

The microfiche card contains 18 frames of data arranged in a 3x6 grid. Each frame contains a table or a block of text. The text is too small to be legible, but the layout suggests a structured document. The frames are separated by thin white lines.

1

IDENTIFICATION

PRODUCT CODE: AC-E517A-MC
PRODUCT NAME: CZKMOA0 KMC-11 ITEP OVRLY
PROGRAM DATE: MAY 1978
MAINTAINER: DIAGNOSTICS-MERRIMACK
AUTHORS: ED BADGER

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) 1978 BY DIGITAL EQUIPMENT CORPORATION

1.0 ABSTRACT.

THIS PROGRAM IS DESIGNED AS A MAINTENANCE AID FOR FIELD SERVICE PERSONEL. IT WILL VERIFY THE PROPER OPERATION OF A COMPLETE COMMUNICATION LINK FROM ONE PDP-11 SYSTEM TO ANOTHER OR TO A COMMUNICATION TEST CENTER.

THIS PROGRAM MUST BE USED IN CONJUNCTION WITH THE INTERPROCESSOR TEST PROGRAM(DZITP) ON A PDP-11 SYSTEM WITH A DL-11 INTERFACE.

:::IMPORTANT:::

PLEASE NOTE: THIS OVERLAY REQUIRES A SYSTEM TO HAