PR2= | 100 | ::PRIORITY LEVEL 2 |
| 754 | 000140 | PR3= | 140 | ::PRIORITY LEVEL 3 |
| 755 | 000200 | PR4= | 200 | ::PRIORITY LEVEL 4 |
| 756 | 000240 | PR5= | 240 | ::PRIORITY LEVEL 5 |
| 757 | 000300 | PR6= | 300 | ::PRIORITY LEVEL 6 |
| 758 | 000340 | PR7= | 340 | ::PRIORITY LEVEL 7 |
| 759 | | | | |
| 760 | | ;*''SWITCH REGISTER'' SWITCH DEFINITIONS | | |
| 761 | 100000 | SW15= | 100000 | |
| 762 | 040000 | SW14= | 40000 | |
| 763 | 020000 | SW13= | 20000 | |
| 764 | 010000 | SW12= | 10000 | |
| 765 | 004000 | SW11= | 4000 | |
| 766 | 002000 | SW10= | 2000 | |
| 767 | 001000 | SW09= | 1000 | |
| 768 | 000400 | SW08= | 400 | |
| 769 | 000200 | SW07= | 200 | |
| 770 | 000100 | SW06= | 100 | |
| 771 | 000040 | SW05= | 40 | |
| 772 | 000020 | SW04= | 20 | |
| 773 | 000010 | SW03= | 10 | |
| 774 | 000004 | SW02= | 4 | |
| 775 | 000002 | SW01= | 2 | |
| 776 | 000001 | SW00= | 1 | |
| 777 | | .EQUIV | SW09,SW9 | |
| 778 | | .EQUIV | SW08,SW8 | |
| 779 | | .EQUIV | SW07,SW7 | |
| 780 | | .EQUIV | SW06,SW6 | |
| 781 | | .EQUIV | SW05,SW5 | |
| 782 | | .EQUIV | SW04,SW4 | |
| 783 | | .EQUIV | SW03,SW3 | |
| 784 | | .EQUIV | SW02,SW2 | |
| 785 | | .EQUIV | SW01,SW1 | |
| 786 | | .EQUIV | SW00,SW0 | |
| 787 | | | | |
| 788 | | ;*DATA BIT DEFINITIONS (BIT00 TO BIT15) | | |
| 789 | 100000 | BIT15= | 100000 | |
| 790 | 040000 | BIT14= | 40000 | |
| 791 | 020000 | BIT13= | 20000 | |
| 792 | 010000 | BIT12= | 10000 | |
| 793 | 004000 | BIT11= | 4000 | |
| 794 | 002000 | BIT10= | 2000 | |
| 795 | 001000 | BIT09= | 1000 | |
| 796 | 000400 | BIT08= | 400 | |
| 797 | 000200 | BIT07= | 200 | |

```
798          000100          BIT06= 100
799          000040          BIT05= 40
800          000020          BIT04= 20
801          000010          BIT03= 10
802          000004          BIT02= 4
803          000002          BIT01= 2
804          000001          BIT00= 1
805          .EQUIV BIT09,BIT9
806          .EQUIV BIT08,BIT8
807          .EQUIV BIT07,BIT7
808          .EQUIV BIT06,BIT6
809          .EQUIV BIT05,BIT5
810          .EQUIV BIT04,BIT4
811          .EQUIV BIT03,BIT3
812          .EQUIV BIT02,BIT2
813          .EQUIV BIT01,BIT1
814          .EQUIV BIT00,BIT0
815
816          ;*BASIC "CPU" TRAP VECTOR ADDRESSES
817          000004          ERRVEC= 4          ;;TIME OUT AND OTHER ERRORS
818          000010          RESVEC= 10         ;;RESERVED AND ILLEGAL INSTRUCTIONS
819          000014          TBITVEC=14        ;;"T" BIT
820          000014          TRTVEC= 14         ;;TRACE TRAP
821          000014          BPTVEC= 14        ;;BREAKPOINT TRAP (BPT)
822          000020          IOTVEC= 20        ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
823          000024          PWRVEC= 24         ;;POWER FAIL
824          000030          EMTVEC= 30        ;;EMULATOR TRAP (EMT) **ERROR**
825          000034          TRAPVEC=34        ;;"TRAP" TRAP
826          000060          TKVEC= 60         ;;TTY KEYBOARD VECTOR
827          000064          TPVEC= 64         ;;TTY PRINTER VECTOR
828          000240          PIRQVEC=240       ;;PROGRAM INTERRUPT REQUEST VECTOR
829
830
831
832
833          ;INSTRUCTION DEFINITIONS
834          ;-----
835
836          005746          PUSH1SP=5746       ;DECREMENT PROCESSOR STACK 1 WORD
837          005726          POP1SP=5726        ;INCREMENT PROCESSOR STACK 1 WORD
838          010046          PUSHRO=10046       ;SAVE R0 ON STACK
839          012600          POPRO=12600        ;RESTORE R0 FROM STACK
840          024646          PUSH2SP=24646     ;DECREMENT STACK TWICE
841          022626          POP2SP=22626      ;INCREMENT STACK TWICE
842          .EQUIV EMT,HLT ;BASIC DEFINITION OF ERROR CALL
843
844
845
```

```

846
847
848
849
850
851
852
853
854
855
856      000000      000000
857      000000      000000      000000
858
859
860
861      000020      000020
862      000020      004134
863      000022      000340
864      000024      007126
865      000026      000340
866      000030      006512
867      000032      000340
868      000034      006414
869      000036      000340
870
871
872
873
874
875
876      000046      000040
877      000046      004070
878      000052      000052
879      000052      000000
880
881
882      000174      000174
883      000174      000000
884      000176      000000
885
886      000200      000200      002402
887
888
889
890      001000      005200      055103      041513
(2)      001010      046513      030503      020061
(2)
891      177570
892      177570

```

```

;*****
;-----
;TRAPCATCAER FOR ILLEGAL INTERRUPTS
;THE STANDARD "TRAP CATCHER" IS PLACED
;BETWEEN ADDRESS 0 TO ADDRESS 776.
;IT LOOKS LIKE "PC+2 HALT".
;-----
;*****
.=0
      .WORD      0,0
;STANDARD INTERRUPT VECTORS
;-----
.=20
      $SCOPE      ; SCOPE LOOP HANDLER.
      PR7         ; SERVICE AT LEVEL 7.
      $PWRDN      ; POWER FAIL HANDLER
      PR7         ; SERVICE AT LEVEL 7
      $ERROR      ; ERROR HANDLER
      PR7         ; SERVICE AT LEVEL 7
      $STRAP      ; GENERAL HANDLER DISPATCH SERVICE
      PR7         ; SERVICE AT LEVEL 7
.SBTTL  ACT11 HOOKS
;*****
;HOOKS REQUIRED BY ACT11
      $SVPC=.      ;SAVE PC
      .=46
      $ENDAD      ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
      .=52
      .WORD      0  ;;2)SET LOC.52 TO ZERO
      .=$SVPC     ;; RESTORE PC
.=174
DISPREG:0      ;SOFTWARE DISPLAY REGISTER
SWREG: 0      ;SOFTWARE SWITCH REGISTER
.=200
      JMP      .START      ;GO TO START OF PROGRAM
.=1000
MTITLE: .ASCII <200><12>/CZKCE/<200>
        .ASCII /KMC11 DDCMP LINE UNIT TESTS/<200>
DSWR =      177570
DDISP =     177570

```

893
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

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.

SCMTAG: .=1200

;; START OF COMMON TAGS
 .WORD 0
 \$TSTNM: .BYTE 0
 \$ERFLG: .BYTE 0
 \$!CNT: .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
 .WORD 0
 \$AUTOB: .BYTE 0
 \$INTAG: .BYTE 0
 .WORD 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>

;; 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
 ;; USER DEFINED
 ;; MAX. NUMBER OF ITERATIONS
 ;; QUESTION MARK
 ;; CARRIAGE RETURN
 ;; LINE FEED

 .SBTTL APT MAILBOX-ETABLE

```

949
950 *****
951 .EVEN
952 001316 $MAIL:          ;;APT MAILBOX
953 001316 000000 $MSGTY: .WORD  AMSGTY ;;MESSAGE TYPE CODE
954 001320 000000 $FATAL: .WORD  AFATAL ;;FATAL ERROR NUMBER
955 001322 000000 $TESTN: .WORD  ATESTN ;;TEST NUMBER
956 001324 000000 $PASS:  .WORD  APASS  ;;PASS COUNT
957 001326 000000 $DEVCT: .WORD  ADEVCT ;;DEVICE COUNT
958 001330 000000 $UNIT:  .WORD  AUNIT  ;;I/O UNIT NUMBER
959 001332 000000 $MSGAD: .WORD  AMSGAD ;;MESSAGE ADDRESS
960 001334 000000 $MSGLG: .WORD  AMSGLG ;;MESSAGE LENGTH
961 001336 $ETABLE:      ;;APT ENVIRONMENT TABLE
962 001336      002 $ENV:  .BYTE  AENV   ;;ENVIRONMENT BYTE
963 001337      000 $ENVM: .BYTE  AENVM  ;;ENVIRONMENT MODE BITS
964 001340 000000 $SWREG: .WORD  ASWREG ;;APT SWITCH REGISTER
965 001342 000000 $USWR:  .WORD  AUSWR  ;;USER SWITCHES
966 001344 000000 $CPUOP: .WORD  ACPUOP ;;CPU TYPE,OPTIONS
967          ;;BITS 15-11=CPU TYPE
968          ;;      11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
969          ;;      11/70=06,PDQ=07,Q=10
970          ;;BIT 10=REAL TIME CLOCK
971          ;;BIT 9=FLOATING POINT PROCESSOR
972          ;;BIT 8=MEMORY MANAGEMENT
973 001346      000 $MAMS1: .BYTE  AMAMS1 ;;HIGH ADDRESS,M.S. BYTE
974 001347      000 $MTYP1: .BYTE  AMTYP1 ;;MEM. TYPE,BLK#1
975          ;;MEM.TYPE BYTE -- (HIGH BYTE)
976          ;;      900 NSEC CORE=001
977          ;;      300 NSEC BIPOLAR=002
978          ;;      500 NSEC MOS=003
979 001350 000000 $MADR1: .WORD  AMADR1 ;;HIGH ADDRESS,BLK#1
980          ;;MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF 'TYPE' ABOVE
981 001352      000 $MAMS2: .BYTE  AMAMS2 ;;HIGH ADDRESS,M.S. BYTE
982 001353      000 $MTYP2: .BYTE  AMTYP2 ;;MEM.TYPE,BLK#2
983 001354 000000 $MADR2: .WORD  AMADR2 ;;MEM.LAST ADDRESS,BLK#2
984 001356      000 $MAMS3: .BYTE  AMAMS3 ;;HIGH ADDRESS,M.S.BYTE
985 001357      000 $MTYP3: .BYTE  AMTYP3 ;;MEM.TYPE,BLK#3
986 001360 000000 $MADR3: .WORD  AMADR3 ;;MEM.LAST ADDRESS,BLK#3
987 001362      000 $MAMS4: .BYTE  AMAMS4 ;;HIGH ADDRESS,M.S.BYTE
988 001363      000 $MTYP4: .BYTE  AMTYP4 ;;MEM.TYPE,BLK#4
989 001364 000000 $MADR4: .WORD  AMADR4 ;;MEM.LAST ADDRESS,BLK#4
990 001366 000000 $VECT1: .WORD  AVECT1 ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
991 001370 000000 $VECT2: .WORD  AVECT2 ;;INTERRUPT VECTOR#2BUS PRIORITY#2
992 001372 000000 $BASE:  .WORD  ABASE  ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
993 001374 000000 $DEVN:  .WORD  ADEVN  ;;DEVICE MAP
994 001376 000000 $CDW1:  .WORD  ACDW1  ;;CONTROLLER DESCRIPTION WORD#1
995 001400 000000 $CDW2:  .WORD  ACDW2  ;;CONTROLLER DESCRIPTION WORD#2
996 001402 000000 $DDW0:  .WORD  ADDW0  ;;DEVICE DESCRIPTOR WORD#0
997 001404 000000 $DDW1:  .WORD  ADDW1  ;;DEVICE DESCRIPTOR WORD#1
998 001406 000000 $DDW2:  .WORD  ADDW2  ;;DEVICE DESCRIPTOR WORD#2
999 001410 000000 $DDW3:  .WORD  ADDW3  ;;DEVICE DESCRIPTOR WORD#3
1000 001412 000000 $DDW4:  .WORD  ADDW4  ;;DEVICE DESCRIPTOR WORD#4
1001 001414 000000 $DDW5:  .WORD  ADDW5  ;;DEVICE DESCRIPTOR WORD#5
1002 001416 000000 $DDW6:  .WORD  ADDW6  ;;DEVICE DESCRIPTOR WORD#6
1003 001420 000000 $DDW7:  .WORD  ADDW7  ;;DEVICE DESCRIPTOR WORD#7
1004 001422 000000 $DDW8:  .WORD  ADDW8  ;;DEVICE DESCRIPTOR WORD#8

```

1005 001424 000000
 1006 001426 000000
 1007 001430 000000
 1008 001432 000000
 1009 001434 000000
 1010 001436 000000
 1011 001440 000000
 1012
 1013
 1014 001442
 1015
 1016
 1017
 1018
 1019 001442 000000
 1020 001444 000000
 1021
 1022
 1023
 1024 001446 000000
 1025 001450 000000
 1026 001452 000000
 1027 001454 000000
 1028 001456 000000
 1029 001460 000000
 1030 001462 000000
 1031 001464 000001
 1032 001466 000000
 1033 001470 000001
 1034 001472 000001
 1035 001474 000001
 1036 001476 000001
 1037 001500 000000
 1038
 1039 001502 002072
 1040 001504 002276
 1041
 1042
 1043
 1044 001506 000
 1045 001510 001510
 1046 001510 000
 1047 001511 000
 1048
 1049

\$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

SETEND:

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
 .EVEN ; ON FIRST PASS OF EACH KMC11 ITERATIONS WILL BE SUPPRES

```

1050      .SBTTL  ERROR POINTER TABLE
1051
1052      ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
1053      ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
1054      ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
1055      ;*NOTE1:      IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
1056      ;*NOTE2:      EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
1057
1058      ;*      EM      ;;POINTS TO THE ERROR MESSAGE
1059      ;*      DH      ;;POINTS TO THE DATA HEADER
1060      ;*      DT      ;;POINTS TO THE DATA
1061      ;*      DF      ;;POINTS TO THE DATA FORMAT
1062
1063
1064      001512      $ERRTB:
1065      .EVEN
1066      ;*      DF      ;; DOES NOT APPLY IN THIS DIAGNOSTIC.
1067      001512      000000      0
1068      001514      000000      0
1069      001516      000000      0
1070      001520      032062      EM1
1071      001522      033225      DH2      ; ERROR 1
1072      001524      033542      DT2
1073      001526      032120      EM2
1074      001530      033225      DH2      ; ERROR 2
1075      001532      033542      DT2
1076      001534      032163      EM3
1077      001536      033225      DH2      ; ERROR 3
1078      001540      033542      DT2
1079      001542      032227      EM4
1080      001544      000000      0      ; ERROR 4
1081      001546      000000      0
1082      001550      032271      EM5
1083      001552      033225      DH2      ; ERROR 5
1084      001554      033542      DT2
1085      001556      032271      EM5
1086      001560      033263      DH3      ; ERROR 6
1087      001562      033560      DT3
1088      001564      032321      EM6
1089      001566      033204      DH1      ; ERROR 7
1090      001570      033530      DT1
1091      001572      032340      EM7
1092      001574      033204      DH1      ; ERROR 10
1093      001576      033530      DT1
1094      001600      032365      EM10
1095      001602      033204      DH1      ; ERROR 11
1096      001604      033530      DT1
1097      001606      032550      EM11
1098      001610      033361      DH5      ; ERROR 12
1099      001612      033604      DT5
1100      001614      032577      EM12
1101      001616      033361      DH5      ; ERROR 13
1102      001620      033604      DT5
1103      001622      032550      EM11
1104      001624      033321      DH4      ; ERROR 14
1105      001626      033572      DT4
  
```


| | | | | |
|------|--------|--------|------|------------|
| 1106 | 001630 | 032623 | EM13 | |
| 1107 | 001632 | 000000 | 0 | ; ERROR 15 |
| 1108 | 001634 | 000000 | 0 | |
| 1109 | 001636 | 032550 | EM11 | |
| 1110 | 001640 | 033361 | DH5 | ; ERROR 16 |
| 1111 | 001642 | 033622 | DT6 | |
| 1112 | 001644 | 032577 | EM12 | |
| 1113 | 001646 | 033361 | DH5 | ; ERROR 17 |
| 1114 | 001650 | 033622 | DT6 | |
| 1115 | 001652 | 032550 | EM11 | |
| 1116 | 001654 | 033413 | DH6 | ; ERROR 20 |
| 1117 | 001656 | 033640 | DT7 | |
| 1118 | 001660 | 032550 | EM11 | |
| 1119 | 001662 | 033413 | DH6 | ; ERROR 21 |
| 1120 | 001664 | 033662 | DT10 | |
| 1121 | 001666 | 032577 | EM12 | |
| 1122 | 001670 | 033413 | DH6 | ; ERROR 22 |
| 1123 | 001672 | 033640 | DT7 | |
| 1124 | 001674 | 032577 | EM12 | |
| 1125 | 001676 | 033413 | DH6 | ; ERROR 23 |
| 1126 | 001700 | 033662 | DT10 | |
| 1127 | 001702 | 032663 | EM14 | |
| 1128 | 001704 | 000000 | 0 | ; ERROR 24 |
| 1129 | 001706 | 000000 | 0 | |
| 1130 | 001710 | 032733 | EM15 | |
| 1131 | 001712 | 033204 | DH1 | ; ERROR 25 |
| 1132 | 001714 | 033530 | DT1 | |
| 1133 | 001716 | 032754 | EM16 | |
| 1134 | 001720 | 033263 | DH3 | ; ERROR 16 |
| 1135 | 001722 | 033704 | DT11 | |
| 1136 | 001724 | 032577 | EM12 | |
| 1137 | 001726 | 033204 | DH1 | ; ERROR 27 |
| 1138 | 001730 | 033716 | DT12 | |
| 1139 | 001732 | 032770 | EM17 | |
| 1140 | 001734 | 000000 | 0 | ; ERROR 30 |
| 1141 | 001736 | 000000 | 0 | |
| 1142 | 001740 | 033034 | EM20 | |
| 1143 | 001742 | 033204 | DH1 | ; ERROR 31 |
| 1144 | 001744 | 033530 | DT1 | |
| 1145 | 001746 | 033055 | EM21 | |
| 1146 | 001750 | 033461 | DH7 | ; ERROR 32 |
| 1147 | 001752 | 000000 | 0 | |
| 1148 | 001754 | 033055 | EM21 | |
| 1149 | 001756 | 033263 | DH3 | ; ERROR 33 |
| 1150 | 001760 | 033560 | DT3 | |
| 1151 | 001762 | 033072 | EM22 | |
| 1152 | 001764 | 033504 | DH10 | ; ERROR 34 |
| 1153 | 001766 | 000000 | 0 | |
| 1154 | 001770 | 033115 | EM23 | |
| 1155 | 001772 | 033225 | DH2 | ; ERROR 35 |
| 1156 | 001774 | 033542 | DT2 | |
| 1157 | 001776 | 033137 | EM24 | |
| 1158 | 002000 | 000000 | 0 | ; ERROR 36 |
| 1159 | 002002 | 000000 | 0 | |
| 1160 | 002004 | 033162 | EM25 | |
| 1161 | 002006 | 000000 | 0 | ; ERROR 37 |

1162 002010 000000
 1163 002012 032321
 1164 002014 033225
 1165 002016 033542
 1166 002020 032271
 1167 002022 033361
 1168 002024 033604
 1169 002026 032623
 1170 002030 033204
 1171 002032 033530
 1172 002034
 1173
 1174
 1175
 1176
 1177
 1178 002034
 1179 000024 000024
 1180 000024 000200
 1181 000044 000044
 1182 000044 002034
 1183 002034
 1184
 1185
 1186
 1187
 1188 002034
 1189 002034 000000
 1190 002036 001316
 1191 002040 000132
 1192 002042 000137
 1193 002044 000137
 1194 002046 000052
 1195

```

0
EM6
DH2          ; ERROR 40
DT2
EM5
DH5          ; ERROR 41
DT5
EM13
DH1          ; ERROR 42
DT1

.=2034
.SBTTL  APT PARAMETER BLOCK

:*****
:SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
:*****
.SX=.      ;;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
.=.SX     ;;RESET LOCATION COUNTER
:*****
:SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
:INTERFACE SPEC.

$APTHD:
$HIBTS: .WORD 0      ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
$MBADR: .WORD $MAIL  ;;ADDRESS OF APT MAILBOX (BITS 0-15)
$TSTM:  .WORD 90     ;;RUN TIM OF LONGEST TEST
$PASTM: .WORD 95     ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
$UNITM: .WORD 95     ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
          .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
  
```

```

1196
1197 ;KMC11 CONTROL INDICATORS FOR CURRENT KMC11 UNDER TEST
1198 ;-----
1199
1200 002050 000000 STAT1: 0
1201 002052 000000 STAT2: 0
1202 002054 000000 STAT3: 0
1203
1204 ;KMC11 VECTOR AND REGISTER INDIRECT POINTERS
1205 ;-----
1206
1207 002056 000000 KMRVEC: 0 ; POINTER TO KMC11 RECEIVER INTERRUPT VECTOR
1208 002060 000000 KMRLVL: 0 ; POINTER TO KMC11 RECEIVER INTERRUPT SERVICE PS
1209 002062 000000 KMTVEC: 0 ; POINTER TO KMC11 TRANSMITTER INTERRUPT VECTOR
1210 002064 000000 KMTLVL: 0 ; POINTER TO KMC11 TRANSMITTER INTERRUPT SERVICE PS
1211 002066 000000 KMCSR: 0 ; POINTER TO KMC11 CONTROL STATUS REGISTER
1212 002070 000000 KMCSRH: 0 ; POINTER TO KMC11 CONTROL STATUS REGISTER HIGH BYTE.
1213 002072 000000 KMCTL: 0 ; POINTER TO KMC11 CONTROL OUT REGISTER
1214 002074 000000 KMP04: 0 ; POINTER TO KMC11 PORT REGISTER(SEL 4)
1215 002076 000000 KMP06: 0 ; POINTER TO KMC11 PORT REGISTER(SEL 6)
1216
1217 ;TEMP STORAGE
1218 ;-----
1219
1220 ;TEMP: 0
1221 ;.=.+40
1222
1223 ;KMC11 STATUS TABLE AND ADDRESS ASSIGNMENTS
1224 ;-----
1225
1226 .=2100
1227 002100 KM.MAP:
1228 002100 000001 KMCR00: .BLKW 1 ; CONTROL STATUS REGISTER FOR KMC11 NUMBER 00
1229 002102 000001 KMS100: .BLKW 1 ; VECTOR FOR KMC11 NUMBER 00
1230 002104 000001 KMS200: .BLKW 1 ; DDCMP LINE# FOR KMC11 NUMBER 00
1231 002106 000001 KMS300: .BLKW 1 ; 3RD STATUS WORD
1232
1233 002110 000001 KMCR01: .BLKW 1 ; CONTROL STATUS REGISTER FOR KMC11 NUMBER 01
1234 002112 000001 KMS101: .BLKW 1 ; VECTOR FOR KMC11 NUMBER 01
1235 002114 000001 KMS201: .BLKW 1 ; DDCMP LINE# FOR KMC11 NUMBER 01
1236 002116 000001 KMS301: .BLKW 1 ; 3RD STATUS WORD
1237
1238 002120 000001 KMCR02: .BLKW 1 ; CONTROL STATUS REGISTER FOR KMC11 NUMBER 02
1239 002122 000001 KMS102: .BLKW 1 ; VECTOR FOR KMC11 NUMBER 02
1240 002124 000001 KMS202: .BLKW 1 ; DDCMP LINE# FOR KMC11 NUMBER 02
1241 002126 000001 KMS302: .BLKW 1 ; 3RD STATUS WORD
1242
1243 002130 000001 KMCR03: .BLKW 1 ; CONTROL STATUS REGISTER FOR KMC11 NUMBER 03
1244 002132 000001 KMS103: .BLKW 1 ; VECTOR FOR KMC11 NUMBER 03
1245 002134 000001 KMS203: .BLKW 1 ; DDCMP LINE# FOR KMC11 NUMBER 03
1246 002136 000001 KMS303: .BLKW 1 ; 3RD STATUS WORD
1247
1248 002140 000001 KMCR04: .BLKW 1 ; CONTROL STATUS REGISTER FOR KMC11 NUMBER 04
1249 002142 000001 KMS104: .BLKW 1 ; VECTOR FOR KMC11 NUMBER 04
1250 002144 000001 KMS204: .BLKW 1 ; DDCMP LINE# FOR KMC11 NUMBER 04
1251 002146 000001 KMS304: .BLKW 1 ; 3RD STATUS WORD

```

| | | | | | |
|------|--------|--------|---------------|---|--|
| 1252 | | | | | |
| 1253 | 002150 | 000001 | KMCR05: .BLKW | 1 | :CONTROL STATUS REGISTER FOR KMC11 NUMBER 05 |
| 1254 | 002152 | 000001 | KMS105: .BLKW | 1 | :VECTOR FOR KMC11 NUMBER 05 |
| 1255 | 002154 | 000001 | KMS205: .BLKW | 1 | :DDCMP LINE# FOR KMC11 NUMBER 05 |
| 1256 | 002156 | 000001 | KMS305: .BLKW | 1 | :3RD STATUS WORD |
| 1257 | | | | | |
| 1258 | 002160 | 000001 | KMCR06: .BLKW | 1 | :CONTROL STATUS REGISTER FOR KMC11 NUMBER 06 |
| 1259 | 002162 | 000001 | KMS106: .BLKW | 1 | :VECTOR FOR KMC11 NUMBER 06 |
| 1260 | 002164 | 000001 | KMS206: .BLKW | 1 | :DDCMP LINE# FOR KMC11 NUMBER 06 |
| 1261 | 002166 | 000001 | KMS306: .BLKW | 1 | :3RD STATUS WORD |
| 1262 | | | | | |
| 1263 | 002170 | 000001 | KMCR07: .BLKW | 1 | :CONTROL STATUS REGISTER FOR KMC11 NUMBER 07 |
| 1264 | 002172 | 000001 | KMS107: .BLKW | 1 | :VECTOR FOR KMC11 NUMBER 07 |
| 1265 | 002174 | 000001 | KMS207: .BLKW | 1 | :DDCMP LINE# FOR KMC11 NUMBER 07 |
| 1266 | 002176 | 000001 | KMS307: .BLKW | 1 | :3RD STATUS WORD |
| 1267 | | | | | |
| 1268 | 002200 | 000001 | KMCR10: .BLKW | 1 | :CONTROL STATUS REGISTER FOR KMC11 NUMBER 10 |
| 1269 | 002202 | 000001 | KMS110: .BLKW | 1 | :VECTOR FOR KMC11 NUMBER 10 |
| 1270 | 002204 | 000001 | KMS210: .BLKW | 1 | :DDCMP LINE# FOR KMC11 NUMBER 10 |
| 1271 | 002206 | 000001 | KMS310: .BLKW | 1 | :3RD STATUS WORD |
| 1272 | | | | | |
| 1273 | 002210 | 000001 | KMCR11: .BLKW | 1 | :CONTROL STATUS REGISTER FOR KMC11 NUMBER 11 |
| 1274 | 002212 | 000001 | KMS111: .BLKW | 1 | :VECTOR FOR KMC11 NUMBER 11 |
| 1275 | 002214 | 000001 | KMS211: .BLKW | 1 | :DDCMP LINE# FOR KMC11 NUMBER 11 |
| 1276 | 002216 | 000001 | KMS311: .BLKW | 1 | :3RD STATUS WORD |
| 1277 | | | | | |
| 1278 | 002220 | 000001 | KMCR12: .BLKW | 1 | :CONTROL STATUS REGISTER FOR KMC11 NUMBER 12 |
| 1279 | 002222 | 000001 | KMS112: .BLKW | 1 | :VECTOR FOR KMC11 NUMBER 12 |
| 1280 | 002224 | 000001 | KMS212: .BLKW | 1 | :DDCMP LINE# FOR KMC11 NUMBER 12 |
| 1281 | 002226 | 000001 | KMS312: .BLKW | 1 | :3RD STATUS WORD |
| 1282 | | | | | |
| 1283 | 002230 | 000001 | KMCR13: .BLKW | 1 | :CONTROL STATUS REGISTER FOR KMC11 NUMBER 13 |
| 1284 | 002232 | 000001 | KMS113: .BLKW | 1 | :VECTOR FOR KMC11 NUMBER 13 |
| 1285 | 002234 | 000001 | KMS213: .BLKW | 1 | :DDCMP LINE# FOR KMC11 NUMBER 13 |
| 1286 | 002236 | 000001 | KMS313: .BLKW | 1 | :3RD STATUS WORD |
| 1287 | | | | | |
| 1288 | 002240 | 000001 | KMCR14: .BLKW | 1 | :CONTROL STATUS REGISTER FOR KMC11 NUMBER 14 |
| 1289 | 002242 | 000001 | KMS114: .BLKW | 1 | :VECTOR FOR KMC11 NUMBER 14 |
| 1290 | 002244 | 000001 | KMS214: .BLKW | 1 | :DDCMP LINE# FOR KMC11 NUMBER 14 |
| 1291 | 002246 | 000001 | KMS314: .BLKW | 1 | :3RD STATUS WORD |
| 1292 | | | | | |
| 1293 | 002250 | 000001 | KMCR15: .BLKW | 1 | :CONTROL STATUS REGISTER FOR KMC11 NUMBER 15 |
| 1294 | 002252 | 000001 | KMS115: .BLKW | 1 | :VECTOR FOR KMC11 NUMBER 15 |
| 1295 | 002254 | 000001 | KMS215: .BLKW | 1 | :DDCMP LINE# FOR KMC11 NUMBER 15 |
| 1296 | 002256 | 000001 | KMS315: .BLKW | 1 | :3RD STATUS WORD |
| 1297 | | | | | |
| 1298 | 002260 | 000001 | KMCR16: .BLKW | 1 | :CONTROL STATUS REGISTER FOR KMC11 NUMBER 16 |
| 1299 | 002262 | 000001 | KMS116: .BLKW | 1 | :VECTOR FOR KMC11 NUMBER 16 |
| 1300 | 002264 | 000001 | KMS216: .BLKW | 1 | :DDCMP LINE# FOR KMC11 NUMBER 16 |
| 1301 | 002266 | 000001 | KMS316: .BLKW | 1 | :3RD STATUS WORD |
| 1302 | | | | | |
| 1303 | 002270 | 000001 | KMCR17: .BLKW | 1 | :CONTROL STATUS REGISTER FOR KMC11 NUMBER 17 |
| 1304 | 002272 | 000001 | KMS117: .BLKW | 1 | :VECTOR FOR KMC11 NUMBER 17 |
| 1305 | 002274 | 000001 | KMS217: .BLKW | 1 | :DDCMP LINE# FOR KMC11 NUMBER 17 |
| 1306 | 002276 | 000001 | KMS317: .BLKW | 1 | :3RD STATUS WORD |
| 1307 | | | | | |

CZKCE MACY11 30A(1052) 08-JUL-80 08:24 PAGE 29
CZKCE.P11 08-JUL-80 08:24 APT PARAMETER BLOCK

C 3

SEQ 0028

1308 002300 000000

KM.END: 000000

| | | | ;KMC11 PASS COUNT AND ERROR COUNT TABLE | |
|------|--------|--------|---|----------------------------------|
| | | | ;----- | |
| | | | CNT.MAP: | |
| 1309 | | | | |
| 1310 | | | | |
| 1311 | | | | |
| 1312 | | | | |
| 1313 | 002302 | | PACT00: 0 | ;PASS COUNT FOR KMC11 NUMBER 00 |
| 1314 | 002302 | 000000 | ERCT00: 0 | ;ERROR COUNT FOR KMC11 NUMBER 00 |
| 1315 | 002304 | 000000 | | |
| 1316 | | | | |
| 1317 | 002306 | 000000 | PACT01: 0 | ;PASS COUNT FOR KMC11 NUMBER 01 |
| 1318 | 002310 | 000000 | ERCT01: 0 | ;ERROR COUNT FOR KMC11 NUMBER 01 |
| 1319 | | | | |
| 1320 | 002312 | 000000 | PACT02: 0 | ;PASS COUNT FOR KMC11 NUMBER 02 |
| 1321 | 002314 | 000000 | ERCT02: 0 | ;ERROR COUNT FOR KMC11 NUMBER 02 |
| 1322 | | | | |
| 1323 | 002316 | 000000 | PACT03: 0 | ;PASS COUNT FOR KMC11 NUMBER 03 |
| 1324 | 002320 | 000000 | ERCT03: 0 | ;ERROR COUNT FOR KMC11 NUMBER 03 |
| 1325 | | | | |
| 1326 | 002322 | 000000 | PACT04: 0 | ;PASS COUNT FOR KMC11 NUMBER 04 |
| 1327 | 002324 | 000000 | ERCT04: 0 | ;ERROR COUNT FOR KMC11 NUMBER 04 |
| 1328 | | | | |
| 1329 | 002326 | 000000 | PACT05: 0 | ;PASS COUNT FOR KMC11 NUMBER 05 |
| 1330 | 002330 | 000000 | ERCT05: 0 | ;ERROR COUNT FOR KMC11 NUMBER 05 |
| 1331 | | | | |
| 1332 | 002332 | 000000 | PACT06: 0 | ;PASS COUNT FOR KMC11 NUMBER 06 |
| 1333 | 002334 | 000000 | ERCT06: 0 | ;ERROR COUNT FOR KMC11 NUMBER 06 |
| 1334 | | | | |
| 1335 | 002336 | 000000 | PACT07: 0 | ;PASS COUNT FOR KMC11 NUMBER 07 |
| 1336 | 002340 | 000000 | ERCT07: 0 | ;ERROR COUNT FOR KMC11 NUMBER 07 |
| 1337 | | | | |
| 1338 | 002342 | 000000 | PACT10: 0 | ;PASS COUNT FOR KMC11 NUMBER 10 |
| 1339 | 002344 | 000000 | ERCT10: 0 | ;ERROR COUNT FOR KMC11 NUMBER 10 |
| 1340 | | | | |
| 1341 | 002346 | 000000 | PACT11: 0 | ;PASS COUNT FOR KMC11 NUMBER 11 |
| 1342 | 002350 | 000000 | ERCT11: 0 | ;ERROR COUNT FOR KMC11 NUMBER 11 |
| 1343 | | | | |
| 1344 | 002352 | 000000 | PACT12: 0 | ;PASS COUNT FOR KMC11 NUMBER 12 |
| 1345 | 002354 | 000000 | ERCT12: 0 | ;ERROR COUNT FOR KMC11 NUMBER 12 |
| 1346 | | | | |
| 1347 | 002356 | 000000 | PACT13: 0 | ;PASS COUNT FOR KMC11 NUMBER 13 |
| 1348 | 002360 | 000000 | ERCT13: 0 | ;ERROR COUNT FOR KMC11 NUMBER 13 |
| 1349 | | | | |
| 1350 | 002362 | 000000 | PACT14: 0 | ;PASS COUNT FOR KMC11 NUMBER 14 |
| 1351 | 002364 | 000000 | ERCT14: 0 | ;ERROR COUNT FOR KMC11 NUMBER 14 |
| 1352 | | | | |
| 1353 | 002366 | 000000 | PACT15: 0 | ;PASS COUNT FOR KMC11 NUMBER 15 |
| 1354 | 002370 | 000000 | ERCT15: 0 | ;ERROR COUNT FOR KMC11 NUMBER 15 |
| 1355 | | | | |
| 1356 | 002372 | 000000 | PACT16: 0 | ;PASS COUNT FOR KMC11 NUMBER 16 |
| 1357 | 002374 | 000000 | ERCT16: 0 | ;ERROR COUNT FOR KMC11 NUMBER 16 |
| 1358 | | | | |
| 1359 | 002376 | 000000 | PACT17: 0 | ;PASS COUNT FOR KMC11 NUMBER 17 |
| 1360 | 002400 | 000000 | ERCT17: 0 | ;ERROR COUNT FOR KMC11 NUMBER 17 |
| 1361 | | | | |

1362
 1363
 1364
 1365
 1366
 1367

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 | 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 | 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 | I | I | I | I | I | I | I | I | I | STAT2 |
| I | * | B | M | 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 | 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 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 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)

CZKCE MACY11 30A(1052) 08-JUL-80 08:24 PAGE 32
CZKCE.P11 08-JUL-80 08:24 APT PARAMETER BLOCK

SEQ 0031

\$


```

1419
1420 ;PROGRAM INITIALIZATION
1421 ;LOCK OUT INTERRUPTS
1422 ;SET UP PROCESSOR STACK
1423 ;SET UP POWER FAIL VECTOR
1424 ;CLEAR PROGRAM CONTROL FLAGS AND COUNTS
1425 ;TYPE TITLE MESSAGE
1426
1427 002402 012737 000340 177776 .START: MOV #340,PS ;LOCK OUT INTERRUPTS
1428 002410 012706 001200 MOV #STACK,SP ;SET UP STACK
1429 002414 012737 007126 000024 MOV #SPWRDN,@#24 ;SET UP POWER FAIL VECTOR
1430 002422 013737 001472 001476 MOV KNUM,SAVNUM ;SAVE NUMBER OF DEVICES IN SYSTEM.
1431 002430 005037 011544 CLR SWFLG ;CLEAR SOFT TYPEOUT FLAG
1432 002434 105037 001203 CLR $ERFLG ;CLEAR ERROR FLAG
1433 002440 105037 001511 CLR QB.FLG ;ZERO QUICK VERIFY FLAG
1434 002444 012737 002070 001502 MOV #KM.MAP-10,CREAM;GET MAP POINTER.
1435 002452 012737 002276 001504 MOV #CNT.MAP-4,MILK ;GET PASS COUNT MAP POINTER
1436 002460 012737 100000 001500 MOV #BIT15,RUN ;POINT POINTER TO FIRST DEVICE.
1437 002466 012700 002302 MOV #CNT.MAP,RO ;PASS COUNT POINTER TO RO
1438 002472 005020 23$: CLR (RO)+ ;CLEAR TABLE
1439 002474 02700 002402 CMP #CNT.MAP+100,RO ;DONE YET?
1440 002500 001374 BNE 23$ ;KEEP GOING
1441 002502 005037 001216 CLR $ERRPC ;CLEAR LAST ERROR POINTER
1442 002506 012737 000001 001202 MOV #1,$STSTM ;SET UP FOR TEST 1
1443 002514 012737 002402 001206 MOV #.START,$LPADR ;SET UP FOR POWER FAIL BEFORE
1444 ;TESTING STARTS
1445 002522 132737 000001 001336 BITB #1,$ENV ; IS IT RUNNING UNDER APT?
1446 002530 001404 BEQ 3$ ; IF NOT CHECK FOR TYPE OF SWITCH REGISTER.
1447 002532 013737 001340 000176 MOV $SWREG,SWREG ; LOAD SOFTWARE SWITCH REG.
1448 002540 000423 BR 6$+2 ; GO SET UP SOFTWARE SWITCH REG.
1449 002542 013746 000006 3$: MOV @#6,-(SP) ;SAVE CURRENT VECTORS
1450 002546 013746 000004 MOV @#4,-(SP)
1451 002552 012737 002606 000004 MOV #6,@#4 ;SET UP FOR TIMEOUT
1452 002560 012737 177570 001240 MOV #177570,SWR ;SET SWR TO HARD SWR ADDRESS
1453 002566 012737 177570 001242 MOV #177570,DISPLAY ;SET DISPLAY TO HARD SWR ADDRESS
1454 002574 022777 177777 176436 CMP #-1,@SWR ;REFERENCE HARDWARE SWITCH REGISTER
1455 002602 001402 BEQ 6$+2 ;IF = -1 USE SOFT SWR ANYWAY
1456 002604 000407 BR 7$ ;IF IT EXISTS AND NOT = -1 USE HARD SWR
1457 002606 022626 6$: CMP (SP)+,(SP)+ ;ADJUST STACK
1458 002610 012737 000176 001240 MOV #SWREG,SWR ;POINTER TO SOFT SWR
1459 002616 012737 000174 001242 MOV #DISPREG,DISPLAY;POINTER TO SOFT DISPLAY REG
1460 002624 012637 000004 7$: MOV (SP)+,@#4 ;RESTORE VECTORS
1461 002630 012637 000006 MOV (SP)+,@#6
1462 002634 105737 001506 TSTB INIFLG ;HAS INITIALIZATION BEEN PERFORMED
1463 002640 001006 BNE 20$ ;BR IF YES
1464 002642 022737 004070 000042 CMP #SENDAD,@#42 ;IF ACT-11 AUTOMATIC MODE, DON'T TYPE ID
1465 002650 001402 BEQ 20$
1466 002652 104401 001000 TYPE ,MTITLE ;TYPE TITLE MESSAGE
1467 002656 004737 011340 20$: JSR PC,CKSWR ;CHECK FOR SOFT SWR
1468 002662 017737 176352 001446 MOV @SWR,STRTSW ;STORE STARTING SWITCHES
1469 002670 005737 000042 TST @#42 ;IS IT RUNNING IN AUTO MODE?
1470 002674 001402 BEQ .+6 ;BR IF NO
1471 002676 005037 001446 CLR STRTSW ;IF YES, CLEAR SWITCHES
1472 002702 032737 000001 001446 BIT #SW00,STRTSW ;IF SW00=1, QUESTIONS ARE ASKED.
1473 002710 001012 BNE 17$ ;BR IF SW00=1
1474 002712 105737 001446 TSTB STRTSW ;BIT7=1??
    
```

```
1475 002716 100007          BPL      17$          ;BR IF SW07=0
1476 002720 005737 001470   TST      KMACTV     ;ARE ANY DEVICES SELECTED?
1477 002724 001027          BNE      16$          ;BR IF YES
1478 002726 104401 011057   TYPE,    NOACT      ;NO DEVICES SELECTED.
1479 002732 000000          HALT                     ;STOP THE SHOW
1480 002734 000776          BR        -2          ;DISQUALIFY CONTINUE SWITCH
1481 002736 105737 001336   17$:    TSTB     $ENV   ; IS IT UNDER APT DUMP MODE?
1482 002742 001405          BEQ      27$          ; YES, CHECK IF APT SIZED IT?
1483 002744 132737 000001 001336   BITB     #1,$ENV     ; IS IT UNDER Q,V OR RUN MODE?
1484 002752 001012          BNE      30$          ; YES, NEEDS ONLY APT SIZING.
1485 002754 000406          BR        33$          ; NO, NEEDS REGULAR AUTO.SIZE.
1486 002756 105737 001337   27$:    TSTB     $ENVM  ; IS IT SIZED BY APT?
1487 002762 100406          BMI      30$          ; YES, NEEDS ONLY APT SIZING.
1488 002764 042737 000001 001446   BIC      #SW00,STRTSW ; SIZE ONLY IN AUTO MODE.
1489 002772 004737 012236   33$:    JSR      PC,AUTO.SIZE ; GO DO THE AUTO.SIZE.
1490 002776 000402          BR        16$          ; GO PRINT THE MAP.
1491 003000 004737 013716   30$:    JSR      PC,APT.SIZE ; GO DO THE APT SIZING.
1492 003004 105737 001506   16$:    TSTB     INIFLG   ;FIRST TIME?
1493 003010 001410          BEQ      21$          ;BR IF YES
1494 003012 105737 001446   TSTB     STRTSW     ;IF USING SAME PARAMETERS DONT TYPE MAP
1495 003016 100431          BMI      1$          ;
1496 003020 032737 000006 001446   BIT      #BIT1!BIT2,STRTSW;IS TEST NO. OR LOCK SELECTED
1497 003026 001403          BEQ      24$          ;IF NO THEN TYPE STATUS
1498 003030 000424          BR        1$          ;IF YES DO NOT TYPE STATUS
1499 003032 105137 001506   21$:    COMB     INIFLG   ;SET FLAG
1500 003036 104401 010100   24$:    TYPE     ,XHEAD   ;TYPE HEADER
1501 003042 012704 002100          MOV      #KM.MAP,R4  ;SET POINTER
1502 003046 010437 001276   5$:    MOV      R4,$TMP0  ;SET ADDRESS
1503 003052 012437 001300          MOV      (R4)+,$TMP1 ;SET CSR
1504 003056 001411          BEQ      1$          ;ALL DONE IF ZERO
1505 003060 012437 001302          MOV      (R4)+,$TMP2 ;SET STAT1
1506 003064 012437 001304          MOV      (R4)+,$TMP3 ;SET STAT2
1507 003070 012437 001306          MOV      (R4)+,$TMP4 ;SET STAT3
1508 003074 104416          CONVRT                     ;TYPE OUT STATUS MAP
1509 003076 011206          XSTATQ                     ;
1510 003100 000762          BR        5$          ;
1511 003102 012700 002100   1$:    MOV      #KM.MAP,R0 ;R0 POINTS TO STATUS TABLE
1512
1513   ;*****
1514   ;*AUTO SIZE TEST
1515   ;*THIS TEST VERIFYS THAT THE KMC11S AND/OR KMC11S ARE AT THE CORRECT FLOATING
1516   ;*ADDRESSES FOR YOUR SYSTEM. IF THIS TEST FAILS, IT IS NOT A HARDWARE ERROR.
1517   ;*CHECK THE ADDRESSES OF ALL FLOATING DEVICES (DJ,DH,DQ,DU,DUP,LK,DMC,DZ,KMC).
1518   ;*IF THERE ARE NO OTHER FLOATING DEVICES BEFORE THE KMC11, THE FIRST
1519   ;* KMC11 IS 760110. NO DEVICE SHOULD EVER BE AT
1520   ;*ADDRESS 760000.
1521   ;*****
1522
1523 003106 013746 000004          MOV      @#4,-(SP)    ;SAVE LOC 4
1524 003112 013746 000006          MOV      @#6,-(SP)    ;SAVE LOC 6
1525 003116 005037 000006          CLR      @#6          ;CLEAR VEC+2
1526 003122 005037 001302          CLR      $TMP2       ;CLEAR FLAG
1527 003126 011037 002066   AUSTRT: MOV      (R0),KMCSR ;GET NEXT KMC CSR
1528 003132 001510          BEQ      AUDONE      ;BR IF DONE
1529 003134 012737 003240 000004   2$:    MOV      #NODEV,@#4 ;SET UP FOR TIMEOUT
1530 003142 012703 000010   3$:    MOV      #10,R3    ;R3 IS COUNT OF DEVICES BEFORE KMC
```

```

1531 003146 012702 003342      4$:  MOV      #DEVTAB,R2      ;R2 IS DEVICE TABLE PONTER
1532 003152 012701 160010      MOV      #160010,R1      ;START WITH ADDRESS 160010
1533 003156 005711      FLOAT: TST      (R1)      ;CHECK ADDRESS IN R1
1534 003160 111204      MOV      (R2),R4      ;IF NO TIMEOUT, GET NEXT ADDRESS
1535 003162 060401      ADD      R4,R1      ;IN R1
1536 003164 005201      INC      R1
1537 003166 040401      BIC      R4,R1
1538 003170 005703      TST      R3
1539 003172 001371      BNE      FLOAT      ;ANY MORE DEVICES TO CHECK FOR?
1540 003174 012737 003244 000004      MOV      #ERR,@#4      ;BR IF YES
1541 003202 005711      FY:  TST      (R1)      ;OK ONLY KMC'S ARE LEFT, SET UP FOR TIMEOUT
1542 003204 020137 002066      CMP      R1,KMCSR      ;CHECK KMC ADDRESS
1543 003210 001403      BEQ      OK      ;DOES IT MATCH
1544 003212 062701 000010      ADD      #10,R1      ;BR IF YES
1545 003216 000771      BR      FY      ;GET NEXT KMC ADDRESS
1546 003220 062700 000010      OK:  ADD      #10,R0      ;DO IT AGAIN
1547 003224 062701 000010      ADD      #10,R1      ;SKIP TO NEXT KMC CSR
1548 003230 011037 002066      MOV      (R0),KMCSR      ;GET NEXT KMC ADDRESS
1549 003234 001447      BEQ      AUDONE      ;GET NEXT KMC CSR
1550 003236 000761      BR      FY      ;BRANCH IF ALL DONE.
1551 003240 122243      NODEV: CMPB     (R2)+,-(R3) ;DO IT AGAIN.
1552 003242 000002      RTI
1553 003244 005737 001302      ERR:  TST      $TMP2      ;ON TIMEOUT, INC R2, DEC R3
1554 003250 001014      BNE      1$
1555 003252 104401      TYPE     $LADR      ;CHECK FLAG IF = 0 TYPE HEADER
1556 003254 011110      CONERR   ;SKIP HEADER
1557 003256 012737 003244 001460      MOV      #ERR,SAVPC      ;TYPEOUT HEADER MESSAGE
1558 003264 104417      CNVRT    ;CONFIGURATION ERROR!!!!
1559 003266 003322      ERRPC   ;SAVE PC FOR TYPEOUT
1560 003270 104401      TYPE     ;TYPE OUT ERROR PC
1561 003272 011155      CNERR   ;TYPE REST OF HEADER
1562 003274 012737 177777 001302      MOV      #-1,$TMP2      ;SET FLAG SO IT ONLY GETS TYPED ONCE
1563 003302 010137 001264      1$:  MOV      R1,$REG1      ;SAVE R1 FOR TYPEOUT
1564 003306 104416      CONVRT  ;TYPE CSR VALUES
1565 003310 003330      CONTAB  ;TYPE CSR VALUES
1566 003312 104401      TYPE
1567 003314 011176      KCM
1568 003316 022626      4$:  CMP      (SP)+,(SP)+ ;ADJUST STACK
1569 003320 000737      BR      OK      ;BR TO GET OUT
1570 003322 000001      ERRPC: 1
1571 003324 006 002      .BYTE   6,2
1572 003326 001460      SAVPC
1573 003330 000002      CONTAB: 2
1574 003332 006 004      .BYTE   6,4
1575 003334 001264      $REG1
1576 003336 006 002      .BYTE   6,2
1577 003340 002066      KMCSR
1578 003342 007      DEVTAB: .BYTE   7 ;DJ
1579 003343 017      .BYTE   17 ;DH
1580 003344 007      .BYTE   7 ;DQ
1581 003345 007      .BYTE   7 ;DU
1582 003346 007      .BYTE   7 ;DUP
1583 003347 007      .BYTE   7 ;LK
1584 003350 007      .BYTE   7 ;DMC
1585 003351 007      .BYTE   7 ;DZ
1586 003352 007      .BYTE   7 ;KMC

```

```

1587      003354      003354      .EVEN
1588      003354      012637      000006      AUDONE:
1589      003354      012637      000004      1$:  MOV      (SP)+, @#6      ;RESTORE LOC 6
1590      003360      012637      000004      MOV      (SP)+, @#4      ;RESTORE LOC 4
1591      003364      032737      000010      001446      BIT      #SW03, STRTSW    ;SELECT SPECIFIC DEVICES??
1592      003372      001422      BEQ      3$              ;BR IF NO.
1593      003374      104401      010020      TYPE    ,MNEW           ;TYPE THE MESSAGE.
1594      003400      005000      CLR      RO              ;ZERO DATA LIGHTS
1595      003402      000000      HALT                    ;WAIT FOR USER TO TELL WHAT DEVICES TO RUN
1596      003404      027737      175630      001474      CMP      @SWR, SAVACT    ;IS THE NUMBER VALID?
1597      003412      101404      BLOS    2$              ;BR IF NUMBER IS OK.
1598      003414      104401      007673      TYPE    ,MERR3          ;TELL USER OF INVALID NUMBER.
1599      003420      000000      HALT                    ;STOP EVERY THING.
1600      003422      000776      BR      -2              ;RESTART THE PROGRAM AGAIN.
1601      003424      017737      175610      001470      2$:  MOV      @SWR, KMACTV   ;GET NEW DEVICE PATTERN
1602      003432      013700      001470      MOV      KMACTV, RO      ;SHOW THE USER WHAT HE SELECTED.
1603      003436      000000      HALT                    ;CONTINUE DYNAMIC SWITCHES.
1604      003440      012700      000300      3$:  MOV      #300, RO      ;PREPARE TO CLEAR THE FLOATING
1605      003444      012701      000302      MOV      #302, R1        ;VECTOR AREA. 300-776
1606      003450      010120      4$:  MOV      R1, (RO)+      ;START PUTTING 'PC+2 - HALT'
1607      003452      005021      CLR      (R1)+          ;IN VECTOR AREA.
1608      003454      022021      CMP      (RO)+, (R1)+    ;POP POINTERS
1609      003456      022700      001000      CMP      #1000, RO      ;ALL DONE??
1610      003462      001372      BNE     4$              ;BR IF NO.
1611
1612      ;TEST START AND RESTART
1613      ;-----
1614
1615      003464      012706      001200      .BEGIN: MOV      #STACK, SP ;SET UP STACK
1616      003470      013746      000006      MOV      @#6, -(SP)      ;SAVE LOC 6
1617      003474      013746      000004      MOV      @#4, -(SP)      ;SAVE LOC 4
1618      003500      005000      CLR      RO              ;START AT 0
1619      003502      012737      003546      000004      MOV      #2$, @#4        ;SET UP FOR TIME OUT
1620      003510      005037      000006      CLR      @#6            ;TO AUTOSIZE MEMORY
1621      003514      005720      6$:  TST      (RO)+          ;CHECK ADDRESS IN RO
1622      003516      022700      157776      CMP      #157776, RO     ;IS IT AT LEAST 28K
1623      003522      001374      BNE     6$              ;BR IF NO
1624      003524      162700      007776      SUB      #7776, RO       ;SAVE 2K FOR MONITORS
1625      003530      010037      001466      7$:  MOV      RO, MEMLIM     ;STORE MEMORY LIMIT
1626      003534      012637      000004      MOV      (SP)+, @#4      ;RESTORE LOC 4
1627      003540      012637      000006      MOV      (SP)+, @#6      ;RESTORE LOC 6
1628      003544      000413      BR      10$            ;CONTINUE
1629      003546      022626      2$:  CMP      (SP)+, (SP)+    ;ADJUST STACK
1630      003550      162700      000004      SUB      #4, RO          ;GET LAST GOOD ADDRESS
1631      003554      162700      007776      SUB      #7776, RO       ;SAVE 2K FOR MONITORS
1632      003560      022700      030000      CMP      #30000, RO      ;IS IT 8K?
1633      003564      001361      BNE     7$              ;BR IF NO
1634      003566      012700      037400      MOV      #37400, RO      ;IF 8K DON'T SAVE 2K
1635      003572      000756      BR      7$              ;
1636      003574      012737      000340      177776      10$: MOV      #340, PS       ;LOCK OUT INTERRUPTS
1637      003602      032737      000004      001446      BIT      #BIT2, STRTSW   ;CHECK FOR LOCK ON TES
1638      003610      001406      BEQ     1$              ;BR IF NO LOCK DESIRED.
1639      003612      104401      007717      TYPE    ,MLOCK          ;TYPE LOCK SELECTED.
1640      003616      012737      000240      004146      MOV      #NOP, TTST      ;SET UP TO LOCK
1641      003624      000403      BR      3$              ;CONTINUE ALONG.
1642      003626      013737      004360      004146      1$:  MOV      BRW, TTST       ;PREPARE NORMAL SCOPE ROUTINE
    
```

| | | | | | | | | |
|------|--------|--------|--------|--------|------|------|----------------|---|
| 1643 | 003634 | 012737 | 011606 | 001206 | 3\$: | MOV | #CYCLE,\$LPADR | :START AT "CYCLE" FIND WHICH DEVICE TO TEST |
| 1644 | 003642 | 032737 | 000002 | 001446 | 4\$: | BIT | #SW01,STRTSW | :IS TEST NO. SELECTED? |
| 1645 | 003650 | 001002 | | | | BNE | 5\$ | :BR IF YES |
| 1646 | 003652 | 104401 | 007643 | | | TYPE | ,MR | :TYPE R |
| 1647 | 003656 | 000177 | 175324 | | 5\$: | JMP | @\$LPADR | :START TESTING |

```

1648 ;END OF PASS
1649 ;TYPE NAME OF TEST
1650 ;UPDATE PASS COUNT
1651 ;CHECK FOR EXIT TO ACT-11
1652 ;RESTART TEST
1653
1654 .SBTTL END OF PASS ROUTINE
1655
1656 ;*****
1657 ;+INCREMENT THE PASS NUMBER ($PASS)
1658 ;*IF THERES A MONITOR GO TO IT
1659 ;*IF THERE ISN'T JUMP TO CYCLE
1660
1661 $EOP:
1662 003662 000005 RESET
1663 003662 005237 001324 INC $PASS ; INCREMENT THE PASS COUNT
1664 003670 105037 001203 CLR $ERFLG ; CLEAR ERROR FLAG
1665 003674 104401 007620 TYPE ,MEPASS ; TYPE END PASS.
1666 003700 104401 007746 TYPE ,MCSRX ; TYPE "CSR"
1667 003704 104417 004104 CNVRT ,XCSR ; SHOW IT.
1668 003710 104401 007754 TYPE ,MVECX ; TYPE VECTOR.
1669 003714 104417 004112 CNVRT ,XVEC ; SHOW IT.
1670 003720 104401 007762 TYPE ,MPASSX ; TYPE " PASSES "
1671 003724 104417 004120 CNVRT ,XPASS ; SHOW IT.
1672 003730 104401 007773 TYPE ,MERRX ; TYPE " ERRORS "
1673 003734 104417 004126 CNVRT ,XERR ; SHOW IT.
1674 003740 013700 001504 MOV MILK,RO ; SET POINTER TO PASSCNT.
1675 003744 013720 001324 MOV $PASS,(RO)+ ; SAVE THE PASS COUNT.
1676 003750 013720 001212 MOV $ERTTL,(RO)+ ; SAVE ERROR COUNT
1677 003754 013777 002060 176074 MOV KMRLVL,@KMRVEC ; RESTORE THE RECEIVER INTERRUPT VECTOR.
1678 003762 005077 176072 CLR @KMRLVL ; RESTORE RECEIVER LEVEL
1679 003766 013777 002064 176066 MOV KMTLVL,@KMTVEC ; RESTORE THE TRANSMIT INTERRUPT VECTOR.
1680 003774 005077 176064 CLR @KMTLVL ; RESTORE TRANSMITTER LEVEL
1681 004000 005337 001476 DEC SAVNUM ; ALL DEVICE TESTED?
1682 004004 001035 BNE $DOAGN ; BRANCH IF NO.
1683 004006 112737 000377 001511 MOVB #377,QV.FLG ; SET QUICK VERIFY FLAG.
1684 004014 013737 001472 001476 MOV KMNUM,SAVNUM ; RESTORE DEVICE COUNT.
1685 004022 005037 001216 CLR $ERRPC ; CLEAR LAST ERROR PC
1686 004026 005037 001310 CLR $TIMES ; ZERO THE NUMBER OF ITERATIONS
1687 004032 005237 001324 INC $PASS ; INCREMENT THE PASS NUMBER
1688 004036 042737 100000 001324 BIC #100000,$PASS ; DON'T ALLOW A NEG. NUMBER
1689 004044 005327 DEC (PC)+ ; LOOP?
1690 004046 000001 SEOPCT: .WORD 1
1691 004050 003013 BGT $DOAGN ; YES
1692 004052 012737 MOV (PC)+,@(PC)+ ; RESTORE COUNTER
1693 004054 000001 SENDCT: .WORD 1
1694 004056 004046 SEOPCT
1695 004060 013700 000042 $GET42: MOV @#42,RO ; GET MONITOR ADDRESS
1696 004064 001405 BEQ $DOAGN ; BRANCH IF NO MONITOR
1697 004066 000005 RESET ; CLEAR THE WORLD
1698 004070 004710 SENDAD: JSR PC,(RO) ; GO TO MONITOR
1699 004072 000240 NOP ; SAVE ROOM
1700 004074 000240 NOP ; FOR
1701 004076 000240 NOP ; ACT11
1702 004100 SDOAGN:
1703 004100 000137 JMP @ (PC)+ ; RETURN

```

1704 004102 011606
 1705 004104 000001
 1706 004106 006 002
 1707 004110 002066
 1708 004112 000001
 1709 004114 004 002
 1710 004116 002056
 1711 004120 000001
 1712 004122 006 002
 1713 004124 001324
 1714 004126 000001
 1715 004130 006 002
 1716 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

1717
 1718
 1719
 1720
 1721
 1722
 1723
 1724
 1725
 1726
 1727
 1728
 1729
 1730
 1731
 1732
 1733
 1734
 1735
 1736
 1737
 1738
 1739
 1740
 1741
 1742
 1743
 1744
 1745
 1746
 1747
 1748
 1749
 1750
 1751
 1752
 1753
 1754
 1755
 1756
 1757
 1758
 1759

```

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

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

004134
 004134 005037 001216
 004140 023716 014142
 004144 001413
 004146 000406
 004150 105777 175070
 004154 100067
 004156 017766 175064 177776
 004164 032777 040000 175046
 004172 001060
 004174 000416
 004176 013746 000004
 004202 012737 004222 000004
 004210 005737 177060
 004214 012637 000004
 004220 000436
 004222 022626
 004224 012637 000004
 004230 000441
 004232
 004232 105737 001203
 004236 001404
 004240 105037 001203
 004244 005037 001310
 004250 032777 004000 174762


```

1872      .SBTTL  APT COMMUNICATIONS ROUTINE
1873
1874      ::*****
1875 004676 112737 000001 005142 $ATY1:  MOV  #1,$FFLG      ::TO REPORT FATAL ERROR
1876 004704 112737 000001 005140 $ATY3:  MOV  #1,$MFLG      ::TO TYPE A MESSAGE
1877 004712 000403                BR      $ATYC
1878 004714 112737 000001 005142 $ATY4:  MOV  #1,$FFLG      ::TO ONLY REPORT FATAL ERROR
1879 004722                $ATYC:
1880 004722 010046                MOV  R0,-(SP)      ::PUSH R0 ON STACK
1881 004724 010146                MOV  R1,-(SP)      ::PUSH R1 ON STACK
1882 004726 105737 005140                TSTB $MFLG      ::SHOULD TYPE A MESSAGE?
1883 004732 001450                BEQ   5$          ::IF NOT: BR
1884 004734 122737 000001 001336                CMPB #APTENV,$ENV  ::OPERATING UNDER APT?
1885 004742 001031                BNE   3$          ::IF NOT: BR
1886 004744 132737 000100 001337                BITB #APTSPOOL,$ENVM ::SHOULD SPOOL MESSAGES?
1887 004752 001425                BEQ   3$          ::IF NOT: BR
1888 004754 017600 000004                MOV  @4(SP),R0     ::GET MESSAGE ADDR.
1889 004760 062766 000002 000004                ADD  #2,4(SP)      ::BUMP RETURN ADDR.
1890 004766 005737 001316                1$:  TST  $MSGTYPE     ::SEE IF DONE W/ LAST XMISSION?
1891 004772 001375                BNE   1$          ::IF NOT: WAIT
1892 004774 010037 001332                MOV  R0,$MSGAD     ::PUT ADDR IN MAILBOX
1893 005000 105720                2$:  TSTB (R0)+      ::FIND END OF MESSAGE
1894 005002 001376                BNE   2$
1895 005004 163700 001332                SUB  $MSGAD,R0     ::SUB START OF MESSAGE
1896 005010 006200                ASR  R0            ::GET MESSAGE LNTH IN WORDS
1897 005012 010037 001334                MOV  R0,$MSGGLT    ::PUT LENGTH IN MAILBOX
1898 005016 012737 000004 001316                MOV  #4,$MSGTYPE   ::TELL APT TO TAKE MSG.
1899 005024 000413                BR   5$
1900 005026 017637 000004 005052 3$:  MOV  @4(SP),4$     ::PUT MSG ADDR IN JSR LINKAGE
1901 005034 062766 000002 000004                ADD  #2,4(SP)      ::BUMP RETURN ADDRESS
1902 005042 013746 177776                MOV  177776,-(SP)  ::PUSH 177776 ON STACK
1903 005046 004737 004414                JSR  PC,$TYPE      ::CALL TYPE MACRO
1904 005052 000000                4$:  .WORD 0
1905 005054                5$:
1906 005054 105737 005142                10$: TSTB $FFLG      ::SHOULD REPORT FATAL ERROR?
1907 005060 001416                BEQ   12$         ::IF NOT: BR
1908 005062 005737 001336                TST  $ENV          ::RUNNING UNDER APT?
1909 005066 001413                BEQ   12$         ::IF NOT: BR
1910 005070 005737 001316                11$: TST  $MSGTYPE     ::FINISHED LAST MESSAGE?
1911 005074 001375                BNE   11$        ::IF NOT: WAIT
1912 005076 017637 000004 001320                MOV  @4(SP),$FATAL ::GET ERROR #
1913 005104 062766 000002 000004                ADD  #2,4(SP)      ::BUMP RETURN ADDR.
1914 005112 005237 001316                INC  $MSGTYPE     ::TELL APT TO TAKE ERROR
1915 005116 105037 005142                12$: CLRB $FFLG      ::CLEAR FATAL FLAG
1916 005122 105037 005141                CLRB $LFLG        ::CLEAR LOG FLAG
1917 005126 105037 005140                CLRB $MFLG        ::CLEAR MESSAGE FLAG
1918 005132 012601                MOV  (SP)+,R1     ::POP STACK INTO R1
1919 005134 012600                MOV  (SP)+,R0     ::POP STACK INTO R0
1920 005136 000207                RTS  PC           ::RETURN
1921 005140                $MFLG: .BYTE 0    ::MESSG. FLAG
1922 005141                $LFLG: .BYTE 0    ::LOG FLAG
1923 005142                $FFLG: .BYTE 0    ::FATAL FLAG
1924                .EVEN
1925                APTSIZE=200
1926                APTENV=001
1927                APTSPOOL=100
    
```

1928 000040

APTCSUP=040

1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983

005144 011646
005146 016666 000004 000002
005154 105777 174064
005160 100375
005162 117766 174060 000004
005170 042766 177600 000004
005176 026627 000004 000023
005204 001013
005206 105777 174032
005212 100375
005214 117746 174026
005220 042716 177600
005224 022627 000021
005230 001366
005232 000750
005234 026627 000004 000140
005242 002407
005244 026627 000004 000175
005252 003003
005254 042766 000040 000004
005262 000002
005264 010346
005266 005046
005270 012703 005520
005274 022703 005527
005300 101456
005302 104402
005304 112613
005306 122713 000177
005312 001022

```

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

```

```

1984 005314 005716          TST      (SP)          ;; IS THIS THE FIRST RUBOUT?
1985 005316 001007          BNE      6$           ;; BR IF NO
1986 005320 112737 000134 005516  MOVB     #' \,9$      ;; TYPE A BACK SLASH
1987 005326 104401 005516          TYPE     ,9$
1988 005332 012716 177777          MOV      #-1,(SP)     ;; SET THE RUBOUT KEY
1989 005336 005303          DEC      R3           ;; BACKUP BY ONE
1990 005340 020327 005520 6$:    CMP      R3,#$TTYIN  ;; STACK EMPTY?
1991 005344 103434          BLO     4$           ;; BR IF YES
1992 005346 111337 005516  MOVB     (R3),9$      ;; SETUP TO TYPEOUT THE DELETED CHAR.
1993 005352 104401 005516          TYPE     ,9$
1994 005356 000746          BR       2$           ;; GO TYPE
1995 005360 005716          TST     (SP)         ;; GO READ ANOTHER CHAR.
1996 005362 001406          BEQ     7$           ;; RUBOUT KEY SET?
1997 005364 112737 000134 005516  MOVB     #' \,9$      ;; BR IF NO
1998 005372 104401 005516          TYPE     ,9$
1999 005376 005016          CLR     (SP)         ;; TYPE A BACK SLASH
2000 005400 122713 000025 7$:    CMPB    #25,(R3)     ;; CLEAR THE RUBOUT KEY
2001 005404 001003          BNE     8$           ;; IS CHARACTER A CTRL U?
2002 005406 104401 005527          TYPE    ,%CNTLU      ;; BR IF NO
2003 005412 000726          BR      1$           ;; TYPE A CONTROL 'U'
2004 005414 122713 000022 8$:    CMPB    #22,(R3)     ;; GO START OVER
2005 005420 001011          BNE     3$           ;; IS CHARACTER A '^R'?
2006 005422 105013          CLRB   (R3)         ;; BRANCH IF NO
2007 005424 104401 001313          TYPE    ,%SCLF       ;; CLEAR THE CHARACTER
2008 005430 104401 005520          TYPE    ,%TTYIN      ;; TYPE A 'CR' & 'LF'
2009 005434 000717          BR      2$           ;; TYPE THE INPUT STRING
2010 005436 104401 001312 4$:    TYPE    ,%QUES       ;; GO PICKUP ANOTHER CHACTER
2011 005442 000712          BR      1$           ;; TYPE A '?'
2012 005444 111337 005516 3$:    MOVB     (R3),9$     ;; CLEAR THE BUFFER AND LOOP
2013 005450 104401 005516          TYPE     ,9$
2014 005454 122723 000015          CMPB    #15,(R3)+    ;; ECHO THE CHARACTER
2015 005460 001305          BNE     2$           ;; CHECK FOR RETURN
2016 005462 105063 177777          CLRB   -1(R3)       ;; LOOP IF NOT RETURN
2017 005466 104401 001314          TYPE    ,%LF         ;; CLEAR RETURN (THE 15)
2018 005472 005726          TST     (SP)+        ;; TYPE A LINE FEED
2019 005474 012603          MOV     (SP)+,R3     ;; CLEAN RUBOUT KEY FROM THE STACK
2020 005476 011646          MOV     (SP),-(SP)   ;; RESTORE R3
2021 005500 016666 000004 000002  MOV     4(SP),2(SP)  ;; ADJUST THE STACK AND PUT ADDRESS OF THE
2022 005506 012766 005520 000004  MOV     #$TTYIN,4(SP) ;; FIRST ASCII CHARACTER ON IT
2023 005514 000002          RTI
2024 005516 000          9$:    .BYTE   0           ;; RETURN
2025 005517 000          .BYTE   0           ;; STORAGE FOR ASCII CHAR. TO TYPE
2026 005520 000007          $TTYIN: .BLKB   7     ;; TERMINATOR
2027 005527 136 006525 000012  %CNTLU: .ASCIZ  /^U/<15><12> ;; RESERVE 7 BYTES FOR TTY INPUT
2028 005534 043536 005015 000  %CNTLG: .ASCIZ  /^G/<15><12> ;; CONTROL 'U'
2029 005541 015 051412 051127  %MSWR:  .ASCIZ  <15><12>/SWR = / ;; CONTROL 'G'
2030 005546 036440 000040          $MNEW:  .ASCIZ  / NEW = /
2031 005552 020040 042516 020127
2032 005560 020075 000
2033 005564
2034          .EVEN
2035          .SBTTL READ AN OCTAL NUMBER FROM THE TTY
2036          ;;*****
2037          ;;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
2038          ;;*CHANGE IT TO BINARY.
2039          ;;*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL

```

```

2040 ;*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED
2041 ;*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
2042 ;*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
2043 ;*CALL:
2044 ;*      RDOCT          ;;READ AN OCTAL NUMBER
2045 ;*      RETURN HERE   ;;LOW ORDER BITS ARE ON TOP OF THE STACK
2046 ;*                   ;;HIGH ORDER BITS ARE IN $HIOCT
2047
2048 005564 011646          $RDOCT: MOV      (SP),-(SP)      ;;PROVIDE SPACE FOR THE
2049 005566 016666 000004 000002 MOV      4(SP),2(SP)      ;;INPUT NUMBER
2050 005574 010046          MOV      R0,-(SP)          ;;PUSH R0 ON STACK
2051 005576 010146          MOV      R1,-(SP)          ;;PUSH R1 ON STACK
2052 005600 010246          MOV      R2,-(SP)          ;;PUSH R2 ON STACK
2053 005602 104403          1$:  RDLIN          ;;READ AN ASCII LINE
2054 005604 012600          MOV      (SP)+,R0          ;;GET ADDRESS OF 1ST CHARACTER
2055 005606 010037 005712 MOV      R0,5$          ;;AND SAVE IT
2056 005612 005001          CLR      R1          ;;CLEAR DATA WORD
2057 005614 005002          CLR      R2
2058 005616 112046          2$:  MOVB      (R0)+,-(SP)      ;;PICKUP THIS CHARACTER
2059 005620 001420          BEQ      3$          ;;IF ZERO GET OUT
2060 005622 122716 000060 CMPB      #'0,(SP)      ;;MAKE SURE THIS CHARACTER
2061 005626 003026          BGT      4$          ;;IS AN OCTAL DIGIT
2062 005630 122716 000067 CMPB      #'7,(SP)
2063 005634 002423          BLT      4$
2064 005636 006301          ASL      R1          ;;*2
2065 005640 006102          ROL      R2
2066 005642 006301          ASL      R1          ;;*4
2067 005644 006102          ROL      R2
2068 005646 006301          ASL      R1          ;;*8
2069 005650 006102          ROL      R2
2070 005652 042716 177770 BIC      #'C7,(SP)      ;;STRIP THE ASCII JUNK
2071 005656 062601          ADD      (SP)+,R1      ;;ADD IN THIS DIGIT
2072 005660 000756          BR       2$          ;;LOOP
2073 005662 005726          3$:  TST      (SP)+          ;;CLEAN TERMINATOR FROM STACK
2074 005664 010166 000012 MOV      R1,12(SP)      ;;SAVE THE RESULT
2075 005670 010237 005722 MOV      R2,$HIOCT
2076 005674 012602          MOV      (SP)+,R2      ;;POP STACK INTO R2
2077 005676 012601          MOV      (SP)+,R1      ;;POP STACK INTO R1
2078 005700 012600          MOV      (SP)+,R0      ;;POP STACK INTO R0
2079 005702 000002          RTI          ;;RETURN
2080 005704 005726          4$:  TST      (SP)+          ;;CLEAN PARTIAL FROM STACK
2081 005706 105010          CLRB     (R0)          ;;SET A TERMINATOR
2082 005710 104401          TYPE          ;;TYPE UP THRU THE BAD CHAR.
2083 005712 000000          5$:  .WORD     0
2084 005714 104401 001312 TYPE      ,SQUES      ;;'"?' 'CR' & 'LF'
2085 005720 000730          BR       1$          ;;TRY AGAIN
2086 005722 000000          $HIOCT: .WORD     0      ;;HIGH ORDER BITS GO HERE
2087
2088 ;-----
2089 ; INPUT OCTAL NUMBER ROUTINE
2090 ;-----
2091 005724 010546          $INPUT: MOV      R5,-(SP)      ; SAVE REGISTER R5.
2092 005726 016605 000002 MOV      2(SP),R5      ; GET FIRST PARAMETER ADDRESS.
2093 005732 012537 005770 MOV      (R5)+,WHAT    ; GET MESSAGE ADDRESS.
2094 005736 012537 006050 MOV      (R5)+,LOLIM   ; GET LOW LIMIT FOR THE #
2095 005742 012537 006052 MOV      (R5)+,HILIM   ; GET HIGH LIMIT FOR THE #.
  
```

```

2096 005746 012537 006054      MOV      (R5)+,WHERE      ; GET ADDRESS OF INBUFFER
2097 005752 112537 006056      MOV      (R5)+,LOBITS    ; GET LOWMASK BITS.
2098 005756 112537 006057      MOV      (R5)+,ADRCNT    ; GET # OF #'S TO BE GENERATED.
2099 005762 010566 000002      MOV      R5,2(SP)        ; SAVE THE RETURN ADDRESS.
2100 005766 104401                INLP1:  TYPE              ; TYPE THE MESSAGE.
2101 005770 000000                WHAT:   .WORD            0
2102 005772 104404                RDOCT
2103 005774 021637 006052      CMP      (SP),HILIM      ; READ OCTAL # FROM KEYBOARD.
2104 006000 003003                BGT      2$              ; IS IT IN HIGH LIMIT?
2105 006002 021637 006050      CMP      (SP),LOLIM      ; BRANCH IF NO.
2106 006006 002005                BGE      3$              ; IS IT MORE THAN LOW LIMIT.
2107 006010 104401 001312      2$:     TYPE              ; BRANCH IF YES.
2108 006014 104401 001313      TYPE      , $QUES        ; TYPE "' ? '"
2109 006020 000762                TYPE      , $CRLF        ; TYPE <CR>,<LF>
2110 006022 013705 006054      BR       INLP1
2111 006026 011625                3$:     MOV      WHERE,R5  ; GET BUFFER ADDRESS.
2112 006030 062716 000002      4$:     MOV      (SP),(R5)+ ; SAVE THE # IN RIGHT PLACE.
2113 006034 105337 006057      ADD      #2,(SP)         ; NEXT SEQUENTIAL NUMBER.
2114 006040 001372                DECB     ADRCNT          ; COUNT BY 1.
2115 006042 005726                BNE      4$              ; BRANCH IF NOT DONE.
2116 006044 012605                TST      (SP)+           ; POP THE STACK POINTER.
2117 006046 000002                MOV      (SP)+,R5       ; POP THE REG.5
2118 006050 000000                RTI
2119 006052 000000                LOLIM:  .WORD            0
2120 006054 000000                HILIM:  .WORD            0
2121 006056          000                WHERE:  .WORD            0
2122 006057          000                LOBITS: .BYTE            0
2123                                ADRCNT: .BYTE            0
2124                                ; ADVANCE TO NEXT TEST HANDLER
2125                                ;-----
2126                                ;
2127 006060 013716 001442      .ADVANCE: MOV      NEXT,(SP) ; CRUNCH STACK WITH ADDRESSOF SCOPE CALL
2128 006064 005037 001444      CLR      LOCK            ; RESET TIGHT LOOP ADDRESS
2129 006070 000002                RTI                      ; CHECK TO SEE IF OLD TEST GETS REPEATED
2130                                ;
2131                                ;SAVE PC OF TEST THAT FAILED AND R0-R5
2132                                ;-----
2133                                ;
2134 006072 016637 000004 001460 .SAV05: MOV      4(SP),SAVPC ;SAVE R7 (PC)
2135                                ;
2136                                ;SAVE R0-R5
2137                                ;
2138 006100 010537 001274      SV05:  MOV      R5,$REG5   ;SAVE R5
2139 006104 010437 001272      MOV      R4,$REG4   ;SAVE R4
2140 006110 010337 001270      MOV      R3,$REG3   ;SAVE R3
2141 006114 010237 001266      MOV      R2,$REG2   ;SAVE R2
2142 006120 010137 001264      MOV      R1,$REG1   ;SAVE R1
2143 006124 010037 001262      MOV      R0,$REG0   ;SAVE R0
2144 006130 000002                RTI                      ;LEAVE.
2145                                ;
2146                                ;RESTORE R0-R5
2147                                ;
2148 006132 013700 001262      .RES05: MOV      $REG0,R0 ;RESTORE R0
2149 006136 013701 001264      MOV      $REG1,R1   ;RESTORE R1
2150 006142 013702 001266      MOV      $REG2,R2   ;RESTORE R2
2151 006146 013703 001270      MOV      $REG3,R3   ;RESTORE R3

```

| | | | | | | | |
|------|--------|--------|--------|--------|---------|-----------|--|
| 2152 | 006152 | 013704 | 001272 | | MOV | \$REG4,R4 | :RESTORE R4 |
| 2153 | 006156 | 013705 | 001274 | | MOV | \$REG5,R5 | :RESTORE R5 |
| 2154 | 006162 | 000002 | | | RTI | | :LEAVE |
| 2155 | | | | | | | |
| 2156 | | | | | : | | :CONVERT OCTAL NUMBER TO ASCII AND OUTPUT TO TELEPRINTER |
| 2157 | | | | | : | | ----- |
| 2158 | | | | | : | | |
| 2159 | 006164 | 104401 | 001313 | | .CONVR: | TYPE | ,\$CRLF |
| 2160 | 006170 | 010046 | | | .CNVRT: | MOV | R0,-(SP) |
| 2161 | 006172 | 010146 | | | | MOV | R1,-(SP) |
| 2162 | 006174 | 010346 | | | | MOV | R3,-(SP) |
| 2163 | 006176 | 010446 | | | | MOV | R4,-(SP) |
| 2164 | 006200 | 010546 | | | | MOV | R5,-(SP) |
| 2165 | 006202 | 017601 | 000012 | | | MOV | @12(SP),R1 |
| 2166 | 006206 | 062766 | 000002 | 000012 | | ADD | #2,12(SP) |
| 2167 | 006214 | 012137 | 006406 | | | MOV | (R1)+,WRDCNT |
| 2168 | 006220 | 112137 | 006410 | | 1\$: | MOVB | (R1)+,CHRCNT |
| 2169 | 006224 | 112137 | 006411 | | | MOVB | (R1)+,SPACNT |
| 2170 | 006230 | 013137 | 006412 | | | MOV | @(R1)+,BINWRD |
| 2171 | 006234 | 122737 | 000003 | 006410 | | CMPB | #3,CHRCNT |
| 2172 | 006242 | 001003 | | | | BNE | 2\$ |
| 2173 | 006244 | 042737 | 177400 | 006412 | | BIC | #177400,BINWRD |
| 2174 | 006252 | 013704 | 006412 | | 2\$: | MOV | BINWRD,R4 |
| 2175 | 006256 | 113705 | 006410 | | | MOVB | CHRCNT,R5 |
| 2176 | 006262 | 012700 | 011234 | | | MOV | #TEMP,R0 |
| 2177 | 006266 | 010403 | | | 3\$: | MOV | R4,R3 |
| 2178 | 006270 | 042703 | 177770 | | | BIC | #177770,R3 |
| 2179 | 006274 | 062703 | 000060 | | | ADD | #060,R3 |
| 2180 | 006300 | 110320 | | | | MOVB | R3,(R0)+ |
| 2181 | 006302 | 000241 | | | | CLC | |
| 2182 | 006304 | 006004 | | | | ROR | R4 |
| 2183 | 006306 | 000241 | | | | CLC | |
| 2184 | 006310 | 006004 | | | | ROR | R4 |
| 2185 | 006312 | 000241 | | | | CLC | |
| 2186 | 006314 | 006004 | | | | ROR | R4 |
| 2187 | 006316 | 005305 | | | | DEC | R5 |
| 2188 | 006320 | 001362 | | | | BNE | 3\$ |
| 2189 | 006322 | 012703 | 011276 | | | MOV | #MDATA,R3 |
| 2190 | 006326 | 114023 | | | 4\$: | MOVB | -(R0),(R3)+ |
| 2191 | 006330 | 105337 | 006410 | | | DECB | CHRCNT |
| 2192 | 006334 | 001374 | | | | BNE | 4\$ |
| 2193 | 006336 | 105737 | 006411 | | | TSTB | SPACNT |
| 2194 | 006342 | 001405 | | | | BEQ | 6\$ |
| 2195 | 006344 | 112723 | 000040 | | 5\$: | MOVB | #040,(R3)+ |
| 2196 | 006350 | 105337 | 006411 | | | DECB | SPACNT |
| 2197 | 006354 | 001373 | | | | BNE | 5\$ |
| 2198 | 006356 | 105013 | | | 6\$: | CLRB | (R3) |
| 2199 | 006360 | 104401 | 011276 | | | TYPE | ,MDATA |
| 2200 | 006364 | 005337 | 006406 | | | DEC | WRDCNT |
| 2201 | 006370 | 001313 | | | | BNE | 1\$ |
| 2202 | 006372 | 012605 | | | | MOV | (SP)+,R5 |
| 2203 | 006374 | 012604 | | | | MOV | (SP)+,R4 |
| 2204 | 006376 | 012603 | | | | MOV | (SP)+,R3 |
| 2205 | 006400 | 012601 | | | | MOV | (SP)+,R1 |
| 2206 | 006402 | 012600 | | | | MOV | (SP)+,R0 |
| 2207 | 006404 | 000002 | | | | RTI | |

2208 006406 000000
 2209 006410 000000
 2210 006411 000000
 2211 006412 000000
 2212
 2213
 2214
 2215
 2216
 2217
 2218
 2219
 2220
 2221
 2222
 2223
 2224
 2225
 2226
 2227 006414 010046
 2228 006416 016600 000002
 2229 006422 005740
 2230 006424 111000
 2231 006426 006300
 2232 006430 016000 006450
 2233 006434 000200
 2234
 2235
 2236
 2237
 2238 006436 011646
 2239 006440 016666 000004 000002
 2240 006446 000002
 2241
 2242
 2243
 2244
 2245
 2246
 2247
 2248
 2249 006450 006436
 2250 006452 004414
 2251
 2252
 2253 006454 005144
 2254 006456 005264
 2255 006460 005564
 2256 006462 004364
 2257 006464 006072
 2258 006466 006132
 2259 006470 007362
 2260 006472 007332
 2261 006474 007400
 2262 006476 007446
 2263 006500 007512

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

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

.SBTTL TRAP DECODER

 :*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
 :*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
 :*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
 :*GO TO THAT ROUTINE.

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

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

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

.SBTTL TRAP TABLE

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

```

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

$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
```



```

2264 006502 005724          $INPUT  ;;CALL=INPUT  TRAP+15(104415) CALL TO OCTAL # INPUT ROUTINE
2265 006504 006164          .CONVRT ;;CALL=CONVRT TRAP+16(104416) CALL TO .....
2266 006506 006170          .CNVRT  ;;CALL=CNVRT  TRAP+17(104417) CALL TO .....
2267 006510 006060          .ADVANCE ;;CALL=ADVANCE TRAP+20(104420) CALL TO ADVANCE TO NEXT TEST
2268
2269
2270
2271
2272
2273
2274 006512 004737 011340          $ERROR: JSR      PC,CKSWR          ;CHECK FOR SOFT SWR
2275 006516 032777 010000 172514  BIT      #SW12,@SWR          ;BELL ON ERROR?
2276 006524 001406          BEQ      XBX                ;BR IF NO BELL
2277 006526 105777 172516          TSTB    @$TPS              ;TTY READY.
2278 006532 100003          BPL      XBX                ;DON'T WAIT IF TTY NOT READY.
2279 006534 112777 000207 172510  MOVB    #207,@$TPB         ;PUSH A BELL AT THE TTY.
2280 006542 032777 020000 172470  XBX:    BIT      #SW13,@SWR          ;DELETE ERROR PRINT OUT?
2281 006550 001107          BNE      HALTS              ;BR IF NO PRINT OUT WANTED.
2282 006552 021637 001216          CMP     (SP),$ERRPC         ;WAS THIS ERROR FOUND LAST TIME?
2283 006556 001404          BEQ      1$                ;BR IF YES
2284 006560 011637 001216          MOV     (SP),$ERRPC         ;RECORD BEING HERE
2285 006564 105037 001203          CLRB    $ERFLG             ;PREPARE HEADER
2286 006570 104406          1$:    SAVO5                ;SAVE ALL PROC REGISTERS
2287 006572 011605          MOV     (SP),R5             ;GET THE PC OF ERROR
2288 006574 162705 000002          SUB     #2,R5               ;GET ADDRESS OF TRAP CALL
2289 006600 011504          MOV     (R5),R4            ;GET ERROR INSTRUCTION
2290 006602 110437 001214          MOVB    R4,$ITEMB          ; COPY ERROR # FOR APT HANDLING
2291 006606 006304          ASL     R4                  ;MULT BY TWO
2292 006610 061504          ADD     (R5),R4            ;DOUBLE IT
2293 006612 006304          ASL     R4                  ;MULT AGAIN
2294 006614 042704 177001          BIC     #177001,R4         ;CLEAR JUNK
2295 006620 062704 001512          ADD     #$ERRTB,R4         ;GET POINTER
2296 006624 012437 006740          MOV     (R4)+,ERRMSG       ;GET ERROR MESSAGE
2297 006630 012437 006752          MOV     (R4)+,DATAHD       ;GET DATA HEADRER
2298 006634 011437 006764          MOV     (R4),DATABP        ;GET DATA TABLE
2299 006640 105737 001203          TSTB    $ERFLG             ;TYPE HEADREER
2300 006644 001403          BEQ     TYPMSG              ;BR IF YES
2301 006646 005737 006764          TST     DATABP              ;DOES DATA TABLE EXIST?
2302 006652 001040          BNE     TYPDAT              ;BR IF YES.
2303 006654 104401 001313          TYPMSG: TYPE    ,SCRLF
2304 006660 104401 001313          TYPE    ,SCRLF
2305 006664 005737 001444          TST     LOCK
2306 006670 001402          BEQ     1$
2307 006672 104401 010016          TYPE    ,MASTEK
2308 006676 104401 010004          1$:    TYPE    ,MTSTN
2309 006702 104417 007120          CNVRT   ,XTSTN              ;SHOW IT
2310 006706 104401 010073          TYPE    ,MERRPC            ;TYPE PC.
2311 006712 104417 007112          CNVRT   ,ERTABO            ;SHOW IT
2312 006716 104401 001313          TYPE    ,SCRLF             ;GIVE A CR/LF
2313 006722 112737 177777 001203  MOVB    #-1,$ERFLG         ;NO MORE HEADER UNLESS NO DATA TABLE.
2314 006730 005737 006740          TST     ERRMSG              ;IS THERE AN ERROR MESSAGE?
2315 006734 001402          BEQ     WRKO.FM             ;BR IF NO.
2316 006736 104401          TYPE
2317 006740 000000          ERRMSG: 0                   ; ERROR MESSAGE
2318 006742          WRKO.FM:
2319 006742 005737 006752          TST     DATAHD            ;DATA HEADER?
  
```

```

2320 006746 001402          BEQ      TYPDAT      ;BR IF NO
2321 006750 104401          TYPE
2322 006752 000000          DATAHD: 0          ;TYPE
2323 006754 005737 006764  TYPDAT: TST      DATABP ; DATA HEADER
2324 006760 001402          BEQ      RESREG      ;DATA TABLE?
2325 006762 104416          CONVRT
2326 006764 000000          DATABP: 0          ;BR IF NO.
2327 006766 104407          RESREG: RES05      ;SHOW
2328 006770 122737 000001 001336 HALTS:  CMPB      #APTENV,$ENV ; DATA TABLE
2329 006776 001007          BNE      3$          ;RESTORE PROC REGISTERS
2330 007000 113737 001214 007012  MOVB     $ITEMB,6$ ; IS APT RUNNING ?
2331 007006 004737 004714          JSR      PC,$ATY4   ; SKIP APT CALL IF NOT.
2332 007012 000000          .WORD   0           ; COPY ERROR #.
2333 007014 000777          BR       9$          ; CALL APT SERVICES.
2334 007016 022737 004070 000042 3$:     CMP      #SENDAD,@#42 ; ERROR # GOES HERE.
2335 007024 001403          BEQ      1$          ; LOCK HERE.
2336 007026 005777 172206          TST      @SWR       ;IF ACT-11 AUTOMATIC MODE, HALT!!
2337 007032 100005          BPL      EXITER     ;HALT ON ERROR?
2338 007034 010046          1$:     PUSHRO ;BR IF NO HALT ON ERROR
2339 007036 016600 000002          MOV      2(SP),RO  ;SAVE RO
2340 007042 000000          HALT          ;SHOW ERROR PC IN DATA LIGHTS
2341 007044 012600          POPRO        ;HALT
2342 007046 005237 001212          EXITER: INC      $ERTTL ;GET RO
2343 007052 032777 000400 172160  BIT     #SW08,@SWR ;UPDATE ERROR COUNT
2344 007060 001007          BNE      1$          ;GOTO TOP OF TEST?
2345 007062 032777 002000 172150  BIT     #SW10,@SWR ;BR IF YES
2346 007070 001407          BEQ      2$          ;GOTO NEXT TEST?
2347 007072 013737 001442 001206 1$:     MOV     NEXT,$LPADR ;BR IF NO
2348 007100 012706 001200          MOV     #STACK,SP ;SET FOR NEXT TEST
2349 007104 000177 172076          JMP     @SLPADR    ;RESET SP
2350 007110 000002          2$:     RTI          ;GOTO SPECIFIED TEST
2351 007112 000001          ERTABO: 1          ;$LPADR
2352 007114          006          002          .BYTE   6,2
2353 007116 001460          SAVPC
2354 007120 000001          XTSTN: 1
2355 007122          003          002          .BYTE   3,2
2356 007124 001202          $STSTM
2357          ;ENTER HERE ON POWER FAILURE
2358          ;-----
2359
2360          .SBTTL  POWER DOWN AND UP ROUTINES
2361
2362          ;*****
2363          ;POWER DOWN ROUTINE
2364 007126 012737 007316 000024 $PWRDN: MOV     #SILLUP,@#PWRVEC ;;SET FOR FAST UP
2365 007134 012737 000340 000026  MOV     #340,@#PWRVEC+2 ;;PRIO:7
2366 007142 010046          MOV     R0,-(SP)    ;;PUSH R0 ON STACK
2367 007144 010146          MOV     R1,-(SP)    ;;PUSH R1 ON STACK
2368 007146 010246          MOV     R2,-(SP)    ;;PUSH R2 ON STACK
2369 007150 010346          MOV     R3,-(SP)    ;;PUSH R3 ON STACK
2370 007152 010446          MOV     R4,-(SP)    ;;PUSH R4 ON STACK
2371 007154 010546          MOV     R5,-(SP)    ;;PUSH R5 ON STACK
2372 007156 017746 172056          MOV     @SWR,-(SP)  ;;PUSH @SWR ON STACK
2373 007162 010637 007322          MOV     SP,$SAVR6  ;;SAVE SP
2374 007166 012737 007200 000024  MOV     #SPWRUP,@#PWRVEC ;;SET UP VECTOR
2375 007174 000000          HALT

```

```

2376 007176 000776          BR      -2          ;;HANG UP
2377
2378
2379          ;;*****
          :POWER UP ROUTINE
2380 007200 012737 007316 000024 $PWRUP: MOV    #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
2381 007206 013706 007322          MOV    $SAVR6,SP ;;GET SP
2382 007212 005037 007322          CLR    $SAVR6 ;;WAIT LOOP FOR THE TTY
2383 007216 005237 007322 1$: INC    $SAVR6 ;;WAIT FOR THE INC
2384 007222 001375          BNE    1$ ;;OF WORD
2385 007224 104401 007562          TYPE   ,MPFAIL
2386 007230 104417 007324          CNVRT  ,PFTAB
2387 007234 105037 001203          CLRB  $ERFLG ;;CLEAR ERROR FLAG.
2388 007240 005037 001216          CLR   $ERRPC ;;CLEAR LAST ERROR PC
2389 007244 013701 002066          MOV   KMCSR,R1 ;;RESTORE DEVICE ADDRESS.
2390 007250 005011          CLR   (R1) ;;CLEAR THE CSR.
2391 007252 104410          MSTCLR
2392 007254 012677 171760          MOV   (SP)+,@SWR ;;POP STACK INTO @SWR
2393 007260 012605          MOV   (SP)+,R5 ;;POP STACK INTO R5
2394 007262 012604          MOV   (SP)+,R4 ;;POP STACK INTO R4
2395 007264 012603          MOV   (SP)+,R3 ;;POP STACK INTO R3
2396 007266 012602          MOV   (SP)+,R2 ;;POP STACK INTO R2
2397 007270 012601          MOV   (SP)+,R1 ;;POP STACK INTO R1
2398 007272 012600          MOV   (SP)+,R0 ;;POP STACK INTO R0
2399 007274 012737 007126 000024          MOV   #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
2400 007302 012737 000340 000026          MOV   #340,@#PWRVEC+2 ;;PRIO:7
2401 007310 104401          TYPE   ,MPFAIL ;;REPORT THE POWER FAILURE
2402 007312 007562          $PWRMG: .WORD MPFAIL ;;POWER FAIL MESSAGE POINTER
2403 007314 000002          RTI
2404 007316 000000          $ILLUP: HALT ;;THE POWER UP SEQUENCE WAS STARTED
2405 007320 000776          BR      -2 ;;BEFORE THE POWER DOWN WAS COMPLETE
2406 007322 000000          $SAVR6: 0 ;;PUT THE SP HERE
2407
2408 007324 000001          PFTAB: 1
2409 007326 003 002          .BYTE 3,2
2410 007330 001202          $TSTNM
2411
2412 007332          .DELAY:
2413 007332 012777 000020 172534          MOV   #20,@KMP04
2414 007340 104412          ROMCLK 121111 ;;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2415 007342 121111          ;;POKE CLOCK DELAY BIT
2416 007344          1$:
2417 007344 104412          ROMCLK 121224 ;;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2418 007346 121224          ;;PORT4 IBUS*11
2419 007350 032777 000020 172516          BIT   #BIT4,@KMP04 ;;IS CLOCK BIT SET?
2420 007356 001772          BEQ   1$ ;;BR IF NO
2421 007360 000002          RTI
2422
2423 007362          .MSTCLR:
2424 007362 152777 000100 172500          BISB  #BIT6,@KMCSRH ;;SET MASTER CLEAR
2425 007370 142777 000300 172472          BICB  #BIT6!BIT7,@KMCSRH ;;CLEAR MASTER CLEAR AND RUN
2426 007376 000002          RTI ;;RETURN
2427
2428 007400          .ROMCLK:
2429 007400 152777 000002 172462          BISB  #BIT1,@KMCSRH ;;SET ROMI
2430 007406 013677 172464          MOV   @ (SP)+,@KMP06 ;;LOAD INSTRUCTION IN SEL6
2431 007412 062746 000002          ADD   #2,-(SP) ;;ADJUST STACK
    
```

```

2432 007416 032777 000100 171614      BIT      #SW06,@SWR      ;HALT IF SW06 =1
2433 007424 001401                    BEQ      1$           ;BR IF SW06 =0
2434 007426 000000                    HALT                                ;HALT BEFORE CLOCKING INSTRUCTION
2435 007430 152777 000003 172432 1$:  BISB    #BIT1!BIT0,@KMCSRH ;CLOCK INSTRUCTION
2436 007436 142777 000007 172424  BICB    #BIT2!BIT1!BIT0,@KMCSRH ;CLEAR ROMO, ROMI, STEP
2437 007444 000002                    RTI
2438
2439 007446                    .DATACLK:
2440 007446 013637 011234      MOV      @(SP)+,TEMP      ;PUT TICK COUNT IN TEMP
2441 007452 062746 000002      ADD      #2,-(SP)        ;ADJUST STACK
2442 007456 152777 000020 172404 1$:  BISB    #BIT4,@KMCSRH    ;SET STEP LU
2443 007464 027777 172376 172374  CMP      @KMCSR,@KMCSR    ;WASTE TIME
2444 007472 142777 000020 172370  BICB    #BIT4,@KMCSRH    ;CLEAR STEP LU
2445 007500 005337 011234      DEC      TEMP            ;DEC TICK COUNT
2446 007504 001364                    BNE      1$           ;BR IF NOT DONE
2447 007506 000002                    RTI                    ;RETURN
2448 007510 000001      3$:  .BLKW 1
2449
2450 007512                    .TIMER:
2451 007512 013637 011234      MOV      @(SP)+,TEMP      ;MOVE COUNT TO TEMP
2452 007516 062746 000002      ADD      #2,-(SP)        ;ADJUST STACK
2453 007522                    1$:
2454 007522 104412                    ROMCLK                                ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2455 007524 021364                    021364                                ;PORT4 IBUS* REG11
2456 007526 032777 000002 172340  BIT      #2,@KMP04        ;IS PGM CLOCK BIT CLEAR?
2457 007534 001772                    BEQ      1$           ;BR IF YES
2458 007536                    2$:
2459 007536 104412                    ROMCLK                                ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2460 007540 021364                    021364                                ;PORT4 IBUS* REG11
2461 007542 032777 000002 172324  BIT      #2,@KMP04        ;IS PGM CLOCK BIT SET?
2462 007550 001372                    BNE      2$           ;BR IF YES
2463 007552 005337 011234      DEC      TEMP            ;DEC COUNT
2464 007556 001361                    BNE      1$           ;BR IF NOT DONE
2465 007560 000002                    RTI                    ;RETURN
2466
2467 007562 050200 051127 043040  MPFAIL: .ASCIZ <200>/PWR FAILED. RESTART AT TEST /
(2) 007620 042600 042116 050040  MEPASS: .ASCIZ <200>/END PASS CZCKEB /
(2) 007643      200 000122      MR:     .ASCIZ <200>/R/
(2) 007646 047200 020117 042504  MERR2: .ASCIZ <200>/NO DEVICES PRESENT./
(2) 007673      200 047111 052523  MERR3: .ASCIZ <200>/INSUFFICIENT DATA!/
(2) 007717      200 047514 045503  MLOCK: .ASCIZ <200>/LOCK ON SELECTED TEST/
(2) 007746 051503 035122 000040  MCSRX: .ASCIZ /CSR: /
(2) 007754 042526 035103 000040  MVECX: .ASCIZ /VEC: /
(2) 007762 040520 051523 051505  MPASSX: .ASCIZ /PASSES: /
(2) 007773      105 051122 051117  MERRX: .ASCIZ /ERRORS: /
(2) 010004 042524 052123 047040  MTSTN: .ASCIZ /TEST NO: /
(2) 010016 000052      MASTEK: .ASCIZ /*/
(2) 010020 051600 052105 051440  MNEW:  .ASCIZ <200>/SET SWITCH REG TO KMC11'S DESIRED ACTIVE./
(2) 010073      120 035103 000040  MERRPC: .ASCIZ /PC: /
(2) 010100 020200 020040 020040  XHEAD: .ASCII <200>/
(2) 010137      200 020040 020040      .ASCII <200>/
(2) 010176 020200 050040 020103      .ASCII <200>/ PC      CSR      STAT1      STAT2      STAT3/
(2) 010250 026600 026455 026455      .ASCII <200>/-----
(2) 010324 044200 053517 046440  NUM:   .ASCIZ <200>/HOW MANY KMC11'S TO BE TESTED?/
(2) 010364 041600 051123 040440  CSR:   .ASCIZ <200>/CSR ADDRESS?/
(2) 010402 053200 041505 047524  VEC:   .ASCIZ <200>/VECTOR ADDRESS?/

```

```

(2) 010423      200 051102 050040  PRI0:  .ASCIZ  <200>/BR PRIORITY LEVEL? (4,5,6,7)?/
(2) 010462 053600 044510 044103  MODU:  .ASCIZ  <200>/WHICH LINE UNIT? IF NONE TYPE 'N', IF M8201 TYPE '1', IF M8202 TYP
(2) 010574 053600 044510 044103  MV35:  .ASCII  <200>/WHICH MODEM TYPE, TYPE 'D' FOR KMC11-DA (RS232C),OR/
(2) 010660 052200 050131 020105  .ASCIZ  <200>/TYPE 'F' FOR KMC11-FA (V.35) ? /
(2)
(2) 010721      200 053523 052111  LINE:  .ASCIZ  <200>/SWITCH PAC#1 (DDCMP LINE #)?/
(2) 010757      200 053523 052111  BM:    .ASCIZ  <200>/SWITCH PAC#2 (BM873 BOOT ADD)?/
(2) 011017      200 051511 052040  CONN:  .ASCIZ  <200>/IS THE LOOP BACK CONNECTOR ON?/
(2) 011057      200 047516 042040  NOACT: .ASCIZ  <200>/NO DEVICES ARE SELECTED/
(2) 011110 100200 046513 030503  CONERR: .ASCIZ  <200><200>/KMC11 AT NONSTANDARD ADDRESS PC: /
(2) 011155      200 054105 042520  CNERR: .ASCIZ  <200>/EXPECTED FOUND/
(2) 011176 024040 046513 024503  KMC11: .ASCIZ  / (KMC) /
(2)
(2) 011206 000005  XSTATQ: 5
2468 011210      006      003      .BYTE  6,3
2469 011212 001276  STMP0
2470 011214      006      003      .BYTE  6,3
2471 011216 001300  STMP1
2472 011220      006      003      .BYTE  6,3
2473 011222 001302  STMP2
2474 011224      006      003      .BYTE  6,3
2475 011226 001304  STMP3
2476 011230      006      002      .BYTE  6,2
2477 011232 001306  STMP4
2478
2479      .EVEN
2480      ;BUFFERS FOR INPUT-OUTPUT
2481
2482 011234 000000  TEMP:  0
2483      .=.+40
2484 011276 000000  MDATA: 0
2485      .=.+40
2486
2487
2488      ;ROUTINE USED TO CHANGE SOFTWARE SWITCH
2489      ;REGISTER USING THE CONSOLE TERMINAL
2490      ;-----
2491
2492 011340 022737 000176 001240  CKSWR:  CMP      #SWREG,SWR      ;IS THE SOFT SWR BEING USED?
2493 011346 001075      BNE      CKSWR5      ;BR IF NO
2494 011350 132737 000001 001336  BITB    #1,$ENV      ; IS IT RUNNING UNDER APT?
2495 011356 001071      BNE      CKSWR5      ; EXIT IF YES.
2496 011360 022777 000007 167660  CMP     #7,@$TKB     ;WAS CTRL G TYPED? (7 BIT ASCII)
2497 011366 001404      BEQ     1$          ;BR IF YES
2498 011370 022777 000207 167650  CMP     #207,@$TKB  ;WAS CTRL G TYPED? (8 BIT ASCII)
2499 011376 001061      BNE      CKSWR5      ;BR IF NO
2500 011400 010246      1$:     MOV     R2,-(SP)    ;STORE R2
2501 011402 010346      MOV     R3,-(SP)    ;STORE R3
2502 011404 010446      MOV     R4,-(SP)    ;STORE R4
2503 011406 012737 177777 011544  MOV     #-1,$WFLG   ;SET SOFT TYPE OUT FLAG
2504 011414 005002      CKSWR1: CLR     R2      ;CLEAR NEW SWR CONTENTS
2505 011416 012704 177777      MOV     #-1,R4     ;SET FLAG TO ALL ONES
2506 011422 104401 005541      TYPE   ,$MSWR     ;TYPE "SWR= "
2507 011426 104417      CKSWR2: CNVRT     ;TYPE OUT PRESENT CONTENTS
2508 011430 011600      SOFTSW  ;OF SOFT SWITCH REGISTER
2509 011432 104401 005552      CKSWR3: TYPE     ,$MNEW ;TYPE "NEW? "
  
```

| | | | | | | | | |
|------|--------|--------|--------|---------|---------|------------|--------------------------------------|---|
| 2510 | 011436 | 004737 | 011546 | CKSWR4: | JSR | PC, INCHAR | :GET RESPONSE | |
| 2511 | 011442 | 022703 | 000015 | | CMP | #15,R3 | :WAS IT A CR? | |
| 2512 | 011446 | 001424 | | | BEQ | 5\$ | :BR IF YES | |
| 2513 | 011450 | 022703 | 000012 | | CMP | #12,R3 | :WAS IT A LF? | |
| 2514 | 011454 | 001416 | | | BEQ | 4\$ | :BR IF YES | |
| 2515 | 011456 | 022703 | 000025 | | CMP | #25,R3 | :WAS IT CTRL U? | |
| 2516 | 011462 | 001754 | | | BEQ | CKSWR1 | :BR IF YES(START OVER) | |
| 2517 | 011464 | 022703 | 000007 | | CMP | #7,R3 | :IF CNTL G GET NEXT CHAR | |
| 2518 | 011470 | 001762 | | | BEQ | CKSWR4 | | |
| 2519 | 011472 | 005004 | | | CLR | R4 | :IT MUST BE A DIGIT SO CLR FLAG | |
| 2520 | 011474 | 042703 | 177770 | | BIC | #177770,R3 | :ONLY 0-7 ARE LEGAL SO MASK OFF BITS | |
| 2521 | 011500 | 006302 | | | ASL | R2 | :SHIFT R2 3 TIMES | |
| 2522 | 011502 | 006302 | | | ASL | R2 | | |
| 2523 | 011504 | 006302 | | | ASL | R2 | | |
| 2524 | 011506 | 050302 | | | BIS | R3,R2 | :ADD LAST DIGIT | |
| 2525 | 011510 | 000752 | | | BR | CKSWR4 | :GET NEXT CHARACTER | |
| 2526 | 011512 | 012766 | 002402 | 000006 | 4\$: | MOV | #.START,6(SP) | :LF WAS TYPED SO GO TO START |
| 2527 | 011520 | 005704 | | | 5\$: | TST | R4 | :IS FLAG CLEAR? |
| 2528 | 011522 | 001002 | | | | BNE | 6\$ | :IF NOT DON'T CHANGE SOFT SWR |
| 2529 | 011524 | 010277 | 167510 | | | MOV | R2,@SWR | :IF YES THEN WRITE NEW CONTENTS TO SOFT SWR |
| 2530 | 011530 | 005037 | 011544 | | 6\$: | CLR | SWFLG | :CLEAR TYPEOUT FLAG |
| 2531 | 011534 | 012604 | | | | MOV | (SP)+,R4 | :RESTORE R4 |
| 2532 | 011536 | 012603 | | | | MOV | (SP)+,R3 | :RESTORE R3 |
| 2533 | 011540 | 012602 | | | | MOV | (SP)+,R2 | :RESTORE R2 |
| 2534 | 011542 | 000207 | | | CKSWR5: | RTS | PC | :RETURN |
| 2535 | | | | | | | | |
| 2536 | 011544 | 000000 | | | SWFLG: | 0 | | |
| 2537 | | | | | | | | |
| 2538 | 011546 | 105777 | 167472 | | INCHAR: | TSTB | @\$TKS | |
| 2539 | 011552 | 100375 | | | | BPL | .-4 | |
| 2540 | 011554 | 017703 | 167466 | | | MOV | @\$TKB,R3 | |
| 2541 | 011560 | 105777 | 167464 | | | TSTB | @\$TPS | |
| 2542 | 011564 | 100375 | | | | BPL | .-4 | |
| 2543 | 011566 | 010377 | 167460 | | | MOV | R3,@\$TPB | |
| 2544 | 011572 | 042703 | 000200 | | | BIC | #BIT7,R3 | |
| 2545 | 011576 | 000207 | | | | RTS | PC | |
| 2546 | | | | | | | | |
| 2547 | 011600 | 000001 | | | SOFTSW: | 1 | | |
| 2548 | 011602 | 006 | 002 | | | .BYTE | 6,2 | |
| 2549 | 011604 | 000176 | | | | SWREG | | |

```

2550
2551
2552
2553
2554
2555
2556
2557
2558
2559 011606 005737 001470      CYCLE: TST      KMACTV      ;ARE ANY KMC11'S TO BE TESTED?
2560 011612 001004          BNE      1$          ;BR IF OK.
2561 011614 104401 011057      TYPE     ,NOACT     ;NO KMC11'S SELECTED!!
2562 011620 000000          HALT     ;STOP THE SHOW.
2563 011622 000776          BR       -2         ;DISQUALIFY CONT. SW.
2564 011624 000241          1$:      CLC          ;CLEAR PROC. CARRY BIT.
2565 011626 006137 001500      ROL     RUN        ;UPDATE POINTER
2566 011632 005537 001500      ADC     RUN        ;CATCH CARRY FROM RUN
2567 011636 062737 000004 001504  ADD     #4,MILK    ;UPDATE POINTER
2568 011644 062737 000010 001502  ADD     #10,CREAM  ;UPDATE ADDRESS POINTER.
2569 011652 022737 002300 001502  CMP     #KM.MAP+200,CREAM
2570 011660 001006          BNE     2$          ;KEEP GOING; NOT ALL TESTED FOR.
2571 011662 012737 002100 001502  MOV     #KM.MAP,CREAM ;RESET ADDRESS POINTER.
2572 011670 012737 002302 001504  MOV     #CNT.MAP,MILK ;RESET PASS COUNT POINTER
2573 011676 033737 001500 001470  2$:     BIT     RUN,KMACTV ;IS THIS ONE ACTIVE?
2574 011704 001747          BEQ     1$          ;BR IF NO
2575 011706 013700 001502          MOV     CREAM,R0   ;GET ADDRESS POINTER
2576 011712 013702 001504          MOV     MILK,R2    ;GET PASS COUNT POINTER
2577 011716 012037 002066          MOV     (R0)+,KMCSR ;LOAD SYSTEM CTRL. REG
2578 011722 011037 002056          MOV     (R0),KMRVEC ;LOAD VECTOR
2579 011726 042737 177000 002056  BIC     #177000,KMRVEC ;CLEAR UNWANTED BITS
2580 011734 012037 002050          MOV     (R0)+,STAT1 ;LOAD STAT1
2581 011740 012037 002052          MOV     (R0)+,STAT2 ;LOAD STAT2
2582 011744 012037 002054          MOV     (R0)+,STAT3 ;LOAD STAT3
2583 011750 012237 001324          MOV     (R2)+,$PASS ;LOAD PASS COUNT
2584 011754 012237 001212          MOV     (R2)+,$ERTL ;LOAD ERROR COUNT
2585 011760 012700 000002          MOV     #2,R0     ;SAVE CORE THIS WAY!
2586 011764 013737 002066 002070  MOV     KMCSR,KMCSRH
2587 011772 005237 002070          INC     KMCSRH
2588 011776 013737 002070 002072  MOV     KMCSRH,KMCTL
2589 012004 005237 002072          INC     KMCTL
2590 012010 013737 002072 002074  MOV     KMCTL,KMPO4
2591 012016 060037 002074          ADD     R0,KMPO4
2592 012022 013737 002074 002076  MOV     KMPO4,KMPO6
2593 012030 060037 002076          ADD     R0,KMPO6
2594
2595 012034 013737 002056 002060  MOV     KMRVEC,KMRLVL ;PTY LVL
2596 012042 060037 002060          ADD     R0,KMRLVL
2597 012046 013737 002060 002062  MOV     KMRLVL,KMTVEC ;TX VEC
2598 012054 060037 002062          ADD     R0,KMTVEC
2599 012060 013737 002062 002064  MOV     KMTVEC,KMTLVL ;TX LVL
2600 012066 060037 002064          ADD     R0,KMTLVL
2601
2602 012072 032737 000002 001446  BIT     #SW01,STRTSW ;IS TEST NO. SELECTED
2603 012100 001447          BEQ     7$          ;BR IF NO
2604 012102
2605 012102 005737 000042          4$:     TST     @#42      ;RUNNING IN AUTO MODE?
    
```

```

2606 012106 001044          BNE      7$          :BR IF YES
2607 012110 104401 001313  TYPE      ,SCRLF
2608 012114 104415          INPUT
2609 012116 010004          MTSTN
2610 012120 000001          1
2611 012122 001000          1000
2612 012124 001202          $STNM
2613 012126 000          .BYTE 0
2614 012127 001          .BYTE 1
2615 012130 012700 014140  MOV      #TST1,R0
2616 012134 022710 5$:    CMP      (PC)+,(R0)    :CMP FIRST WORD TO 12737
2617 012136 012737          MOV      (PC)+,@(PC)+
2618 012140 001020          BNE      6$          :BR IF NOT SAME
2619 012142 023760 001202 000002  CMP      $STNM,2(R0)  :DOES $STNM MATCH?
2620 012150 001014          BNE      6$          :BR IF NO
2621 012152 022760 001202 000004  CMP      #$STNM,4(R0) :IS LAST WORD OK?
2622 012160 001010          BNE      6$          :BR IF NO
2623 012162 010037 001206  MOV      R0,$LPADR    :IT IS A LEGAL TEST SO DO IT
2624 012166 104401 007643  TYPE      ,MR
2625 012172 042737 000002 001446  BIC      #SW01,STRTSW
2626 012200 000412          BR       8$
2627 012202 005720 6$:    TST      (R0)+        :POP R0
2628 012204 020027 027716  CMP      R0,#T!LAST+10 :AT END YET?
2629 012210 001351          BNE      5$          :BR IF NO
2630 012212 104401 001312  TYPE      ,SQUES      :YES ILLEGAL TEST NO.
2631 012216 000731          BR       4$          :TRY AGAIN
2632
2633 012220 012737 014140 001206 7$:    MOV      #TST1,$LPADR  :PREPARE $LPADR ADDRESS
2634 012226 013701 002066 8$:    MOV      KMC11,R1     :R1 = BASE KMC11 ADDRESS
2635 012232 000177 166750  JMP      @B$LPADR     :GO START TESTING.
2636
2637
2638          :ROUTINE USED TO "AUTO SIZE" THE KMC11
2639          :CSR AND VECTOR.
2640          :NOTE: THE CSR MAY BE ANY WHERE IN THE FLOATING
2641          :      ADDRESS RANGE (160000:164000)
2642          :      AND THE VECTOR MAY BE ANY WHERE IN THE
2643          :      FLOATING VECTOR RANGE (300:770)
2644          :
2645          :
2646          AUTO.SIZE:
2647 012236 000005          RESET
2648 012240 012702 002100  CSRMAP: MOV      #KM.MAP,R2    :INSURE A BUS INIT.
2649 012244 005022 1$:    CLR      (R2)+        :LOAD MAP POINTER.
2650 012246 022702 002300  CMP      #KM.END,R2    :ZERO ENTIRE MAP
2651 012252 001374          BNE      1$          :ALL DONE?
2652 012254 005037 001472  CLR      KMNUM         :BR IF NO
2653 012260 012702 002100  MOV      #KM.MAP,R2    :SET OCTAL NUMBER OF KMC11'S TO 0
2654 012264 005037 001470  CLR      KMACTV        :R2 POINTS TO KMC MAP
2655 012270 032737 000001 001446  BIT      #SW00,STRTSW  :CLEAR ACTIVE
2656 012276 001002          BNE      .+6         :QUESTIONS?
2657 012300 000137 012740  JMP      7$          :BR IF YES
2658 012304 012737 000001 001306  MOV      #1,$TMP4     :IF NO SKIP QUESTIONS
2659 012312 104415          INPUT
2660 012314 010324          NUM
2661 012316 000001          1
  
```



```

2662 012320 000020      16.
2663 012322 001302      $TMP2
2664 012324      000      .BYTE 0
2665 012325      001      .BYTE 1
2666 012326 013737 001302 001472      MOV $TMP2,KMNUM ;KMNUM = HOW MANY
2667 012334 104401 001313      TYPE ,SCRLF
2668 012340 104416      CONVRT ;TYPE WHICH KMC IS BEING DONE
2669 012342 013372      WHICH ;$TMP4 IS WHICH KMC
2670 012344 005237 001306      INC $TMP4
2671 012350 104415      INPUT
2672 012352 010364      CSR
2673 012354 160000      160000
2674 012356 164000      164000
2675 012360 001304      $TMP3
2676 012362      000      .BYTE 0
2677 012363      001      .BYTE 1
2678 012364 013722 001304      MOV $TMP3,(R2)+ ;STORE CSR IN MAP
2679 012370 104415      INPUT
2680 012372 010402      VEC
2681 012374 000000      0
2682 012376 000776      776
2683 012400 001304      $TMP3
2684 012402      000      .BYTE 0
2685 012403      001      .BYTE 1
2686 012404 013712 001304      MOV $TMP3,(R2) ;STORE VECTOR IN MAP
2687 012410 104401      TYPE
2688 012412 010423      PRIO ;ASK WHAT BR LEVEL
2689 012414 004737 013664      JSR PC,INTTY ;GET RESPONSE
2690 012420 022703 000024      CMP #24,R3
2691 012424 101014      BHI 50$ ;BR IF LESS THAN 4
2692 012426 022703 000027      CMP #27,R3
2693 012432 103411      BLO 50$ ;BR IF GREATER THAN 7
2694 012434 012704 000011      MOV #11,R4 ;R4 = NUMBER OF SHIFTS
2695 012440 006303      ASL R3 ;SHIFT R3 LEFT
2696 012442 005304      DEC R4 ;DEC SHIFT COUNT
2697 012444 001375      BNE -4 ;BR IF NOT DONE
2698 012446 042703 170777      BIC #170777,R3 ;BIC UNWANTED BITS
2699 012452 050312      BIS R3,(R2) ;PUT BR LEVEL IN STATUS MAP
2700 012454 000403      BR 8$ ;CONTINUE
2701 012456 104401      TYPE
2702 012460 001312      $QUES ;RESPONSE IS OUT OF LIMITS
2703 012462 000752      BR 10$ ;TRY AGAIN
2704 012464
2705 012464
2706 012464
2707
2708 012464 104401      TYPE
2709 012466 010462      MODU
2710 012470 004737 013664      JSR PC,INTTY ;ASK WHICH LINE UNIT
2711 012474 022703 000021      CMP #21,R3 ;GET REPLY
2712 012500 001417      BEQ 30$ ;'1'
2713 012502 022703 000022      CMP #22,R3 ;'2'
2714 012506 001412      BEQ 31$
2715 012510 022703 000116      CMP #116,R3 ;'N'
2716 012514 001403      BEQ 32$
2717 012516 104401      TYPE

```

| | | | | | | | | | |
|------|--------|--------|--------|--------|--------|--------|---------------|--|-----------------------------------|
| 2718 | 012520 | 001312 | | | | \$QUES | | | ;IF NOT A 1,2 OR N TYPE "'?" |
| 2719 | 012522 | 000760 | | | | BR | 16\$ | | ;TRY AGAIN |
| 2720 | 012524 | 052722 | 010000 | | 32\$: | BIS | #BIT12,(R2)+ | | ;SET BIT 12 IN STAT2 IF NO LU |
| 2721 | 012530 | 022222 | | | | CMP | (R2)+,(R2)+ | | ;POP OVER STAT2 AND STAT3 |
| 2722 | 012532 | 000475 | | | | BR | 33\$ | | |
| 2723 | 012534 | 052712 | 020000 | | 31\$: | BIS | #BIT13,(R2) | | ;SET BIT 13 IN STAT2 IF M8202 |
| 2724 | 012540 | 104401 | | | 30\$: | TYPE | | | |
| 2725 | 012542 | 011017 | | | | CONN | | | ;ASK IF LOOP-BACK IS ON |
| 2726 | 012544 | 004737 | 013664 | | | JSR | PC,INTTY | | ;GET REPLY |
| 2727 | 012550 | 022703 | 000131 | | | CMP | #131,R3 | | ;Y |
| 2728 | 012554 | 001406 | | | | BEQ | 17\$ | | |
| 2729 | 012556 | 022703 | 000116 | | | CMP | #116,R3 | | ;N |
| 2730 | 012562 | 001436 | | | | BEQ | 18\$ | | |
| 2731 | 012564 | 104401 | | | | TYPE | | | |
| 2732 | 012566 | 001312 | | | | \$QUES | | | ;IF NOT Y OR N TYPE "'?" |
| 2733 | 012570 | 000763 | | | | BR | 30\$ | | ;TRY AGAIN |
| 2734 | 012572 | 052722 | 040000 | | 17\$: | BIS | #BIT14,(R2)+ | | ;TURNAROUND IS CONNECTED |
| 2735 | 012576 | 032762 | 020000 | 177776 | | BIT | #BIT13,-2(R2) | | ; M8202? |
| 2736 | 012604 | 001027 | | | | BNE | 19\$ | | |
| 2737 | 012606 | | | | 440\$: | | | | |
| 2738 | | | | | | | | | |
| 2739 | 012606 | 104401 | | | | TYPE | | | ; ASK QUESTION |
| 2740 | 012610 | 010574 | | | | MV35 | | | ; ABOUT MODEM TTYPE |
| 2741 | 012612 | 004737 | 013664 | | | JSR | PC,INTTY | | ; GET ANSWER. |
| 2742 | 012616 | 122703 | 000104 | | | CMPB | #'D,R3 | | ; IS IT DMC11-DA? |
| 2743 | 012622 | 001004 | | | | BNE | 442\$ | | ; NO. |
| 2744 | | | | | | | | | |
| 2745 | 012624 | 042762 | 000004 | 000002 | | BIC | #BIT2,2(R2) | | ; YES INDICATE IT IN STAT3 |
| 2746 | 012632 | 000411 | | | | BR | 441\$ | | |
| 2747 | | | | | | | | | |
| 2748 | 012634 | 122703 | 000106 | | 442\$: | CMPB | #'F,R3 | | ;IS IT A DMC11-FA (V.35)? |
| 2749 | 012640 | 001403 | | | | BEQ | 443\$ | | ; YES-TAKE CARE OF IT. |
| 2750 | | | | | | | | | |
| 2751 | 012642 | 104401 | | | | TYPE | | | ; NO ASK OPERATER WHATS GOING ON. |
| 2752 | 012644 | 001312 | | | | \$QUES | | | |
| 2753 | 012646 | 000757 | | | | BR | 440\$ | | ; REASK QUESTION |
| 2754 | | | | | | | | | |
| 2755 | 012650 | 052762 | 000004 | 000002 | 443\$: | BIS | #BIT2,2(R2) | | ; YES V.35, RECORD IN STAT3 |
| 2756 | | | | | | | | | |
| 2757 | 012656 | | | | 441\$: | | | | ; END DECISION POINT. |
| 2758 | 012656 | 000402 | | | | BR | 19\$ | | |
| 2759 | 012660 | 042722 | 040000 | | 18\$: | BIC | #BIT14,(R2)+ | | ;NO TURNAROUND |
| 2760 | 012664 | | | | 19\$: | | | | |
| 2761 | 012664 | 104415 | | | | INPUT | | | |
| 2762 | 012666 | 010721 | | | | LINE | | | |
| 2763 | 012670 | 000000 | | | | 0 | | | |
| 2764 | 012672 | 000377 | | | | 377 | | | |
| 2765 | 012674 | 001304 | | | | \$TMP3 | | | |
| 2766 | 012676 | 000 | | | | .BYTE | 0 | | |
| 2767 | 012677 | 001 | | | | .BYTE | 1 | | |
| 2768 | 012700 | 113722 | 001304 | | | MOVB | \$TMP3,(R2)+ | | ;STORE SWITCH PAC IN MAP |
| 2769 | 012704 | 104415 | | | | INPUT | | | |
| 2770 | 012706 | 010757 | | | | BM | | | |
| 2771 | 012710 | 000000 | | | | 0 | | | |
| 2772 | 012712 | 000377 | | | | 377 | | | |
| 2773 | 012714 | 001304 | | | | \$TMP3 | | | |

```

2774 012716 000 .BYTE 0
2775 012717 001 .BYTE 1
2776 012720 113722 001304 MOVB $TMP3,(R2)+ ;STORE SWITCH PAC IN MAP
2777 012724 005722 TST (R2)+ ;POP OVER STAT3
2778 012726 005337 001302 33$: DEC $TMP2 ;DEC KMC COUNT
2779 012732 001200 BNE 12$ ;BR IF MORE TO DO
2780 012734 000137 013272 JMP 13$ ;CONTINUE
2781 012740 012701 160000 7$: MOV #160000,R1 ;SET FOR FIRST ADDRESS TO BE TESTED
2782 012744 012737 013364 000004 MOV #6$,@#4 ;SET FOR NON-EXISTANT DEVICE TIME OUT
2783 012752 005011 2$: CLR (R1) ;CLEAR SEL0
2784 012754 005711 TST (R1) ;IF KMC11 KMCSR S/B 0
2785 012756 001135 BNE 3$ ;IF NO DEV ; TRAP TO 4. IF NO BIT 8 THEN NO KMC11
2786 012760 005061 000006 CLR 6(R1) ;CLEAR SEL6
2787 012764 005761 000006 TST 6(R1) ;IF KMC11 THEN KMRIC S/B =0!
2788 012770 001130 BNE 3$ ;BR IF NOT KMC11
2789 012772 012711 002000 MOV #BIT10,(R1) ;SET ROMO
2790 012776 005061 000004 CLR 4(R1) ;CLEAR SEL4
2791 013002 012761 125252 000006 MOV #125252,6(R1) ;WRITE THIS TO SEL6
2792 013010 052711 020000 BIS #BIT13,(R1) ;WRITE IT!
2793 013014 022761 125252 000004 CMP #125252,4(R1) ;WAS IT WRITTEN?
2794 013022 001113 BNE 3$ ;IF NO IT IS NOT CRAM
2795 ;AT THIS POINT IT IS ASSUMED THAT R1 HOLDS A KMC11 CSR ADDRESS.
2796 013024 21$:
2797 013024 010122 22$: MOV R1,(R2)+ ;STORE CSR IN CORE TABLE.
2798 013026 012711 001000 15$: MOV #BIT9,(R1) ;CLEAR LINE UNIT LOOP
2799 013032 005061 000004 CLR 4(R1) ;CLEAR PORT4
2800 013036 012761 122113 000006 MOV #122113,6(R1) ;LOAD INSTRUCTION (CLR DTR)
2801 013044 052711 000400 BIS #BIT8,(R1) ;CLOCK INSTRUCTION
2802 013050 012761 021264 000006 MOV #021264,6(R1) ;LOAD INSTRUCTION
2803 013056 052711 000400 BIS #BIT8,(R1) ;CLOCK INSTRUCTION
2804 013062 122761 000377 000004 CMPB #377,4(R1) ;IS IT ALL ONES?
2805 013070 001003 BNE .+10 ;BR IF NO
2806 013072 052712 010000 BIS #BIT12,(R2) ;IF YES, NO LINE UNIT, SET STATUS BIT
2807 013076 000436 BR 20$
2808 013100 032761 000002 000004 BIT #BIT1,4(R1) ;IS SWITCH A ONE?
2809 013106 001403 BEQ .+10 ;BR IF M8201
2810 013110 052712 060000 BIS #BIT13!BIT14,(R2) ;M8202 ASSUME CONNECTOR
2811 013114 000427 BR 20$ ;CONNECTOR ON)
2812 013116 032761 000010 000004 BIT #BIT3,4(R1) ;IS MRDY SET
2813 013124 001023 BNE 20$ ;BR IF M8201 NO CONNECTOR (ON LINE)
2814 013126 012761 000100 000004 MOV #BIT6,4(R1) ;LOAD PORT4
2815 013134 012761 122113 000006 MOV #122113,6(R1) ;LOAD INSTRUCTION
2816 013142 052711 000400 BIS #BIT8,(R1) ;CLOCK INSTRUCTION(SET DTR)
2817 013146 012761 021264 000006 MOV #021264,6(R1) ;LOAD INSTRUCTION
2818 013154 052711 000400 BIS #BIT8,(R1) ;CLOCK INSTRUCTION(READ MODEM REG)
2819 013160 032761 000010 000004 BIT #BIT3,4(R1) ;IS MRDY SET NOW?
2820 013166 001402 BEQ 20$ ;BR IF NO CONNECTOR
2821 013170 052712 040000 BIS #BIT14,(R2) ;SET STATUS BIT FOR CONNECTOR
2822 013174 005722 20$: TST (R2)+ ;POP POINTER
2823 013176 012761 021324 000006 MOV #021324,6(R1) ;PUT INSTRUCTION IN PORT6
2824 013204 012711 001400 MOV #BIT9!BIT8,(R1) ;PORT4_LU 15
2825 013210 156122 000004 BISB 4(R1),(R2)+ ;STORE DDCMP LINE # IN TABLE
2826 013214 012761 021344 000006 MOV #021344,6(R1) ;PORT6_INSTRUCTION
2827 013222 012711 001400 MOV #BIT8!BIT9,(R1) ;CLOCK INSTR.
2828 013226 156122 000004 BISB 4(R1),(R2)+ ;STORE BM873 ADD IN TABLE
2829 013232 005722 TST (R2)+ ;POP OVER STAT3

```

| | | | | | | | |
|------|--------|--------|--------|--------|-------------|-----------------|---|
| 2830 | 013234 | 005011 | | | CLR | (R1) | :CLEAR ROMI |
| 2831 | 013236 | 005237 | 001472 | | INC | KMNUM | :UPDATE DEVICE COUNTER |
| 2832 | 013242 | 022737 | 000020 | 001472 | CMP | #20,KMNUM | :ARE MAX. NO. OF DEV FOUND? |
| 2833 | 013250 | 001410 | | | BEQ | 13\$ | :YES DON'T LOOK FOR ANY MORE. |
| 2834 | 013252 | 005011 | | | 3\$: CLR | (R1) | :CLEAR BIT 10 |
| 2835 | 013254 | 005061 | 000006 | | CLR | 6(R1) | :CLEAR SEL 6 |
| 2836 | 013260 | 062701 | 000010 | | 14\$: ADD | #10,R1 | :UPDATE CSR POINTER ADDRESS |
| 2837 | 013264 | 022701 | 164000 | | CMP | #164000,R1 | |
| 2838 | 013270 | 001230 | | | BNE | 2\$ | :BR IF MORE ADDRESS TO CHECK. |
| 2839 | 013272 | 005037 | 001470 | | 13\$: CLR | KMACTV | |
| 2840 | 013276 | 005737 | 001472 | | TST | KMNUM | :WERE ANY KMC11'S FOUND AT ALL? |
| 2841 | 013302 | 001423 | | | BEQ | 5\$ | :ERROR AUTO SIZER FOUND NO KMC11'S IN THIS SYS. |
| 2842 | 013304 | 013701 | 001472 | | MOV | KMNUM,R1 | |
| 2843 | 013310 | 010137 | 001476 | | MOV | R1,SAVNUM | :SAVE NUMBER OF DEVICES |
| 2844 | 013314 | 000241 | | | 4\$: CLC | | |
| 2845 | 013316 | 006137 | 001470 | | ROL | KMACTV | :GENERATE ACTIVE REGISTER OF DEVICES. |
| 2846 | 013322 | 005237 | 001470 | | INC | KMACTV | :SET THE BIT |
| 2847 | 013326 | 005301 | | | DEC | R1 | |
| 2848 | 013330 | 001371 | | | BNE | 4\$ | :BR IF MORE TO GENERATE |
| 2849 | 013332 | 012737 | 000006 | 000004 | MOV | #6,@#4 | :RESTORE TRAP VECTOR |
| 2850 | 013340 | 013737 | 001470 | 001474 | MOV | KMACTV,SAVACT | :SAVE ACTIVE REGISTER |
| 2851 | 013346 | 000137 | 013400 | | JMP | VECMAP | :GO FIND THE VECTOR NOW. |
| 2852 | 013352 | 104401 | 007646 | | 5\$: TYPE | ,MERR2 | :NOTIFY OPR THAT NO KMC11'S FOUND. |
| 2853 | 013356 | 005000 | | | CLR | RO | :MAKE DATA LIGHTS ZERO |
| 2854 | 013360 | 000000 | | | HALT | | :STOP THE SHOW |
| 2855 | 013362 | 000776 | | | BR | .-2 | :DISABLE CONT. SW. |
| 2856 | 013364 | 012716 | 013260 | | 6\$: MOV | #14\$, (SP) | :ENTERED BY NON-EXISTANT TIME-OUT. |
| 2857 | 013370 | 000002 | | | RTI | | :RETURN TO MAINSTREAM |
| 2858 | | | | | | | |
| 2859 | 013372 | 000001 | | | WHICH: 1 | | |
| 2860 | 013374 | 002 | 002 | | .BYTE | 2,2 | |
| 2861 | 013376 | 001306 | | | \$TMP4 | | |
| 2862 | | | | | | | |
| 2863 | 013400 | 032737 | 000001 | 001446 | VECMAP: BIT | #SW00,STRTSW | |
| 2864 | 013406 | 001114 | | | BNE | 5\$ | |
| 2865 | 013410 | 012737 | 000340 | 000022 | MOV | #340,@#22 | :SET IOT TRAP PRIO TO 7 |
| 2866 | 013416 | 012737 | 013572 | 000020 | MOV | #4\$,@#20 | :SET IOT TRAP VECTOR |
| 2867 | 013424 | 012702 | 002100 | | MOV | #KM,MAP,R2 | :SET SOFTWARE POINTER |
| 2868 | 013430 | 012700 | 000300 | | MOV | #300,R0 | :FLOATING VECTORS START HERE. |
| 2869 | 013434 | 012701 | 000302 | | MOV | #302,R1 | :PC OF IOT INSTR. |
| 2870 | 013440 | 010120 | | | 1\$: MOV | R1,(R0)+ | :START FILLING VECTOR AREA |
| 2871 | 013442 | 012721 | 000004 | | MOV | #4,(R1)+ | :WITH +2: IOT |
| 2872 | 013446 | 022021 | | | CMP | (R0)+,(R1)+ | :ADD 2 TO R0 +R1 |
| 2873 | 013450 | 020127 | 001000 | | CMP | R1,#1000 | |
| 2874 | 013454 | 101771 | | | BLOS | 1\$ | :BR IF MORE TO FILL |
| 2875 | 013456 | 013737 | 001470 | 001276 | MOV | KMACTV,\$TMP0 | :STORE TEMPORALLY |
| 2876 | 013464 | 006037 | 001276 | | 2\$: ROR | \$TMP0 | :BRING OUT A BIT |
| 2877 | 013470 | 103063 | | | BCC | 5\$ | :BR IF ALL DONE |
| 2878 | 013472 | 012704 | 000012 | | MOV | #12,R4 | :R4 IS INDEX REGISTER |
| 2879 | 013476 | 016437 | 013650 | 177776 | MOV | BRLVL(R4),PS | :SET PS TO 7 |
| 2880 | 013504 | 011201 | | | MOV | (R2),R1 | |
| 2881 | 013506 | 012761 | 000200 | 000004 | MOV | #200,4(R1) | |
| 2882 | 013514 | 012711 | 001000 | | MOV | #BIT9,(R1) | :SET ROMI |
| 2883 | 013520 | 012761 | 121111 | 000006 | MOV | #121111,6(R1) | :PUT INSTRUCTION IN PORT6 |
| 2884 | 013526 | 012711 | 001400 | | MOV | #BIT9!BIT8,(R1) | :FORCE AN INTERRUPT |
| 2885 | 013532 | 105200 | | | 7\$: INCB | RO | :STALL |

```

2886 013534 001376          BNE      .-2          ;FOR TIME TO INTERUPT
2887 013536 162704 000002    SUB      #2,R4        ;GET NEXT LOWEST PS LEVEL
2888 013542 001404          BEQ      6$           ;BR IF R4 = 0
2889 013544 016437 013650 177776  MOV     BRLVL(R4),PS ;MOVE NEXT LOWER LEVEL IN PS
2890 013552 000767          BR       7$           ;BR TO DELAY
2891 013554 052762 005300 000002 6$:    BIS     #5300,2(R2)   ;NO INTERUPT ASSUME 300 AT LEVEL 5 AND FIX KMC11 LATER
2892 013562 005011          CLR     (R1)         ;CLEAR ROMI
2893 013564 062702 000010          ADD     #10,R2       ;POP SOFTWARE POINTER
2894 013570 000735          BR       2$           ;KEEP GOING
2895 013572 051662 000002          BIS     (SP),2(R2)   ;GET VECTOR ADDRESS
2896 013576 042762 000007 000002 4$:    BIC     #7,2(R2)     ;CLEAR JUNK
2897 013604 016405 013652          MOV     BRLVL+2(R4),R5 ;GET BR LEVEL OF KMC11
2898 013610 006305          ASL     R5           ;SHIFT LEVEL 4 PLACES
2899 013612 006305          ASL     R5           ;TO THE LEFT FOR THE
2900 013614 006305          ASL     R5           ;STATUS TABLE
2901 013616 006305          ASL     R5
2902 013620 042705 170777          BIC     #170777,R5   ;CLEAR UNWANTED BITS
2903 013624 050562 000002          BIS     R5,2(R2)    ;PUT BR LEVEL IN STATUS TABLE
2904 013630 022626          CMP     (SP)+,(SP)+ ;POP IOT JUNK OFF STACK
2905 013632 012716 013562          MOV     #3$,(SP)    ;SET FOR RETURN
2906 013636 000002          RTI
2907 013640 012737 004134 000020 5$:    MOV     #$$SCOPE,@#20 ; RESTORE SCOPE VECTOR
2908 013646 000207          RTS     PC          ;ALL DONE WITH 'AUTO SIZING'
2909
2910 013650 000000          BRLVL:  PRO         ;LEVEL 0
2911 013652 000000          PRO         ;LEVEL 0
2912 013654 000200          PR4        ;LEVEL 4
2913 013656 000240          PR5        ;LEVEL 5
2914 013660 000300          PR6        ;LEVEL 6
2915 013662 000340          PR7        ;LEVEL 7
2916
2917
2918 013664 105777 165354          INTTY:  TSTB     @STKS ;WAIT FOR DONE
2919 013670 100375          BPL     .-4
2920 013672 017703 165350          MOV     @STKB,R3    ;PUT CHAR IN R3
2921 013676 105777 165346          TSTB     @STPS     ;WAIT UNTIL PRINTER IS READY
2922 013702 100375          BPL     .-4
2923 013704 010377 165342          MOV     R3,@STPB   ;ECHO CHAR
2924 013710 042703 000240          BIC     #BIT7!BIT5,R3 ;MASK OFF LOWER CASE
2925 013714 000207          RTS     PC          ;RETURN
2926
2927 013716          APT.SIZE:
2928 013716 000005          RESET
2929 013720 010046          MOV     R0,-(SP)    ;;PUSH R0 ON STACK
2930 013722 010146          MOV     R1,-(SP)    ;;PUSH R1 ON STACK
2931 013724 010246          MOV     R2,-(SP)    ;;PUSH R2 ON STACK
2932 013726 010346          MOV     R3,-(SP)    ;;PUSH R3 ON STACK
2933 013730 005037 014132          CLR     VECTR      ; CLEAR THE LOCAL VARIABLE
2934 013734 005037 014136          CLR     PRIRTY     ; CLEAN UP LOCAL VARIABLE
2935 013740 013700 001376          MOV     $CDW1,R0   ; GET THE DEVICE COUNT
2936 013744 010037 001476          MOV     R0,SAVNUM  ; SAVE THE NO. OF DEVICES
2937 013750 012701 001346          MOV     #SMAMS1,R1 ; GET EXTRA INFO, BITS POINTER
2938 013754 013737 001372 014134          MOV     $BASE,BASE ; GET BASE CSR ADDRESS
2939 013762 113737 001366 014132          MOVB   $VECT1,VECTR ; GET THE VECTOR
2940 013770 113737 001367 014136          MOVB   $VECT1+1,PRIRTY ; GET THE PRIORITY
2941 013776 013737 001374 001470          MOV     $DEVN,KMACTV ; SAVE THE KMC'S SELECTED ACTIVE
    
```

```

2942 014004 013737 001470 001474      MOV    KMACTV,SAVACT      ; SAVE THE ACTIVE REGISTER
2943 014012 012702 001402              MOV    #SDDW0,R2         ; GET ADDRESS OF FIRST DEVICE DESCRIPTOR WORD
2944 014016 012703 002100              MOV    #KM.MAP,R3        ; GET POINTER TO DEVICE MAP
2945 014022 005023 3$: CLR    (R3)+             ; CLEAR DEVICE MAP
2946 014024 022703 002300              CMP    #KM.END,R3        ; IS WHOLE DEV.MAP CLEARED?
2947 014030 003374 3$: BGT    3$                ; NO, THEN GO ON.
2948 014032 012703 002100              MOV    #KM.MAP,R3        ; RESTORE DEV.MAP POINTER.
2949 014036 013723 014134 1$: MOV    BASE,(R3)+         ; LOAD CSR ADDRESS
2950 014042 112163 000001              MOV    (R1)+,1(R3)       ; GET EXTRA INFO. BITS
2951 014046 006213 1$: ASR    (R3)             ; SET IT IN RIGHT POSITION.
2952 014050 006213 1$: ASR    (R3)             ; SET IT IN RIGHT POSITION.
2953 014052 053713 014136              BIS    PRIRTY,(R3)       ; GET PRIORITY IN STAT1
2954 014056 006313 1$: ASL    (R3)             ; SET THEM IN RIGHT POSITION
2955 014060 006313 1$: ASL    (R3)             ; " " " " " "
2956 014062 006313 1$: ASL    (R3)             ; " " " " " "
2957 014064 006313 1$: ASL    (R3)             ; " " " " " "
2958 014066 053723 014132              BIS    VECTR,(R3)+       ; GET THE VECTOR IN STAT1.
2959 014072 012223 1$: MOV    (R2)+,(R3)+     ; GET THE STAT2 FROM DDWXX
2960 014074 005723 1$: TST    (R3)+             ; SKIP OVER STAT3
2961 014076 005300 1$: DEC    R0                ; COUNT BY 1
2962 014100 001407 1$: BEQ    2$                ; ALL DONE?
2963 014102 062737 000010 014134      ADD    #10,BASE          ; INCREMENT BASE CSR ADDRESS BY 10
2964 014110 062737 000010 014132      ADD    #10,VECTR         ; INCREMENT VECTOR ADDRESS BY 10
2965 014116 000747 1$: BR     1$                ; SET THE NEXT MAP ENTRY
2966 014120 2$:
2967 014120 012603 2$: MOV    (SP)+,R3         ;; POP STACK INTO R3
2968 014122 012602 2$: MOV    (SP)+,R2         ;; POP STACK INTO R2
2969 014124 012601 2$: MOV    (SP)+,R1         ;; POP STACK INTO R1
2970 014126 012600 2$: MOV    (SP)+,R0         ;; POP STACK INTO R0
2971 014130 000207 2$: RTS    PC                ; RETURN
2972 014132 000000 2$: VECTR: .WORD 0
2973 014134 000000 2$: BASE:  .WORD 0
2974 014136 000000 2$: PRIRTY: .WORD 0

```

```

2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997

```

```

:***** TEST 1 *****
:*OUT CONTROL REGISTER READ/ONLY TEST
:*DO A MASTER CLEAR, VERIFY THAT ALL READ/ONLY
:*BITS ARE IN THE CORRECT STATE
:*****

```

```

: TEST 1
:-----

```

```

2987 014140 000004 2987: TST1: SCOPE
2988 014142 012737 000001 001202      MOV    #1,$TSTNM        ; LOAD THE NO. OF THIS TEST
2989 014150 012737 014214 001442      MOV    #TST2,NEXT       ; POINT TO THE START OF NEXT TEST.
2990
2991 014156 005077 165704              CLR    @KMCSR           ; R1 CONTAINS BASE KMC11 ADDRESS
2992 014162 012702 000011              MOV    #11,R2          ; CLEAR SELO
2993 014166 104412              ROMCLK                 ; SAVE R2 FOR TYPEOUT
2994 014170 021224 021004!<20*11>          ; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2995 014172 016104 000004              MOV    4(R1),R4         ; PORT4 LINE UNIT REG 11
2996 014176 042704 000054              BIC    #54,R4          ; PUT "FOUND" IN R4
2997 014202 012705 000020              MOV    #20,R5          ; CLEAR UNKNOWN BITS
                          ; PUT "EXPECTED" IN R5

```

2998 014206 120504
2999 014210 001401
3000 014212 104002
3001 014214

1\$:

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

3002
3003
3004
3005
3006
3007
3008
3009
3010
3011

***** 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

3012
3013 014214 000004
3014 014216 012737 000002 001202
3015 014224 012737 014262 001442
3016
3017 014232 012702 000012
3018 014236 104412
3019 014240 021244
3020 014242 016104 000004
3021 014246 042704 000017
3022 014252 005005
3023 014254 120504
3024 014256 001401
3025 014260 104002
3026 014262

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
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\$:

3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037

***** 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

3038 014262 000004
3039 014264 012737 000003 001202
3040 014272 012737 014334 001442
3041
3042 014300 104410
3043 014302 012702 000013
3044 014306 104412
3045 014310 021264
3046 014312 016104 000004
3047 014316 042704 000213
3048 014322 012705 000100
3049 014326 120504
3050 014330 001401
3051 014332 104002
3052 014334
3053

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
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\$:

```

3054
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064 014334 000004
3065 014336 012737 000004 001202
3066 014344 012737 014426 001442
3067
3068 014352 104410
3069 014354 012702 000017
3070 014360 104412
3071 014362 021364
3072 014364 016104 000004
3073 014370 042704 000206
3074 014374 012705 000051
3075 014400 032737 020000 002050
3076 014406 001404
3077 014410 042704 000040
3078 014414 042705 000040
3079 014420 120504
3080 014422 001401
3081 014424 104002
3082 014426
3083
3084
3085
3086
3087
3088
3089
3090
3091
3092
3093
3094 014426 000004
3095 014430 012737 000005 001202
3096 014436 012737 014566 001442
3097 014444 012737 014460 001444
3098
3099 014452 104410
3100 014454 012702 000012
3101 014460 012761 000040 000004 1$:
3102 014466 104412
3103 014470 122112
3104 014472 104412
3105 014474 021245
3106 014476 012705 000040
3107 014502 116104 000005
3108 014506 042704 000337
3109 014512 120504

:***** 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
MOV #4,$STSTM ; LOAD THE NO. OF THIS TEST
MOV #TST5,NEXT ; POINT TO THE START OF NEXT TEST.
;R1 CONTAINS BASE KMC11 ADDRESS
MSTCLR ;MASTER CLEAR KMC11
MOV #17,R2 ;SAVE R2 FOR TYPEOUT
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
021004!<20*17> ;PORT4 LINE UNIT REG 17
MOV 4(R1),R4 ;PUT 'FOUND' IN R4
BIC #206,R4 ;CLEAR UNKNOWN BITS
MOV #51,R5 ;PUT 'EXPECTED' IN R5
BIT #BIT13,STAT1 ;IS LU AN M8202 OR M8201?
BEQ .+12 ;BR IF M8201
BIC #40,R4 ;MASK OFF SI BIT IF M8202
BIC #BIT5,R5 ;SI BIT IS UNKNOWN ON AN M8202
CMPB R5,R4 ;ARE SI AND ICIR SET?
BEQ 1$ ;BR IF YES
ERROR 2 ;ERROR IN LU 17

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
MOV #5,$STSTM ; LOAD THE NO. OF THIS TEST
MOV #TST6,NEXT ; POINT TO THE START OF NEXT TEST.
MOV #1$,LOCK ; ADDRESS FOR LOCK ON DATA.
;R1 CONTAINS BASE KMC11 ADDRESS
MSTCLR ;MASTER CLEAR KMC11
MOV #12,R2 ;SAVE REGISTER ADDRESS FOR TYPEOUT
MOV #40,4(R1) ;LOAD PORT4
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
122112 ;SET BITS IN LU-12
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
021245 ;READ LU-12
MOV #40,R5 ;PUT 'EXPECTED' IN R5
MOVB 5(R1),R4 ;PUT 'FOUND' IN R4
BIC #337,R4 ;CLEAR UNWANTED BITS
CMPB R5,R4 ;IS BITS SET?

```



```

3110 014514 001401          BEQ      2$          ;BR IF YES
3111 014516 104003          ERROR    3          ;ERROR, BIT 5 IS NOT SET
3112 014520 104405          2$: SCOPE1         ;SCOPE SUBTEST (SW09=1)
3113 014522 012737 014530 001444  MOV     #3$,LOCK    ;NEW SCOPE1
3114 014530 005061 000004  3$: CLR      4(R1)    ;LOAD PORT4
3115 014534 104412          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3116 014536 122112          122112         ;CLEAR BIT 5 IN LU-12
3117 014540 104412          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3118 014542 021245          021245         ;READ LU-12
3119 014544 005005          CLR      R5       ;PUT 'EXPECTED' IN R5
3120 014546 116104 000005  MOVB    5(R1),R4   ;PUT 'FOUND' IN R4
3121 014552 042704 000337  BIC     #337,R4    ;CLEAR UNWANTED BITS
3122 014556 120504          CMPB    R5,R4     ;IS BIT5 CLEAR?
3123 014560 001401          BEQ      4$          ;BR IF YES
3124 014562 104003          ERROR    3          ;ERROR, BIT5 IS NOT CLEAR
3125 014564 104405          4$: SCOPE1         ;SCOPE SUBTEST (SW09=1)
3126
3127
3128
3129
3130
3131
3132
3133
3134
3135
3136
3137 014566 000004          ;*****
3138 014570 012737 000006 001202  TST6: SCOPE          ;LOAD THE NO. OF THIS TEST
3139 014576 012737 014726 001442  MOV     #6,$STNM   ; POINT TO THE START OF NEXT TEST.
3140 014604 012737 014620 001444  MOV     #TST7,NEXT ; ADDRESS FOR LOCK ON DATA.
3141
3142 014612 104410          MOV     #1$,LOCK   ; R1 CONTAINS BASE KMC11 ADDRESS
3143 014614 012702 000017          MSTCLR          ;MASTER CLEAR KMC11
3144 014620 012761 000001 000004  1$: MOV     #17,R2   ;SAVE REGISTER ADDRESS FOR TYPEOUT
3145 014626 104412          MOV     #1,4(R1)   ;LOAD PORT4
3146 014630 122117          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3147 014632 104412          122117         ;SET BIT1 IN LU-17
3148 014634 021365          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3149 014636 012705 000001          021365         ;READ LU-17
3150 014642 116104 000005  MOV     #1,R5      ;PUT 'EXPECTED' IN R5
3151 014646 042704 000376  MOVB    5(R1),R4   ;PUT 'FOUND' IN R4
3152 014652 120504          BIC     #376,R4    ;CLEAR UNWANTED BITS
3153 014654 001401          CMPB    R5,R4     ;IS BIT1 SET?
3154 014656 104003          BEQ      2$          ;BR IF YES
3155 014660 104405          ERROR    3          ;ERROR, BIT 1 IS NOT SET
3156 014662 012737 014670 001444  2$: SCOPE1         ;SCOPE SUBTEST (SW09=1)
3157 014670 005061 000004  3$: MOV     #3$,LOCK    ;NEW SCOPE1
3158 014674 104412          CLR      4(R1)    ;LOAD PORT4
3159 014676 122117          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3160 014700 104412          122117         ;CLEAR BIT 1 IN LU-17
3161 014702 021365          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3162 014704 005005          021365         ;READ LU-17
3163 014706 116104 000005  CLR      R5       ;PUT 'EXPECTED' IN R5
3164 014712 042704 000376  MOVB    5(R1),R4   ;PUT 'FOUND' IN R4
3165 014716 120504          BIC     #376,R4    ;CLEAR UNWANTED BITS
3165 014716 120504          CMPB    R5,R4     ;IS BIT1 CLEAR?

```

```

:***** TEST 6 *****
:*LINE UNIT REGISTER WRITE/READ TEST
:*SET BIT1 IN LU REGISTER 17, VERIFY IT IS SET
:*CLEAR BIT1 IN LU REGISTER 17, VERIFY IT IS CLEAR
:*****

```

: TEST 6

```

:*****
TST6:
:*****

```

```

3166 014720 001401      BEQ      4$      :BR IF YES
3167 014722 104003      ERROR    3      :ERROR, BIT1 IS NOT CLEAR
3168 014724 104405      4$: SCOPE1    :SCOPE SUBTEST (SW09=1)
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3180 014726 000004      :***** TEST 7 *****
3181 014730 012737 000007 001202  IST7: SCOPE      :*LINE UNIT REGISTER WRITE/READ TEST
3182 014736 012737 015136 001442  MOV      #7,$STSTNM      :*FLOAT A 1 THROUGH LINE UNIT REGISTER 13
3183 014744 012737 014764 001444  MOV      #TST10,NEXT     :*FLOAT A 0 THROUGH LINE UNIT REGISTER 13
3184
3185 014752 104410      MOV      #64$,LOCK      :*****
3186 014754 012702 000013  MSTCLR
3187 014760 012700 000001  MOV      #13,R2          : TEST 7
3188 014764
3189 014764 010061 000004  MOV      #1,R0          :-----
3190 014770 042761 000257 000004  MOV      R0,4(R1)       :*****
3191 014776 104412      BIC      #257,4(R1)     :SCOPE
3192 015000 122113      ROMCLK   :LOAD THE NO. OF THIS TEST
3193 015002 104412      122100!13              :POINT TO THE START OF NEXT TEST.
3194 015004 021265      ROMCLK   :ADDRESS FOR LOCK ON DATA.
3195 015006 010005      21005!<13*20>          :R1 CONTAINS BASE KMC11 ADDRESS
3196 015010 042705 000257  MOV      R0,R5          :MASTER CLEAR KMC11
3197 015014 116104 000005  BIC      #257,R5        :SAVE REGISTER ADDRESS FOR TYPEOUT
3198 015020 042704 000257  MOV      5(R1),R4       :START WITH BIT J
3199 015024 120504      CMPB    R5,R4          :PUT PATTERN INTO PORT4
3200 015026 001401      BEQ      65$           :CLEAR UNWANTED BITS
3201 015030 104003      ERROR    3            :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3202 015032 104405      65$: SCOPE1          :MOV DATA TO IBUS REGISTER 13
3203 015034 000241      CLC
3204 015036 106100      ROLB    R0             :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3205 015040 001351      BNE     64$           :READ FROM IBUS REGISTER 13
3206 015042 012737 015056 001444  MOV      #67$,LOCK     :PUT EXPECTED IN R5
3207 015050 012700 000001  MOV      #1,R0          :CLEAR UNWANTED BITS
3208 015054 005100      69$: COM      R0       :PUT 'FOUND' INTO R4
3209 015056      67$:          :CLEAR UNWANTED BITS
3210 015056 010061 000004  MOV      R0,4(R1)       :DATA CORRECT?
3211 015062 042761 000257 000004  BIC      #257,4(R1)     :BR IF YES
3212 015070 104412      ROMCLK   :SW09=1?
3213 015072 122113      122100!13              :CLEAR CARRY
3214 015074 104412      ROMCLK   :SHIFT BIT IN R0
3215 015076 021265      21005!<13*20>          :IF R0=0 THEN DONE
3216 015100 010005      MOV      R0,R5          :NEW SCOPE1
3217 015102 042705 000257  BIC      #257,R5        :START WITH BIT 0
3218 015106 116104 000005  MOV      5(R1),R4       :CHANGE TO FLOATING ZERO
3219 015112 042704 000257  BIC      #257,R4        :PUT PATTERN INTO PORT4
3220 015116 120504      CMPB    R5,R4          :CLEAR UNWANTED BITS
3221 015120 001401      BEQ      68$           :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304

```

LINE UNIT WRITE/READ TESTS

3222 015122 104003
 3223 015124 104405
 3224 015126 005100
 3225 015130 000241
 3226 015132 106100
 3227 015134 001347
 3228
 3229
 3230
 3231
 3232
 3233
 3234
 3235
 3236
 3237
 3238
 3239 015136 000004
 3240 015140 012737 000010 001202
 3241 015146 012737 015312 001442
 3242 015154 012737 015174 001444
 3243
 3244 015162 104410
 3245 015164 012702 000014
 3246 015170 012700 000001
 3247 015174
 3248 015174 010061 000004
 3249 015200 104412
 3250 015202 122114
 3251 015204 104412
 3252 015206 021305
 3253 015210 010005
 3254 015212 116104 000005
 3255 015216 120504
 3256 015220 001401
 3257 015222 104003
 3258 015224 104405
 3259 015226 000241
 3260 015230 106100
 3261 015232 001360
 3262 015234 012737 015250 001444
 3263 015242 012700 000001
 3264 015246 005100
 3265 015250
 3266 015250 010061 000004
 3267 015254 104412
 3268 015256 122114
 3269 015260 104412
 3270 015262 021305
 3271 015264 010005
 3272 015266 116104 000005
 3273 015272 120504
 3274 015274 001401
 3275 015276 104003
 3276 015300 104405
 3277 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,R0 ;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

65$: ERROR 3 ;ERROR
SCOPI ;SW09=1?
CLC ;CLEAR CARRY
ROLB RO ;SHIFT BIT IN RO
BNE 64$ ;IF RO=0 THEN DONE
MOV #67$,LOCK ;NEW SCOPI
MOV #1,R0 ;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

68$: ERROR 3 ;ERROR
SCOPI ;SW09=1?
COM RO ;CHANGE TO FLOATING 1
    
```

```

3278 015304 000241          CLC          :CLEAR CARRY
3279 015306 106100          ROLB      RO          :SHIFT BIT IN RO
3280 015310 001356          BNE      69$         :IF RO=0 THEN DONE
3281
3282
3283          :***** TEST 11 *****
3284          :*SWITCH PAC TEST
3285          :*THIS TEST READS SWITCH PAC#1
3286          :*THIS SWITCH PAC CONTAINS THE DDCMP LINE #
3287          :*****
3288
3289          ; TEST 11
3290          :-----
3291          :*****
3292 015312 000004          TST11: SCOPE
3293 015314 012737 000011 001202  MOV      #11,$STNM      ; LOAD THE NO. OF THIS TEST
3294 015322 012737 015354 001442  MOV      #TST12,NEXT    ; POINT TO THE START OF NEXT TEST.
3295          :R1 CONTAINS BASE KMC11 ADDRESS
3296 015330 104410          MSTCLR          :MASTER CLEAR KMC11
3297 015332 104412          ROMCLK          :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3298 015334 021324          021324          :PORT4 LU15
3299 015336 016104 000004  MOV      4(R1),R4       :PUT 'FOUND' IN R4
3300 015342 113705 002052  MOVB    STAT2,R5       :PUT 'EXPECTED' IN R5
3301 015346 120504          CMPB    R5,R4         :SW OK?
3302 015350 001401          BEQ     1$           :BR IF YES
3303 015352 104031          ERROR   31          :ERROR, SWITCH PAC READ ERROR
3304 015354
3305
3306
3307          :***** TEST 12 *****
3308          :*SWITCH PAC TEST
3309          :*THIS TEST READS SWITCH PAC#2
3310          :*THIS SWITCH PAC CONTAINS THE BM873 BOOT ADD
3311          :*****
3312
3313          ; TEST 12
3314          :-----
3315          :*****
3316 015354 000004          TST12: SCOPE
3317 015356 012737 000012 001202  MOV      #12,$STNM      ; LOAD THE NO. OF THIS TEST
3318 015364 012737 015416 001442  MOV      #TST13,NEXT    ; POINT TO THE START OF NEXT TEST.
3319          :R1 CONTAINS BASE KMC11 ADDRESS
3320 015372 104410          MSTCLR          :MASTER CLEAR KMC11
3321 015374 104412          ROMCLK          :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3322 015376 021344          021344          :PORT4 LU16
3323 015400 016104 000004  MOV      4(R1),R4       :PUT 'FOUND' IN R4
3324 015404 113705 002053  MOVB    STAT2+1,R5     :PUT 'EXPECTED' IN R5
3325 015410 120504          CMPB    R5,R4         :SW OK?
3326 015412 001401          BEQ     1$           :BR IF YES
3327 015414 104031          ERROR   31          :ERROR, SWITCH PAC READ ERROR
3328 015416
3329
3330
3331          :***** TEST 13 *****
3332          :*LINE UNIT CLOCK TEST
3333          :*THIS TEST VERIFYS THAT THE LU INTERNAL CLOCK

```

```
3334 ;*(BIT 1 IN LU-17) IS WORKING
3335 ;*****
3336
3337 ; TEST 13
3338 ;-----
3339 ;*****
3340 TST13: SCOPE
3341 015416 000004 MOV #13,$TSTNM ; LOAD THE NO. OF THIS TEST
3342 015420 012737 000013 001202 MOV #TST14,NEXT ; POINT TO THE START OF NEXT TEST.
3343 015426 012737 015516 001442 ;R1 CONTAINS BASE KMC11 ADDRESS
3344 015434 104410 MSTCLR ;MASTER CLEAR KMC11
3345 015436 005037 011234 CLR TEMP ;PREPARE FOR DELAY
3346 015442 1$:
3347 015442 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3348 015444 021364 021364 ;PORT4 LU-17
3349 015446 032761 000002 000004 BIT #2,4(R1) ;IS CLOCK BIT SET?
3350 015454 001004 BNE 2$ ;BR IF YES
3351 015456 005237 011234 INC TEMP ;DELAY
3352 015462 001367 BNE 1$ ;DELAY FINISHED?
3353 015464 104004 ERROR 4 ;ERROR BIT IS STUCK CLEAR
3354 015466 005037 011234 2$: CLR TEMP ;PREPARE FOR DELAY
3355 015472 3$:
3356 015472 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3357 015474 021364 021364 ;PORT4 LU-17
3358 015476 032761 000002 000004 BIT #2,4(R1) ;IS CLOCK BIT CLEAR?
3359 015504 001404 BEQ 4$ ;BR IF YES
3360 015506 005237 011234 INC TEMP ;DELAY
3361 015512 001367 BNE 3$ ;BR IF DELAY NOT DONE
3362 015514 104004 ERROR 4 ;ERROR BIT IS STUCK SET
3363 015516 4$:
3364
3365
3366 ;***** TEST 14 *****
3367 ;*OUT DATA SILO TEST
3368 ;*SET SOM AND LOAD OUT DATA SILO
3369 ;*VERIFY THAT OCOR SET, INDICATING THAT THE
3370 ;*CHARACTER IS AT THE BOTTOM OF THE OUT SILO
3371 ;*****
3372
3373 ; TEST 14
3374 ;-----
3375 ;*****
3376 TST14: SCOPE
3377 015516 000004 MOV #14,$TSTNM ; LOAD THE NO. OF THIS TEST
3378 015520 012737 000014 001202 MOV #TST15,NEXT ; POINT TO THE START OF NEXT TEST.
3379 015526 012737 015616 001442 ;R1 CONTAINS BASE KMC11 ADDRESS
3380 015534 104410 MSTCLR ;MASTER CLEAR KMC11
3381 015536 012711 004000 MOV #BIT11,(R1) ;SET LINE UNIT LOOP
3382 015542 012761 000001 000004 MOV #1,4(R1) ;LOAD PORT4 WITH BIT0
3383 015550 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3384 015552 122111 122111 ;SET SOM
3385 015554 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3386 015556 122110 122110 ;LOAD OUT DATA SILO
3387 015560 104414 000002 TIMER, 2 ;WAIT FOR OCOR
3388 015564 012702 000017 MOV #17,R2 ;SAVE ADDRESS FOR TYPEOUT
3389 015570 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
```

```

3390 015572 021364          021364          :PORT4 LU 17
3391 015574 016104 000004    MOV      4(R1),R4      :PUT 'FOUND' IN R4
3392 015600 042704 000357    BIC      #357,R4      :CLEAR UNWANTED BITS
3393 015604 012705 000020    MOV      #20,R5       :PUT 'EXPECTED' IN R5
3394 015610 120504          CMPB     R5,R4        :IS OCOR SET?
3395 015612 001401          BEQ      1$           :BR IF YES
3396 015614 104005          ERROR    5
3397 015616          1$:
3398
3399
3400          :***** TEST 15 *****
3401          :*DDCMP TEST OF RTS AND OUT ACTIVE
3402          :*SET SOM AND LOAD OUT DATA SILO
3403          :*SINGLE STEP 2 DATA CLOCKS, VERIFY
3404          :*THAT RTS AND ACTIVE ARE SET
3405          :*****
3406
3407          : TEST 15
3408          :-----
3409          :*****
3410 015616 000004          TST15: SCOPE
3411 015620 012737 000015 001202    MOV      #15,$STNM    ; LOAD THE NO. OF THIS TEST
3412 015626 012737 015754 001442    MOV      #TST16,NEXT  ; POINT TO THE START OF NEXT TEST.
3413          :R1 CONTAINS BASE KMC11 ADDRESS
3414 015634 104410          MSTCLR
3415 015636 012711 004000          MOV      #BIT11,(R1)  ;MASTER CLEAR KMC11
3416 015642 012761 000001 000004    MOV      #1,4(R1)     ;SET LINE UNIT LOOP
3417 015650 104412          ROMCLK
3418 015652 122111          122111              ;LOAD PORT4 WITH BIT0
3419 015654 104412          ROMCLK              ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3420 015656 122110          122110              ;SET SOM
3421 015660 004737 030260          JSR      PC,OCOR      ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3422 015664 104413 000002          DATACLK, 2          ;LOAD OUT DATA SILO
3423 015670 012702 000011          MOV      #11,R2      ;WAIT FOR OCOR
3424 015674 104412          ROMCLK              ;CLOCK DATA FOUR TIMES
3425 015676 021224          021224              ;SAVE ADDRESS FOR TYPEOUT
3426 015700 016104 000004          MOV      4(R1),R4     ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3427 015704 042704 000257          BIC      #257,R4      ;PORT4 LU 11
3428 015710 012705 000120          MOV      #120,R5     ;PUT 'FOUND' IN R4
3429 015714 120504          CMPB     R5,R4        ;CLEAR UNWANTED BITS
3430 015716 001401          BEQ      1$           ;PUT 'EXPECTED' IN R5
3431 015720 104005          ERROR    5           ;IS ACTIVE SET?
3432          1$:
3433 015722 012702 000013          MOV      #13,R2      ;BR IF YES
3434 015726 104412          ROMCLK              ;SAVE ADDRESS FOR TYPEOUT
3435 015730 021264          021264              ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3436 015732 016104 000004          MOV      4(R1),R4     ;PORT4 LU 13
3437 015736 042704 000337          BIC      #337,R4      ;PUT EXPECTED IN R4
3438 015742 012705 000040          MOV      #BIT5,R5    ;CLEAR UNWANTED BITS
3439 015746 120504          CMPB     R5,R4        ;PUT 'EXPECTED' IN R5, RTS SHOULD BE SET
3440 015750 001401          BEQ      2$           ;IS RTS OK?
3441 015752 104005          ERROR    5           ;BR IF YES
3442          2$:
3443          :RTS ERROR
3444
3445          :***** TEST 16 *****
    
```

```

3446      ;*TEST OF OUT CLEAR
3447      ;*SET SOM AND LOAD OUT DATA SILO
3448      ;*SINGLE STEP DATA CLOCK, SET OUT CLEAR
3449      ;*VERIFY THAT OCOR,RTS, AND ACTIVE ARE CLEARED
3450      ;:*****
3451
3452      ; TEST 16
3453      ;-----
3454      ;:*****
3455      TST16: SCOPE
3456      MOV      #16,$STNM          ; LOAD THE NO. OF THIS TEST
3457      MOV      #TST17,NEXT      ; POINT TO THE START OF NEXT TEST.
3458
3459      MSTCLR          ;R1 CONTAINS BASE KMC11 ADDRESS
3460      MOV      #BIT11,(R1)      ;MASTER CLEAR KMC11
3461      MOV      #1,4(R1)        ;SET LINE UNIT LOOP
3462      ROMCLK          ;LOAD PORT4 WITH BIT0
3463      122111          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3464      ROMCLK          ;SET SOM
3465      122110          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3466      JSR      PC,OCOR        ;LOAD OUT DATA SILO
3467      DATACLK, 2      ;WAIT FOR OCOR
3468      MOV      #BIT7,4(R1)    ;CLOCK DATA FOUR TIMES
3469      ROMCLK          ;SET BIT7 IN PORT4
3470      122111          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3471      DATACLK, 1      ;SET OUT CLEAR
3472      MOV      #17,R2        ;GIVE A TICK TO CLEAR RTS
3473      ROMCLK          ;SAVE ADDRESS FOR TYPEOUT
3474      021364          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3475      MOV      4(R1),R4      ;PORT4 LU 17
3476      BIC      #357,R4      ;PUT 'FOUND' IN R4
3477      CLR      R5           ;CLEAR UNWANTED BITS
3478      CMPB    R5,R4        ;PUT 'EXPECTED' IN R5
3479      BEQ     1$          ;IS OCOR CLEARED?
3480      ERROR   5           ;BR IF YES
3481
3482      1$: MOV      #13,R2      ;SAVE ADDRESS FOR TYPEOUT
3483      ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3484      021264          ;PORT4 LU 13
3485      MOV      4(R1),R4      ;PUT EXPECTED IN R4
3486      BIC      #337,R4      ;CLEAR UNWANTED BITS
3487      CLR      R5           ;PUT 'EXPECTED' IN R5, RTS SHOULD BE CLEARED
3488      CMPB    R5,R4        ;IS RTS OK?
3489      BEQ     2$          ;BR IF YES
3490      ERROR   5           ;RTS ERROR
3491
3492      2$: MOV      #11,R2      ;SAVE ADDRESS FOR TYPEOUT
3493      ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3494      021224          ;PORT4 LU11
3495      MOV      4(R1),R4      ;PUT 'FOUND' IN R4
3496      MOV      #BIT4,R5      ;ONLY OUT READY SHOULD BE SET
3497      CMPB    R5,R4        ;IS ACTIVE CLEAR?
3498      BEQ     3$          ;BR IF YES
3499      ERROR   5           ;ERROR ACTIVE NOT CLEARED
3500
3501
    
```

3502
 3503
 3504
 3505
 3506
 3507
 3508
 3509
 3510
 3511
 3512
 3513
 3514
 3515
 3516
 3517
 3518
 3519
 3520
 3521
 3522
 3523
 3524
 3525
 3526
 3527
 3528
 3529
 3530
 3531
 3532
 3533
 3534
 3535
 3536
 3537
 3538
 3539
 3540
 3541
 3542
 3543
 3544
 3545
 3546
 3547
 3548
 3549
 3550
 3551
 3552
 3553
 3554
 3555
 3556
 3557

```

***** 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,$TSTNM ; LOAD THE NO. OF THIS TEST
MOV #TST20,NEXT ; POINT TO THE START OF NEXT TEST.
;R1 CONTAINS BASE KMC11 ADDRESS
MSTCLR ;MASTER CLEAR KMC11
MOV #BIT11,(R1) ;SET LINE UNIT LOOP
JSR PC,OUTRDY ;WAIT FOR OUT-READY
MOV #1,4(R1) ;SET BIT0 IN PORT4
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
122111 ;SET SOM!
MOV #0,R5 ;LOAD CHARACTER IN R5 FOR TYPEOUT
JSR PC,OUTRDY ;WAIT FOR OUT-READY
MOV R5,4(R1) ;LOAD PORT4 WITH CHARACTER
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
122110 ;LOAD OUT DATA
JSR PC,OCOR ;WAIT FOR OCOR TO SET
CLR R3 ;CLEAR BIT COUNTER
MOV R5,R2 ;LOAD CHARACTER IN R2
DATACLK, 2 ;2 TICKS TO SET UP TRANSMITTER
1$: 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$ ;CONTINE 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
    
```



```

3558 ;*SINGLE CLOCK THE CHARACTER 125
3559 ;*VERIFY EACH BIT POSITION AS IT
3560 ;*PASSES THE BIT WINDOW (SI BIT)
3561 ;*ON AN ERROR, R3 CONTAINS BIT POSITION OF FAILURE
3562 ;*****
3563
3564 ; TEST 20
3565 ;-----
3566 ;*****
3567 016334 000004 TST20: SCOPE
3568 016336 012737 000020 001202 MOV #20,$STNM ; LOAD THE NO. OF THIS TEST
3569 016344 012737 016516 001442 MOV #TST21,NEXT ; POINT TO THE START OF NEXT TEST.
3570 ;R1 CONTAINS BASE KMC11 ADDRESS
3571 016352 104410 MSTCLR ;MASTER CLEAR KMC11
3572 016354 012711 004000 MOV #BIT11,(R1) ;SET LINE UNIT LOOP
3573 016360 004737 030412 JSR PC,OUTRDY ;WAIT FOR OUT-READY
3574 016364 012761 000001 000004 MOV #1,4(R1) ;SET BIT0 IN PORT4
3575 016372 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3576 016374 122111 122111 ;SET SOM!
3577 016376 012705 000125 MOV #125,R5 ;LOAD CHARACTER IN R5 FOR TYPEOUT
3578 016402 004737 030412 JSR PC,OUTRDY ;WAIT FOR OUT-READY
3579 016406 010561 000004 MOV R5,4(R1) ;LOAD PORT4 WITH CHARACTER
3580 016412 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3581 016414 122110 122110 ;LOAD OUT DATA
3582 016416 004737 030260 JSR PC,OCOR ;WAIT FOR OCOR TO SET
3583 016422 005003 CLR R3 ;CLEAR BIT COUNTER
3584 016424 010502 MOV R5,R2 ;LOAD CHARACTER IN R2
3585 016426 104413 000002 DATACLK, 2 ;2 TICKS TO SET UP TRANSMITTER
3586 016432 104413 000001 1$: DATACLK, 1 ;SHIFT NEXT BIT IN THE WINDOW (SI BIT)
3587 016436 106002 RORB R2 ;SHIFT NEXT SOFTWARE BIT IN TO CARRY
3588 016440 103005 BCC 2$ ;BR IF CARRY CLEAR
3589 016442 004737 030226 JSR PC,GETSI ;GET THE WINDOW
3590 016446 103406 BCS 3$ ;BR IF BIT IS A MARK
3591 016450 104006 ERROR 6 ;ERROR BIT WAS A SPACE
3592 016452 000404 BR 3$ ;CONTINUE WITH TEST
3593 016454 004737 030226 2$: JSR PC,GETSI ;GET THE WINDOW
3594 016460 103001 BCC 3$ ;BR IF BIT IS A SPACE
3595 016462 104006 ERROR 6 ;ERROR BIT WAS A MARK
3596 016464 3$:
3597 016464 005203 INC R3 ;NEXT BIT
3598 016466 022703 000010 CMP #10,R3 ;DONE YET?
3599 016472 001357 BNE 1$ ;BR IF NO
3600 016474 104413 000014 DATACLK, 14 ;CLOCK TRANSMITTER 14 MORE TICKS
3601 016500 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3602 016502 021264 021264 ;PORT4 LU-13
3603 016504 032761 000040 000004 BIT #BITS,4(R1) ;RTS SHOULD BE CLEAR NOW
3604 016512 001401 BEQ 4$ ;BR IF YES
3605 016514 104034 ERROR 34 ;ERROR, RTS NOT CLEAR
3606 016516 4$:
3607
3608
3609 ;***** TEST 21 *****
3610 ;*DDCMP TRANSMITTER TEST
3611 ;*SINGLE CLOCK THE CHARACTER 252
3612 ;*VERIFY EACH BIT POSITION AS IT
3613 ;*PASSES THE BIT WINDOW (SI BIT)
    
```



```

3670 ; TEST 22
3671 ;-----
3672 ;*****
3673 016700 000004 TST22: SCOPE
3674 016702 012737 000022 001202 MOV #22,$STNM ; LOAD THE NO. OF THIS TEST
3675 016710 012737 017062 001442 MOV #TST23,NEXT ; POINT TO THE START OF NEXT TEST.
3676 ;R1 CONTAINS BASE KMC11 ADDRESS
3677 016716 104410 MSTCLR ;MASTER CLEAR KMC11
3678 016720 012711 004000 MOV #BIT11,(R1) ;SET LINE UNIT LOOP
3679 016724 004737 030412 JSR PC,OUTRDY ;WAIT FOR OUT-READY
3680 016730 012761 000001 000004 MOV #1,4(R1) ;SET BIT0 IN PORT4
3681 016736 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3682 016740 122111 122111 ;SET SOM!
3683 016742 012705 000377 MOV #377,R5 ;LOAD CHARACTER IN R5 FOR TYPEOUT
3684 016746 004737 030412 JSR PC,OUTRDY ;WAIT FOR OUT-READY
3685 016752 010561 000004 MOV R5,4(R1) ;LOAD PORT4 WITH CHARACTER
3686 016756 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3687 016760 122110 122110 ;LOAD OUT DATA
3688 016762 004737 030260 JSR PC,OCOR ;WAIT FOR OCOR TO SET
3689 016766 005003 CLR R3 ;CLEAR BIT COUNTER
3690 016770 010502 MOV R5,R2 ;LOAD CHARACTER IN R2
3691 016772 104413 000002 DATACLK, 2 ;2 TICKS TO SET UP TRANSMITTER
3692 016776 104413 000001 1$: DATACLK, 1 ;SHIFT NEXT BIT IN THE WINDOW (SI BIT)
3693 017002 106002 RORB R2 ;SHIFT NEXT SOFTWARE BIT IN TO CARRY
3694 017004 10300 BCC 2$ ;BR IF CARRY CLEAR
3695 017006 004737 030226 JSR PC,GETSI ;GET THE WINDOW
3696 017012 103406 BCS 3$ ;BR IF BIT IS A MARK
3697 017014 104006 ERROR 6 ;ERROR BIT WAS A SPACE
3698 017016 000404 BR 3$ ;CONTINUE WITH TEST
3699 017020 004737 030226 2$: JSR PC,GETSI ;GET THE WINDOW
3700 017024 103001 BCC 3$ ;BR IF BIT IS A SPACE
3701 017026 104006 ERROR 6 ;ERROR BIT WAS A MARK
3702 017030 3$:
3703 017030 005203 INC R3 ;NEXT BIT
3704 017032 022703 000010 CMP #10,R3 ;DONE YET?
3705 017036 001357 BNE 1$ ;BR IF NO
3706 017040 104413 000014 DATACLK, 14 ;CLOCK TRANSMITTER 14 MORE TICKS
3707 017044 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3708 017046 021264 021264 ;PORT4 LU-13
3709 017050 032761 000040 000004 BIT #BIT5,4(R1) ;RTS SHOULD BE CLEAR NOW
3710 017056 001401 BEQ 4$ ;BR IF YES
3711 017060 104034 ERROR 34 ;ERROR, RTS NOT CLEAR
3712 017062 4$:
3713
3714
3715 ;***** TEST 23 *****
3716 ;*DDCMP TRANSMITTER TEST
3717 ;*SINGLE CLOCK A BINARY COUNT PATTERN
3718 ;*VERIFY EACH BIT POSITION AS IT
3719 ;*PASSES THE BIT WINDOW (SI BIT)
3720 ;*ON AN ERROR, R3 CONTAINS BIT POSITION OF FAILURE
3721 ;*AND R5 CONTAINS THE CHARACTER THAT FAILED
3722 ;*****
3723
3724 ; TEST 23
3725 ;-----
    
```

```

3726
3727 017062 000004
3728 017064 012737 000023 001202
3729 017072 012737 017270 001442
3730
3731 017100 104410
3732 017102 012711 004000
3733 017106 005003
3734 017110 005004
3735 017112 005005
3736 017114 004737 030412
3737 017120 012761 000001 000004
3738 017126 104412
3739 017130 122111
3740 017132 004737 030412
3741 017136 010461 000004
3742 017142 104412
3743 017144 122110
3744 017146 005204
3745 017150 004737 030412
3746 017154 010461 000004
3747 017160 104412
3748 017162 122110
3749 017164 004737 030260
3750 017170 104413 000002
3751 017174 005003
3752 017176 010502
3753 017200 104413 000001
3754 017204 106002
3755 017206 103005
3756 017210 004737 030226
3757 017214 103406
3758 017216 104006
3759 017220 000404
3760 017222 004737 030226
3761 017226 103001
3762 017230 104006
3763 017232
3764 017232 005203
3765 017234 022703 000010
3766 017240 001357
3767 017242 005204
3768 017244 004737 030412
3769 017250 010461 000004
3770 017254 104412
3771 017256 122110
3772 017260 005205
3773 017262 022705 000400
3774 017266 001342
3775 017270
3776
3777
3778
3779
3780
3781

```

 TST23: SCOPE
 MOV #23,\$TSTNM ; LOAD THE NO. OF THIS TEST
 MOV #TST24,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 ;R3 CONTAINS BIT COUNT
 CLR R4 ;R4 CONTAINS CHAR TO BE LOADED IN SILO
 CLR R5 ;R5 CONTAINS CHARACTER CURRENTLY BEING SHIFTED OUT
 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!
 JSR PC,OUTRDY ;WAIT FOR OUT-READY
 MOV R4,4(R1) ;LOAD PORT4 WITH CHARACTER
 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
 122110 ;LOAD OUT DATA
 INC R4 ;INCREMENT TO NEXT CHARACTER
 JSR PC,OUTRDY ;WAIT FOR OUT-READY
 MOV R4,4(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
 DATACLK, 2 ;2 TICKS TO SET UP TRANSMITTER
 CLR R3 ;CLEAR BIT COUNTER
 MOV R5,R2 ;LOAD CHARACTER IN R2
 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\$;CONTINE WITH TEST
 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
 INC R4 ;NEXT CHARACTER
 JSR PC,OUTRDY ;WAIT FOR OUT-READY
 MOV R4,4(R1) ;LOAD PORT4 WITH CHARACTER
 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
 122110 ;LOAD OUT DATA
 INC R5 ;NEXT CHARACTER
 CMP #400,R5 ;DONE YET?
 BNE 4\$;BR IF NO
 5\$:
 ***** TEST 24 *****
 ;*DDCMP STRIP SYNC TEST
 ;*SET LU LOOP, SINGLE STEP 5 SYNCs,
 ;*VERIFY THAT IN ACTIVE DOES NOT SET

```

3782
3783
3784
3785
3786
3787 017270 000004
3788 017272 012737 000024 001202
3789 017300 012737 017356 001442
3790
3791 017306 104410
3792 017310 012711 004000
3793 017314 012702 000012
3794 017320 004737 030276
3795 017324 000005
3796 017326 104413 000054
3797 017332 104412
3798 017334 021244
3799 017336 016104 000004
3800 017342 042704 000277
3801 017346 005005
3802 017350 120504
3803 017352 001401
3804 017354 104040
3805 017356
3806
3807
3808
3809
3810
3811
3812
3813
3814
3815
3816
3817 017356 000004
3818 017360 012737 000025 001202
3819 017366 012737 017446 001442
3820
3821 017374 104410
3822 017376 012711 004000
3823 017402 012702 000012
3824 017406 004737 030276
3825 017412 000005
3826 017414 104413 000064
3827 017420 104412
3828 017422 021244
3829 017424 016104 000004
3830 017430 042704 000277
3831 017434 012705 000100
3832 017440 120504
3833 017442 001401
3834 017444 104040
3835 017446
3836
3837
    
```

```

:*****
: TEST 24
:-----
:*****
TST24: SCOPE
MOV #24,$STNM ; LOAD THE NO. OF THIS TEST
MOV #TST25,NEXT ; POINT TO THE START OF NEXT TEST.
;R1 CONTAINS BASE KMC11 ADDRESS
MSTCLR ;MASTER CLEAR KMC11
MOV #BIT11,(R1) ;SET LU LOOP
MOV #12,R2 ;SAVE LU REG FOR TYPEOUT
JSR PC,SYNC ;SINGLE CLOCK 5 SYNC CHARACTERS
5
DATACLK, 54
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
021244 ;PORT4 LU12
MOV 4(R1),R4 ;PUT 'FOUND' IN R4
BIC #277,R4 ;CLEAR UNWANTED BITS
CLR R5 ;PUT 'EXPECTED' IN R5
CMPB R5,R4 ;IS ACTIVE CLEAR?
BEQ 1$ ;BR IF YES
ERROR 40 ;ERROR ACTIVE IS NOT CLEAR
1$:
    
```

```

:***** TEST 25 *****
:*DDCMP IN ACTIVE TEST
:*SET LU LOOP, SINGLE STEP 5 SYNC AND A NON-SYNC (301)
:*VERIFY THAT IN ACTIVE IS SET
:*****
    
```

```

: TEST 25
:-----
:*****
TST25: SCOPE
MOV #25,$STNM ; LOAD THE NO. OF THIS TEST
MOV #TST26,NEXT ; POINT TO THE START OF NEXT TEST.
;R1 CONTAINS BASE KMC11 ADDRESS
MSTCLR ;MASTER CLEAR KMC11
MOV #BIT11,(R1) ;SET LU LOOP
MOV #12,R2 ;SAVE LU REG FOR TYPEOUT
JSR PC,SYNC ;SINGLE CLOCK 5 SYNC CHARACTERS
5
DATACLK, 64
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
021244 ;PORT4 LU12
MOV 4(R1),R4 ;PUT 'FOUND' IN R4
BIC #277,R4 ;CLEAR UNWANTED BITS
MOV #BIT6,R5 ;PUT 'EXPECTED' IN R5
CMPB R5,R4 ;IS ACTIVE SET?
BEQ 1$ ;BR IF YES
ERROR 40 ;ERROR ACTIVE IS NOT SET
1$:
    
```

3838
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888
3889
3890
3891
3892
3893

017446 000004
 017450 012737 000026 001202
 017456 012737 017534 001442
 104410
 017466 012711 004000
 017472 012702 000012
 017476 004737 030276
 017502 000001
 017504 104413 000024
 017510 104412
 017512 021244
 017514 016104 000004
 017520 042704 000277
 017524 005005
 017526 120504
 017530 001401
 017532 104040
 017534
 017534 000004
 017536 012737 000027 001202
 017544 012737 017624 001442
 017552 104410
 017554 012711 004000
 017560 012702 000012
 017564 004737 030276
 017570 000002
 017572 104413 000034
 017576 104412
 017600 021244
 017602 016104 000004
 017606 042704 000277
 017612 012705 000100
 017616 120504
 017620 001401

```

***** TEST 26 *****
;DDCMP IN ACTIVE TEST
;SET LU LOOP, SINGLE STEP 1 SYNC AND A NON-SYNC (301)
;VERIFY THAT IN ACTIVE DOES NOT SET
*****

; TEST 26
-----
;*****
TST26: SCOPE
MOV #26,$STNM ; LOAD THE NO. OF THIS TEST
MOV #TST27,NEXT ; POINT TO THE START OF NEXT TEST.
MSTCLR ;R1 CONTAINS BASE KMC11 ADDRESS
;MASTER CLEAR KMC11
MOV #BIT11,(R1) ;SET LU LOOP
MOV #12,R2 ;SAVE LU REG FOR TYPEOUT
JSR PC,SYNC ;SINGLE CLOCK 1 SYNC CHARACTERS
1
DATACLK, 24
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
021244 ;PORT4 LU12
MOV 4(R1),R4 ;PUT 'FOUND' IN R4
BIC #277,R4 ;CLEAR UNWANTED BITS
CLR R5 ;PUT 'EXPECTED' IN R5
CMPB R5,R4 ;IS ACTIVE CLEAR?
BEQ 1$ ;BR IF YES
ERROR 40 ;ERROR ACTIVE IS NOT CLEAR

1$:

***** TEST 27 *****
;DDCMP IN ACTIVE TEST
;SET LU LOOP, SINGLE STEP 2 SYNCs AND A NON-SYNC (301)
;VERIFY THAT IN ACTIVE IS SET
*****

; TEST 27
-----
;*****
TST27: SCOPE
MOV #27,$STNM ; LOAD THE NO. OF THIS TEST
MOV #TST30,NEXT ; POINT TO THE START OF NEXT TEST.
MSTCLR ;R1 CONTAINS BASE KMC11 ADDRESS
;MASTER CLEAR KMC11
MOV #BIT11,(R1) ;SET LU LOOP
MOV #12,R2 ;SAVE LU REG FOR TYPEOUT
JSR PC,SYNC ;SINGLE CLOCK 2 SYNC CHARACTERS
2
DATACLK, 34
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
021244 ;PORT4 LU12
MOV 4(R1),R4 ;PUT 'FOUND' IN R4
BIC #277,R4 ;CLEAR UNWANTED BITS
MOV #BIT6,R5 ;PUT 'EXPECTED' IN R5
CMPB R5,R4 ;IS ACTIVE SET?
BEQ 1$ ;BR IF YES
  
```

3894 017622 104040
3895 017624
3896
3897
3898
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908 017624 000004
3909 017626 012737 000030 001202
3910 017634 012737 017776 001442
3911
3912 017642 104410
3913 017644 012702 000012
3914 017650 012711 004000
3915 017654 004737 030444
3916 017660 000301
3917 017662 104413 000053
3918 017666 104414 000002
3919 017672 104412
3920 017674 021244
3921 017676 016104 000004
3922 017702 042704 000357
3923 017706 012705 000020
3924 017712 120504
3925 017714 001401
3926 017716 104040
3927 017720
3928 017720 012761 000200 000004
3929 017726 104412
3930 017730 122112
3931 017732 104412
3932 017734 021244
3933 017736 016104 000004
3934 017742 042704 000277
3935 017746 005005
3936 017750 120504
3937 017752 001401
3938 017754 104040
3939 017756
3940 017756 016104 000004
3941 017762 042704 000357
3942 017766 005005
3943 017770 120504
3944 017772 001401
3945 017774 104040
3946 017776
3947
3948
3949

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

:***** 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
:-----
:*****
TST30: SCOPE ;
MOV #30,$TSTNM ; LOAD THE NO. OF THIS TEST
MOV #TST31,NEXT ; POINT TO THE START OF NEXT TEST.

MSTCLR ;R1 CONTAINS BASE KMC11 ADDRESS
;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 NON-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

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

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

3$:

:***** TEST 31 *****
```

```

3950 ;*DDCMP BASIC RECEICER TEST
3951 ;*SYNC UP RECEIVER AND SINGLE CLOCK THE CHARACTER 0
3952 ;*VERIFY THAT IN RDY IS SET, AND THAT THE CHARACTER WAS RECEIVED
3953 ;*****
3954
3955 ; TEST 31
3956 -----
3957 ;*****
3958 017776 000004 TST31: SCOPE
3959 020000 012737 000031 001202 MOV #31,$STSTNM ; LOAD THE NO. OF THIS TEST
3960 020006 012737 020112 001442 MOV #TST32,NEXT ; POINT TO THE START OF NEXT TEST.
3961 ;R1 CONTAINS BASE KMC11 ADDRESS
3962 020014 104410 MSTCLR ;MASTER CLEAR KMC11
3963 020016 012702 000012 MOV #12,R2 ;SAVE REG ADDRESS IN R2 FOR TYPEOUT
3964 020022 012711 004000 MOV #BIT11,(R1) ;SET LINE UNIT LOOP
3965 020026 004737 030444 JSR PC,CHAR ;LOAD SILO WITH 3 SYNCs
3966 020032 000000 0 ;AND THE CHARACTER 0
3967 020034 104413 000053 DATACLK, 53 ;SINGLE CLOCK THE DATA
3968 020040 104414 000002 TIMER, 2 ;WAIT FOR INRDY
3969 020044 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3970 020046 021244 021244 ;PORT4 LU 12
3971 020050 016104 000004 MOV 4(R1),R4 ;PUT 'FOUND' IN R4
3972 020054 042704 000357 BIC #357,R4 ;CLEAR UNWANTED BITS
3973 020060 012705 000020 MOV #BIT4,R5 ;PUT 'EXPECTED' IN R5
3974 020064 120504 CMPB R5,R4 ;IS INRDY SET?
3975 020066 001401 BEQ 1$
3976 020070 104040 ERROR 40 ;ERROR, INRDY IS NOT SET
3977 020072 1$:
3978 020072 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3979 020074 021204 021204 ;PORT4 IN DATA
3980 020076 016104 000004 MOV 4(R1),R4 ;PUT 'FOUND' IN R4
3981 020102 005005 CLR R5 ;PUT 'EXPECTED' IN R5
3982 020104 120504 CMPB R5,R4 ;WAS A 0 RECEIVED?
3983 020106 001401 BEQ 2$
3984 020110 104010 ERROR 10 ;ERROR, RECEIVED DATA IS WRONG
3985 020112 2$:
3986
3987
3988 ;***** TEST 32 *****
3989 ;*DDCMP BASIC RECEICER TEST
3990 ;*SYNC UP RECEIVER AND SINGLE CLOCK THE CHARACTER 125
3991 ;*VERIFY THAT IN RDY IS SET, AND THAT THE CHARACTER WAS RECEIVED
3992 ;*****
3993
3994 ; TEST 32
3995 -----
3996 ;*****
3997 020112 000004 TST32: SCOPE
3998 020114 012737 000032 001202 MOV #32,$STSTNM ; LOAD THE NO. OF THIS TEST
3999 020122 012737 020230 001442 MOV #TST33,NEXT ; POINT TO THE START OF NEXT TEST.
4000 ;R1 CONTAINS BASE KMC11 ADDRESS
4001 020130 104410 MSTCLR ;MASTER CLEAR KMC11
4002 020132 012702 000012 MOV #12,R2 ;SAVE REG ADDRESS IN R2 FOR TYPEOUT
4003 020136 012711 004000 MOV #BIT11,(R1) ;SET LINE UNIT LOOP
4004 020142 004737 030444 JSR PC,CHAR ;LOAD SILO WITH 3 SYNCs
4005 020146 000125 125 ;AND THE CHARACTER 125
    
```



```

4006 020150 104413 000053      DATACLK,      53      :SINGLE CLOCK THE DATA
4007 020154 104414 000002      TIMER, 2      :WAIT FOR INRDY
4008 020160 104412      ROMCLK      :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4009 020162 021244      021244      :PORT4 LU 12
4010 020164 016104 000004      MOV 4(R1),R4  :PUT 'FOUND' IN R4
4011 020170 042704 000357      BIC #357,R4   :CLEAR UNWANTED BITS
4012 020174 012705 000020      MOV #BIT4,R5  :PUT 'EXPECTED' IN R5
4013 020200 120504      CMPB R5,R4    :IS INRDY SET?
4014 020202 001401      BEQ 1$        :
4015 020204 104040      ERROR 40      :ERROR, INRDY IS NOT SET
4016 020206      1$:
4017 020206 104412      ROMCLK      :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4018 020210 021204      021204      :PORT4 IN DATA
4019 020212 016104 000004      MOV 4(R1),R4  :PUT 'FOUND' IN R4
4020 020216 012705 000125      MOV #125,R5   :PUT 'EXPECTED' IN R5
4021 020222 120504      CMPB R5,R4    :WAS A 125 RECEIVED?
4022 020224 001401      BEQ 2$        :
4023 020226 104010      ERROR 10      :ERROR, RECEIVED DATA IS WRONG
4024 020230      2$:
4025
4026
4027      :***** TEST 33 *****
4028      :*DDCMP BASIC RECEICER TEST
4029      :*SYNC UP RECEIVER AND SINGLE CLOCK THE CHARACTER 252
4030      :*VERIFY THAT IN RDY IS SET, AND THAT THE CHARACTER WAS RECEIVED
4031      :*****
4032
4033      : TEST 33
4034      :-----
4035      :*****
4036 020230 000004      TST33: SCOPE
4037 020232 012737 000033 001202      MOV #33,$TSTNM      : LOAD THE NO. OF THIS TEST
4038 020240 012737 020346 001442      MOV #TST34,NEXT    : POINT TO THE START OF NEXT TEST.
4039      :R1 CONTAINS BASE KMC11 ADDRESS
4040 020246 104410      MSTCLR      :MASTER CLEAR KMC11
4041 020250 012702 000012      MOV #12,R2      :SAVE REG ADDRESS IN R2 FOR TYPEOUT
4042 020254 012711 004000      MOV #BIT11,(R1) :SET LINE UNIT LOOP
4043 020260 004737 030444      JSR PC,CHAR     :LOAD SILO WITH 3 SYNCs
4044 020264 000252      252            :AND THE CHARACTER 252
4045 020266 104413 000053      DATACLK,      53      :SINGLE CLOCK THE DATA
4046 020272 104414 000002      TIMER, 2      :WAIT FOR INRDY
4047 020276 104412      ROMCLK      :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4048 020300 021244      021244      :PORT4 LU 12
4049 020302 016104 000004      MOV 4(R1),R4  :PUT 'FOUND' IN R4
4050 020306 042704 000357      BIC #357,R4   :CLEAR UNWANTED BITS
4051 020312 012705 000020      MOV #BIT4,R5  :PUT 'EXPECTED' IN R5
4052 020316 120504      CMPB R5,R4    :IS INRDY SET?
4053 020320 001401      BEQ 1$        :
4054 020322 104040      ERROR 40      :ERROR, INRDY IS NOT SET
4055 020324      1$:
4056 020324 104412      ROMCLK      :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4057 020326 021204      021204      :PORT4 IN DATA
4058 020330 016104 000004      MOV 4(R1),R4  :PUT 'FOUND' IN R4
4059 020334 012705 000252      MOV #252,R5   :PUT 'EXPECTED' IN R5
4060 020340 120504      CMPB R5,R4    :WAS A 252 RECEIVED?
4061 020342 001401      BEQ 2$        :
    
```

```

4062 020344 104010          ERROR 10          ;ERROR, RECEIVED DATA IS WRONG
4063 020346          2$:
4064
4065
4066          ;***** TEST 34 *****
4067          ;*DDCMP BASIC RECEICER TEST
4068          ;*SYNC UP RECEIVER AND SINGLE CLOCK THE CHARACTER 377
4069          ;*VERIFY THAT IN RDY IS SET, AND THAT THE CHARACTER WAS RECEIVED
4070          ;*****
4071
4072          ; TEST 34
4073          ;-----
4074          ;*****
4075 020346 000004          TST34: SCOPE
4076 020350 012737 000034 001202          MOV #34,$STSTNM          ; LOAD THE NO. OF THIS TEST
4077 020356 012737 020464 001442          MOV #TST35,NEXT          ; POINT TO THE START OF NEXT TEST.
4078          ;R1 CONTAINS BASE KMC11 ADDRESS
4079 020364 104410          MSTCLR          ;MASTER CLEAR KMC11
4080 020366 012702 000012          MOV #12,R2          ;SAVE REG ADDRESS IN R2 FOR TYPEOUT
4081 020372 012711 004000          MOV #BIT11,(R1)      ;SET LINE UNIT LOOP
4082 020376 004737 030444          JSR PC,CHAR          ;LOAD SILO WITH 3 SYNCs
4083 020402 000377          377          ;AND THE CHARACTER 377
4084 020404 104413 000053          DATACLK, 53        ;SINGLE CLOCK THE DATA
4085 020410 104414 000002          TIMER, 2           ;WAIT FOR INRDY
4086 020414 104412          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4087 020416 021244          021244          ;PORT4 LU 12
4088 020420 016104 000004          MOV 4(R1),R4        ;PUT 'FOUND' IN R4
4089 020424 042704 000357          BIC #357,R4         ;CLEAR UNWANTED BITS
4090 020430 012705 000020          MOV #BIT4,R5        ;PUT 'EXPECTED' IN R5
4091 020434 120504          CMPB R5,R4         ;IS INRDY SET?
4092 020436 001401          BEQ 1$             ;
4093 020440 104040          ERROR 40           ;ERROR, INRDY IS NOT SET
4094 020442          1$:
4095 020442 104412          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4096 020444 021204          021204          ;PORT4 IN DATA
4097 020446 016104 000004          MOV 4(R1),R4        ;PUT 'FOUND' IN R4
4098 020452 012705 000377          MOV #377,R5        ;PUT 'EXPECTED' IN R5
4099 020456 120504          CMPB R5,R4         ;WAS A 377 RECEIVED?
4100 020460 001401          BEQ 2$             ;
4101 020462 104010          ERROR 10           ;ERROR, RECEIVED DATA IS WRONG
4102 020464          2$:
4103
4104
4105          ;***** TEST 35 *****
4106          ;*DDCMP DATA TEST
4107          ;*THIS TEST SINGLE STEPS A BINARY COUNT PATTERN
4108          ;*CHECKING EACH CHARACTER AS IT IS RECEIVED
4109          ;*****
4110
4111          ; TEST 35
4112          ;-----
4113          ;*****
4114 020464 000004          TST35: SCOPE
4115 020466 012737 000035 001202          MOV #35,$STSTNM          ; LOAD THE NO. OF THIS TEST
4116 020474 012737 020514 001442          MOV #TST36,NEXT          ; POINT TO THE START OF NEXT TEST.
4117          ;R1 CONTAINS BASE KMC11 ADDRESS
    
```

| | | | | | | | |
|------|--------|--------|--------|------|----------|-------------|---|
| 4118 | 020502 | 104410 | | | MSTCLR | | :MASTER CLEAR KMC11 |
| 4119 | 020504 | 005037 | 031062 | | CLR | SCHAR | :START BINARY COUNT AT ZERO |
| 4120 | 020510 | 005037 | 031064 | | CLR | STUFLG | :CLEAR BITSTUFF FLAG |
| 4121 | 020514 | 005002 | | | CLR | R2 | :R2 IS "EXPECTED" DATA |
| 4122 | 020516 | 012703 | 000073 | | MOV | #73,R3 | :R3 IS CHARACTER COUNT |
| 4123 | 020522 | 012711 | 004000 | | MOV | #BIT11,(R1) | :SET LINE UNIT LOOP |
| 4124 | 020526 | 004737 | 030622 | | JSR | PC,S:LOLD | :LOAD SILO WITH COUNT PATTERN |
| 4125 | 020532 | 104413 | 000043 | | DATACLK, | 43 | :SYNC RECEIVER AND GET IT ACTIVE |
| 4126 | 020536 | 104413 | 000730 | 1\$: | DATACLK, | 730 | :CLOCK IN 73 CHARACTERS |
| 4127 | 020542 | 004737 | 031066 | 4\$: | JSR | PC,INRDY | :WAIT FOR INRDY |
| 4128 | 020546 | 104412 | | | ROMCLK | | :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 |
| 4129 | 020550 | 021204 | | | 021204 | | :PORT4 IN DATA |
| 4130 | 020552 | 016104 | 000004 | | MOV | 4(R1),R4 | :PUT "FOUND" IN R4 |
| 4131 | 020556 | 010205 | | | MOV | R2,R5 | :PUT "EXPECTED" IN R5 |
| 4132 | 020560 | 120504 | | | CMPB | R5,R4 | :IS DATA CORRECT? |
| 4133 | 020562 | 001401 | | | BEQ | 2\$ | :BR IF YES |
| 4134 | 020564 | 104010 | | | ERROR | 10 | :DATA ERROR |
| 4135 | 020566 | 005202 | | 2\$: | INC | R2 | :NEXT CHARACTER |
| 4136 | 020570 | 022702 | 000400 | | CMP | #400,R2 | :ALL DONE? |
| 4137 | 020574 | 001407 | | | BEQ | 3\$ | :BR IF YES |
| 4138 | 020576 | 005303 | | | DEC | R3 | :DECREMENT CHARACTER COUNT |
| 4139 | 020600 | 001360 | | | BNE | 4\$ | :BR IF SILO NOT EMPTY |
| 4140 | 020602 | 004737 | 030622 | | JSR | PC,SILOLD | :LOAD SILO WITH MORE OF COUNT PATTERN |
| 4141 | 020606 | 012703 | 000073 | | MOV | #73,R3 | :RELOAD CHARACTER COUNT |
| 4142 | 020612 | 000751 | | | BR | 1\$ | :CONTINUE |
| 4143 | 020614 | | | 3\$: | | | |

***** TEST 36 *****
 :*DDCMP DATA TEST
 :*THIS TEST SINGLE STEPS A BINARY COUNT PATTERN
 :*CHECKING EACH CHARACTER AS IT IS RECEIVED
 :*THIS TEST IS EXACTLY THE SAME AS THE LAST TEST,
 :*EXCEPT LINE UNIT LOOP IS SET IN LU REGISTER 12
 :*****

: TEST 36
 :-----

| | | | | | | | |
|------|--------|--------|--------|--------|----------|-------------|---|
| 4156 | | | | | | | |
| 4157 | 020614 | 000004 | | | TST36: | SCOPE | |
| 4158 | 020616 | 012737 | 000036 | 001202 | MOV | #36,\$TSTNM | : LOAD THE NO. OF THIS TEST |
| 4159 | 020624 | 012737 | 020754 | 001442 | MOV | #TST37,NEXT | : POINT TO THE START OF NEXT TEST. |
| 4160 | | | | | | | :R1 CONTAINS BASE KMC11 ADDRESS |
| 4161 | 020632 | 104410 | | | MSTCLR | | :MASTER CLEAR KMC11 |
| 4162 | 020634 | 005037 | 031062 | | CLR | SCHAR | :START BINARY COUNT AT ZERO |
| 4163 | 020640 | 005037 | 031064 | | CLR | STUFLG | :CLEAR BITSTUFF FLAG |
| 4164 | 020644 | 005002 | | | CLR | R2 | :R2 IS "EXPECTED" DATA |
| 4165 | 020646 | 012703 | 000073 | | MOV | #73,R3 | :R3 IS CHARACTER COUNT |
| 4166 | 020652 | 005011 | | | CLR | (R1) | :CLEAR LU LOOP IN MAINT REG |
| 4167 | 020654 | 012761 | 000040 | 000004 | MOV | #BITS,4(R1) | :LOAD PORT4 |
| 4168 | 020662 | 104412 | | | ROMCLK | | :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 |
| 4169 | 020664 | 122112 | | | 122112 | | :SET LU LOOP IN LU REG 12 |
| 4170 | 020666 | 004737 | 030622 | | JSR | PC,SILOLD | :LOAD SILO WITH COUNT PATTERN |
| 4171 | 020672 | 104413 | 000043 | | DATACLK, | 43 | :SYNC RECEIVER AND GET IT ACTIVE |
| 4172 | 020676 | 104413 | 000730 | 1\$: | DATACLK, | 730 | :CLOCK IN 73 CHARACTERS |
| 4173 | 020702 | 004737 | 031066 | 4\$: | JSR | PC,INRDY | :WAIT FOR INRDY |

| | | | | | | |
|------|--------|--------|--------|--------|-----------|---|
| 4174 | 020706 | 104412 | | ROMCLK | | :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 |
| 4175 | 020710 | 021204 | | 021204 | | :PORT4 IN DATA |
| 4176 | 020712 | 016104 | 000004 | MOV | 4(R1),R4 | :PUT 'FOUND' IN R4 |
| 4177 | 020716 | 010205 | | MOV | R2,R5 | :PUT 'EXPECTED' IN R5 |
| 4178 | 020720 | 120504 | | CMPB | R5,R4 | :IS DATA CORRECT? |
| 4179 | 020722 | 001401 | | BEQ | 2\$ | :BR IF YES |
| 4180 | 020724 | 104010 | | ERROR | 10 | :DATA ERROR |
| 4181 | 020726 | 005202 | | INC | R2 | :NEXT CHARACTER |
| 4182 | 020730 | 022702 | 000400 | CMP | #400,R2 | :ALL DONE? |
| 4183 | 020734 | 001407 | | BEQ | 3\$ | :BR IF YES |
| 4184 | 020736 | 005303 | | DEC | R3 | :DECREMENT CHARACTER COUNT |
| 4185 | 020740 | 001360 | | BNE | 4\$ | :BR IF SILO NOT EMPTY |
| 4186 | 020742 | 004737 | 030622 | JSR | PC,SILOLD | :LOAD SILO WITH MORE OF COUNT PATTERN |
| 4187 | 020746 | 012703 | 000073 | MOV | #73,R3 | :RELOAD CHARACTER COUNT |
| 4188 | 020752 | 000751 | | BR | 1\$ | :CONTINUE |
| 4189 | 020754 | | | | | |

4190
 4191
 4192
 4193
 4194
 4195
 4196
 4197
 4198
 4199
 4200
 4201
 4202

```

:***** TEST 37 *****
:*TRANSMITTER MARK TEST
:*SINGLE CLOCK 3 SYNCs AND A 301 AND 20 EXTRA
:*CLOCK TICKS, VERIFY THAT A 301, A 377 AND A 377
:*WERE RECEIVED INDICATING THAT THE TRANSMITTER WENT
:*TO A MARK STATE FOR 16 BITS WHEN OUT SILO WAS EMPTY
:*****
    
```

4203
 4204
 4205
 4206
 4207
 4208
 4209
 4210
 4211
 4212
 4213
 4214
 4215
 4216
 4217
 4218
 4219
 4220
 4221
 4222
 4223
 4224
 4225
 4226
 4227
 4228
 4229

```

: TEST 37
:-----
:*****
TST37: SCOPE
MOV #37,$TSTNM ; LOAD THE NO. OF THIS TEST
MOV #TST40,NEXT ; POINT TO THE START OF NEXT TEST.
MSTCLR ;R1 CONTAINS BASE KMC11 ADDRESS
MOV #BIT11,(R1) ;MASTER CLEAR KMC11
JSR PC,CHAR ;SET LINE UNIT LOOP
301 ;LOAD SILO WITH 3 SYNCs
DATACLK, 73 ;AND A 301
JSR PC,INRDY ;CLOCK THE 301 IN AND 20 EXTRA TICKS
ROMCLK ;WAIT FOR INRDY
021204 ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
MOV 4(R1),R4 ;PORT4 IN DATA
MOV #301,R5 ;PUT 'FOUND' IN R4
CMPB R5,R4 ;PUT 'EXPECTED' IN R5
BEQ 1$ ;WAS A 301 RECEIVED?
ERROR 10 ;ERROR FIRST CHARACTER INCORRECT
JSR PC,INRDY ;WAIT FOR INRDY
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
021204 ;PORT4 IN DATA
MOV 4(R1),R4 ;PUT 'FOUND' IN R4
MOV #377,R5 ;PUT 'EXPECTED' IN R5
CMPB R5,R4 ;WAS A 377 RECEIVED?
BEQ 2$
ERROR 10 ;ERROR, 377 WAS NOT RECEIVED
JSR PC,INRDY ;WAIT FOR INRDY
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
    
```

```

4230 021074 021204          021204          :PORT4 IN DATA
4231 021076 016104 000004  MOV      4(R1),R4      :PUT 'FOUND' IN R4
4232 021102 012705 000377  MOV      #377,R5      :PUT 'EXPECTED' IN R5
4233 021106 120504          CMPB     R5,R4        :WAS A 377 RECEIVED?
4234 021110 001401          BEQ      3$           :
4235 021112 104010          ERROR    10          :ERROR, 177 WAS NOT RECEIVED
4236 021114          3$:
4237
4238
4239
4240          :***** TEST 40 *****
4241          :*CABLE TURNAROUND TEST
4242          :*CLEAR LINE UNIT LOOP, SET DTR
4243          :*VERIFY THAT RING AND MODEM READY ARE SET
4244          :*CLEAR DTR, VERIFY THAT RING AND MRDY ARE CLEARED
4245          :*****
4246          : TEST 40
4247          :-----
4248          :*****
4249 021114 000004          TST40: SCOPE
4250 021116 012737 000040 001202  MOV      #40,$TSTNM   ; LOAD THE NO. OF THIS TEST
4251 021124 012737 021342 001442  MOV      #TST41,NEXT  ; POINT TO THE START OF NEXT TEST.
4252          MSTCLR          ;R1 CONTAINS BASE KMC11 ADDRESS
4253 021132 104410          ;MASTER CLEAR KMC11
4254 021134 032737 020000 002050  BIT      #BIT13,STAT1 ; IS LINE UNIT M8202?
4255 021142 001004          BNE      .+12         ; BR IF YES (DO TEST EVEN IF NO LOOP-BACK CONN)
4256 021144 032737 040000 002050  BIT      #BIT14,STAT1 ; IS TURNAROUND CONNECTOR ON?
4257 021152 001473          BEQ      2$           ; SKIP TEST IF NO
4258 021154 005011          CLR      (R1)         ; CLEAR LINE UNIT LOOP
4259 021156 012761 000100 000004  MOV      #100,4(R1)   ; LOAD PORT4
4260 021164 104412          ROMCLK          ; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4261 021166 122113          122113          ; SET DTR
4262 021170 104414 000002          TIMER, 2          ; WAIT
4263 021174 104412          ROMCLK          ; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4264 021176 021264          021264          ; PORT4 LU13
4265 021200 016104 000004          MOV      4(R1),R4     ; PUT FOUND IN R4      +---+NEW
4266 021204 042704 000023          BIC      #23,R4       ; CLEAR JUNK
4267 021210 012705 000310          MOV      #310,R5     ; GET EXPECTED.
4268 021214 032737 020000 002050  BIT      #BIT13,STAT1 ; IS LINE UNIT M8202?
4269 021222 001402          BEQ      .+6         ; NO RING ON M8202
4270 021224 042705 000200          BIC      #BIT7,R5
4271          ;
4272 021230 032737 000004 002054  BIT      #BIT2,STAT3 ; IS THIS V.35 MODEM?
4273 021236 001402          BEQ      3$           ; NO THEN GO AHEAD.
4274          ;
4275 021240 042705 000200          BIC      #BIT7,R5     ; YES-NO RING ON V.35 MODEM
4276 021244          3$:
4277 021244 020504          CMP      R5,R4
4278 021246 001401          BEQ      1$
4279          ;
4280          ; ARE RING AND MODEM READY SET?
4281 021250 104011          ERROR    11          ; WARNING! IF V.35 AND AUTO STARTED,
4282 021252 005061 000004          CLR      4(R1)       ; YOU WILL GET THIS ERROR. YOU MUST
4283 021256 104412          ROMCLK          ; MANUALL NASWER THE QUESTIONS FOR V.35.
4284 021260 122113          122113          ; ERROR, RING OR MRDY NOT SET
4285 021262 104414 000002          TIMER, 2          ; CLEAR PORT4
                    ; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
                    ; CLEAR DTR
    
```

```

4286 021266 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4287 021270 021264 021264 ;PORT4 LU13
4288 021272 016104 000004 MOV 4(R1),R4 ;PUR FOUND IN R4 +++NEW
4289 021276 042704 000023 BIC #23,R4 ;STRIP JUNK
4290 021302 005005 CLR R5 ;SET EXPECTED.
4291 021304 032737 020000 002050 BIT #BIT13,STAT1 ;IS THIS A M8202?
4292 021312 001402 BEQ .+6
4293 021314 052705 000010 BIS #BIT3,R5 ;YES THEN EXPECT MRDY SET.
4294 021320 032737 000004 002054 BIT #BIT2,STAT3 ;IS THIS V.35?
4295 021326 001402 BEQ 4$
4296 021330 042704 000200 BIC #BIT7,R4
4297 021334 4$: CMPB R4,R5 ;ARE RING AND MRDY CLEAR?
4298 021334 120405 BEQ 2$
4299 021336 001401
4300 ;WARNING! YOU MAY GET THIS ERROR IF V.35
4301 ;AND AUTOSTART. YOU MUST MANNUALLY ANSWER
4302 ;ALL QUESTIONS IF V.35.
4303 021340 104011 ERROR 11 ;ERROR, RING OR MRDY NOT CLEAR
4304 021342 2$:
4305
4306 ;***** TEST 41 *****
4307 ;*CABLE TURNAROUND TEST
4308 ;*CLEAR LINE UNIT LOOP, LOAD OUT DATA SILO
4309 ;*VERIFY THAT ALL MODEM SIGNALS ARE SET
4310 ;*****
4311
4312 ; TEST 41
4313 ;-----
4314
4315 ;*****
4316 021342 000004 TST41: SCOPE
4317 021344 012737 000041 001202 MOV #41,$STNM ;LOAD THE NO. OF THIS TEST
4318 021352 012737 021536 001442 MOV #TST42,NEXT ;POINT TO THE START OF NEXT TEST.
4319 ;R1 CONTAINS BASE KMC11 ADDRESS
4320 021360 104410 MSTCLR ;MASTER CLEAR KMC11
4321 021362 032737 020000 002050 BIT #BIT13,STAT1 ;IS LINE UNIT M8202?
4322 021370 001004 BNE .+12 ;BR IF YES (DO TEST EVEN IF NO LOOP-BACK CONN)
4323 021372 032737 040000 002050 BIT #BIT14,STAT1 ;IS TURNAROUND CONNECTOR ON?
4324 021400 001456 BEQ 1$ ;SKIP TEST IF NO
4325 021402 012711 004000 MOV #BIT11,(R1) ;SET LINE UNIT LOOP
4326 021406 012761 000100 000004 MOV #100, 4(R1) ;LOAD PORT4
4327 021414 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4328 021416 122113 122113 ;CLEAR ALL MODEM SIGNALS,EXCEPT DTR
4329 021420 104414 000002 TIMER, 2 ;WAIT
4330 021424 012761 000001 000004 MOV #1,4(R1) ;LOAD PORT4
4331 021432 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4332 021434 122111 122111 ;SET SOM
4333 021436 004537 031530 JSR R5,MESLD ;FILL OUT DATA SILO
4334 021442 032012 MESDAT ;WITH 64 CHARACTERS
4335 021444 000100 64.
4336 021446 012700 000050 MOV #50,R0 ;PREPARE FOR DELAY
4337 021452 005011 CLR (R1) ;CLEAR LINE UNIT LOOP
4338 021454 2$:
4339 021454 104412 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4340 021456 021264 021264 ;PORT4 LU13
4341 021460 016104 000004 MOV 4(R1),R4 ;PUT 'FOUND' IN R4

```

```

4342 021464 042704 000023      BIC    #23,R4      ;CLEAR UNWANTED BITS
4343 021470 012705 000354      MOV    #354,R5     ;PUT "EXPECTED" IN R5
4344
4345 021474 032737 000004 002054  BIT    #BIT2,STAT3 ; IS THIS V.35?
4346 021502 001402                BEQ    4$          ; NO
4347
4348 021504 042705 000200                BIC    #BIT7,R5   ; YES, GET RID OF RING
4349 021510                4$:
4350 021510 032737 020000 002050  BIT    #BIT13,STAT1 ; IS LINE UNIT M8202?
4351 021516 001402                BEQ    .+6        ;BR IF NO
4352 021520 042705 000200                BIC    #BIT7,R5   ;NO RING ON M8202
4353 021524 120504                CMPB   R5,R4      ;COMPARE EXPECTED AND FOUND
4354 021526 001403                BEQ    1$        ;BR IF OK
4355 021530 005300                DEC    R0        ;DEC DELAY COUNT
4356 021532 001350                BNE    2$        ;BR IF NOT ZERO
4357 021534 104011                ERROR  11        ;ERROR, ALL SIGNALS ARE NOT SET
4358
4359
4360 021536                1$:
4361
4362
4363
4364
4365
4366
4367
4368
4369
4370
4371
4372

```

```

:***** TEST 42 *****
:*TEST OF CRC OPERATION
:*USING THE CRC16 POLYNOMIAL, SINGLE CLOCK THE CHARACTER
:*0, VERIFY THE LSB OF THE BCC ON EACH SHIFT
:*TEST TRANSMITTER FIRST THEN THE RECEIVER BCC
:*****

```

; TEST 42

```

4373 021536 000004      TST42: SCOPE
4374 021540 012737 000042 001202  MOV    #42,$TSTNM ; LOAD THE NO. OF THIS TEST
4375 021546 012737 022052 001442  MOV    #TST43,NEXT ; POINT TO THE START OF NEXT TEST.
4376 021554 012737 021570 001444  MOV    #64$,LOCK   ; ADDRESS FOR LOCK ON DATA.
4377
4378 021562 104410                MSTCLR ;R1 CONTAINS BASE KMC11 ADDRESS
4379 021564 012711 004000                MOV    #BIT11,(R1) ;MASTER CLEAR KMC11
4380 021570 004737 031572                64$: JSR    PC,CLRIO   ;SET LU LOOP
4381 021574 005000                CLR    R0         ;CLEAR BCC REGISTERS
4382 021576 012737 120001 031226  MOV    #CRC16,XPOLY ;START SHIFT COUNTER AT ZERO
4383 021604 012737 000000 021644  MOV    #0,66$     ;LOAD POLYNOMIAL FOR SOFTWARE BCC
4384 021612 005037 021646                CLR    67$       ;LOAD CHAR FOR SOFTWARE BCC
4385 021616 004737 031232                JSR    PC,BCCLD  ;CLEAR OLD SOFTWARE BCC
4386 021622 000000                0           ;LOAD OUT SILO WITH 2 SYNCs
4387 021624 104413 000021                DATACLK, 21    ;AND THE CHARACTER 0
4388 021630 104413 000001                65$: DATACLK, 1 ;GET TRANSMITTER ACTIVE
4389 021634 005200                INC    R0        ;SHIFT BCC ONCE
4390 021636 004537 031122                JSR    R5,SIMBCC ;BUMP SHIFT COUNT
4391 021642 000001                1           ;CALCULATE SOFTWARE BCC LSB
4392 021644 000000                66$: 0          ;ONE SHIFT
4393 021646 000000                67$: 0          ;DATA CHARACTER
4394 021650 103405                BCS    68$      ;OLD BCC
4395 021652 004737 031344                JSR    PC,GETQO ;BR IF SOFT BCC LSB IS SET
4396 021656 103006                BCC    69$     ;GET HARDWARE TRANSMITTER BCC LSB
4397 021660 104012                ERROR  12      ;BR IF HARD BCC LSB IS CLEAR
;ERROR, BCC LSB IS SET

```

```

4398 021662 000404
4399 021664 004737 031344
4400 021670 103401
4401 021672 104016
4402 021674
4403 021674 006037 021644
4404 021700 013737 031230 021646
4405 021706 022700 000010
4406 021712 001346
4407 021714 104405
4408 021716 012737 021724 001444
4409 021724 004737 031572
4410 021730 005000
4411 021732 012737 120001 031226
4412 021740 012737 000000 022000
4413 021746 005037 022002
4414 021752 004737 031232
4415 021756 000000
4416 021760 104413 000032
4417 021764 104413 000001
4418 021770 005200
4419 021772 004537 031122
4420 021776 000001
4421 022000 000000
4422 022002 000000
4423 022004 103405
4424 022006 004737 031356
4425 022012 103006
4426 022014 104013
4427 022016 000404
4428 022020 004737 031356
4429 022024 103401
4430 022026 104017
4431 022030
4432 022030 006037 022000
4433 022034 013737 031230 022002
4434 022042 022700 000010
4435 022046 001346
4436 022050 104405
4437 022052
4438
4439
4440
4441
4442
4443
4444
4445
4446
4447
4448
4449
4450 022052 000004
4451 022054 012737 000043 001202
4452 022062 012737 022366 001442
4453 022070 012737 022104 001444

68$: BR 69$ ;CONTINUE
JSR PC,GETQ0 ;GET HARDWARE TRANSMITTER BCC LSB
BCS 69$ ;BR IF HARD BCC LSB IS SET
ERROR 16 ;ERROR, HARD BCC LSB IS CLEAR

69$: ROR 66$ ;SHIFT SOFT DATA
MOV CALBCC,67$ ;LOAD OLD SOFT BCC
CMP #10,RO ;DONE YET?
BNE 65$ ;BR IF NOT DONE
SCOP1 ;SCOPE SUBTEST (SW09=1)

71$: MOV #71$,LOCK ;NEW SCOPE1
JSR PC,CLRIO ;CLEAR BCC REGISTERS
CLR RO ;START SHIFT COUNTER AT ZERO
MOV #CRC16,XPOLY ;LOAD POLYNOMIAL FOR SOFTWARE BCC
MOV #0,73$ ;LOAD CHAR FOR SOFTWARE BCC
CLR 74$ ;CLEAR OLD SOFTWARE BCC
JSR PC,BCCLD ;LOAD OUT SILO WITH 2 SYNCs
0 ;AND THE CHARACTER 0
DATACLK, 32 ;GET RECEIVER ACTIVE
DATACLK, 1 ;SHIFT BCC ONCE
INC RO ;BUMP SHIFT COUNT
JSR R5,SIMBCC ;CALCULATE SOFTWARE BCC LSB
1 ;ONE SHIFT
0 ;DATA CHARACTER
73$: 0 ;OLD BCC
74$: 0 ;BR IF SOFT BCC LSB IS SET
BCS 75$ ;GET HARDWARE RECEIVER BCC LSB
JSR PC,GETQ1 ;BR IF HARD BCC LSB IS CLEAR
BCC 76$ ;ERROR, BCC LSB IS SET
ERROR 13 ;CONTINUE
BR 76$ ;GET HARDWARE RECEIVER BCC LSB
75$: JSR PC,GETQ1 ;BR IF HARD BCC LSB IS SET
BCS 76$ ;ERROR, BCC LSB IS CLEAR
ERROR 17

76$: ROR 73$ ;SHIFT SOFT DATA
MOV CALBCC,74$ ;LOAD OLD SOFT BCC
CMP #10,RO ;DONE YET?
BNE 72$ ;BR IF NOT DONE
SCOP1 ;SCOPE SUBTEST (SW09=1)

77$:

:***** TEST 43 *****
:*TEST OF CRC OPERATION
:*USING THE CRC16 POLYNOMIAL, SINGLE CLOCK THE CHARACTER
:*377, VERIFY THE LSB OF THE BCC ON EACH SHIFT
:*TEST TRANSMITTER FIRST THEN THE RECEIVER BCC
:*****

: TEST 43
:-----
:*****
TST43: SCOPE
MOV #43,$TSTNM ; LOAD THE NO. OF THIS TEST
MOV #TST44,NEXT ; POINT TO THE START OF NEXT TEST.
MOV #64$,LOCK ; ADDRESS FOR LOCK ON DATA.

```


| | | | | | | | | | | | |
|------|--------|--------|--------|--------|--|-------|--|--|--|--|-----------------------------------|
| 4454 | | | | | | | | | | | :R1 CONTAINS BASE KMC11 ADDRESS |
| 4455 | 022076 | 104410 | | | | | | | | | :MASTER CLEAR KMC11 |
| 4456 | 022100 | 012711 | 004000 | | | | | | | | :SET LU LOOP |
| 4457 | 022104 | 004737 | 031572 | | | 64\$: | | | | | :CLEAR BCC REGISTERS |
| 4458 | 022110 | 005000 | | | | | | | | | :START SHIFT COUNTER AT ZERO |
| 4459 | 022112 | 012737 | 120001 | 031226 | | | | | | | :LOAD POLYNOMIAL FOR SOFTWARE BCC |
| 4460 | 022120 | 012737 | 000377 | 022160 | | | | | | | :LOAD CHAR FOR SOFTWARE BCC |
| 4461 | 022126 | 005037 | 022162 | | | | | | | | :CLEAR OLD SOFTWARE BCC |
| 4462 | 022132 | 004737 | 031232 | | | | | | | | :LOAD OUT SILO WITH 2 SYNC |
| 4463 | 022136 | 000377 | | | | | | | | | :AND THE CHARACTER 377 |
| 4464 | 022140 | 104413 | 000021 | | | | | | | | :GET TRANSMITTER ACTIVE |
| 4465 | 022144 | 104413 | 000001 | | | 65\$: | | | | | :SHIFT BCC ONCE |
| 4466 | 022150 | 005200 | | | | | | | | | :BUMP SHIFT COUNT |
| 4467 | 022152 | 004537 | 031122 | | | | | | | | :CALCULATE SOFTWARE BCC LSB |
| 4468 | 022156 | 000001 | | | | | | | | | :ONE SHIFT |
| 4469 | 022160 | 000000 | | | | 66\$: | | | | | :DATA CHARACTER |
| 4470 | 022162 | 000000 | | | | 67\$: | | | | | :OLD BCC |
| 4471 | 022164 | 103405 | | | | | | | | | :BR IF SOFT BCC LSB IS SET |
| 4472 | 022166 | 004737 | 031344 | | | | | | | | :GET HARDWARE TRANSMITTER BCC LSB |
| 4473 | 022172 | 103006 | | | | | | | | | :BR IF HARD BCC LSB IS CLEAR |
| 4474 | 022174 | 104012 | | | | | | | | | :ERROR, BCC LSB IS SET |
| 4475 | 022176 | 000404 | | | | | | | | | :CONTINUE |
| 4476 | 022200 | 004737 | 031344 | | | 68\$: | | | | | :GET HARDWARE TRANSMITTER BCC LSB |
| 4477 | 022204 | 103401 | | | | | | | | | :BR IF HARD BCC LSB IS SET |
| 4478 | 022206 | 104016 | | | | | | | | | :ERROR, HARD BCC LSB IS CLEAR |
| 4479 | 022210 | | | | | 69\$: | | | | | |
| 4480 | 022210 | 006037 | 022160 | | | | | | | | :SHIFT SOFT DATA |
| 4481 | 022214 | 013737 | 031230 | 022162 | | | | | | | :LOAD OLD SOFT BCC |
| 4482 | 022222 | 022700 | 000010 | | | | | | | | :DONE YET? |
| 4483 | 022226 | 001346 | | | | | | | | | :BR IF NOT DONE |
| 4484 | 022230 | 104405 | | | | | | | | | :SCOPE SUBTEST (SW09=1) |
| 4485 | 022232 | 012737 | 022240 | 001444 | | | | | | | :NEW SCOPE1 |
| 4486 | 022240 | 004737 | 031572 | | | 71\$: | | | | | :CLEAR BCC REGISTERS |
| 4487 | 022244 | 005000 | | | | | | | | | :START SHIFT COUNTER AT ZERO |
| 4488 | 022246 | 012737 | 120001 | 031226 | | | | | | | :LOAD POLYNOMIAL FOR SOFTWARE BCC |
| 4489 | 022254 | 012737 | 000377 | 022314 | | | | | | | :LOAD CHAR FOR SOFTWARE BCC |
| 4490 | 022262 | 005037 | 022316 | | | | | | | | :CLEAR OLD SOFTWARE BCC |
| 4491 | 022266 | 004737 | 031232 | | | | | | | | :LOAD OUT SILO WITH 2 SYNC |
| 4492 | 022272 | 000377 | | | | | | | | | :AND THE CHARACTER 377 |
| 4493 | 022274 | 104413 | 000032 | | | | | | | | :GET RECEIVER ACTIVE |
| 4494 | 022300 | 104413 | 000001 | | | 72\$: | | | | | :SHIFT BCC ONCE |
| 4495 | 022304 | 005200 | | | | | | | | | :BUMP SHIFT COUNT |
| 4496 | 022306 | 004537 | 031122 | | | | | | | | :CALCULATE SOFTWARE BCC LSB |
| 4497 | 022312 | 000001 | | | | | | | | | :ONE SHIFT |
| 4498 | 022314 | 000000 | | | | 73\$: | | | | | :DATA CHARACTER |
| 4499 | 022316 | 000000 | | | | 74\$: | | | | | :OLD BCC |
| 4500 | 022320 | 103405 | | | | | | | | | :BR IF SOFT BCC LSB IS SET |
| 4501 | 022322 | 004737 | 031356 | | | | | | | | :GET HARDWARE RECEIVER BCC LSB |
| 4502 | 022326 | 103006 | | | | | | | | | :BR IF HARD BCC LSB IS CLEAR |
| 4503 | 022330 | 104013 | | | | | | | | | :ERROR, BCC LSB IS SET |
| 4504 | 022332 | 000404 | | | | | | | | | :CONTINUE |
| 4505 | 022334 | 004737 | 031356 | | | 75\$: | | | | | :GET HARDWARE RECEIVER BCC LSB |
| 4506 | 022340 | 103401 | | | | | | | | | :BR IF HARD BCC LSB IS SET |
| 4507 | 022342 | 104017 | | | | | | | | | :ERROR, BCC LSB IS CLEAR |
| 4508 | 022344 | | | | | 76\$: | | | | | |
| 4509 | 022344 | 006037 | 022314 | | | | | | | | :SHIFT SOFT DATA |

```

4510 022350 013737 031230 022316      MOV    CALBCC,74$      ;LOAD OLD SOFT BCC
4511 022356 022700 000010                CMP    #10,RO         ;DONE YET?
4512 022362 001346                BNE    72$           ;BR IF NOT DONE
4513 022364 104405                SCOP1                ;SCOPE SUBTEST (SW09=1)
4514 022366                        77$:
4515
4516
4517
4518 ;***** TEST 44 *****
4519 ;*TEST OF CRC OPERATION
4520 ;*USING THE CRC16 POLYNOMIAL, SINGLE CLOCK THE CHARACTER
4521 ;*125, VERIFY THE LSB OF THE BCC ON EACH SHIFT
4522 ;*TEST TRANSMITTER FIRST THEN THE RECEIVER BCC
4523 ;*****
4524 ; TEST 44
4525 ;-----
4526 ;*****
4527 022366 000004      TST44: SCOPE
4528 022370 012737 000044 001202      MOV    #44,$TSTNM    ; LOAD THE NO. OF THIS TEST
4529 022376 012737 022702 001442      MOV    #TST45,NEXT  ; POINT TO THE START OF NEXT TEST.
4530 022404 012737 022420 001444      MOV    #64$,LOCK    ; ADDRESS FOR LOCK ON DATA.
4531                                     ;R1 CONTAINS BASE KMC11 ADDRESS
4532 022412 104410      MSTCLR      ;MASTER CLEAR KMC11
4533 022414 012711 004000      MOV    #BIT11,(R1)  ;SET LU LOOP
4534 022420 004737 031572      JSR    PC,CLRIO    ;CLEAR BCC REGISTERS
4535 022424 005000      CLR    RO         ;START SHIFT COUNTER AT ZERO
4536 022426 012737 120001 031226      MOV    #CRC16,XPOLY ;LOAD POLYNOMIAL FOR SOFTWARE BCC
4537 022434 012737 000125 022474      MOV    #125,66$    ;LOAD CHAR FOR SOFTWARE BCC
4538 022442 005037 022476      CLR    67$        ;CLEAR OLD SOFTWARE BCC
4539 022446 004737 031232      JSR    PC,BCCLD    ;LOAD OUT SILO WITH 2 SYNC'S
4540 022452 000125 125          ;AND THE CHARACTER 125
4541 022454 104413 000021      DATACLK, 21      ;GET TRANSMITTER ACTIVE
4542 022460 104413 000001      DATACLK, 1       ;SHIFT BCC ONCE
4543 022464 005200      INC    RO         ;BUMP SHIFT COUNT
4544 022466 004537 031122      JSR    R5,SIMBCC   ;CALCULATE SOFTWARE BCC LSB
4545 022472 000001 1          ;ONE SHIFT
4546 022474 000000      66$: 0           ;DATA CHARACTER
4547 022476 000000      67$: 0           ;OLD BCC
4548 022500 103405      BCS    68$        ;BR IF SOFT BCC LSB IS SET
4549 022502 004737 031344      JSR    PC,GETQO    ;GET HARDWARE TRANSMITTER BCC LSB
4550 022506 103006      BCC    69$        ;BR IF HARD BCC LSB IS CLEAR
4551 022510 104012      ERROR  12        ;ERROR, BCC LSB IS SET
4552 022512 000404      BR     69$        ;CONTINUE
4553 022514 004737 031344      JSR    PC,GETQO    ;GET HARDWARE TRANSMITTER BCC LSB
4554 022520 103401      BCS    69$        ;BR IF HARD BCC LSB IS SET
4555 022522 104016      ERROR  16        ;ERROR, HARD BCC LSB IS CLEAR
4556 022524
4557 022524 006037 022474      ROR    66$        ;SHIFT SOFT DATA
4558 022530 013737 031230 022476      MOV    CALBCC,67$  ;LOAD OLD SOFT BCC
4559 022536 022700 000010      CMP    #10,RO     ;DONE YET?
4560 022542 001346      BNE    65$        ;BR IF NOT DONE
4561 022544 104405      SCOP1                ;SCOPE SUBTEST (SW09=1)
4562 022546 012737 022554 001444      MOV    #71$,LOCK   ;NEW SCOPE1
4563 022554 004737 031572      JSR    PC,CLRIO    ;CLEAR BCC REGISTERS
4564 022560 005000      CLR    RO         ;START SHIFT COUNTER AT ZERO
4565 022562 012737 120001 031226      MOV    #CRC16,XPOLY ;LOAD POLYNOMIAL FOR SOFTWARE BCC
    
```

| | | | | | | | |
|------|--------|--------|--------|--------|----------|--------------|-----------------------------------|
| 4566 | 022570 | 012737 | 000125 | 022630 | MOV | #125,73\$; | :LOAD CHAR FOR SOFTWARE BCC |
| 4567 | 022576 | 005037 | 022632 | | CLR | 74\$ | :CLEAR OLD SOFTWARE BCC |
| 4568 | 022602 | 004737 | 031232 | | JSR | PC,BCCLD | :LOAD OUT SILO WITH 2 SYNC |
| 4569 | 022606 | 000125 | | | 125 | | :AND THE CHARACTER 125 |
| 4570 | 022610 | 104413 | 000032 | | DATACLK, | 32 | :GET RECEIVER ACTIVE |
| 4571 | 022614 | 104413 | 000001 | 72\$: | DATACLK, | 1 | :SHIFT BCC ONCE |
| 4572 | 022620 | 005200 | | | INC | RO | :BUMP SHIFT COUNT |
| 4573 | 022622 | 004537 | 031122 | | JSR | R5,SIMBCC | :CALCULATE SOFTWARE BCC LSB |
| 4574 | 022626 | 000001 | | | 1 | | :ONE SHIFT |
| 4575 | 022630 | 000000 | | 73\$: | 0 | | :DATA CHARACTER |
| 4576 | 022632 | 000000 | | 74\$: | 0 | | :OLD BCC |
| 4577 | 022634 | 103405 | | | BCS | 75\$ | :BR IF SOFT BCC LSB IS SET |
| 4578 | 022636 | 004737 | 031356 | | JSR | PC,GETQI | :GET HARDWARE RECEIVER BCC LSB |
| 4579 | 022642 | 103006 | | | BCC | 76\$ | :BR IF HARD BCC LSB IS CLEAR |
| 4580 | 022644 | 104013 | | | ERROR | 13 | :ERROR, BCC LSB IS SET |
| 4581 | 022646 | 000404 | | | BR | 76\$ | :CONTINUE |
| 4582 | 022650 | 004737 | 031356 | 75\$: | JSR | PC,GETQI | :GET HARDWARE RECEIVER BCC LSB |
| 4583 | 022654 | 103401 | | | BCS | 76\$ | :BR IF HARD BCC LSB IS SET |
| 4584 | 022656 | 104017 | | | ERROR | 17 | :ERROR, BCC LSB IS CLEAR |
| 4585 | 022660 | | | 76\$: | | | |
| 4586 | 022660 | 006037 | 022630 | | ROR | 73\$ | :SHIFT SOFT DATA |
| 4587 | 022664 | 013737 | 031230 | 022632 | MOV | CALBCC,74\$ | :LOAD OLD SOFT BCC |
| 4588 | 022672 | 022700 | 000010 | | CMP | #10,RO | :DONE YET? |
| 4589 | 022676 | 001346 | | | BNE | 72\$ | :BR IF NOT DONE |
| 4590 | 022700 | 104405 | | | SCOPE | | :SCOPE SUBTEST (SW09=1) |
| 4591 | 022702 | | | 77\$: | | | |
| 4592 | | | | | | | |
| 4593 | | | | | | | |
| 4594 | | | | | | | |
| 4595 | | | | | | | |
| 4596 | | | | | | | |
| 4597 | | | | | | | |
| 4598 | | | | | | | |
| 4599 | | | | | | | |
| 4600 | | | | | | | |
| 4601 | | | | | | | |
| 4602 | | | | | | | |
| 4603 | | | | | | | |
| 4604 | 022702 | 000004 | | | TST45: | SCOPE | |
| 4605 | 022704 | 012737 | 000045 | 001202 | MOV | #45,\$TSTNM | :LOAD THE NO. OF THIS TEST |
| 4606 | 022712 | 012737 | 023216 | 001442 | MOV | #TST46,NEXT | :POINT TO THE START OF NEXT TEST. |
| 4607 | 022720 | 012737 | 022734 | 001444 | MOV | #64\$,LOCK | :ADDRESS FOR LOCK ON DATA. |
| 4608 | | | | | | | :R1 CONTAINS BASE KMC11 ADDRESS |
| 4609 | 022726 | 104410 | | | MSTCLR | | :MASTER CLEAR KMC11 |
| 4610 | 022730 | 012711 | 004000 | | MOV | #BIT11,(R1) | :SET LU LOOP |
| 4611 | 022734 | 004737 | 031572 | 64\$: | JSR | PC,CLRIO | :CLEAR BCC REGISTERS |
| 4612 | 022740 | 005000 | | | CLR | RO | :START SHIFT COUNTER AT ZERO |
| 4613 | 022742 | 012737 | 120001 | 031226 | MOV | #CRC16,XPOLY | :LOAD POLYNOMIAL FOR SOFTWARE BCC |
| 4614 | 022750 | 012737 | 000252 | 023010 | MOV | #252,66\$; | :LOAD CHAR FOR SOFTWARE BCC |
| 4615 | 022756 | 005037 | 023012 | | CLR | 67\$ | :CLEAR OLD SOFTWARE BCC |
| 4616 | 022762 | 004737 | 031232 | | JSR | PC,BCCLD | :LOAD OUT SILO WITH 2 SYNC |
| 4617 | 022766 | 000252 | | | 252 | | :AND THE CHARACTER 252 |
| 4618 | 022770 | 104413 | 000021 | | DATACLK, | 21 | :GET TRANSMITTER ACTIVE |
| 4619 | 022774 | 104413 | 000001 | 65\$: | DATACLK, | 1 | :SHIFT BCC ONCE |
| 4620 | 023000 | 005200 | | | INC | RO | :BUMP SHIFT COUNT |
| 4621 | 023002 | 004537 | 031122 | | JSR | R5,SIMBCC | :CALCULATE SOFTWARE BCC LSB |

```

:***** 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
:-----
:*****

```

```

TST45: SCOPE
MOV #45,$TSTNM ; LOAD THE NO. OF THIS TEST
MOV #TST46,NEXT ; POINT TO THE START OF NEXT TEST.
MOV #64$,LOCK ; ADDRESS FOR LOCK ON DATA.
;R1 CONTAINS BASE KMC11 ADDRESS
MSTCLR ;MASTER CLEAR KMC11
MOV #BIT11,(R1) ;SET LU LOOP
64$: JSR PC,CLRIO ;CLEAR BCC REGISTERS
CLR RO ;START SHIFT COUNTER AT ZERO
MOV #CRC16,XPOLY ;LOAD POLYNOMIAL FOR SOFTWARE BCC
MOV #252,66$; ;LOAD CHAR FOR SOFTWARE BCC
CLR 67$ ;CLEAR OLD SOFTWARE BCC
JSR PC,BCCLD ;LOAD OUT SILO WITH 2 SYNC
252 ;AND THE CHARACTER 252
65$: DATACLK, 21 ;GET TRANSMITTER ACTIVE
DATACLK, 1 ;SHIFT BCC ONCE
INC RO ;BUMP SHIFT COUNT
JSR R5,SIMBCC ;CALCULATE SOFTWARE BCC LSB

```

```

4622 023006 000001          1          ;ONE SHIFT
4623 023010 000000      66$: 0          ;DATA CHARACTER
4624 023012 000000      67$: 0          ;OLD BCC
4625 023014 103405          BCS      68$          ;BR IF SOFT BCC LSB IS SET
4626 023016 004737 031344  JSR      PC,GETQO    ;GET HARDWARE TRANSMITTER BCC LSB
4627 023022 103006          BCC      69$          ;BR IF HARD BCC LSB IS CLEAR
4628 023024 104012          ERROR    12          ;ERROR, BCC LSB IS SET
4629 023026 000404          BR       69$          ;CONTINUE
4630 023030 004737 031344  68$: JSR      PC,GETQO    ;GET HARDWARE TRANSMITTER BCC LSB
4631 023034 103401          BCS      69$          ;BR IF HARD BCC LSB IS SET
4632 023036 104016          ERROR    16          ;ERROR, HARD BCC LSB IS CLEAR
4633 023040
4634 023040 006037 023010      69$: ROR      66$          ;SHIFT SOFT DATA
4635 023044 013737 031230 023012  MOV      CALBCC,67$  ;LOAD OLD SOFT BCC
4636 023052 022700 000010      CMP      #10,R0     ;DONE YET?
4637 023056 001346          BNE      65$          ;BR IF NOT DONE
4638 023060 104405          SCOPE1  ;SCOPE SUBTEST (SW09=1)
4639 023062 012737 023070 001444  MOV      #71$,LOCK  ;NEW SCOPE1
4640 023070 004737 031572      71$: JSR      PC,CLRIO    ;CLEAR BCC REGISTERS
4641 023074 005000          CLR      R0         ;START SHIFT COUNTER AT ZERO
4642 023076 012737 120001 031226  MOV      #CRC16,XPOLY ;LOAD POLYNOMIAL FOR SOFTWARE BCC
4643 023104 012737 000252 023144  MOV      #252,73$;  ;LOAD CHAR FOR SOFTWARE BCC
4644 023112 005037 023146      CLR      74$        ;CLEAR OLD SOFTWARE BCC
4645 023116 004737 031232      JSR      PC,BCCLD   ;LOAD OUT SILO WITH 2 SYNCs
4646 023122 000252          252        ;AND THE CHARACTER 252
4647 023124 104413 000032      DATACLK, 32       ;GET RECEIVER ACTIVE
4648 023130 104413 000001      72$: DATACLK, 1    ;SHIFT BCC ONCE
4649 023134 005200          INC      R0         ;BUMP SHIFT COUNT
4650 023136 004537 031122      JSR      R5,SIMBCC  ;CALCULATE SOFTWARE BCC LSB
4651 023142 000001          1          ;ONE SHIFT
4652 023144 000000      73$: 0          ;DATA CHARACTER
4653 023146 000000      74$: 0          ;OLD BCC
4654 023150 103405          BCS      75$          ;BR IF SOFT BCC LSB IS SET
4655 023152 004737 031356  JSR      PC,GETQI    ;GET HARDWARE RECEIVER BCC LSB
4656 023156 103006          BCC      76$          ;BR IF HARD BCC LSB IS CLEAR
4657 023160 104013          ERROR    13          ;ERROR, BCC LSB IS SET
4658 023162 000404          BR       76$          ;CONTINUE
4659 023164 004737 031356  75$: JSR      PC,GETQI    ;GET HARDWARE RECEIVER BCC LSB
4660 023170 103401          BCS      76$          ;BR IF HARD BCC LSB IS SET
4661 023172 104017          ERROR    17          ;ERROR, BCC LSB IS CLEAR
4662 023174
4663 023174 006037 023144      76$: ROR      73$          ;SHIFT SOFT DATA
4664 023200 013737 031230 023146  MOV      CALBCC,74$  ;LOAD OLD SOFT BCC
4665 023206 022700 000010      CMP      #10,R0     ;DONE YET?
4666 023212 001346          BNE      72$          ;BR IF NOT DONE
4667 023214 104405          SCOPE1  ;SCOPE SUBTEST (SW09=1)
4668 023216      77$:
4669
4670
4671
4672
4673
4674
4675
4676
4677
:***** 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
    
```

```

4678
4679
4680 023216 000004
4681 023220 012737 000046 001202
4682 023226 012737 023454 001442
4683
4684 023234 104410
4685 023236 012711 004000
4686 023242 005003
4687 023244 005004
4688 023246 005005
4689 023250 005037 023352
4690 023254 012737 120001 031226
4691 023262 004737 031374
4692 023266 010461 000004
4693 023272 104412
4694 023274 122110
4695 023276 005204
4696 023300 010461 000004
4697 023304 104412
4698 023306 122110
4699 023310 005204
4700 023312 010461 000004
4701 023316 104412
4702 023320 122110
4703 023322 004737 030260
4704 023326 104413 000021
4705 023332 010537 023350
4706 023336 104413 000001
4707 023342 004537 031122
4708 023346 000001
4709 023350 000000
4710 023352 000000
4711 023354 103405
4712 023356 004737 031344
4713 023362 103006
4714 023364 104020
4715 023366 000404
4716 023370 004737 031344
4717 023374 103401
4718 023376 104021
4719
4720 023400
4721 023400 006037 023350
4722 023404 013737 031230 023352
4723 023412 005203
4724 023414 022703 000010
4725 023420 001346
4726 023422 005003
4727 023424 005204
4728 023426 022704 000400
4729 023432 003404
4730 023434 010461 000004
4731 023440 104412
4732 023442 122110
4733 023444 005205

```

 :*****
 TST46: SCOPE
 MOV #46,STSTNM ; LOAD THE NO. OF THIS TEST
 MOV #TST47,NEXT ; POINT TO THE START OF NEXT TEST.
 ;R1 CONTAINS BASE KMC11 ADDRESS
 ;MASTER CLEAR KMC11
 ;SET LINE UNIT LOOP
 ;ZERO BIT COUNT
 ;R4 CONTAINS CHAR TO BE LOADED IN SILO
 ;R5 CONTAINS CHAR CURRENTLY BEING SHIFTED OUT
 ;CLEAR SOFT BCC
 ;LOAD POLYNOMIAL
 ;LOAD SILO WITH 2 SYNCs, SOM SET
 ;PORT4 CHAR
 ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
 ;LOAD OUT DATA
 ;INCREMENT TO NEXT CHARACTER
 ;PORT4 CHAR
 ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
 ;LOAD OUT DATA
 ;INCREMENT TO NEXT CHARACTER
 ;PORT4 CHAR
 ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
 ;LOAD OUT DATA
 ;WAIT FOR OCOR
 ;CLOCK DATA
 ;LOAD CHAR FOR SOFT CRC
 ;SHIFT BCC ONCE
 ;CALCULATE SOFT BCC
 ;SOFT SHIFT COUNT
 ;SOFT CHARACTER
 ;OLD SOFT BCC
 ;BR IF SOFT BCC LSB IS SET
 ;GET HARDWARE TRANSMITTER BCC LSB
 ;BR IF OK (CLEARED)
 ;ERROR, BCC LSB WAS SET
 ;CONTINUE WITH TEST
 ;GET HARDWARE TRANSMITTER BCC LSB
 ;BR IF OK (SET)
 ;ERROR, BCC LSB WAS CLEAR
 ;SHIFT SOFT DATA
 ;LOAD OLD SOFT BCC
 ;INCREMENT BIT COUNTER
 ;DONE A FULL CHARACTER YET?
 ;BR IF NO
 ;RESTART BIT COUNTER
 ;INCREMENT DATA FOR SILO
 ;DONE BINARY COUNT YET?
 ;BR IF YES
 ;PORT4 DATA
 ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
 ;LOAD OUT DATA
 ;INCREMENT DATA

```

4734 023446 022705 000400          CMP    #400,R5          ;DONE BINARY PATTERN YET?
4735 023452 001327          BNE    1$              ;BR IF NO
4736 023454          7$:
4737
4738
4739
4740          ;***** TEST 47 *****
4741          ;*RECEIVER CRC TEST
4742          ;*USING THE CRC16 POLYNOMIAL, SINGLE CLOCK A BINARY
4743          ;*COUNT PATTERN, VERIFY THE LSB OF THE RECEIVER BCC ON EACH SHIFT
4744          ;*****
4745          ; TEST 47
4746          ;-----
4747          ;*****
4748 023454 000004          1$T47: SCOPE
4749 023456 012737 000047 001202    MOV    #47,$TSTNM      ; LOAD THE NO. OF THIS TEST
4750 023464 012737 023712 001442    MOV    #TST50,NEXT    ; POINT TO THE START OF NEXT TEST.
4751          ;R1 CONTAINS BASE KMC11 ADDRESS
4752 023472 104410          MSTCLR          ;MASTER CLEAR KMC11
4753 023474 012711 004000          MOV    #BIT11,(R1)   ;SET LINE UNIT LOOP
4754 023500 005003          CLR    R3           ;ZERO BIT COUNT
4755 023502 005004          CLR    R4           ;R4 CONTAINS CHAR TO BE LOADED IN SILO
4756 023504 005005          CLR    R5           ;R5 CONTAINS CHAR CURRENTLY BEING SHIFTED OUT
4757 023506 005037 023610          CLR    4$          ;CLEAR SOFT BCC
4758 023512 012737 120001 031226    MOV    #CRC16,XPOLY  ;LOAD POLYNOMIAL
4759 023520 004737 031374          JSR    PC,SYNLD     ;LOAD SILO WITH 2 SYNCs, SOM SET
4760 023524 010461 000004          MOV    R4,4(R1)    ;PORT4 CHAR
4761 023530 104412          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4762 023532 122110          122110          ;LOAD OUT DATA
4763 023534 005204          INC    R4           ;INCREMENT TO NEXT CHARACTER
4764 023536 010461 000004          MOV    R4,4(R1)    ;PORT4 CHAR
4765 023542 104412          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4766 023544 122110          122110          ;LOAD OUT DATA
4767 023546 005204          INC    R4           ;INCREMENT TO NEXT CHARACTER
4768 023550 010461 000004          MOV    R4,4(R1)    ;PORT4 CHAR
4769 023554 104412          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4770 023556 122110          122110          ;LOAD OUT DATA
4771 023560 004737 030260          JSR    PC,OCOR     ;WAIT FOR OCOR
4772 023564 104413 000032          DATACLK,32       ;CLOCK DATA
4773 023570 010537 023606          1$: MOV    R5,3$     ;LOAD CHAR FOR SOFT CRC
4774 023574 104413 000001          2$: DATACLK,1    ;SHIFT BCC ONCE
4775 023600 004537 031122          JSR    R5,SIMBCC  ;CALCULATE SOFT BCC
4776 023604 000001          1           ;SOFT SHIFT COUNT
4777 023606 000000          3$: 0           ;SOFT CHARACTER
4778 023610 000000          4$: 0           ;OLD SOFT BCC
4779 023612 103405          BCS    5$         ;BR IF SOFT BCC LSB IS SET
4780 023614 004737 031356          JSR    PC,GETQI   ;GET HARDWARE RECEIVER BCC LSB
4781 023620 103006          BCC    6$         ;BR IF OK (CLEARED)
4782 023622 104022          ERROR  22        ;ERROR, BCC LSB WAS SET
4783 023624 000404          BR     6$         ;CONTINUE WITH TEST
4784 023626 004737 031356          5$: JSR    PC,GETQI   ;GET HARDWARE RECEIVER BCC LSB
4785 023632 103401          BCS    6$         ;BR IF OK (SET)
4786 023634 104023          ERROR  23        ;ERROR, BCC LSB WAS CLEAR
4787
4788 023636          6$:
4789 023636 006037 023606          ROR    3$         ;SHIFT SOFT DATA
  
```

BASIC RECEIVER TESTS

SEQ 0094

| | | | | | | | |
|------|--------|--------|--------|--------|--------|------------|---|
| 4790 | 023642 | 013737 | 031230 | 023610 | MOV | CALBCC,4\$ | :LOAD OLD SOFT BCC |
| 4791 | 023650 | 005203 | | | INC | R3 | :INCREMENT BIT COUNTER |
| 4792 | 023652 | 022703 | 000010 | | CMP | #10,R3 | :DONE A FULL CHARACTER YET? |
| 4793 | 023656 | 001346 | | | BNE | 2\$ | :BR IF NO |
| 4794 | 023660 | 005003 | | | CLR | R3 | :RESTART BIT COUNTER |
| 4795 | 023662 | 005204 | | | INC | R4 | :INCREMENT DATA FOR SILO |
| 4796 | 023664 | 022704 | 000400 | | CMP | #400,R4 | :DONE BINARY COUNT YET? |
| 4797 | 023670 | 003404 | | | BLE | 9\$ | :BR IF YES |
| 4798 | 023672 | 010461 | 000004 | | MOV | R4,4(R1) | :PORT4 DATA |
| 4799 | 023676 | 104412 | | | ROMCLK | | :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 |
| 4800 | 023700 | 122110 | | | 122110 | | :LOAD OUT DATA |
| 4801 | 023702 | 005205 | | 9\$: | INC | R5 | :INCREMENT DATA |
| 4802 | 023704 | 022705 | 000400 | | CMP | #400,R5 | :DONE BINARY PATTERN YET? |
| 4803 | 023710 | 001327 | | | BNE | 1\$ | :BR IF NO |
| 4804 | 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

| | | | | | | | |
|------|--------|--------|--------|--------|------------|--------------|---------------------------------------|
| 4817 | | | | | ***** | | |
| 4818 | 023712 | 000004 | | | TST50: | SCOPE | |
| 4819 | 023714 | 012737 | 000050 | 001202 | MOV | #50,\$TSTNM | : LOAD THE NO. OF THIS TEST |
| 4820 | 023722 | 012737 | 024244 | 001442 | MOV | #TST51,NEXT | : POINT TO THE START OF NEXT TEST. |
| 4821 | | | | | | | :R1 CONTAINS BASE KMC11 ADDRESS |
| 4822 | 023730 | 104410 | | | MSTCLR | | :MASTER CLEAR KMC11 |
| 4823 | | | | | | | |
| 4824 | | | | | | | :LOAD OUT DATA SILO |
| 4825 | | | | | | | |
| 4826 | 023732 | 012711 | 004000 | | MOV | #BIT11,(R1) | :SET LINE UNIT LOOP |
| 4827 | 023736 | 012704 | 032012 | | MOV | #MESDAT,R4 | :LOAD POINTER TO DATA |
| 4828 | 023742 | 005037 | 024036 | | CLR | 10\$ | :CLEAR SOFT BCC |
| 4829 | 023746 | 012700 | 000004 | | MOV | #4,R0 | :LOAD CHARACTER COUNT |
| 4830 | 023752 | 004737 | 031374 | | JSR | PC,SYNLD | :LOAD 2 SYNCs IN OUT SILO |
| 4831 | 023756 | 004737 | 030412 | | JSR | PC,OUTRDY | :WAIT FOR OUTRDY |
| 4832 | 023762 | 004537 | 031530 | | JSR | R5,MESLD | :LOAD SILO WITH 4 CHAR MESS |
| 4833 | 023766 | 032012 | | | MESDAT | | :ADDRESS OF MESSAGE |
| 4834 | 023770 | 000004 | | | 4 | | :NUMBER OF CHARACTERS |
| 4835 | 023772 | 004737 | 031504 | | JSR | PC,EOM | :LOAD GARBAGE CHARACTER, WITH EOM SET |
| 4836 | 023776 | 004737 | 030260 | | JSR | PC,OCOR | :WAIT FOR OCOR |
| 4837 | 024002 | 005003 | | | CLR | R3 | :CLEAR BIT COUNTER |
| 4838 | 024004 | 104413 | 000022 | | DATACLK,22 | | :CLOCK DATA |
| 4839 | 024010 | 112405 | | 12\$: | MOVB | (R4;+,R5 | :LOAD R5 WITH CHAR |
| 4840 | 024012 | 010502 | | | MOV | R5,R2 | :LOAD R2 WITH CHAR |
| 4841 | | | | | | | |
| 4842 | | | | | | | :CHECK FIRST FOUR CHARACTER MESSAGE |
| 4843 | | | | | | | :IN THE BIT WINDOW (0,125,252,377) |
| 4844 | | | | | | | |
| 4845 | 024014 | 012737 | 120001 | 031226 | MOV | #CRC16,XPOLY | :LOAD POLYNOMIAL |

| | | | | | | | | |
|------|--------|--------|--------|--------|----------|-------------|--|--|
| 4846 | 024022 | 010537 | 024034 | | MOV | R5,67\$ | | :LOAD SOFT CHAR FOR BCC |
| 4847 | 024026 | 004537 | 031122 | | JSR | R5,SIMBCC | | :CALCULATE SOFT BCC |
| 4848 | 024032 | 000010 | | | 10 | | | :SHIFT COUNT |
| 4849 | 024034 | 000000 | | 67\$: | 0 | | | :CHARACTER |
| 4850 | 024036 | 000000 | | 10\$: | 0 | | | :OLD BCC |
| 4851 | 024040 | 013737 | 031230 | 024036 | MOV | CALBCC,10\$ | | :LOAD SOFT BCC FOR NEXT SHIFT |
| 4852 | 024046 | 104413 | 000001 | | DATACLK, | 1 | | :SHIFT DATA IN TO BIT WINDOW |
| 4853 | 024052 | 106002 | | 64\$: | RORB | R2 | | :SHIFT SOFT DATA |
| 4854 | 024054 | 103005 | | | BCC | 65\$ | | :BR IF A SPACE |
| 4855 | 024056 | 004737 | 030226 | | JSR | PC,GETSI | | :LOOK AT BIT WINDOW |
| 4856 | 024062 | 103406 | | | BCS | 66\$ | | :BR IF OK (MARK) |
| 4857 | 024064 | 104006 | | | ERROR | 6 | | :ERROR, BIT WINDOW WAS A SPACE |
| 4858 | 024066 | 000404 | | | BR | 66\$ | | :CONTINUE |
| 4859 | 024070 | 004737 | 030226 | 65\$: | JSR | PC,GETSI | | :LOOK AT BIT WINDOW |
| 4860 | 024074 | 103001 | | | BCC | 66\$ | | :BR IF OK (SPACE) |
| 4861 | 024076 | 104006 | | | ERROR | 6 | | :ERROR, BIT WINDOW WAS A MARK |
| 4862 | 024100 | | | 66\$: | | | | |
| 4863 | 024100 | 005203 | | | INC | R3 | | :BUMP BIT COUNTER |
| 4864 | 024102 | 022703 | 000010 | | CMP | #10,R3 | | :DONE FULL 8 BITS YET |
| 4865 | 024106 | 001357 | | | BNE | 64\$ | | :BR IF NO |
| 4866 | 024110 | 005003 | | | CLR | R3 | | :CLEAR BIT COUNTER |
| 4867 | 024112 | 005300 | | | DEC | R0 | | :DEC CHARACTER COUNT |
| 4868 | 024114 | 001335 | | | BNE | 12\$ | | :BR IF NOT DONE YET |
| 4869 | | | | | | | | |
| 4870 | | | | | | | | :CHECK BCC FOR PRECEDING MESSAGE IN THE BIT WINDOW |
| 4871 | | | | | | | | |
| 4872 | 024116 | 013700 | 031230 | | MOV | CALBCC,R0 | | :PUT BCC IN R0 |
| 4873 | 024122 | 104413 | 000001 | 68\$: | DATACLK, | 1 | | :SHIFT HARDWARE BCC |
| 4874 | 024126 | 006000 | | | ROR | R0 | | :SHIFT SOFT BCC |
| 4875 | 024130 | 103005 | | | BCC | 69\$ | | :BR IF CARRY CLEAR |
| 4876 | 024132 | 004737 | 030226 | | JSR | PC,GETSI | | :LOOK AT BIT WINDOW |
| 4877 | 024136 | 103406 | | | BCS | 70\$ | | :BR IF OK (MARK) |
| 4878 | 024140 | 104014 | | | ERROR | 14 | | :ERROR, CRC WRONG (SPACE) |
| 4879 | 024142 | 000404 | | | BR | 70\$ | | :CONTINUE |
| 4880 | 024144 | 004737 | 030226 | 69\$: | JSR | PC,GETSI | | :LOOK AT BIT WINDOW |
| 4881 | 024150 | 103001 | | | BCC | 70\$ | | :BR IF OK (SPACE) |
| 4882 | 024152 | 104014 | | | ERROR | 14 | | :ERROR, CRC WRONG (MARK) |
| 4883 | 024154 | | | 70\$: | | | | |
| 4884 | 024154 | 005203 | | | INC | R3 | | :BUMP BIT COUNTER |
| 4885 | 024156 | 022703 | 000020 | | CMP | #20,R3 | | :FINISHED BCC YET? |
| 4886 | 024162 | 001357 | | | BNE | 68\$ | | :BR IF NO |
| 4887 | 024164 | 005003 | | | CLR | R3 | | :CLEAR BIT COUNTER |
| 4888 | | | | | | | | |
| 4889 | | | | | | | | :CHECK TO SEE IF TRANSMITTER IS MARKING |
| 4890 | | | | | | | | |
| 4891 | 024166 | 104413 | 000001 | 2\$: | DATACLK, | 1 | | :CLOCK TRANSMITTER |
| 4892 | 024172 | 004737 | 030226 | | JSR | PC,GETSI | | :LOOK AT WINDOW |
| 4893 | 024176 | 103401 | | | BCS | 3\$ | | :IT SHOULD BE MARKING |
| 4894 | 024200 | 104024 | | | ERROR | 24 | | :ERROR, BIT WAS A SPACE |
| 4895 | 024202 | 005203 | | 3\$: | INC | R3 | | :BUMP BIT COUNTER |
| 4896 | 024204 | 022703 | 000007 | | CMP | #7,R3 | | :DONE YET |
| 4897 | 024210 | 001366 | | | BNE | 2\$ | | :BR IF NO |
| 4898 | 024212 | 104413 | 000010 | | DATACLK, | 10 | | :GIVE ENOUGH TICKS TO CLEAR OUT ACTIVE |
| 4899 | 024216 | 005003 | | | CLR | R3 | | :CLEAR BIT COUNTER |
| 4900 | 024220 | 104413 | 000001 | 4\$: | DATACLK, | 1 | | :SHIFT OUT NEXT BIT |
| 4901 | 024224 | 004737 | 030226 | | JSR | PC,GETSI | | :LOOK AT BIT WINDOW |

4902 024230 103401
 4903 024232 104024
 4904 024234 005203
 4905 024236 022703 000020
 4906 024242 001366
 4907 024244

5\$:

BCS +4 ;BR IF IT IS A MARK
 ERROR 24 ;ERROR, TRANSMITTER IS NOT MARKING
 INC R3 ;INC BIT COUNT
 CMP #20,R3 ;DONE YET?
 BNE 4\$;BR IF NO

4908
 4909

4910

4911

4912

4913

4914

4915

4916

4917

4918

4919

4920 024244 000004

4921 024246 012737 000051 001202

4922 024254 012737 024446 001442

4923

4924 024262 104410

4925 024264 012711 004000

4926 024270 012702 032012

4927 024274 012700 000004

4928 024300 004737 031374

4929 024304 004737 030412

4930 024310 004537 031530

4931 024314 032012

4932 024316 000004

4933 024320 004737 031504

4934 024324 004737 030260

4935 024330 104413 000114

4936 024334 004737 031066

4937 024340 104412

4938 024342 021204

4939 024344 016104 000004

4940 024350 112205

4941 024352 120504

4942 024354 001401

4943 024356 104010

4944 024360 005300

4945 024362 001364

4946

4947

4948

4949 024364 004737 031066

4950 024370 104412

4951 024372 021204

4952 024374 116137 000004 001302

4953 024402 042737 177400 001302

4954 024410 004737 031066

4955 024414 104412

4956 024416 021244

4957 024420 016104 000004

TST51:

3\$:

1\$:

***** 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

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
 MSTCLR ;MASTER CLEAR KMC11
 MOV #BIT11,(R1) ;SET LINE UNIT LOOP
 MOV #MESDAT,R2 ;LOAD POINTER TO DATA
 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
 DATACLK,114 ;CLOCK DATA
 JSR PC,INRDY ;WAIT FOR INRDY
 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
 021204 ;GET IN DATA
 MOV 4(R1),R4 ;PUT "FOUND" IN R4
 MOVB (R2)+,R5 ;PUT "EXPECTED" IN R5
 CMPB R5,R4 ;COMPARE RECEIVED DATA
 BEQ 1\$;BR IF OK
 ERROR 10 ;DATA ERROR
 DEC R0 ;DEC CHARACTER COUNT
 BNE 3\$;BR IF NOT DONE YET

;CHECK TO SEE THAT IN BCC MATCH IS SET

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

| | | | | | | |
|------|--------|--------|--------|--------|---------|---|
| 4958 | 024424 | 042704 | 000376 | BIC | #376,R4 | :CLEAR UNWANTED BITS |
| 4959 | 024430 | 012705 | 000001 | MOV | #1,R5 | :PUT 'EXPECTED' IN R5 |
| 4960 | 024434 | 120504 | | CMPB | R5,R4 | :IS IN BCC MATCH SET? |
| 4961 | 024436 | 001401 | | BEQ | 25\$ | |
| 4962 | 024440 | 104015 | | ERROR | 15 | :IN BCC MATCH ERROR |
| 4963 | 024442 | | | 25\$: | | |
| 4964 | 024442 | 104412 | | ROMCLK | | :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 |
| 4965 | 024444 | 021204 | | 021204 | | :GET LAST HALF |
| 4966 | 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
:*****
    
```

4980 : TEST 52
 4981 :-----

| | | | | | | |
|------|--------|--------|--------|--------|----------|--|
| 4982 | | | | | | :***** |
| 4983 | 024446 | 000004 | | TST52: | SCOPE | |
| 4984 | 024450 | 012737 | 000052 | 001202 | MOV | #52,\$TSTNM ; LOAD THE NO. OF THIS TEST |
| 4985 | 024456 | 012737 | 025546 | 001442 | MOV | #TST53,NEXT ; POINT TO THE START OF NEXT TEST. |
| 4986 | | | | | | :R1 CONTAINS BASE KMC11 ADDRESS |
| 4987 | 024464 | 104410 | | | MSTCLR | :MASTER CLEAR KMC11 |
| 4988 | | | | | | |
| 4989 | | | | | | :LOAD OUT DATA SILO |
| 4990 | | | | | | |
| 4991 | 024466 | 012711 | 004000 | | MOV | #BIT11,(R1) ;SET LINE UNIT LOOP |
| 4992 | 024472 | 012704 | 032012 | | MOV | #MESDAT,R4 ;LOAD POINTER TO DATA |
| 4993 | 024476 | 005037 | 024606 | | CLR | 10\$;CLEAR SOFT BCC |
| 4994 | 024502 | 012700 | 000004 | | MOV | #4,R0 ;LOAD CHARACTER COUNT |
| 4995 | 024506 | 004737 | 031374 | | JSR | PC,SYNLD ;LOAD 2 SYNCs IN OUT SILO |
| 4996 | 024512 | 004737 | 030412 | | JSR | PC,OUTRDY ;WAIT FOR OUTRDY |
| 4997 | 024516 | 004537 | 031530 | | JSR | R5,MESLD ;LOAD SILO WITH 4 CHAR MESS |
| 4998 | 024522 | 032012 | | | MESDAT | :ADDRESS OF MESSAGE |
| 4999 | 024524 | 000004 | | | 4 | :NUMBER OF CHARACTERS |
| 5000 | 024526 | 004737 | 031504 | | JSR | PC,EOM ;LOAD GARBAGE CHARACTER, WITH EOM SET |
| 5001 | 024532 | 004537 | 031530 | | JSR | R5,MESLD ;LOAD FOUR MORE CHARACTERS |
| 5002 | 024536 | 032012 | | | MESDAT | :ADDRESS OF MESSAGE |
| 5003 | 024540 | 000004 | | | 4 | :NUMBER OF CHACTERS |
| 5004 | 024542 | 004737 | 031504 | | JSR | PC,EOM ;SET EOM |
| 5005 | 024546 | 004737 | 030260 | | JSR | PC,OCOR ;WAIT FOR OCOR |
| 5006 | 024552 | 005003 | | | CLR | R3 ;CLEAR BIT COUNTER |
| 5007 | 024554 | 104413 | 000022 | | DATACLK, | 22 ;CLOCK DATA |
| 5008 | 024560 | 112405 | | 12\$: | MOVB | (R4)+,R5 ;LOAD R5 WITH CHAR |
| 5009 | 024562 | 010502 | | | MOV | R5,R2 ;LOAD R2 WITH CHAR |

```

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

```

5014 024564 012737 120001 031226      MOV      #CRC16,XPOLY      ;LOAD POLYNOMIAL
5015 024572 010537 024604              MOV      R5,67$           ;LOAD SOFT CHAR FOR BCC
5016 024576 004537 031122              JSR      R5,SIMBCC        ;CALCULATE SOFT BCC
5017 024602 000010              10                        ;SHIFT COUNT
5018 024604 000000              67$: 0                    ;CHARACTER
5019 024606 000000              10$: 0                    ;OLD BCC
5020 024610 013737 031230 024606      MOV      CALBCC,10$       ;LOAD SOFT BCC FOR NEXT SHIFT
5021 024616 104413 000001              64$: DATACLK, 1         ;SHIFT DATA IN TO BIT WINDOW
5022 024622 106002              RORB     R2               ;SHIFT SOFT DATA
5023 024624 103005              BCC     65$              ;BR IF A SPACE
5024 024626 004737 030226              JSR     PC,GETSI         ;LOOK AT BIT WINDOW
5025 024632 103406              BCS     66$              ;BR IF OK (MARK)
5026 024634 104006              ERROR   6                ;ERROR, BIT WINDOW WAS A SPACE
5027 024636 000404              BR      66$              ;CONTINUE
5028 024640 004737 030226              65$: JSR     PC,GETSI         ;LOOK AT BIT WINDOW
5029 024644 103001              BCC     66$              ;BR IF OK (SPACE)
5030 024646 104006              ERROR   6                ;ERROR, BIT WINDOW WAS A MARK
5031 024650              66$:
5032 024650 005203              INC     R3                ;BUMP BIT COUNTER
5033 024652 022703 000010              CMP     #10,R3           ;DONE FULL 8 BITS YET
5034 024656 001357              BNE     64$              ;BR IF NO
5035 024660 005003              CLR     R3                ;CLEAR BIT COUNTER
5036 024662 005300              DEC     R0                ;DEC CHARACTER COUNT
5037 024664 001335              BNE     12$              ;BR IF NOT DONE YET
5038
5039              ;CHECK BCC FOR PRECEDING MESSAGE IN THE BIT WINDOW
5040
5041 024666 013700 031230      MOV      CALBCC,R0        ;PUT BCC IN R0
5042 024672 104413 000001              68$: DATACLK,1         ;SHIFT HARDWARE BCC
5043 024676 006000              ROR     R0                ;SHIFT SOFT BCC
5044 024700 103005              BCC     69$              ;BR IF CARRY CLEAR
5045 024702 004737 030226              JSR     PC,GETSI         ;LOOK AT BIT WINDOW
5046 024706 103406              BCS     70$              ;BR IF OK (MARK)
5047 024710 104014              ERROR   14               ;ERROR, CRC WRONG (SPACE)
5048 024712 000404              BR      70$              ;CONTINUE
5049 024714 004737 030226              69$: JSR     PC,GETSI         ;LOOK AT BIT WINDOW
5050 024720 103001              BCC     70$              ;BR IF OK (SPACE)
5051 024722 104014              ERROR   14               ;ERROR, CRC WRONG (MARK)
5052 024724              70$:
5053 024724 005203              INC     R3                ;BUMP BIT COUNTER
5054 024726 022703 000020              CMP     #20,R3           ;FINISHED BCC YET?
5055 024732 001357              BNE     68$              ;BR IF NO
5056 024734 005003              CLR     R3                ;CLEAR BIT COUNTER
5057 024736 012700 000004              MOV     #4,R0            ;RESET CHARACTER COUNTER
5058 024742 012704 032012              MOV     #MESDAT,R4       ;LOAD MESSAGE POINTER
5059 024746 005037 025000              CLR     11$              ;CLR SOFT BCC
5060 024752 112405              13$: MOV     (R4)+,R5      ;LOAD CHAR IN R5
5061 024754 010502              MOV     R5,R2            ;LOAD CHAR IN R2
5062
5063              ;CHECK SECOND MESSAGE IN THE BIT WINDOW (0,125,252,377)
5064
5065 024756 012737 120001 031226      MOV      #CRC16,XPOLY      ;LOAD POLYNOMIAL
5066 024764 010537 024776              MOV      R5,76$           ;LOAD SOFT CHAR FOR BCC
5067 024770 004537 031122              JSR      R5,SIMBCC        ;CALCULATE SOFT BCC
5068 024774 000010              10                        ;SHIFT COUNT
5069 024776 000000              76$: 0                    ;CHARACTER

```

| | | | | | | | |
|------|--------|--------|--------|--------|----------|-------------|--|
| 5070 | 025000 | 000000 | | 11\$: | 0 | | :OLD BCC |
| 5071 | 025002 | 013737 | 031230 | 025000 | MOV | CALBCC,11\$ | :LOAD SOFT BCC FOR NEXT SHIFT |
| 5072 | 025010 | 104413 | 000001 | 73\$: | DATACLK, | 1 | :SHIFT DATA IN TO BIT WINDOW |
| 5073 | 025014 | 106002 | | | RORB | R2 | :SHIFT SOFT DATA |
| 5074 | 025016 | 103005 | | | BCC | 74\$ | :BR IF A SPACE |
| 5075 | 025020 | 004737 | 030226 | | JSR | PC,GETSI | :LOOK AT BIT WINDOW |
| 5076 | 025024 | 103406 | | | BCS | 75\$ | :BR IF OK (MARK) |
| 5077 | 025026 | 104006 | | | ERROR | 6 | :ERROR, BIT WINDOW WAS A SPACE |
| 5078 | 025030 | 000404 | | | BR | 75\$ | :CONTINUE |
| 5079 | 025032 | 004737 | 030226 | 74\$: | JSR | PC,GETSI | :LOOK AT BIT WINDOW |
| 5080 | 025036 | 103001 | | | BCC | 75\$ | :BR IF OK (SPACE) |
| 5081 | 025040 | 104006 | | | ERROR | 6 | :ERROR, BIT WINDOW WAS A MARK |
| 5082 | 025042 | | | 75\$: | | | |
| 5083 | 025042 | 005203 | | | INC | R3 | :BUMP BIT COUNTER |
| 5084 | 025044 | 022703 | 000010 | | CMP | #10,R3 | :DONE FULL 8 BITS YET |
| 5085 | 025050 | 001357 | | | BNE | 73\$ | :BR IF NO |
| 5086 | 025052 | 005003 | | | CLR | R3 | :CLEAR BIT COUNTER |
| 5087 | 025054 | 005300 | | | DEC | R0 | :DEC CHARACTER COUNT |
| 5088 | 025056 | 001335 | | | BNE | 13\$ | :BR IF NOT DONE YET |
| 5089 | | | | | | | |
| 5090 | | | | | | | :CHECK BCC FOR PRECEDING MESSAGE IN THE BIT WINDOW |
| 5091 | | | | | | | |
| 5092 | 025060 | 013700 | 031230 | | MOV | CALBCC,R0 | :PUT BCC IN R0 |
| 5093 | 025064 | 104413 | 000001 | 77\$: | DATACLK, | 1 | :SHIFT HARDWARE BCC |
| 5094 | 025070 | 006000 | | | ROR | R0 | :SHIFT SOFT BCC |
| 5095 | 025072 | 103005 | | | BCC | 78\$ | :BR IF CARRY CLEAR |
| 5096 | 025074 | 004737 | 030226 | | JSR | PC,GETSI | :LOOK AT BIT WINDOW |
| 5097 | 025100 | 103406 | | | BCS | 79\$ | :BR IF OK (MARK) |
| 5098 | 025102 | 104014 | | | ERROR | 14 | :ERROR, CRC WRONG (SPACE) |
| 5099 | 025104 | 000404 | | | BR | 79\$ | :CONTINUE |
| 5100 | 025106 | 004737 | 030226 | 78\$: | JSR | PC,GETSI | :LOOK AT BIT WINDOW |
| 5101 | 025112 | 103001 | | | BCC | 79\$ | :BR IF OK (SPACE) |
| 5102 | 025114 | 104014 | | | ERROR | 14 | :ERROR, CRC WRONG (MARK) |
| 5103 | 025116 | | | 79\$: | | | |
| 5104 | 025116 | 005203 | | | INC | R3 | :BUMP BIT COUNTER |
| 5105 | 025120 | 022703 | 000020 | | CMP | #20,R3 | :FINISHED BCC YET? |
| 5106 | 025124 | 001357 | | | BNE | 77\$ | :BR IF NO |
| 5107 | 025126 | 005003 | | | CLR | R3 | :CLEAR BIT COUNTER |
| 5108 | | | | | | | |
| 5109 | | | | | | | :CHECK TO SEE IF TRANSMITTER IS MARKING |
| 5110 | | | | | | | |
| 5111 | 025130 | 104413 | 000001 | 2\$: | DATACLK, | 1 | :CLOCK TRANSMITTER |
| 5112 | 025134 | 004737 | 030226 | | JSR | PC,GETSI | :LOOK AT WINDOW |
| 5113 | 025140 | 103401 | | | BCS | 3\$ | :IT SHOULD BE MARKING |
| 5114 | 025142 | 104024 | | | ERROR | 24 | :ERROR, BIT WAS A SPACE |
| 5115 | 025144 | 005203 | | 3\$: | INC | R3 | :BUMP BIT COUNTER |
| 5116 | 025146 | 022703 | 000007 | | CMP | #7,R3 | :DONE YET |
| 5117 | 025152 | 001366 | | | BNE | 2\$ | :BR IF NO |
| 5118 | 025154 | 104413 | 000010 | | DATACLK, | 10 | :GIVE ENOUGH TICKS TO CLEAR OUT ACTIVE |
| 5119 | 025160 | 005003 | | | CLR | R3 | :CLEAR BIT COUNTER |
| 5120 | 025162 | 104413 | 000001 | 4\$: | DATACLK, | 1 | :SHIFT OUT NEXT BIT |
| 5121 | 025166 | 004737 | 030226 | | JSR | PC,GETSI | :LOOK AT BIT WINDOW |
| 5122 | 025172 | 103401 | | | BCS | +4 | :BR IF IT IS A MARK |
| 5123 | 025174 | 104024 | | | ERROR | 24 | :ERROR, TRANSMITTER IS NOT MARKING |
| 5124 | 025176 | 005203 | | | INC | R3 | :INC BIT COUNT |
| 5125 | 025200 | 022703 | 000020 | | CMP | #20,R3 | :DONE YET? |

```

5126 025204 001366          BNE      4$          ;BR IF NO
5127
5128                          ;CHECK TO SEE THAT FIRST FOUR CHARACTER MESSAGE
5129                          ;WAS RECEIVED CORRECTLY (0,125,252,377)
5130
5131 025206 104413 000001    DATACLK,      1          ;GET LAST BIT IN RECEIVER
5132 025212 012703 000004    MOV      #4,R3          ;R3=CHARACTER COUNT
5133 025216 012702 032012    MOV      #MESDAT,R2     ;LOAD MESSAGE POINTER IN R2
5134 025222 004737 031066    JSR      PC,INRDY       ;WAIT FOR INRDY
5135 025226 104412          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5136 025230 021204          021204
5137 025232 016104 000004    MOV      4(R1),R4       ;PUT "FOUND" IN R4
5138 025236 112205          MOVB     (R2)+,R5       ;PUT "EXPECTED" IN R5
5139 025240 120504          CMPB    R5,R4           ;IS RECEIVED DATA CORRECT?
5140 025242 001401          BEQ     41$            ;BR IF YES
5141 025244 104010          ERROR   10             ;RECEIVE DATA ERROR
5142 025246 005303          41$: DEC     R3          ;DEC CHARACTER COUNT
5143 025250 001364          BNE     40$            ;BR IF NOT DONE YET
5144
5145                          ;CHECK TO SEE THAT IN BCC MATCH IS SET
5146                          ;AND THAT THE BCC WAS RECEIVED CORRECTLY
5147
5148 025252 004737 031066    JSR      PC,INRDY       ;WAIT FOR INRDY
5149 025256 104412          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5150 025260 021204          021204             ;GET FIRST HALF OF CRC
5151 025262 116137 000004 001302  MOVB     4(R1),$TMP2     ;PUT IN $TMP2
5152 025270 042737 177400 001302  BIC     #177400,$TMP2   ;CLEAR HI BYTE
5153 025276 004737 031066    JSR      PC,INRDY       ;WAIT FOR INRDY
5154 025302 104412          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5155 025304 021244          021244
5156 025306 016104 000004    MOV      4(R1),R4       ;PUT "FOUND" IN R4
5157 025312 042704 000376    BIC     #376,R4         ;CLEAR UNWANTED BITS
5158 025316 012705 000001    MOV      #1,R5          ;PUT "EXPECTED" IN R5
5159 025322 120504          CMPB    R5,R4           ;IS IN BCC MATCH SET?
5160 025324 001401          BEQ     50$            ;BR IF YES
5161 025326 104015          ERROR   15             ;IN BCC MATCH ERROR
5162 025330          50$: ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5163 025330 104412          021204             ;GET LAST HALF
5164 025332 021204          021204
5165 025334 116137 000004 001301  MOVB     4(R1),$TMP1+1   ;PUT IN $TMP1
5166 025342 042737 000377 001300  BIC     #377,$TMP1     ;CLEAR LO BYTE
5167 025350 053737 001300 001302  BIS     $TMP1,$TMP2     ;16 BIT BCC NOW IN $TMP2
5168 025356 023737 031230 001302  CMP     CALBCC,$TMP2    ;IS IT CORRECT?
5169 025364 001401          BEQ     42$            ;BR IF OK
5170 025366 104027          ERROR   27
5171
5172                          ;CHECK TO SEE THAT SECOND FOUR CHARACTER MESSAGE
5173                          ;WAS RECEIVED CORRECTLY (0,125,252,377)
5174
5175 025370 012703 000004          42$: MOV      #4,R3          ;R3=CHARACTER COUNT
5176 025374 012702 032012          MOV      #MESDAT,R2     ;LOAD MESSAGE POINTER IN R2
5177 025400 004737 031066          43$: JSR      PC,INRDY       ;WAIT FOR INRDY
5178 025404 104412          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5179 025406 021204          021204
5180 025410 016104 000004    MOV      4(R1),R4       ;PUT "FOUND" IN R4
5181 025414 112205          MOVB     (R2)+,R5       ;PUT "EXPECTED" IN R5
    
```

```

5182 025416 120504          CMPB   R5,R4          ;IS RECEIVED DATA CORRECT?
5183 025420 001401          BEQ    44$           ;BR IF YES
5184 025422 104010          ERROR  10           ;RECEIVE DATA ERROR
5185 025424 005303          44$: DEC   R3        ;DEC CHARACTER COUNT
5186 025426 001364          BNE    43$           ;BR IF NOT DONE YET
5187
5188                          ;CHECK TO SEE THAT IN BCC MATCH IS SET
5189                          ;AND THAT THE BCC WAS RECEIVED CORRECTLY
5190
5191 025430 004737 031066    JSR    PC,INRDY      ;WAIT FOR INRDY
5192 025434 104412          ROMCLK  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5193 025436 021204          021204 ;GET FIRST HALF OF CRC
5194 025440 116137 000004 001302  MOVB   4(R1),$TMP2   ;PUT IN $TMP2
5195 025446 042737 177400 001302  BIC    #177400,$TMP2 ;CLEAR HI BYTE
5196 025454 004737 031066    JSR    PC,INRDY      ;WAIT FOR INRDY
5197 025460 104412          ROMCLK  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5198 025462 021244          021244
5199 025464 016104 000004    MOV    4(R1),R4      ;PUT "FOUND" IN R4
5200 025470 042704 000376    BIC    #376,R4       ;CLEAR UNWANTED BITS
5201 025474 012705 000001    MOV    #1,R5        ;PUT "EXPECTED" IN R5
5202 025500 120504          CMPB   R5,R4        ;IS IN BCC MATCH SET?
5203 025502 001401          BEQ    51$           ;IN BCC MATCH ERROR
5204 025504 104015          51$: ERROR  15
5205 025506
5206 025506 104412          ROMCLK  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5207 025510 021204          021204 ;GET LAST HALF
5208 025512 116137 000004 001301  MOVB   4(R1),$TMP1+1 ;PUT IN $TMP1
5209 025520 042737 000377 001300  BIC    #377,$TMP1   ;CLEAR LO BYTE
5210 025526 053737 001300 001302  BIS    $TMP1,$TMP2  ;16 BIT BCC NOW IN $TMP2
5211 025534 023737 031230 001302  CMP    CALBCC,$TMP2 ;IS IT CORRECT?
5212 025542 001401          BEQ    5$            ;BR IF OK
5213 025544 104027          5$: ERROR  27
5214 025546

```

```

5215
5216
5217 ***** TEST 53 *****
5218 ;*DDCMP EOM FUNCTION TEST
5219 ;*THIS TEST LOADS OUT SILO WITH: 2 SYNCs,4 CHAR MESSAGE,EOM
5220 ;*SOM,4 CHAR MESS,EOM. THE DATA STREAM IS CHECKED TO BE
5221 ;*4 CHAR,BCC,4 CHAR,BCC,MARKS. THIS TEST VERIFYS THAT
5222 ;*THE CHARCTERS LOADED WITH EOM SET ARE LOST
5223 ;*ALSO THAT THE CHAR LOADED WITH SOM IS NOT IN THE BCC
5224 ;*ALL DATA AND BCC'S ARE CHECKED IN THE BIT WINDOW
5225 ;*THE FOUR CHARACTER MESSAGE IS 0,125,252,377
5226 ;*RECEIVED DATA IS VERIFIED, AND IN BCC MATCH IS CHECKED
5227 ;:*****
5228

```

```

5229 : TEST 53
5230 :-----
5231 :*****
5232 TST53: SCOPE
5233 025546 000004          MOV    #53,$STNM    ; LOAD THE NO. OF THIS TEST
5234 025550 012737 000053 001202  MOV    #TST54,NEXT  ; POINT TO THE START OF NEXT TEST.
5235 025556 012737 026746 001442
5236 025564 104410          MSTCLR ;R1 CONTAINS BASE KMC11 ADDRESS
5237 ;MASTER CLEAR KMC11

```


| | | | | | | | | |
|------|--------|--------|--------|--------|----------|--------------|--|---|
| 5294 | 026004 | 103005 | | | BCC | 69\$ | | :BR IF CARRY CLEAR |
| 5295 | 026006 | 004737 | 030226 | | JSR | PC,GETSI | | :LOOK AT BIT WINDOW |
| 5296 | 026012 | 103406 | | | BCS | 70\$ | | :BR IF OK (MARK) |
| 5297 | 026014 | 104014 | | | ERROR | 14 | | :ERROR, CRC WRONG (SPACE) |
| 5298 | 026016 | 000404 | | | BR | 70\$ | | :CONTINUE |
| 5299 | 026020 | 004737 | 030226 | 69\$: | JSR | PC,GETSI | | :LOOK AT BIT WINDOW |
| 5300 | 026024 | 103001 | | | BCC | 70\$ | | :BR IF OK (SPACE) |
| 5301 | 026026 | 104014 | | | ERROR | 14 | | :ERROR, CRC WRONG (MARK) |
| 5302 | 026030 | | | 70\$: | | | | |
| 5303 | 026030 | 005203 | | | INC | R3 | | :BUMP BIT COUNTER |
| 5304 | 026032 | 022703 | 000020 | | CMP | #20,R3 | | :FINISHED BCC YET? |
| 5305 | 026036 | 001357 | | | BNE | 68\$ | | :BR IF NO |
| 5306 | 026040 | 005003 | | | CLR | R3 | | :CLEAR BIT COUNTER |
| 5307 | | | | | | | | |
| 5308 | | | | | | | | :CHECK CHARACTER LOADED WITH SOM (000), IN THE BIT WINDOW |
| 5309 | | | | | | | | |
| 5310 | 026042 | 005005 | | | CLR | R5 | | :CHARACTER LOADED WITH SOM |
| 5311 | 026044 | 010502 | | | MOV | R5,R2 | | :LOAD R2 WITH CHAR |
| 5312 | 026046 | 104413 | 000001 | 32\$: | DATACLK, | 1 | | :CLOCK TRANSMITTER |
| 5313 | 026052 | 106002 | | | RORB | R2 | | :SHIFT SOFT DATA |
| 5314 | 026054 | 103005 | | | BCC | 30\$ | | :BR IF SPACE |
| 5315 | 026056 | 004737 | 030226 | | JSR | PC,GETSI | | :LOOK AT BIT WINDOW |
| 5316 | 026062 | 103406 | | | BCS | 31\$ | | :BR IF OK (MARK) |
| 5317 | 026064 | 104006 | | | ERROR | 6 | | :ERROR,BIT WINDOW WAS A SPACE |
| 5318 | 026066 | 000404 | | | BR | 31\$ | | :CONTINUE |
| 5319 | 026070 | 004737 | 030226 | 30\$: | JSR | PC,GETSI | | :LOOK AT BIT WINDOW |
| 5320 | 026074 | 103001 | | | BCC | 31\$ | | :BR IF OK (SPACE) |
| 5321 | 026076 | 104006 | | | ERROR | 6 | | :ERROR,BIT WINDOW WAS A MARK |
| 5322 | 026100 | 005203 | | 31\$: | INC | R3 | | :BUMP BIT COUNTER |
| 5323 | 026102 | 022703 | 000010 | | CMP | #10,R3 | | :DONE CHARACTER YET? |
| 5324 | 026106 | 001357 | | | BNE | 32\$ | | :BR IF NO |
| 5325 | 026110 | 005003 | | | CLR | R3 | | :RESET BIT COUNTER |
| 5326 | 026112 | 012700 | 000004 | | MOV | #4,R0 | | :RESET CHARACTER COUNTER |
| 5327 | 026116 | 012704 | 032012 | | MOV | #MESDAT,R4 | | :LOAD MESSAGE POINTER |
| 5328 | 026122 | 005037 | 026154 | | CLR | 11\$ | | :CLR SOFT BCC |
| 5329 | 026126 | 112405 | | 13\$: | MOVB | (R4)+,R5 | | :LOAD CHAR IN R5 |
| 5330 | 026130 | 010502 | | | MOV | R5,R2 | | :LOAD CHAR IN R2 |
| 5331 | | | | | | | | |
| 5332 | | | | | | | | :CHECK SECOND MESSAGE IN THE BIT WINDOW (0,125,252,377) |
| 5333 | | | | | | | | |
| 5334 | 026132 | 012737 | 120001 | 031226 | MOV | #CRC16,XPOLY | | :LOAD POLYNOMIAL |
| 5335 | 026140 | 010537 | 026152 | | MOV | R5,76\$ | | :LOAD SOFT CHAR FOR BCC |
| 5336 | 026144 | 004537 | 031122 | | JSR | R5,SIMBCC | | :CALCULATE SOFT BCC |
| 5337 | 026150 | 000010 | | | 10 | | | :SHIFT COUNT |
| 5338 | 026152 | 000000 | | 76\$: | 0 | | | :CHARACTER |
| 5339 | 026154 | 000000 | | 11\$: | 0 | | | :OLD BCC |
| 5340 | 026156 | 013737 | 031230 | 026154 | MOV | CALBCC,11\$ | | :LOAD SOFT BCC FOR NEXT SHIFT |
| 5341 | 026164 | 104413 | 000001 | 73\$: | DATACLK, | 1 | | :SHIFT DATA IN TO BIT WINDOW |
| 5342 | 026170 | 106002 | | | RORB | R2 | | :SHIFT SOFT DATA |
| 5343 | 026172 | 103005 | | | BCC | 74\$ | | :BR IF A SPACE |
| 5344 | 026174 | 004737 | 030226 | | JSR | PC,GETSI | | :LOOK AT BIT WINDOW |
| 5345 | 026200 | 103406 | | | BCS | 75\$ | | :BR IF OK (MARK) |
| 5346 | 026202 | 104006 | | | ERROR | 6 | | :ERROR, BIT WINDOW WAS A SPACE |
| 5347 | 026204 | 000404 | | | BR | 75\$ | | :CONTINUE |
| 5348 | 026206 | 004737 | 030226 | 74\$: | JSR | PC,GETSI | | :LOOK AT BIT WINDOW |
| 5349 | 026212 | 103001 | | | BCC | 75\$ | | :BR IF OK (SPACE) |

| | | | | | | | | |
|------|--------|--------|--------|-------|----------|------------|--|--|
| 5350 | 026214 | 104006 | | | ERROR | 6 | | :ERROR, BIT WINDOW WAS A MARK |
| 5351 | 026216 | | | 75\$: | | | | |
| 5352 | 026216 | 005203 | | | INC | R3 | | :BUMP BIT COUNTER |
| 5353 | 026220 | 022703 | 000010 | | CMP | #10,R3 | | :DONE FULL 8 BITS YET |
| 5354 | 026224 | 001357 | | | BNE | 73\$ | | :BR IF NO |
| 5355 | 026226 | 005003 | | | CLR | R3 | | :CLEAR BIT COUNTER |
| 5356 | 026230 | 005300 | | | DEC | R0 | | :DEC CHARACTER COUNT |
| 5357 | 026232 | 001335 | | | BNE | 13\$ | | :BR IF NOT DONE YET |
| 5358 | | | | | | | | |
| 5359 | | | | | | | | :CHECK BCC FOR PRECEDING MESSAGE IN THE BIT WINDOW |
| 5360 | | | | | | | | |
| 5361 | 026234 | 013700 | 031230 | | MOV | CALBCC,R0 | | :PUT BCC IN R0 |
| 5362 | 026240 | 104413 | 000001 | 77\$: | DATACLK, | 1 | | :SHIFT HARDWARE BCC |
| 5363 | 026244 | 006000 | | | ROR | R0 | | :SHIFT SOFT BCC |
| 5364 | 026246 | 103005 | | | BCC | 78\$ | | :BR IF CARRY CLEAR |
| 5365 | 026250 | 004737 | 030226 | | JSR | PC,GETSI | | :LOOK AT BIT WINDOW |
| 5366 | 026254 | 103406 | | | BCS | 79\$ | | :BR IF OK (MARK) |
| 5367 | 026256 | 104014 | | | ERROR | 14 | | :ERROR, CRC WRONG (SPACE) |
| 5368 | 026260 | 000404 | | | BR | 79\$ | | :CONTINUE |
| 5369 | 026262 | 004737 | 030226 | 78\$: | JSR | PC,GETSI | | :LOOK AT BIT WINDOW |
| 5370 | 026266 | 103001 | | | BCC | 79\$ | | :BR IF OK (SPACE) |
| 5371 | 026270 | 104014 | | | ERROR | 14 | | :ERROR, CRC WRONG (MARK) |
| 5372 | 026272 | | | 79\$: | | | | |
| 5373 | 026272 | 005203 | | | INC | R3 | | :BUMP BIT COUNTER |
| 5374 | 026274 | 022703 | 000020 | | CMP | #20,R3 | | :FINISHED BCC YET? |
| 5375 | 026300 | 001357 | | | BNE | 77\$ | | :BR IF NO |
| 5376 | 026302 | 005003 | | | CLR | R3 | | :CLEAR BIT COUNTER |
| 5377 | | | | | | | | |
| 5378 | | | | | | | | :CHECK TO SEE IF TRANSMITTER IS MARKING |
| 5379 | | | | | | | | |
| 5380 | 026304 | 104413 | 000001 | 2\$: | DATACLK, | 1 | | :CLOCK TRANSMITTER |
| 5381 | 026310 | 004737 | 030226 | | JSR | PC,GETSI | | :LOOK AT WINDOW |
| 5382 | 026314 | 103401 | | | BCS | 3\$ | | :IT SHOULD BE MARKING |
| 5383 | 026316 | 104024 | | | ERROR | 24 | | :ERROR, BIT WAS A SPACE |
| 5384 | 026320 | 005203 | | 3\$: | INC | R3 | | :BUMP BIT COUNTER |
| 5385 | 026322 | 022703 | 000007 | | CMP | #7,R3 | | :DONE YET |
| 5386 | 026326 | 001366 | | | BNE | 2\$ | | :BR IF NO |
| 5387 | 026330 | 104413 | 000010 | | DATACLK, | 10 | | :GIVE ENOUGH TICKS TO CLEAR OUT ACTIVE |
| 5388 | 026334 | 005003 | | | CLR | R3 | | :CLEAR BIT COUNTER |
| 5389 | 026336 | 104413 | 000001 | 4\$: | DATACLK, | 1 | | :SHIFT OUT NEXT BIT |
| 5390 | 026342 | 004737 | 030226 | | JSR | PC,GETSI | | :LOOK AT BIT WINDOW |
| 5391 | 026346 | 103401 | | | BCS | .+4 | | :BR IF IT IS A MARK |
| 5392 | 026350 | 104024 | | | ERROR | 24 | | :ERROR, TRANSMITTER IS NOT MARKING |
| 5393 | 026352 | 005203 | | | INC | R3 | | :INC BIT COUNT |
| 5394 | 026354 | 022703 | 000020 | | CMP | #20,R3 | | :DONE YET? |
| 5395 | 026360 | 001366 | | | BNE | 4\$ | | :BR IF NO |
| 5396 | | | | | | | | |
| 5397 | | | | | | | | :CHECK TO SEE THAT FIRST FOUR CHARACTER MESSAGE |
| 5398 | | | | | | | | :WAS RECEIVED CORRECTLY (0,125,252,377) |
| 5399 | | | | | | | | |
| 5400 | 026362 | 104413 | 000001 | | DATACLK, | 1 | | :GET LAST BIT IN RECEIVER |
| 5401 | 026366 | 012703 | 000004 | | MOV | #4,R3 | | :R3=CHARACTER COUNT |
| 5402 | 026372 | 012702 | 032012 | | MOV | #MESDAT,R2 | | :LOAD MESSAGE POINTER IN R2 |
| 5403 | 026376 | 004737 | 031066 | 40\$: | JSR | PC,INRDY | | :WAIT FOR INRDY |
| 5404 | 026402 | 104412 | | | ROMCLK | | | :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 |
| 5405 | 026404 | 021204 | | | 021204 | | | |

| | | | | | | | |
|------|--------|--------|--------|--------|--------|----------------|--|
| 5406 | 026406 | 016104 | 000004 | | MOV | 4(R1),R4 | :PUT "FOUND" IN R4 |
| 5407 | 026412 | 112205 | | | MOVB | (R2)+,R5 | :PUT "EXPECTED" IN R5 |
| 5408 | 026414 | 120504 | | | CMPB | R5,R4 | :IS RECEIVED DATA CORRECT? |
| 5409 | 026416 | 001401 | | | BEQ | 41\$ | :BR IF YES |
| 5410 | 026420 | 104010 | | | ERROR | 10 | :RECEIVE DATA ERROR |
| 5411 | 026422 | 005303 | | 41\$: | DEC | R3 | :DEC CHARACTER COUNT |
| 5412 | 026424 | 001364 | | | BNE | 40\$ | :BR IF NOT DONE YET |
| 5413 | | | | | | | |
| 5414 | | | | | | | :CHECK TO SEE THAT IN BCC MATCH IS SET |
| 5415 | | | | | | | :AND THAT THE BCC WAS RECEIVED CORRECTLY |
| 5416 | | | | | | | |
| 5417 | 026426 | 004737 | 031066 | | JSR | PC,INRDY | :WAIT FOR INRDY |
| 5418 | 026432 | 104412 | | | ROMCLK | | :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 |
| 5419 | 026434 | 021204 | | | 021204 | | :GET FIRST HALF OF CRC |
| 5420 | 026436 | 116137 | 000004 | 001302 | MOVB | 4(R1),\$TMP2 | :PUT IN \$TMP2 |
| 5421 | 026444 | 042737 | 177400 | 001302 | BIC | #177400,\$TMP2 | :CLEAR HI BYTE |
| 5422 | 026452 | 004737 | 031066 | | JSR | PC,INRDY | :WAIT FOR INRDY |
| 5423 | 026456 | 104412 | | | ROMCLK | | :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 |
| 5424 | 026460 | 021244 | | | 021244 | | |
| 5425 | 026462 | 016104 | 000004 | | MOV | 4(R1),R4 | :PUT "FOUND" IN R4 |
| 5426 | 026466 | 042704 | 000376 | | BIC | #376,R4 | :CLEAR UNWANTED BITS |
| 5427 | 026472 | 012705 | 000001 | | MOV | #1,R5 | :PUT "EXPECTED" IN R5 |
| 5428 | 026476 | 120504 | | | CMPB | R5,R4 | :IS IN BCC MATCH SET? |
| 5429 | 026500 | 001401 | | | BEQ | 50\$ | |
| 5430 | 026502 | 104015 | | | ERROR | 15 | :IN BCC MATCH ERROR |
| 5431 | 026504 | | | 50\$: | | | |
| 5432 | 026504 | 104412 | | | ROMCLK | | :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 |
| 5433 | 026506 | 021204 | | | 021204 | | :GET LAST HALF |
| 5434 | 026510 | 116137 | 000004 | 001301 | MOVB | 4(R1),\$TMP1+1 | :PUT IN \$TMP1 |
| 5435 | 026516 | 042737 | 000377 | 001300 | BIC | #377,\$TMP1 | :CLEAR LO BYTE |
| 5436 | 026524 | 053737 | 001300 | 001302 | BIS | \$TMP1,\$TMP2 | :16 BIT BCC NOW IN \$TMP2 |
| 5437 | 026532 | 023737 | 031230 | 001302 | CMP | CALBCC,\$TMP2 | :IS IT CORRECT? |
| 5438 | 026540 | 001401 | | | BEQ | 45\$ | :BR IF OK |
| 5439 | 026542 | 104027 | | | ERROR | 27 | |
| 5440 | | | | | | | |
| 5441 | | | | | | | :CHECK THAT CHARACTER LOADED WITH SOM WAS RECEIVED (000) |
| 5442 | | | | | | | |
| 5443 | 026544 | 004737 | 031066 | | JSR | PC,INRDY | :WAIT FOR INRDY |
| 5444 | 026550 | 104412 | | 45\$: | ROMCLK | | :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 |
| 5445 | 026552 | 021204 | | | 021204 | | :GET RECEIVE DATA |
| 5446 | 026554 | 016104 | 000004 | | MOV | 4(R1),R4 | :PUT "FOUND" IN R4 |
| 5447 | 026560 | 005005 | | | CLR | R5 | :PUT "EXPECTED" IN R5 |
| 5448 | 026562 | 120504 | | | CMPB | R5,R4 | :IS RECEIVED DATA CORRECT? |
| 5449 | 026564 | 001401 | | | BEQ | 42\$ | :BR IF YES |
| 5450 | 026566 | 104010 | | | ERROR | 10 | :RECEIVE DATA ERROR |
| 5451 | | | | | | | |
| 5452 | | | | | | | :CHECK TO SEE THAT SECOND FOUR CHARACTER MESSAGE |
| 5453 | | | | | | | :WAS RECEIVED CORRECTLY (0,125,252,377) |
| 5454 | | | | | | | |
| 5455 | 026570 | 012703 | 000004 | | MOV | #4,R3 | :R3=CHARACTER COUNT |
| 5456 | 026574 | 012702 | 032012 | | MOV | #MESDAT,R2 | :LOAD MESSAGE POINTER IN R2 |
| 5457 | 026600 | 004737 | 031066 | | JSR | PC,INRDY | :WAIT FOR INRDY |
| 5458 | 026604 | 104412 | | 43\$: | ROMCLK | | :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 |
| 5459 | 026606 | 021204 | | | 021204 | | |
| 5460 | 026610 | 016104 | 000004 | | MOV | 4(R1),R4 | :PUT "FOUND" IN R4 |
| 5461 | 026614 | 112205 | | | MOVB | (R2)+,R5 | :PUT "EXPECTED" IN R5 |

```

5462 026616 120504          CMPB   R5,R4          ;IS RECEIVED DATA CORRECT?
5463 026620 001401          BEQ    44$           ;BR IF YES
5464 026622 104010          ERROR  10           ;RECEIVE DATA ERROR
5465 026624 005303          DEC    R3           ;DEC CHARACTER COUNT
5466 026626 001364          BNE    43$           ;BR IF NOT DONE YET
5467
5468                          ;CHECK TO SEE THAT IN BCC MATCH IS SET
5469                          ;AND THAT THE BCC WAS RECEIVED CORRECTLY
5470
5471 026630 004737 031066     JSR    PC,INRDY      ;WAIT FOR INRDY
5472 026634 104412          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5473 026636 021204          021204          ;GET FIRST HALF OF CRC
5474 026640 116137 000004 001302  MOVB   4(R1),$TMP2   ;PUT IN $TMP2
5475 026646 042737 177400 001302  BIC    #177400,$TMP2 ;CLEAR HI BYTE
5476 026654 004737 031066     JSR    PC,INRDY      ;WAIT FOR INRDY
5477 026660 104412          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5478 026662 021244          021244
5479 026664 016104 000004     MOV    4(R1),R4      ;PUT 'FOUND' IN R4
5480 026670 042704 000376     BIC    #376,R4       ;CLEAR UNWANTED BITS
5481 026674 012705 000001     MOV    #1,R5         ;PUT 'EXPECTED' IN R5
5482 026700 120504          CMPB   R5,R4          ;IS IN BCC MATCH SET?
5483 026702 001401          BEQ    51$           ;IN BCC MATCH ERROR
5484 026704 104015          ERROR  15
5485 026706          51$:
5486 026706 104412          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5487 026710 021204          021204          ;GET LAST HALF
5488 026712 116137 000004 001301  MOVB   4(R1),$TMP1+1 ;PUT IN $TMP1
5489 026720 042737 000377 001300  BIC    #377,$TMP1    ;CLEAR LO BYTE
5490 026726 053737 001300 001302  BIS    $TMP1,$TMP2   ;16 BIT BCC NOW IN $TMP2
5491 026734 023737 031230 001302  CMP    CALBCC,$TMP2  ;IS IT CORRECT?
5492 026742 001401          BEQ    5$            ;BR IF OK
5493 026744 104027          ERROR  27
5494 026746          5$:
5495
5496
5497
5498                          ;***** TEST 54 *****
5499                          ;*EMPTY SILO TEST
5500                          ;*LOAD SILO WITH 2 SYNCs, 4 CHAR MESSAGE, SINGLE CLOCK
5501                          ;*UNTIL THE SILO IS EMPTY, LOAD 4 MORE CHARACTERS IN THE
5502                          ;*SILO. GIVE MORE TICKS, AND VERIFY THAT ONLY THE FIRST
5503                          ;*4 CHARACTER MESSAGE WAS RECEIVED AND THAT RTS IS CLEAR
5504                          ;:*****
5505
5506                          ; TEST 54
5507                          ;-----
5508 026746 000004          TST54: SCOPE
5509 026750 012737 000054 001202  MOV    #54,$STNM     ; LOAD THE NO. OF THIS TEST
5510 026756 012737 027200 001442  MOV    #TST55,NEXT   ; POINT TO THE START OF NEXT TEST.
5511
5512 026764 104410          ;R1 CONTAINS BASE KMC11 ADDRESS
5513 026766 012711 004000     MSTCLR          ;MASTER CLEAR KMC11
5514 026772 012702 032012     MOV    #BIT11,(R1)   ;SET LINE UNIT LOOP
5515 026776 012700 000004     MOV    #MESDAT,R2    ;R2 POINTS TO MESSAGE
5516 027002 004737 031374     MOV    #4,R0         ;R0 = CHAR COUNT
5517 027006 004737 030412     JSR    PC,SYNLD      ;LOAD SILO WITH TWO SYNCs
5518                          JSR    PC,OUTRDY     ;WAIT FOR OUTRDY
    
```

| | | | | | | | | |
|------|--------|--------|--------|--------|----------|-------------|--|---|
| 5518 | 027012 | 004537 | 031530 | | JSR | R5,MESLD | | :LOAD MESSAGE IN SILO |
| 5519 | 027016 | 032012 | | | MESDAT | | | :START OF MESSAGE |
| 5520 | 027020 | 000004 | | | 4 | | | :CHARACTER COUNT |
| 5521 | 027022 | 004737 | 030260 | | JSR | PC,OCOR | | :WAIT FOR OCOR |
| 5522 | 027026 | 104413 | 000065 | | DATACLK, | 65 | | :CLOCK DATA (EMPTY SILO) |
| 5523 | 027032 | 004537 | 031530 | | JSR | R5,MESLD | | :PUT MORE CHARACTERS IN SILO |
| 5524 | 027036 | 032012 | | | MESDAT | | | |
| 5525 | 027040 | 000004 | | | 4 | | | |
| 5526 | 027042 | 004737 | 030260 | | JSR | PC,OCOR | | |
| 5527 | 027046 | 104413 | 000005 | | DATACLK, | 5 | | :CLOCK UNTIL RTS IS CLEARED |
| 5528 | 027052 | 104412 | | | ROMCLK | | | :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 |
| 5529 | 027054 | 021264 | | | 021264 | | | :GET RTS |
| 5530 | 027056 | 032761 | 000040 | 000004 | BIT | #BIT5,4(R1) | | :IS IT CLEAR? |
| 5531 | 027064 | 001401 | | | BEQ | 5\$ | | :BR IF YES |
| 5532 | 027066 | 104034 | | | ERROR | 34 | | :ERROR, RTS NOT CLEAR |
| 5533 | 027070 | 104413 | 000041 | 5\$: | DATACLK, | 41 | | :CLOCK XMITTER SOME MORE |
| 5534 | 027074 | 004737 | 031066 | 1\$: | JSR | PC,INRDY | | :OK LETS CHECK WHAT WAS RECEIVED |
| 5535 | 027100 | 104412 | | | ROMCLK | | | :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 |
| 5536 | 027102 | 021204 | | | 021204 | | | :GET RECEIVE DATA |
| 5537 | 027104 | 016104 | 000004 | | MOV | 4(R1),R4 | | :PUT IT IN R4 |
| 5538 | 027110 | 112205 | | | MOVB | (R2)+,R5 | | :R5 = 'EXPECTED' |
| 5539 | 027112 | 120504 | | | CMPB | R5,R4 | | :IS DATA CORRECT? |
| 5540 | 027114 | 001401 | | | BEQ | 2\$ | | :BR IF OK |
| 5541 | 027116 | 104010 | | | ERROR | 10 | | :DATA ERROR |
| 5542 | 027120 | 005300 | | 2\$: | DEC | R0 | | :DEC CHAR COUNT |
| 5543 | 027122 | 001364 | | | BNE | 1\$ | | :BR IF NOT DONE YET |
| 5544 | 027124 | 004737 | 031066 | 3\$: | JSR | PC,INRDY | | :WAIT FOR INRDY |
| 5545 | 027130 | 104412 | | | ROMCLK | | | :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 |
| 5546 | 027132 | 021204 | | | 021204 | | | :GET RECEIVE DATA |
| 5547 | 027134 | 016104 | 000004 | | MOV | 4(R1),R4 | | :PUT IT IN 'FOUND' |
| 5548 | 027140 | 012705 | 000377 | | MOV | #377,R5 | | :R5 = 'EXPECTED' |
| 5549 | 027144 | 120504 | | | CMPB | R5,R4 | | :SHOULD SEE 377 |
| 5550 | 027146 | 001401 | | | BEQ | 4\$ | | :BR IF OK |
| 5551 | 027150 | 104010 | | | ERROR | 10 | | :ERROR, TRANSMITTER DID NOT ABORT |
| 5552 | 027152 | 004737 | 031066 | 4\$: | JSR | PC,INRDY | | :WAIT FOR INRDY |
| 5553 | 027156 | 104412 | | | ROMCLK | | | :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 |
| 5554 | 027160 | 021204 | | | 021204 | | | :GET RECEIVE DATA |
| 5555 | 027162 | 016104 | 000004 | | MOV | 4(R1),R4 | | :PUT IT IN 'FOUND' |
| 5556 | 027166 | 012705 | 000377 | | MOV | #377,R5 | | :R5 = 'EXPECTED' |
| 5557 | 027172 | 120504 | | | CMPB | R5,R4 | | :SHOULD SEE 377 |
| 5558 | 027174 | 001401 | | | BEQ | 10\$ | | :BR IF OK |
| 5559 | 027176 | 104010 | | | ERROR | 10 | | :ERROR, TRANSMITTER DID NOT ABORT |
| 5560 | 027200 | | | 10\$: | | | | |

```

:***** TEST 55 *****
:*HALF DUPLEX TEST
:*SET LINE UNIT LOOP AND HALF DUPLEX, SEND SYNCs AND A
:*MESSAGE. VERIFY THAT IN-ACTIVE AND IN-READY ARE CLEAR
:*****

```

: TEST 55

```

:-----
:*****
TST55: SCOPE
MOV #55,$TSTNM ; LOAD THE NO. OF THIS TEST

```

5571
 5572 027200 000004
 5573 027202 012737 000055 001202

```

5574 027210 012737 027316 001442      MOV      #TST56,NEXT          ; POINT TO THE START OF NEXT TEST.
5575                                     ;R1 CONTAINS BASE KMC11 ADDRESS
5576 027216 104410                       MSTCLR                       ;MASTER CLEAR KMC11
5577 027220 012702 000012                 MOV      #12,R2              ;SAVE R2 FOR TYPEOUT
5578 027224 012711 004000                 MOV      #BIT11,(R1)         ;SET LINE UNIT LOOP
5579 027230 012761 000020 000004         MOV      #BIT4,4(R1)         ;LOAD PORT4
5580 027236 104412                       ROMCLK                        ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5581 027240 122113                       122113                       ;SET H/D BIT
5582 027242 004737 031374                 JSR      PC,SYNLD            ;LOAD 2 SYNCs
5583 027246 004737 030412                 JSR      PC,OUTRDY          ;WAIT FOR OUTRDY
5584 027252 004537 031530                 JSR      R5,MESLD           ;LOAD 4 CHAR MESSAGE
5585 027256 032012                       MESDAT                        ;ADDRESS OF MESSAGE
5586 027260 000004 4                       ;CHARACTER COUNT
5587 027262 004737 030260                 JSR      PC,OCOR            ;WAIT FOR OCOR
5588 027266 104413 000073                 DATACLK, 73                ;SEND MESSAGE
5589 027272 104412                       ROMCLK                        ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5590 027274 021244                       021244                       ;READ LU-12
5591 027276 016104 000004                 MOV      4(R1),R4            ;PUT 'FOUND' IN R4
5592 027302 042704 000257                 BIC      #257,R4            ;CLEAR UNWANTED BITS
5593 027306 005005                       CLR      R5                  ;R5 = 'EXPECTED'
5594 027310 120504                       CMPB    R5,R4               ;IN-ACTIVE AND IN-RDY SHOULD BE CLEAR
5595 027312 001401                       BEQ     1$                  ;BR IF OK
5596 027314 104035                       ERROR   35                  ;ERROR BOTH ARE NOT CLEAR
5597 027316                                     1$:

```

```

5598
5599
5600                                     :***** TEST 56 *****
5601                                     ;*DDCMP CABLE DATA TEST
5602                                     ;*THIS TEST LOADS OUT SILO WITH THE FOLLOWING:
5603                                     ;*4 SYNCs,16 CHAR,EOM,16 CHAR,EOM,16 CHAR,EOM
5604                                     ;*THE 16 CHARACTERS INCLUDE A FLOATING ONE AND ZERO
5605                                     ;*THE DATA IS TRANSMITTED OVER THE CABLE USING THE INTERNAL CLOCK
5606                                     ;*RECEIVED DATA IS VERIFIED AS IS IN BCC MATCH
5607                                     ;*LOOP-BACK CONNECTOR MUST BE ON TO RUN THIS TEST
5608                                     :*****
5609

```

```

5610                                     ; TEST 56
5611                                     ;-----
5612

```

```

5613 027316 000004 1$TST56: SCOPE
5614 027320 012737 000056 001202      MOV      #56,$TSTNM          ; LOAD THE NO. OF THIS TEST
5615 027326 012737 027706 001442      MOV      #TST57,NEXT        ; POINT TO THE START OF NEXT TEST.
5616                                     ;R1 CONTAINS BASE KMC11 ADDRESS
5617 027334 104410                       MSTCLR                       ;MASTER CLEAR KMC11
5618 027336 032737 040000 002050         BIT      #BIT14,STAT1       ;SKIP TEST IF NO
5619 027344 001557 3$                      BEQ     3$                   ;LOOPBACK CONNECTOR ON
5620 027346 012711 004000                 MOV      #BIT11,(R1)         ;SET LINE UNIT LOOP
5621 027352 004737 031374                 JSR      PC,SYNLD            ;LOAD 2 SYNCs
5622 027356 004737 031374                 JSR      PC,SYNLD            ;LOAD 2 MORE SYNCs
5623 027362 012737 120001 031226         MOV      #CRC16,XPOLY        ;LOAD POLYNOMIAL FOR SOFT CRC CALC
5624 027370 005037 027420                 CLR      6$                  ;CLEAR OLD BCC
5625 027374 012703 000020                 MOV      #16.,R3            ;CHARACTER COUNT
5626 027400 012702 032016                 MOV      #FLTDAT,R2         ;R2= POINTER
5627 027404 112237 027416 7$: MOVB    (R2)+,5$            ;LOAD CHAR FOR SOFT BCC CALC.
5628 027410 004537 031122                 JSR      R5,SIMBCC          ;CALC SOFT BCC
5629 027414 000010                       10                          ;SHIFT COUNT

```

| | | | | | | |
|------|--------|--------|--------|--------|---------------------|--|
| 5630 | 027416 | 000000 | | | 5\$: 0 | : CHARACTER |
| 5631 | 027420 | 000000 | | | 6\$: 0 | : OLD BCC |
| 5632 | 027422 | 013737 | 031230 | 027420 | MOV CALBCC,6\$ | : LOAD OLD BCC |
| 5633 | 027430 | 005303 | | | DEC R3 | : DEC COUNT |
| 5634 | 027432 | 001364 | | | BNE 7\$ | : BR IF NOT DONE YET |
| 5635 | 027434 | 004537 | 031530 | | JSR R5,MESLD | : LOAD SILO |
| 5636 | 027440 | 032016 | | | FLTDAT | : MESSAGE ADDRESS |
| 5637 | 027442 | 000020 | | | 16. | : CHARACTER COUNT |
| 5638 | 027444 | 004737 | 031504 | | JSR PC,EOM | : LOAD AN EOM |
| 5639 | 027450 | 004537 | 031530 | | JSR R5,MESLD | : LOAD SILO |
| 5640 | 027454 | 032016 | | | FLTDAT | : MESSAGE ADDRESS |
| 5641 | 027456 | 000020 | | | 16. | : CHARACTER COUNT |
| 5642 | 027460 | 004737 | 031504 | | JSR PC,EOM | : LOAD AN EOM |
| 5643 | 027464 | 004537 | 031530 | | JSR R5,MESLD | : LOAD SILO |
| 5644 | 027470 | 032016 | | | FLTDAT | : MESSAGE ADDRESS |
| 5645 | 027472 | 000020 | | | 16. | : CHARACTER COUNT |
| 5646 | 027474 | 004737 | 031504 | | JSR PC,EOM | : LOAD AN EOM |
| 5647 | 027500 | 004737 | 030260 | | JSR PC,OCOR | : WAIT FOR OCOR |
| 5648 | 027504 | 005011 | | | CLR (R1) | : CLEAR LINE UNIT LOOP |
| 5649 | 027506 | 012700 | 000003 | | MOV #3,R0 | : R0 = MESSAGE COUNT |
| 5650 | 027512 | 012703 | 000020 | | MOV #16.,R3 | : R3 = CHARACTER COUNT |
| 5651 | 027516 | 012702 | 032016 | | MOV #FLTDAT,R2 | : LOAD MESSAGE POINTER IN R2 |
| 5652 | 027522 | 004737 | 031066 | | 1\$: JSR PC,INRDY | : WAIT FOR INRDY |
| 5653 | 027526 | 104412 | | | ROMCLK | : NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 |
| 5654 | 027530 | 021204 | | | 021204 | : GET DATA FROM IN SILO |
| 5655 | 027532 | 016104 | 000004 | | MOV 4(R1),R4 | : PUT CHARACTER IN 'FOUND' |
| 5656 | 027536 | 112205 | | | MOVB (R2)+,R5 | : PUT 'EXPECTED' IN R5 |
| 5657 | 027540 | 120504 | | | CMPB R5,R4 | : IS RECEIVED DATA CORRECT |
| 5658 | 027542 | 001401 | | | BEQ 2\$ | : BR IF OK |
| 5659 | 027544 | 104025 | | | ERROR 25 | : DATA ERROR |
| 5660 | 027546 | | | | 2\$: | |
| 5661 | 027546 | 005303 | | | DEC R3 | : DEC CHARACTER COUNT |
| 5662 | 027550 | 001364 | | | BNE 1\$ | : BR IF NOT DONE THIS MESSAGE |
| 5663 | 027552 | 012703 | 000020 | | MOV #16.,R3 | : RESET CHARACTER COUNT |
| 5664 | | | | | | |
| 5665 | | | | | | |
| 5666 | | | | | | : CHECK TO SEE THAT IN BCC MATCH IS SET |
| 5667 | | | | | | : AND THAT THE BCC WAS RECEIVED CORRECTLY |
| 5668 | 027556 | 004737 | 031066 | | JSR PC,INRDY | : WAIT FOR INRDY |
| 5669 | 027562 | 104412 | | | ROMCLK | : NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 |
| 5670 | 027564 | 021204 | | | 021204 | : GET FIRST HALF OF CRC |
| 5671 | 027566 | 116137 | 000004 | 001302 | MOVB 4(R1),\$TMP2 | : PUT IN \$TMP2 |
| 5672 | 027574 | 042737 | 177400 | 001302 | BIC #177400,\$TMP2 | : CLEAR HI BYTE |
| 5673 | 027602 | 004737 | 031066 | | JSR PC,INRDY | : WAIT FOR INRDY |
| 5674 | 027606 | 104412 | | | ROMCLK | : NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 |
| 5675 | 027610 | 021244 | | | 021244 | |
| 5676 | 027612 | 016104 | 000004 | | MOV 4(R1),R4 | : PUT 'FOUND' IN R4 |
| 5677 | 027616 | 042704 | 000376 | | BIC #376,R4 | : CLEAR UNWANTED BITS |
| 5678 | 027622 | 012705 | 000001 | | MOV #1,R5 | : PUT 'EXPECTED' IN R5 |
| 5679 | 027626 | 120504 | | | CMPB R5,R4 | : IS IN BCC MATCH SET? |
| 5680 | 027630 | 001401 | | | BEQ 25\$ | |
| 5681 | 027632 | 104015 | | | ERROR 15 | : IN BCC MATCH ERROR |
| 5682 | 027634 | | | | 25\$: | |
| 5683 | 027634 | 104412 | | | ROMCLK | : NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 |
| 5684 | 027636 | 021204 | | | 021204 | : GET LAST HALF |
| 5685 | 027640 | 116137 | 000004 | 001301 | MOVB 4(R1),\$TMP1+1 | : PUT IN \$TMP1 |


```

5742 030066 016104 000004      MOV    4(R1),R4      ;PUT CHARACTER IN "FOUND"
5743 030072 112205      MOVB   (R2)+,R5     ;PUT "EXPECTED" IN R5
5744 030074 120504      CMPB   R5,R4        ;IS RECEIVED DATA CORRECT
5745 030076 001401      BEQ    2$           ;BR IF OK
5746 030100 104025      ERROR  25          ;DATA ERROR
5747 030102      2$:
5748 030102 005300      DEC    R0           ;DECREMENT COUNTER
5749 030104 001364      BNE    1$           ;BR IF NOT DONE
5750
5751      ;CHECK TO SEE THAT IN BCC MATCH IS SET
5752      ;AND THAT THE BCC WAS RECEIVED CORRECTLY
5753
5754 030106 004737 031066      JSR    PC,INRDY     ;WAIT FOR INRDY
5755 030112 104412      ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5756 030114 021204      021204 ;GET FIRST HALF OF CRC
5757 030116 116137 000004 001302      MOVB   4(R1),$TMP2  ;PUT IN $TMP2
5758 030124 042737 177400 001302      BIC    #177400,$TMP2 ;CLEAR HI BYTE
5759 030132 004737 031066      JSR    PC,INRDY     ;WAIT FOR INRDY
5760 030136 104412      ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5761 030140 021244      021244
5762 030142 016104 000004      MOV    4(R1),R4     ;PUT "FOUND" IN R4
5763 030146 042704 000376      BIC    #376,R4      ;CLEAR UNWANTED BITS
5764 030152 012705 000001      MOV    #1,R5        ;PUT "EXPECTED" IN R5
5765 030156 120504      CMPB   R5,R4        ;IS IN BCC MATCH SET?
5766 030160 001401      BEQ    25$          ;IN BCC MATCH ERROR
5767 030162 104015      ERROR  15
5768 030164      25$:
5769 030164 104412      ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5770 030166 021204      021204 ;GET LAST HALF
5771 030170 116137 000004 001301      MOVB   4(R1),$TMP1+1 ;PUT IN $TMP1
5772 030176 042737 000377 001300      BIC    #377,$TMP1   ;CLEAR LO BYTE
5773 030204 053737 001300 001302      BIS    $TMP1,$TMP2  ;16 BIT BCC NOW IN $TMP2
5774 030212 023737 031230 001302      CMP    CALBCC,$TMP2 ;IS IT CORRECT?
5775 030220 001401      BEQ    3$           ;BR IF OK
5776 030222 104027      ERROR  27
5777 030224 104420      3$: ADVANCE      ; ADVANCE TO NEXT TEST
5778
5779
5780      ;SUBROUTINES
5781      ;-----
5782
5783 030226      GETSI:
5784      ;THIS SUBROUTINE READS LU 17, AND PUTS IT INTO NITCH.
5785      ;NITCH IS ROTATED LEFT UNTILL THE SI BIT IS IN CARRY
5786
5787 030226 104412      ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5788 030230 021364      021364 ;PORT4_LU 17
5789 030232 017737 151636 030256      MOV    @KMP04,NITCH ;STORE LU 17
5790 030240 106137 030256      ROLB   NITCH
5791 030244 106137 030256      ROLB   NITCH
5792 030250 106137 030256      ROLB   NITCH      ;PUT SI IN THE CARRY BIT
5793 030254 000207      RTS    PC
5794 030256 000000      NITCH: 0
5795
5796
5797 030260      OCOR:
    
```



```

5798                                     ;THIS SUBROUTINE SPINS ON OCOR
5799
5800 030260 104412 ROMCLK                ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5801 030262 021364 021364                ;PORT4 LU 17
5802 030264 032777 000020 151602 BIT #BIT4,@KMP04 ;IS OCOR SET?
5803 030272 001772 BEQ OCOR                ;BR IF NO
5804 030274 000207 RTS PC                ;OK OCOR IS SET, GO BACK
5805
5806
5807 030276 SYNC:
5808                                     ;THIS SUBROUTINE LOADS THE SILO WITH THE NUMBER OF SYNC
5809                                     ;CHARACTERS PASSED TO IT IN THE WORD AFTER THE JSR CALL
5810                                     ;AND A NON-SYNC CHARACTER (301)
5811
5812 030276 013637 001276 MOV @ (SP)+,$TMP0 ;GET COUNT
5813 030302 062746 000002 ADD #2,-(SP)      ;ADJUST STACK
5814 030306 012761 000026 000004 MOV #26,4(R1)    ;LOAD PORT4
5815 030314 104412 ROMCLK                ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5816 030316 122114 122114                ;LOAD SYNC REGISTER
5817 030320 004737 030412 1$: JSR PC,OUTRDY    ;WAIT FOR OUTRDY
5818 030324 012761 000001 000004 MOV #1,4(R1)     ;LOAD PORT4
5819 030332 104412 ROMCLK                ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5820 030334 122111 122111                ;SET SOM
5821 030336 012761 000026 000004 MOV #26,4(R1)    ;LOAD PORT4
5822 030344 104412 ROMCLK                ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5823 030346 122110 122110                ;LOAD OUT DATA
5824 030350 005337 001276 DEC $TMP0        ;ALL DONE?
5825 030354 001361 BNE 1$                ;BR IF NOT
5826 030356 004737 030412 JSR PC,OUTRDY    ;WAIT FOR OUTRDY
5827 030362 005061 000004 CLR 4(R1)        ;LOAD PORT4
5828 030366 104412 ROMCLK                ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5829 030370 122111 122111                ;SET SOM
5830 030372 012761 000301 000004 MOV #301,4(R1)   ;LOAD PORT4
5831 030400 104412 ROMCLK                ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5832 030402 122110 122110                ;LOAD OUT DATA
5833 030404 004737 030260 JSR PC,OCOR     ;WAIT FOR OCOR
5834 030410 000207 RTS PC
5835
5836
5837 030412 OUTRDY:
5838                                     ;THIS SUBROUTINE SPINS ON OUT READY
5839
5840 030412 005037 001306 CLR $TMP4        ;CLEAR TIMER
5841 030416 1$:
5842 030416 104412 ROMCLK                ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5843 030420 021224 021224                ;PORT4 LU11
5844 030422 032777 000020 151444 BIT #BIT4,@KMP04 ;IS OUT RDY SET?
5845 030430 001004 BNE 2$                ;BR IF YES
5846 030432 005237 001306 INC $TMP4        ;INC TIMER
5847 030436 001367 BNE 1$                ;KEEP CHECKING IF NOT DONE
5848 030440 104036 ERROR 36             ;ERROR, OUT READY NOT SET
5849 030442 000207 2$: RTS PC
5850
5851
5852 030444 CHAR:
5853                                     ;THIS SUBROUTINE LOADS THE SILO WITH 3 SYNC

```

```

5854                                     ;AND THE CHARACTER PASSED TO IT.
5855
5856 030444 013637 001300                MOV     @ (SP)+,$TMP1      ;GET CHARACTER
5857 030450 062746 000002                ADD     #2,-(SP)          ;ADJUST STACK
5858 030454 012737 000003 001276        MOV     #3,$TMP0          ;SET FOR 3 SYNCs
5859 030462 012761 000026 000004        MOV     #26,4(R1)        ;LOAD PORT4
5860 030470 104412                        ROMCLK                    ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5861 030472 122114                        122114                   ;LOAD SYNC REGISTER
5862 030474 004737 030412 1$:          JSR     PC,OUTRDY        ;WAIT FOR OUTRDY
5863 030500 012761 000001 000004        MOV     #1,4(R1)         ;LOAD PORT4
5864 030506 104412                        ROMCLK                    ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5865 030510 122111                        122111                   ;SET SOM
5866 030512 012761 000026 000004        MOV     #26,4(R1)        ;LOAD PORT4
5867 030520 104412                        ROMCLK                    ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5868 030522 122110                        122110                   ;LOAD OUT DATA
5869 030524 005337 001276                DEC     $TMP0             ;ALL DONE?
5870 030530 001361                        BNE     1$               ;BR IF NOT
5871 030532 004737 030412                JSR     PC,OUTRDY        ;WAIT FOR OUTRDY
5872 030536 013761 001300 000004        MOV     $TMP1,4(R1)      ;LOAD PORT4
5873 030544 104412                        ROMCLK                    ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5874 030546 122110                        122110                   ;LOAD OUT DATA
5875 030550 004737 030260                JSR     PC,OCOR          ;WAIT FOR OCOR
5876 030554 000207                        RTS     PC
5877
5878
5879 030556                                CHARSD:
5880                                     ;THIS SUBROUTINE LOADS THE SILO WITH THE CHARACTER PASSED TO IT.
5881
5882 030556 013637 001300                MOV     @ (SP)+,$TMP1      ;GET CHARACTER
5883 030562 062746 000002                ADD     #2,-(SP)          ;ADJUST STACK
5884 030566 004737 030412                JSR     PC,OUTRDY        ;WAIT FOR OUTRDY
5885 030572 013761 001300 000004        MOV     $TMP1,4(R1)      ;LOAD PORT4
5886 030600 104412                        ROMCLK                    ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5887 030602 122110                        122110                   ;LOAD OUT DATA
5888 030604 004737 030412                JSR     PC,OUTRDY        ;WAIT FOR OUTRDY
5889 030610 104412                        ROMCLK                    ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5890 030612 122110                        122110                   ;LOAD GARBAGE CHAR
5891 030614 004737 030260                JSR     PC,OCOR          ;WAIT FOR OCOR
5892 030620 000207                        RTS     PC
5893
5894
5895 030622                                SILOLD:
5896                                     ;THIS SUBROUTINE FILLS THE OUT SILO
5897                                     ; WITH A BINARY COUNT PATTERN
5898
5899 030622 012737 000073 001300        MOV     #73,$TMP1        ;LOAD COUNT
5900 030630 005737 031062                TST     SCHAR            ;FIRST TIME HERE?
5901 030634 100470                        BMI     4$               ;BR IF BITSTUFF
5902 030636 001032                        BNE     2$               ;BR IF NO
5903 030640 062737 000002 001300        ADD     #2,$TMP1         ;ADD 2 TO CHARACTER COUNT
5904 030646 012737 000003 001276        MOV     #3,$TMP0          ;SET FOR 3 SYNCs
5905 030654 012761 000026 000004        MOV     #26,4(R1)        ;LOAD PORT4
5906 030662 104412                        ROMCLK                    ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
5907 030664 122114                        122114                   ;LOAD SYNC REGISTER
5908 030666 004737 030412 1$:          JSR     PC,OUTRDY        ;WAIT FOR OUTRDY
5909 030672 012761 000001 000004        MOV     #1,4(R1)         ;LOAD PORT4
  
```

| | | | | | | | |
|------|--------|--------|--------|--------|---------|--------------|---|
| 5910 | 030700 | 104412 | | | ROMCLK | | :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 |
| 5911 | 030702 | 122111 | | | 122111 | | :SET SOM |
| 5912 | 030704 | 012761 | 000026 | 000004 | MOV | #26,4(R1) | :LOAD PORT4 |
| 5913 | 030712 | 104412 | | | ROMCLK | | :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 |
| 5914 | 030714 | 122110 | | | 122110 | | :LOAD OUT DATA |
| 5915 | 030716 | 005337 | 001276 | | DEC | \$TMP0 | :ALL DONE? |
| 5916 | 030722 | 001361 | | | BNE | 1\$ | :BR IF NOT |
| 5917 | 030724 | 004737 | 030412 | | JSR | PC,OUTRDY | :WAIT FOR OUTRDY |
| 5918 | 030730 | 013761 | 031062 | 000004 | MOV | SCHAR,4(R1) | :LOAD PORT4 |
| 5919 | 030736 | 104412 | | | ROMCLK | | :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 |
| 5920 | 030740 | 122110 | | | 122110 | | :LOAD OUT DATA |
| 5921 | 030742 | 005737 | 031064 | | TST | STUFLG | :BITSTUFF??? |
| 5922 | 030746 | 001407 | | | BEQ | 6\$ | :BR IF NO |
| 5923 | 030750 | 013737 | 031062 | 030762 | MOV | SCHAR,5\$ | :IT IS SDLD SO CHECK BITSTUFFING |
| 5924 | 030756 | 004537 | 031612 | | JSR | R5,STFFCL | :ADD ANY BIT STUFF CLOCK TICKS |
| 5925 | 030762 | 000000 | | | 0 | | :CHARACTER |
| 5926 | 030764 | 000010 | | | 10 | | :CHIFT COUNT |
| 5927 | 030766 | 005237 | 031062 | | INC | SCHAR | :NEXT CHARACTER |
| 5928 | 030772 | 022737 | 000400 | 031062 | CMP | #400,SCHAR | :ALL DONE? |
| 5929 | 031000 | 001403 | | | BEQ | 3\$ | |
| 5930 | 031002 | 005337 | 001300 | | DEC | \$TMP1 | :DECREMENT COUNT |
| 5931 | 031006 | 001346 | | | BNE | 2\$ | :BR IF NOT DONE |
| 5932 | 031010 | 004737 | 030260 | | JSR | PC,OCOR | :WAIT FOR OCOR |
| 5933 | 031014 | 000207 | | | RTS | PC | |
| 5934 | 031016 | 005037 | 031062 | | CLR | SCHAR | :START PATTERN AT ZERO |
| 5935 | 031022 | 012737 | 177777 | 031064 | MOV | #-1,STUFLG | :SET BITSTUFF FLAG |
| 5936 | 031030 | 005037 | 032010 | | CLR | BITCON | :CLEAR STUFF COUNT |
| 5937 | 031034 | 062737 | 000002 | 001300 | ADD | #2,\$TMP1 | :ADD 2 TO CHARACTER COUNT |
| 5938 | 031042 | 012761 | 000001 | 000004 | MOV | #1,4(R1) | :SET BIT0 IN PORT4 |
| 5939 | 031050 | 104412 | | | ROMCLK | | :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 |
| 5940 | 031052 | 122111 | | | 122111 | | :SET SOM! |
| 5941 | 031054 | 104412 | | | ROMCLK | | :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 |
| 5942 | 031056 | 122110 | | | 122110 | | :LOAD GARBAGE CHAR |
| 5943 | 031060 | 000721 | | | BR | 2\$ | :GO LOAD SILO |
| 5944 | 031062 | 000000 | | | SCHAR: | 0 | |
| 5945 | 031064 | 000000 | | | STUFLG: | 0 | |
| 5946 | | | | | | | |
| 5947 | | | | | | | |
| 5948 | 031066 | | | | INRDY: | | |
| 5949 | | | | | | | :THIS SUBROUTINE SPINS ON INRDY |
| 5950 | | | | | | | :IF INRDY FAILS TO SET THE DELAY TIMES OUT AND AN |
| 5951 | | | | | | | :ERROR IS REPORTED. FOR BETTER SCOPE LOOPS THIS |
| 5952 | | | | | | | :DELAY CAN BE MADE SHORTER BY ALTERING THE NUMBER |
| 5953 | | | | | | | :INITIALLY LOADED INTO \$TMP0, THE SMALLER THE NUMBER |
| 5954 | | | | | | | :THE SHORTER THE DELAY. 0 IS THE LONGEST DELAY. |
| 5955 | | | | | | | |
| 5956 | 031066 | 012737 | 000000 | 001276 | MOV | #0,\$TMP0 | :SET UP DELAY COUNTER |
| 5957 | 031074 | | | | 1\$: | | |
| 5958 | 031074 | 104412 | | | ROMCLK | | :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 |
| 5959 | 031076 | 021244 | | | 021244 | | :PORT4 LU12 |
| 5960 | 031100 | 032777 | 000020 | 150766 | BIT | #BIT4,@KMP04 | :IS INRDY SET? |
| 5961 | 031106 | 001004 | | | BNE | 2\$ | :BR IF YES |
| 5962 | 031110 | 005237 | 001276 | | INC | \$TMP0 | :INC DELAY |
| 5963 | 031114 | 001367 | | | BNE | 1\$ | :TRY AGAIN |
| 5964 | 031116 | 104037 | | | ERROR | 37 | :ERROR,NO INRDY |
| 5965 | 031120 | 000207 | | | 2\$: | RTS | PC |

```

5966
5967
5968 031122      SIMBCC:
5969              ;THIS SUBROUTINE CALCULATES THE CRC USING POLYNOMIAL GIVEN
5970              ;IN XPOLY. THE CORRECT CRC IS $LPADRED IN CALBCC, AND THE
5971              ;STATE OF THE LSB OF THE BCC IS $LPADRED IN THE C BIT.
5972
5973 031122 010046      MOV      RO,-(SP)      ;SAVE RO ON STACK
5974 031124 012537 001276  MOV      (R5)+,$TMP0    ;$TMP0 = SHIFT COUNT
5975 031130 012537 001300  MOV      (R5)+,$TMP1    ;$TMP1 = CHARACTER
5976 031134 012537 031230  MOV      (R5)+,CALBCC   ;CALBCC = OLD BCC
5977 031140 013700 031230 1$:      MOV      CALBCC,RO    ;PUT OLD BCC IN RO
5978 031144 000241      CLC
5979 031146 006037 031230  ROR      CALBCC        ;SHIFT OLD BCC
5980 031152 006037 001300  ROR      $TMP1        ;SHIFT CHARACTER
5981 031156 005500      ADC      RO            ;ADD CHAR CARRY TO OLD BCC
5982 031160 006000      ROR      RO            ;PUT BITO TO CARRY BIT
5983 031162 103011      BCC      2$           ;CARRY IS FEEDBACK BIT
5984 031164 013700 031226  MOV      XPOLY,RO      ;IF FEEDBACK = 1
5985 031170 043700 031230  BIC      CALBCC,RO     ;EXCLUSIVLY OR XPOLY TO CALBCC
5986 031174 043737 031226  BIC      XPOLY,CALBCC
5987 031202 050037 031230  BIS      RO,CALBCC
5988 031206 005337 001276 2$:      DEC      $TMP0        ;DEC SHIFT COUNT
5989 031212 001352      BNE      1$           ;BR IF NOT DONE
5990 031214 013700 031230  MOV      CALBCC,RO    ;PUT RESULT IN RO
5991 031220 006000      ROR      RO            ;SHIFT BITO TO CARRY
5992 031222 012600      MOV      (SP)+,RO      ;RESTORE RO
5993 031224 000205      RTS      R5            ;$LPADR
5994 031226 000000      XPOLY: 0
5995 031230 000000      CALBCC: 0
5996              LRC8=200
5997              CRC16=120001
5998              CRC.CCITT=102010
5999
6000
6001 031232      BCCLD:
6002              ;THIS SUBROUTINE LOADS THE OUT SILO WITH 2 SYNCs
6003              ;WITH SOM SET, AND ONE CHARACTER PASSED TO IT
6004              ;WITH THE SOM BIT CLEAR (ENABLE CRC)
6005
6006 031232 013637 001300  MOV      @($P)+,$TMP1  ;GET CHARACTER
6007 031236 062746 000002  ADD      #2,-(SP)     ;ADJUST STACK
6008 031242 012737 000002  MOV      #2,$TMP0     ;SET FOR 2 SYNCs
6009 031250 012761 000026  MOV      #26,4(R1)    ;LOAD PORT4
6010 031256 104412      ROMCLK  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
6011 031260 122114      122114  ;LOAD SYNC REGISTER
6012 031262 004737 030412 1$:      JSR      PC,OUTRDY    ;WAIT FOR OUTRDY
6013 031266 012761 000001  MOV      #1,4(R1)    ;LOAD PORT4
6014 031274 104412      ROMCLK  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
6015 031276 122111      122111  ;SET SOM
6016 031300 012761 000026  MOV      #26,4(R1)    ;LOAD PORT4
6017 031306 104412      ROMCLK  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
6018 031310 122110      122110  ;LOAD OUT DATA
6019 031312 005337 001276  DEC      $TMP0        ;ALL DONE?
6020 031316 001361      BNE      1$           ;BR IF NOT
6021 031320 004737 030412  JSR      PC,OUTRDY    ;WAIT FOR OUTRDY
  
```

```

6022 031324 013761 001300 000004      MOV    $TMP1,4(R1)      ;LOAD PORT4
6023 031332 104412                    ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
6024 031334 122110                    122110                  ;LOAD OUT DATA
6025 031336 004737 030260            JSR    PC,OCOR          ;WAIT FOR OCOR
6026 031342 000207                    RTS    PC
6027
6028
6029 031344                          GETQO:
6030                                ;THIS SUBROUTINE READS THE STATE OF THE TRANSMIT
6031                                ;BCC LSB AND PUTS IT IN THE CARRY BIT
6032
6033 031344 104412                    ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
6034 031346 021364                    021364                  ;PORT4 LU-17
6035 031350 106177 150520            ROLB   @KMP04          ;PUT Q0 IN CARRY
6036 031354 000207                    RTS    PC                ;RETURN
6037
6038
6039 031356                          GETQI:
6040                                ;THIS SUBROUTINE READS THE STATE OF THE RECEIVE
6041                                ;BCC LSB AND PUTS IT IN THE CARRY BIT
6042
6043 031356 104412                    ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
6044 031360 021364                    021364                  ;PORT4 LU-17
6045 031362 106177 150506            ROLB   @KMP04          ;PUT Q0 IN CARRY
6046 031366 106177 150502            ROLB   @KMP04          ;PUT QI IN CARRY
6047 031372 000207                    RTS    PC                ;RETURN
6048
6049
6050 031374                          SYNLD:
6051                                ;THIS SUBROUTINE LOADS OUT SILO WITH
6052                                ;2 SYNC CHARACTERS WITH SOM SET
6053
6054 031374 012737 000002 001276      MOV    #2,$TMP0        ;LOAD COUNTER FOR 2 SYNC
6055 031402 012761 000026 000004      MOV    #26,4(R1)       ;PORT4 26
6056 031410 104412                    ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
6057 031412 122114                    122114                  ;LOAD SYNC REG
6058 031414 004737 030412 1$:        JSR    PC,OUTRDY       ;WAIT FOR OUTRDY
6059 031420 012761 000001 000004      MOV    #1,4(R1)        ;LOAD PORT4
6060 031426 104412                    ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
6061 031430 122111                    122111                  ;SET SOM
6062 031432 012761 000026 000004      MOV    #26,4(R1)       ;PORT 26
6063 031440 104412                    ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
6064 031442 122110                    122110                  ;LOAD OUT DATA WITH SYNC
6065 031444 005337 001276            DEC    $TMP0           ;DECREMENT COUNTER
6066 031450 001361                    BNE    1$              ;BR IF NOT DONE
6067 031452 000207                    RTS    PC                ;RETURN
6068
6069
6070 031454                          SOM:
6071                                ;THIS SUBROUTINE LOADS SOM AND OUT DATA WITH A
6072                                ;GARBAGE CHARACTER (0)
6073
6074 031454 004737 030412 000004      JSR    PC,OUTRDY       ;WAIT FOR OUTRDY
6075 031460 012761 000001 000004      MOV    #1,4(R1)        ;PORT4 1
6076 031466 104412                    ROMCLK                  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
6077 031470 122111                    122111                  ;SET SOM

```

```

6078 031472 005061 000004      CLR      4(R1)      ;CLEAR DATA CHAR
6079 031476 104412      ROMCLK      ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
6080 031500 122110      122110      ;LOAD GARBAGE CHARACTER
6081 031502 000207      RTS        PC      ;RETURN
6082
6083
6084 031504      EOM:
6085      ;THIS SUBROUTINE LOADS EOM AND OUT DATA WITH A
6086      ;GARBAGE CHARACTER (2) TO ENABLE TRANSMISSION OF BCC
6087
6088 031504 004737 030412      JSR      PC,OUTRDY  ;WAIT FOR OUTRDY
6089 031510 012761 000002 000004  MOV      #2,4(R1)  ;PORT4 2
6090 031516 104412      ROMCLK      ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
6091 031520 122111      122111      ;SET EOM
6092 031522 104412      ROMCLK      ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
6093 031524 122110      122110      ;LOAD GARBAGE CHARACTER
6094 031526 000207      RTS        PC      ;RETURN
6095
6096
6097 031530      MESLD:
6098      ;THIS SUBROUTINE LOADS SILO WITH MESSAGE
6099      ;THE FIRST ARGUMENT IS THE ADDRESS OF THE MESSAGE
6100      ;THE SECOND ARGUMENT IS THE NUMBER OF CHARACTERS IN THE MESSAGE
6101
6102 031530 010046      MOV      R0,-(SP)  ;SAVE R0
6103 031532 012500      MOV      (R5)+,R0  ;R0=MESSAGE POINTER
6104 031534 012537 001276  MOV      (R5)+,$TMPO ;$TMPO=CHARACTER COUNT
6105 031540 004737 030412  JSR      PC,OUTRDY  ;WAIT FOR OUT RDY
6106 031544 112061 000004  MOV      (R0)+,4(R1) ;LOAD PORT4 WITH CHARACTER
6107 031550 104412      ROMCLK      ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
6108 031552 122110      122110      ;LOAD OUT DATA SILO
6109 031554 005337 001276  DEC      $TMPO     ;DEC CHAR COUNT
6110 031560 001367      BNE      1$       ;BR IF NOT DONE
6111 031562 004737 030260  JSR      PC,OCOR   ;WAIT FOR OCOR
6112 031566 012600      MOV      (SP)+,R0  ;RESTORE R0
6113 031570 000205      RTS        R5      ;RETURN
6114
6115
6116 031572      CLRIO:
6117      ;THIS SUBROUTINE SETS IN CLR AND OUT CLR TO
6118      ;CLEAR THE TRANSMIT AND RECEIVE BCC REGISTERS
6119
6120 031572 012761 000200 000004  MOV      #BIT7,4(R1) ;LOAD PORT4
6121 031600 104412      ROMCLK      ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
6122 031602 122112      122112      ;SET IN CLR!
6123 031604 104412      ROMCLK      ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
6124 031606 122111      122111      ;SET OUT CLR!
6125 031610 000207      RTS        PC      ;RETURN
6126
6127
6128 031612      STFFCL:
6129      ;THIS SUBROUTINE ADDS ANY NECESSARY BIT STUFF CLOCK TICKS
6130      ;FIRST ARGUMENT IS CHAR, SECOND ARGUMENT IS SHIFT COUNT.
6131
6132 031612 010046      MOV      R0,-(SP)  ;SAVE R0
6133 031614 012500      MOV      (R5)+,R0  ;PUT CHAR IN R0

```

```

6134 031616 012537 001302      MOV      (R5)+,$TMP2      ;PUT SHIFT COUNT IN $TMP2
6135 031622 106000              1$: RORB      RO              ;LOOK AT NEXT BIT
6136 031624 103403              BCS      2$              ;BR IF A MARK
6137 031626 005037 032010      CLR      BITCON          ;IT WAS A SPACE, CLEAR 1'S COUNTER
6138 031632 000412              BR       3$              ;CONTINUE
6139 031634 005237 032010      2$: INC      BITCON          ;INC CONSECUTIVE 1'S COUNTER
6140 031640 022737 000005 032010  CMP      #5,BITCON        ;IS IT 5 YET?
6141 031646 001004              BNE      3$              ;BR IF NO
6142 031650 005037 032010      CLR      BITCON          ;YES! SO START AGAIN
6143 031654 104413 000001      DATACLK, 1              ;GIVE EXTRA TICK TO STUFF ZERO
6144 031660 005337 001302      3$: DEC      $TMP2          ;DEC SHIFT COUNT
6145 031664 001356              BNE      1$              ;BR IF NOT DONE
6146 031666 012600              MOV      (SP)+,RO        ;RESTORE RO
6147 031670 000205              RTS       R5              ;RETURN
6148
6149

```

```

6150 031672      STFFCK:
6151              ;THIS SUBROUTINE CHECKS TO SEE IF TRANSMITTER
6152              ;IS STUFFING ZEROS WHEN IT SHOULD. FIRST ARGUMENT
6153              ;IS THE CHARACTER, SECOND ARGUMENT IS SHIFT COUNT.
6154

```

```

6155 031672 010046      MOV      RO,-(SP)         ;SAVE RO
6156 031674 012500      MOV      (R5)+,RO        ;PUT CHAR IN RO
6157 031676 012537 001302      MOV      (R5)+,$TMP2    ;PUT SHIFT COUNT IN $TMP2
6158 031702 106000              1$: RORB      RO              ;SHIFT OUT NEXT BIT
6159 031704 103403              BCS      2$              ;BR IF IT IS A MARK
6160 031706 005037 032010      CLR      BITCON          ;IT WAS A SPACE, CLEAR 1'S COUNTER
6161 031712 000416              BR       3$              ;CONTINUE
6162 031714 005237 032010      2$: INC      BITCON          ;INC CONSECUTIVE I'S COUNTER
6163 031720 022737 000005 032010  CMP      #5,BITCON        ;5 IN A ROW YET?
6164 031726 001010              BNE      3$              ;BR IF NO
6165 031730 005037 032010      CLR      BITCON          ;YES, SO START OVER
6166 031734 104413 000001      DATACLK, 1              ;EXTRA TICK TO STUFF ZERO
6167 031740 004737 030226      JSR      PC,GETSI        ;LOOK AT WINDOW
6168 031744 103001              BCC      3$              ;IS IT A ZERO, BR IF YES
6169 031746 104030              ERROR    30              ;NO, ERROR ZERO WAS NOT STUFFED
6170 031750 005337 001302      3$: DEC      $TMP2          ;DEC SHIFT COUNT
6171 031754 001352              BNE      1$              ;BR IF NOT DONE
6172 031756 012600              MOV      (SP)+,RO        ;RESTORE RO
6173 031760 000205              RTS       R5              ;RETURN
6174
6175

```

```

6176 031762      CTSDLY:
6177              ;THIS SUBROUTINE WASTES TIME UNTIL CTS SETS,
6178              ;BUT HOPEFULLY NOT SO LONG THAT THE SILO RUNS OUT
6179

```

```

6180 031762 010046      MOV      RO,-(SP)         ;SAVE RO
6181 031764 012700      MOV      #32,RO          ;LOAD RO WITH COUNT
6182 031770 027777 000032 147250 147246 1$: CMP      @STKS,@STKS      ;WASTE TIME
6183 031776 005300      DEC      RO              ;DECREMENT COUNTER
6184 032000 001373              BNE      1$              ;DO IT AGAIN IF NOT = 0
6185 032002 012600      MOV      (SP)+,RO        ;RESTORE RO
6186 032004 000207      RTS       PC              ;RETURN
6187
6188

```

```

6189 032006 000176      FLAG:  *B<01111110>    ;FLAG CHARACTER

```

| | | | | | | |
|------|--------|--------|-----|-----|---------------|--|
| 6190 | 032010 | 000000 | | | BITCON: 0 | |
| 6191 | 032012 | 000 | 125 | 252 | MESDAT: .BYTE | 0,125,252,377 |
| 6192 | 032015 | 377 | | | | |
| 6193 | 032016 | 001 | 002 | 004 | FLTDAT: .BYTE | 1,2,4,10,20,40,100,200,376,375,373,367,357,337,277,177 |
| 6194 | 032021 | 010 | 020 | 040 | | |
| 6195 | 032024 | 100 | 200 | 376 | | |
| 6196 | 032027 | 375 | 373 | 367 | | |
| 6197 | 032032 | 357 | 337 | 277 | | |
| 6198 | 032035 | 177 | | | | |
| 6199 | 032036 | 100 | 140 | 160 | STUFDT: .BYTE | 100,140,160,170,3,300,174,176,177,1 |
| 6200 | 032041 | 170 | 003 | 300 | | |
| 6201 | 032044 | 174 | 176 | 177 | | |
| 6202 | 032047 | 001 | | | | |
| 6203 | 032050 | 363 | 347 | 317 | .BYTE | 363,347,317,200,0,377,377,377,200,37 |
| 6204 | 032053 | 200 | 000 | 377 | | |
| 6205 | 032056 | 377 | 377 | 200 | | |
| 6206 | 032061 | 037 | | | | |

| | | | | | | |
|------|--------|--------|--------|--------|-------|--|
| 6207 | | | | | .EVEN | |
| 6208 | 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 | | | | EM10: | |
| | 032365 | 200 | 047515 | 042504 | | .ASCII <200>/MODEM SIGNAL ERROR/ |
| | 032410 | 052200 | 044510 | 020123 | | .ASCII <200>/THIS ERROR COULD BE CAUSED IF YOU HAVE V.35 AND / |
| | 032471 | 200 | 052501 | 047524 | | .ASCIZ <200>/AUTOSIZED. IF V.35, MANUALLY ANSWER QUESTIONS/ |
| | 032550 | 052200 | 040522 | 051516 | EM11: | .ASCIZ <200>/TRANSMITTER CRC ERROR/ |
| | 032577 | 200 | 042522 | 042503 | EM12: | .ASCIZ <200>/RECEIVER CRC ERROR/ |
| | 032623 | 200 | 047111 | 041040 | EM13: | .ASCIZ <200>/IN BCC MATCH ERROR (LU REG 12)/ |
| | 032663 | 200 | 051124 | 047101 | EM14: | .ASCIZ <200>/TRANSMITTER FAILED TO GO TO MARK STATE/ |
| | 032733 | 200 | 040503 | 046102 | EM15: | .ASCIZ <200>/CABLE DATA TEST/ |
| | 032754 | 043200 | 040514 | 020107 | EM16: | .ASCIZ <200>/FLAG ERROR/ |
| | 032770 | 052200 | 040522 | 051516 | EM17: | .ASCIZ <200>/TRANSMITTER FAILED TO STUFF A ZERO/ |
| | 033034 | 051600 | 044527 | 041524 | EM20: | .ASCIZ <200>/SWITCH PAC TEST/ |
| | 033055 | 200 | 041101 | 051117 | EM21: | .ASCIZ <200>/ABORT ERROR/ |
| | 033072 | 052200 | 040522 | 051516 | EM22: | .ASCIZ <200>/TRANSMITTER ERROR/ |
| | 033115 | 200 | 040510 | 043114 | EM23: | .ASCIZ <200>/HALF DUPLEX TEST/ |
| | 033137 | 200 | 052517 | 020124 | EM24: | .ASCIZ <200>/OUT READY NOT SET/ |
| | 033162 | 044600 | 020116 | 042522 | EM25: | .ASCIZ <200>/IN READY NOT SET/ |

| | | | | | | |
|--|--------|--------|--------|--------|-------|--|
| | 033204 | 042600 | 050130 | 041505 | DH1: | .ASCIZ <200>/EXPECTED FOUND/ |
| | 033225 | 200 | 054105 | 042520 | DH2: | .ASCIZ <200>/EXPECTED FOUND LU-REGISTER/ |
| | 033263 | 200 | 044103 | 051101 | DH3: | .ASCIZ <200>/CHARACTER BIT THAT FAILED/ |
| | 033321 | 200 | 047503 | 051122 | DH4: | .ASCIZ <200>/CORRECT CRC BIT THAT FAILED/ |
| | 033361 | 200 | 054105 | 042520 | DH5: | .ASCIZ <200>/EXPECTED FOUND SHIFT/ |
| | 033413 | 200 | 054105 | 042520 | DH6: | .ASCIZ <200>/EXPECTED FOUND CHARACTER SHIFT/ |
| | 033461 | 200 | 046102 | 041517 | DH7: | .ASCIZ <200>/BLOCK END NOT SET/ |
| | 033504 | 051200 | 051524 | 042040 | DH10: | .ASCIZ <200>/RTS DID NOT CLEAR/ |
| | | 033530 | | | .EVEN | |

| | | | | | | |
|--|--------|--------|-----|--|--------|-----|
| | 033530 | 000002 | | | DT1: | 2 |
| | 033532 | 003 | 007 | | .BYTE | 3,7 |
| | 033534 | 001274 | | | \$REG5 | |

| | | | | | |
|--------|--------|-----|-------|--------|------|
| 033536 | 003 | 002 | | .BYTE | 3,2 |
| 033540 | 001272 | | | \$REG4 | |
| 033542 | 000003 | | DT2: | 3 | |
| 033544 | 003 | 007 | | .BYTE | 3,7 |
| 033546 | 001274 | | | \$REG5 | |
| 033550 | 003 | 010 | | .BYTE | 3,10 |
| 033552 | 001272 | | | \$REG4 | |
| 033554 | 003 | 002 | | .BYTE | 3,2 |
| 033556 | 001266 | | | \$REG2 | |
| 033560 | 000002 | | DT3: | 2 | |
| 033562 | 003 | 017 | | .BYTE | 3,17 |
| 033564 | 001274 | | | \$REG5 | |
| 033566 | 002 | 002 | | .BYTE | 2,2 |
| 033570 | 001270 | | | \$REG3 | |
| 033572 | 000002 | | DT4: | 2 | |
| 033574 | 006 | 021 | | .BYTE | 6,21 |
| 033576 | 031230 | | | CALBCC | |
| 033600 | 002 | 002 | | .BYTE | 2,2 |
| 033602 | 001270 | | | \$REG3 | |
| 033604 | 000003 | | DT5: | 3 | |
| 033606 | 001 | 011 | | .BYTE | 1,11 |
| 033610 | 001462 | | | ZERO | |
| 033612 | 001 | 011 | | .BYTE | 1,11 |
| 033614 | 001464 | | | ONE | |
| 033616 | 002 | 002 | | .BYTE | 2,2 |
| 033620 | 001262 | | | \$REG0 | |
| 033622 | 000003 | | DT6: | 3 | |
| 033624 | 001 | 011 | | .BYTE | 1,11 |
| 033626 | 001464 | | | ONE | |
| 033630 | 001 | 011 | | .BYTE | 1,11 |
| 033632 | 001462 | | | ZERO | |
| 033634 | 002 | 002 | | .BYTE | 2,2 |
| 033636 | 001262 | | | \$REG0 | |
| 033640 | 000004 | | DT7: | 4 | |
| 033642 | 001 | 011 | | .BYTE | 1,11 |
| 033644 | 001462 | | | ZERO | |
| 033646 | 001 | 011 | | .BYTE | 1,11 |
| 033650 | 001464 | | | ONE | |
| 033652 | 003 | 007 | | .BYTE | 3,7 |
| 033654 | 001274 | | | \$REG5 | |
| 033656 | 002 | 001 | | .BYTE | 2,1 |
| 033660 | 001270 | | | \$REG3 | |
| 033662 | 000004 | | DT10: | 4 | |
| 033664 | 001 | 011 | | .BYTE | 1,11 |
| 033666 | 001464 | | | ONE | |
| 033670 | 001 | 011 | | .BYTE | 1,11 |
| 033672 | 001462 | | | ZERO | |
| 033674 | 003 | 007 | | .BYTE | 3,7 |
| 033676 | 001274 | | | \$REG5 | |
| 033700 | 002 | 001 | | .BYTE | 2,1 |
| 033702 | 001270 | | | \$REG3 | |
| 033704 | 000002 | | DT11: | 2 | |
| 033706 | 003 | 007 | | .BYTE | 3,7 |
| 033710 | 032006 | | | FLAG | |
| 033712 | 002 | 002 | | .BYTE | 2,2 |
| 033714 | 001270 | | | \$REG3 | |

CZKCE MACY11 30A(1052) 08-JUL-80 08:24 PAGE 122
CZKCE.P11 08-JUL-80 08:24 SUBROUTINES

E 10

SEQ 0121

| | | | | | |
|--------|--------|-----|-------|--------|-----|
| 033716 | 000002 | | DT12: | 2 | |
| 033720 | 006 | 004 | | .BYTE | 6,4 |
| 033722 | 031230 | | | CALBCC | |
| 033724 | 006 | 002 | | .BYTE | 6,2 |
| 033726 | 001302 | | | \$TMP2 | |

| | | | | |
|--------|--------|--|---------|--|
| 033730 | 000001 | | CORMAX: | |
| | | | .END | |

| | | | | |
|------------------|-------|-------|-------|------|
| ABASE = 000000 | 951 | 992 | | |
| ACDW1 = 000000 | 951 | 994 | | |
| ACDW2 = 000000 | 951 | 995 | | |
| ACPUOP = 000000 | 951 | 966 | | |
| ADDW0 = 000000 | 951 | 996 | | |
| ADDW1 = 000000 | 951 | 997 | | |
| ADDW10 = 000000 | 951 | 1006 | | |
| ADDW11 = 000000 | 951 | 1007 | | |
| ADDW12 = 000000 | 951 | 1008 | | |
| ADDW13 = 000000 | 951 | 1009 | | |
| ADDW14 = 000000 | 951 | 1010 | | |
| ADDW15 = 000000 | 951 | 1011 | | |
| ADDW2 = 000000 | 951 | 998 | | |
| ADDW3 = 000000 | 951 | 999 | | |
| ADDW4 = 000000 | 951 | 1000 | | |
| ADDW5 = 000000 | 951 | 1001 | | |
| ADDW6 = 000000 | 951 | 1002 | | |
| ADDW7 = 000000 | 951 | 1003 | | |
| ADDW8 = 000000 | 951 | 1004 | | |
| ADDW9 = 000000 | 951 | 1005 | | |
| ADEVCT = 000000 | 951 | 957 | | |
| ADEVN = 000000 | 951 | 993 | | |
| ADRCNT 006057 | 2098* | 2113* | 2122# | |
| ADVANC = 104420 | 2267# | 5694 | 5777 | |
| AENV = 000002 | 1# | 951 | 962 | |
| AENVN = 000000 | 951 | 963 | | |
| AFATAL = 000000 | 951 | 954 | | |
| AMADR1 = 000000 | 951 | 979 | | |
| AMADR2 = 000000 | 951 | 983 | | |
| AMADR3 = 000000 | 951 | 986 | | |
| AMADR4 = 000000 | 951 | 989 | | |
| AMAMS1 = 000000 | 951 | 973 | | |
| AMAMS2 = 000000 | 951 | 981 | | |
| AMAMS3 = 000000 | 951 | 984 | | |
| AMAMS4 = 000000 | 951 | 987 | | |
| AMSGAD = 000000 | 951 | 959 | | |
| AMSGLG = 000000 | 951 | 960 | | |
| AMSGTY = 000000 | 951 | 953 | | |
| AMTYP1 = 000000 | 951 | 974 | | |
| AMTYP2 = 000000 | 951 | 982 | | |
| AMTYP3 = 000000 | 951 | 985 | | |
| AMTYP4 = 000000 | 951 | 988 | | |
| APASS = 000000 | 951 | 956 | | |
| APRIOR = 000000 | 951 | | | |
| APTC SU = 000040 | 1823 | 1928# | | |
| APTENV = 000001 | 1816 | 1884 | 1926# | 2328 |
| APTSIZ = 000200 | 1925# | | | |
| APTSPO = 000100 | 1818 | 1886 | 1927# | |
| APT.SI 013716 | 1491 | 2927# | | |
| ASWREG = 000000 | 951 | 964 | | |
| ATESTN = 000000 | 951 | 955 | | |
| AUDONE 003354 | 1528 | 1549 | 1588# | |
| AUNIT = 000000 | 951 | 958 | | |
| AUSTRY 003126 | 1527# | | | |
| AUSWR = 000000 | 951 | 965 | | |
| AUTO.S 012236 | 1489 | 2646# | | |

CZKCE MACY11 30A(1052) 08-JUL-80 08:24 PAGE 136
CZKCE.P11 08-JUL-80 08:24

CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0134

| | | | | | | | | | | | | | | |
|---------|----------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| SMSGTY | 001316 | 953# | 1890 | 1898* | 1910 | 1914* | | | | | | | | |
| SMSWR | 005541 | 2029# | 2506 | | | | | | | | | | | |
| SMTYP1 | 001347 | 974# | | | | | | | | | | | | |
| SMTYP2 | 001353 | 982# | | | | | | | | | | | | |
| SMTYP3 | 001357 | 985# | | | | | | | | | | | | |
| SMTYP4 | 001363 | 988# | | | | | | | | | | | | |
| SMXCNT | 004362 | 1767 | 1777# | | | | | | | | | | | |
| SN | = 000057 | 1# | 2976 | 2982 | 2984 | 2991# | 3002 | 3008 | 3010 | 3017# | 3027 | 3033 | 3035 | 3042 |
| | | 3043# | 3053 | 3059 | 3061 | 3068 | 3069# | 3083 | 3089 | 3091 | 3099 | 3100# | 3126 | 3132 |
| | | 3134 | 3142 | 3143# | 3169 | 3175 | 3177 | 3185 | 3186# | 3228 | 3234 | 3236 | 3244 | 3245# |
| | | 3281 | 3287 | 3289 | 3296 | 3297# | 3305 | 3311 | 3313 | 3320 | 3321# | 3329 | 3335 | 3337 |
| | | 3344 | 3345# | 3364 | 3371 | 3373 | 3380 | 3381# | 3398 | 3405 | 3407 | 3414 | 3415# | 3443 |
| | | 3450 | 3452 | 3459 | 3460# | 3501 | 3509 | 3511 | 3518 | 3519# | 3554 | 3562 | 3564 | 3571 |
| | | 3572# | 3607 | 3615 | 3617 | 3624 | 3625# | 3660 | 3668 | 3670 | 3677 | 3678# | 3713 | 3722 |
| | | 3724 | 3731 | 3732# | 3776 | 3782 | 3784 | 3791 | 3792# | 3806 | 3812 | 3814 | 3821 | 3822# |
| | | 3836 | 3842 | 3844 | 3851 | 3852# | 3866 | 3872 | 3874 | 3881 | 3882# | 3896 | 3903 | 3905 |
| | | 3912 | 3913# | 3947 | 3953 | 3955 | 3962 | 3963# | 3986 | 3992 | 3994 | 4001 | 4002# | 4025 |
| | | 4031 | 4033 | 4040 | 4041# | 4064 | 4070 | 4072 | 4079 | 4080# | 4103 | 4109 | 4111 | 4118 |
| | | 4119# | 4144 | 4152 | 4154 | 4161 | 4162# | 4190 | 4198 | 4200 | 4207 | 4208# | 4237 | 4244 |
| | | 4246 | 4253 | 4254# | 4305 | 4311 | 4313 | 4320 | 4321# | 4361 | 4368 | 4370 | 4378 | 4379# |
| | | 4438 | 4445 | 4447 | 4455 | 4456# | 4515 | 4522 | 4524 | 4532 | 4533# | 4592 | 4599 | 4601 |
| | | 4609 | 4610# | 4669 | 4675 | 4677 | 4684 | 4685# | 4737 | 4743 | 4745 | 4752 | 4753# | 4805 |
| | | 4813 | 4815 | 4822 | 4823# | 4908 | 4915 | 4917 | 4924 | 4925# | 4967 | 4978 | 4980 | 4987 |
| | | 4988# | 5215 | 5227 | 5229 | 5236 | 5237# | 5495 | 5503 | 5505 | 5512 | 5513# | 5561 | 5567 |
| | | 5569 | 5576 | 5577# | 5598 | 5608 | 5610 | 5617 | 5618# | 5695 | 5704 | 5706 | 5713 | 5714# |
| | | 5778# | | | | | | | | | | | | |
| SNULL | 001254 | 926# | 1843 | 1872 | | | | | | | | | | |
| SNWTST= | 000000 | 2986# | 3012# | 3037# | 3063# | 3093# | 3136# | 3179# | 3238# | 3291# | 3315# | 3339# | 3375# | 3409# |
| | | 3454# | 3513# | 3566# | 3619# | 3672# | 3726# | 3786# | 3816# | 3846# | 3876# | 3907# | 3957# | 3996# |
| | | 4035# | 4074# | 4113# | 4156# | 4202# | 4248# | 4315# | 4372# | 4449# | 4526# | 4603# | 4679# | 4747# |
| | | 4817# | 4919# | 4982# | 5231# | 5507# | 5571# | 5612# | 5708# | | | | | |
| SOVER | 004334 | 1739 | 1742 | 1753 | 1765 | 1771# | | | | | | | | |
| SPASS | 001324 | 956# | 1663* | 1675 | 1687* | 1688* | 1705 | 1713 | 1761 | 1778 | 2583* | | | |
| SPASTM | 002042 | 1192# | | | | | | | | | | | | |
| SPWRDN | 007126 | 864 | 1429 | 2364# | 2399 | | | | | | | | | |
| SPWRMG | 007312 | 2402# | | | | | | | | | | | | |
| SPWRUP | 007200 | 2374 | 2380# | | | | | | | | | | | |
| SQUES | 001312 | 944# | 1872 | 2010 | 2027 | 2084 | 2087 | 2107 | 2630 | 2702 | 2718 | 2732 | 2752 | |
| SRDCHR | 005144 | 1947# | 2253 | | | | | | | | | | | |
| SRDDEC= | ***** U | 2256 | | | | | | | | | | | | |
| SRDLIN | 005264 | 1975# | 2254 | | | | | | | | | | | |
| SRDOCT | 005564 | 2048# | 2255 | | | | | | | | | | | |
| SRDSZ = | 000007 | 1968# | | | | | | | | | | | | |
| SREGAD | 001260 | 930# | | | | | | | | | | | | |
| SREG0 | 001262 | 932# | 2143* | 2148 | 6208 | | | | | | | | | |
| SREG1 | 001264 | 933# | 1563* | 1575 | 2142* | 2149 | | | | | | | | |
| SREG2 | 001266 | 934# | 2141* | 2150 | 6208 | | | | | | | | | |
| SREG3 | 001270 | 935# | 2140* | 2151 | 6208 | | | | | | | | | |
| SREG4 | 001272 | 936# | 2139* | 2152 | 6208 | | | | | | | | | |
| SREG5 | 001274 | 937# | 2138* | 2153 | 6208 | | | | | | | | | |
| SRTNAD | 004102 | 1704# | | | | | | | | | | | | |
| SR2A = | ***** U | 2256 | | | | | | | | | | | | |
| SS | = 000061 | 1# | 2989 | 2991# | 3015 | 3017# | 3040 | 3043# | 3066 | 3069# | 3096 | 3100# | 3139 | 3143# |
| | | 3182 | 3186# | 3241 | 3245# | 3294 | 3297# | 3318 | 3321# | 3342 | 3345# | 3378 | 3381# | 3412 |
| | | 3415# | 3457 | 3460# | 3516 | 3519# | 3569 | 3572# | 3622 | 3625# | 3675 | 3678# | 3729 | 3732# |
| | | 3789 | 3792# | 3819 | 3822# | 3849 | 3852# | 3879 | 3882# | 3910 | 3913# | 3960 | 3963# | 3999 |

CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0135

| | | | | | | | | | | | | | |
|-------------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| | 4002# | 4038 | 4041# | 4077 | 4080# | 4116 | 4119# | 4159 | 4162# | 4205 | 4208# | 4251 | 4254# |
| | 4318 | 4321# | 4375 | 4379# | 4452 | 4456# | 4529 | 4533# | 4606 | 4610# | 4682 | 4685# | 4750 |
| | 4753# | 4820 | 4823# | 4922 | 4925# | 4985 | 4988# | 5234 | 5237# | 5510 | 5513# | 5574 | 5577# |
| | 5615 | 5618# | 5711 | 5714# | | | | | | | | | |
| \$SAVRE = ***** U | 2256 | | | | | | | | | | | | |
| \$SAVR6 007322 | 2373* | 2381 | 2382* | 2383* | 2406# | | | | | | | | |
| \$SCOPE 004134 | 862 | 1733# | 2907 | | | | | | | | | | |
| \$SETUP= 000000 | 1686 | 1734 | 1936 | 2033 | | | | | | | | | |
| \$SVLAD 004316 | 1750 | 1768# | | | | | | | | | | | |
| \$SVPC = 000040 | 874# | 879 | | | | | | | | | | | |
| \$SWR = 164000 | 1# | 696 | 943 | 944 | 1658 | 1686 | 1697 | 1703 | 1705 | 1727 | 1728 | 1729 | 1730 |
| | 1741 | 1753 | 1755 | 1756 | 1757 | 1758 | 1759 | 1771 | 1777 | 2403 | 2988 | 3014 | 3039 |
| | 3065 | 3095 | 3138 | 3181 | 3240 | 3293 | 3317 | 3341 | 3377 | 3411 | 3456 | 3515 | 3568 |
| | 3621 | 3674 | 3728 | 3788 | 3818 | 3848 | 3878 | 3909 | 3959 | 3998 | 4037 | 4076 | 4115 |
| | 4158 | 4204 | 4250 | 4317 | 4374 | 4451 | 4528 | 4605 | 4681 | 4749 | 4819 | 4921 | 4984 |
| | 5233 | 5509 | 5573 | 5614 | 5710 | | | | | | | | |
| \$SWREG 001340 | 964# | 1447 | | | | | | | | | | | |
| \$SWRMK= 000000 | 1730 | | | | | | | | | | | | |
| \$TESTN 001322 | 955# | 1769* | | | | | | | | | | | |
| \$TIMES 001310 | 943# | 1686* | 1758* | 1764 | 1767* | 1777 | | | | | | | |
| \$TKB 001246 | 923# | 1740 | 1934 | 1951 | 1957 | 2496 | 2498 | 2540 | 2920 | | | | |
| \$TKS 001244 | 922# | 1738 | 1934 | 1949 | 1955 | 2538 | 2918 | 6182 | | | | | |
| \$TMP0 001276 | 938# | 1502* | 2469 | 2875* | 2876* | 5812* | 5824* | 5858* | 5869* | 5904* | 5915* | 5956* | 5962* |
| | 5974* | 5988* | 6008* | 6019* | 6054* | 6065* | 6104* | 6109* | | | | | |
| \$TMP1 001300 | 939# | 1503* | 2471 | 5165* | 5166* | 5167 | 5208* | 5209* | 5210 | 5434* | 5435* | 5436 | 5488* |
| | 5489* | 5490 | 5685* | 5686* | 5687 | 5771* | 5772* | 5773 | 5856* | 5872 | 5882* | 5885 | 5899* |
| \$TMP2 001302 | 5903* | 5930* | 5937* | 5975* | 5980* | 6006* | 6022 | | | | | | |
| | 940# | 1505* | 1526* | 1553 | 1562* | 2473 | 2663 | 2666 | 2778* | 4952* | 4953* | 5151* | 5152* |
| | 5167* | 5168 | 5194* | 5195* | 5210* | 5211 | 5420* | 5421* | 5436* | 5437 | 5474* | 5475* | 5490* |
| | 5491 | 5671* | 5672* | 5687* | 5688 | 5757* | 5758* | 5773* | 5774 | 6134* | 6144* | 6157* | 6170* |
| | 6208 | | | | | | | | | | | | |
| \$TMP3 001304 | 941# | 1506* | 2475 | 2675 | 2678 | 2683 | 2686 | 2765 | 2768 | 2773 | 2776 | | |
| \$TMP4 001306 | 942# | 1507* | 2477 | 2658* | 2670* | 2861 | 5840* | 5846* | | | | | |
| \$STN = 000060 | 1# | 696 | 2986 | 2988# | 3012 | 3014# | 3037 | 3039# | 3063 | 3065# | 3093 | 3095# | 3136 |
| | 3138# | 3179 | 3181# | 3238 | 3240# | 3291 | 3293# | 3315 | 3317# | 3339 | 3341# | 3375 | 3377# |
| | 3409 | 3411# | 3454 | 3456# | 3513 | 3515# | 3566 | 3568# | 3619 | 3621# | 3672 | 3674# | 3726 |
| | 3728# | 3786 | 3788# | 3816 | 3818# | 3846 | 3848# | 3876 | 3878# | 3907 | 3909# | 3957 | 3959# |
| | 3996 | 3998# | 4035 | 4037# | 4074 | 4076# | 4113 | 4115# | 4156 | 4158# | 4202 | 4204# | 4248 |
| | 4250# | 4315 | 4317# | 4372 | 4374# | 4449 | 4451# | 4526 | 4528# | 4603 | 4605# | 4679 | 4681# |
| | 4747 | 4749# | 4817 | 4819# | 4919 | 4921# | 4982 | 4984# | 5231 | 5233# | 5507 | 5509# | 5571 |
| | 5573# | 5612 | 5614# | 5708 | 5710# | | | | | | | | |
| \$TPB 001252 | 925# | 1861* | 1872 | 2279* | 2543* | 2923* | | | | | | | |
| \$TPFLG 001257 | 929# | 1810 | 1872 | | | | | | | | | | |
| \$TPS 001250 | 924# | 1859 | 1872 | 2277 | 2541 | 2921 | | | | | | | |
| \$TRAP 006414 | 868 | 2227# | | | | | | | | | | | |
| \$TRAP2 006436 | 2238# | 2249 | | | | | | | | | | | |
| \$TRP = 000021 | 2242# | 2251# | 2253 | 2254# | 2255# | 2256# | 2257# | 2258# | 2259# | 2260# | 2261# | 2262# | 2263# |
| | 2264# | 2265# | 2266# | 2267# | 2268# | | | | | | | | |
| \$TRPAD 006450 | 2232 | 2249# | | | | | | | | | | | |
| \$TSTM 002040 | 1191# | | | | | | | | | | | | |
| \$TSTM 001202 | 902# | 1442* | 1726 | 1768* | 1769 | 1771 | 1778 | 2356 | 2410 | 2612 | 2619 | 2621 | 2988* |
| | 3014* | 3039* | 3065* | 3095* | 3138* | 3181* | 3240* | 3293* | 3317* | 3341* | 3377* | 3411* | 3456* |
| | 3515* | 3568* | 3621* | 3674* | 3728* | 3788* | 3818* | 3848* | 3878* | 3909* | 3959* | 3998* | 4037* |
| | 4076* | 4115* | 4158* | 4204* | 4250* | 4317* | 4374* | 4451* | 4528* | 4605* | 4681* | 4749* | 4819* |
| | 4921* | 4984* | 5233* | 5509* | 5573* | 5614* | 5710* | | | | | | |
| \$TIYIN 005520 | 1977 | 1978 | 1990 | 2008 | 2022 | 2026# | | | | | | | |

| | | | | | | | | | | | | | | | |
|----------|-------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| \$STUFF | 686# | | | | | | | | | | | | | | |
| \$SWPAC | 686# | 3281 | 3305 | | | | | | | | | | | | |
| \$TCHAR | 686# | 4826 | 4925 | 4991 | 5240 | | | | | | | | | | |
| \$TCRC | 686# | 4805 | 4967 | 5215 | | | | | | | | | | | |
| \$STRANW | 686# | 4845 | 5014 | 5065 | 5264 | 5334 | | | | | | | | | |
| \$STRAN1 | 686# | 3364 | 3398 | 3443 | | | | | | | | | | | |
| \$TSTN | 1# | 2984 | 3010 | 3035 | 3061 | 3091 | 3134 | 3177 | 3236 | 3289 | 3313 | 3337 | 3373 | 3407 | 3452 |
| | 3511 | 3564 | 3617 | 3670 | 3724 | 3784 | 3814 | 3844 | 3874 | 3905 | 3955 | 3994 | 4033 | 4072 | 4111 |
| | 4154 | 4200 | 4246 | 4313 | 4370 | 4447 | 4524 | 4601 | 4677 | 4745 | 4815 | 4917 | 4980 | 5229 | 5505 |
| | 5569 | 5610 | 5706 | | | | | | | | | | | | |
| \$UPADD | 1# | 2385 | | | | | | | | | | | | | |
| \$VARIA | 1# | 888 | | | | | | | | | | | | | |
| \$WINDO | 686# | 3501 | 3554 | 3607 | 3660 | | | | | | | | | | |
| \$XZ | 1# | 2976 | 2982 | 3002 | 3008 | 3027 | 3033 | 3053 | 3059 | 3083 | 3089 | 3126 | 3132 | 3169 | 3175 |
| | 3228 | 3234 | 3281 | 3287 | 3305 | 3311 | 3329 | 3335 | 3364 | 3371 | 3398 | 3405 | 3443 | 3450 | 3501 |
| | 3509 | 3554 | 3562 | 3607 | 3615 | 3660 | 3668 | 3713 | 3722 | 3776 | 3782 | 3806 | 3812 | 3836 | 3842 |
| | 3866 | 3872 | 3896 | 3903 | 3947 | 3953 | 3986 | 3992 | 4025 | 4031 | 4064 | 4070 | 4103 | 4109 | 4144 |
| | 4152 | 4190 | 4198 | 4237 | 4244 | 4305 | 4311 | 4361 | 4368 | 4438 | 4445 | 4515 | 4522 | 4592 | 4599 |
| | 4669 | 4675 | 4737 | 4743 | 4805 | 4813 | 4908 | 4915 | 4967 | 4978 | 5215 | 5227 | 5495 | 5503 | 5561 |
| | 5567 | 5598 | 5608 | 5695 | 5704 | | | | | | | | | | |
| \$ZEROS | 686# | | | | | | | | | | | | | | |
| \$SCMRE | 893# | 932 | 933 | 934 | 935 | 936 | 937 | | | | | | | | |
| \$SCMTM | 893# | 938 | 939 | 940 | 941 | 942 | | | | | | | | | |
| \$SESCA | 829# | | | | | | | | | | | | | | |
| \$SNEW | 829# | 2986 | 3012 | 3037 | 3063 | 3093 | 3136 | 3179 | 3238 | 3291 | 3315 | 3339 | 3375 | 3409 | 3454 |
| | 3513 | 3566 | 3619 | 3672 | 3726 | 3786 | 3816 | 3846 | 3876 | 3907 | 3957 | 3996 | 4035 | 4074 | 4113 |
| | 4156 | 4202 | 4248 | 4315 | 4372 | 4449 | 4526 | 4603 | 4679 | 4747 | 4817 | 4919 | 4982 | 5231 | 5507 |
| | 5571 | 5612 | 5708 | | | | | | | | | | | | |
| \$SSCOP | 1# | 1717 | | | | | | | | | | | | | |
| \$SSET | 2242# | 2253 | 2254 | 2255 | 2256 | 2257 | 2258 | 2259 | 2260 | 2261 | 2262 | 2263 | 2264 | 2265 | 2266 |
| | 2267 | | | | | | | | | | | | | | |
| \$SSKIP | 829# | | | | | | | | | | | | | | |
| .EQUAT | 1# | 719 | | | | | | | | | | | | | |
| .HEADE | 1# | 686 | | | | | | | | | | | | | |
| .SETUP | 1# | | | | | | | | | | | | | | |
| .SACT1 | 1# | 870 | | | | | | | | | | | | | |
| .SAPT8 | 1# | 948# | | | | | | | | | | | | | |
| .SAPTH | 1# | 1173 | | | | | | | | | | | | | |
| .SAPTY | 1# | 1872 | | | | | | | | | | | | | |
| .SCATC | 1# | | | | | | | | | | | | | | |
| .SCMTA | 1# | 893 | | | | | | | | | | | | | |
| .SEOP | 1# | 1654 | | | | | | | | | | | | | |
| .SERRO | 1# | | | | | | | | | | | | | | |
| .SERRT | 1# | | | | | | | | | | | | | | |
| .SPOWE | 1# | 2360 | | | | | | | | | | | | | |
| .SRDOC | 1# | 2034 | | | | | | | | | | | | | |
| .SREAD | 1# | 1931 | | | | | | | | | | | | | |
| .SSCOP | 1# | 1721 | | | | | | | | | | | | | |
| .STRAP | 1# | 2219 | | | | | | | | | | | | | |
| .STYPE | 1# | 1793 | | | | | | | | | | | | | |
| .STYPO | 1# | | | | | | | | | | | | | | |

CZKCE MACY11 30A(1052) 08-JUL-80 08:24 PAGE 143
CZKCE.P11 08-JUL-80 08:24 CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0140

ERRORS DETECTED: 0

CZKCE,CZKCE/NL:TOC/SOL/CRF_CZKCE.MAC,CZKCE.P11/EQ:DZDME
RUN-TIME: 30 25 2 SECONDS
RUN-TIME RATIO: 103/58=1.7
CORE USED: 53K (105 PAGES)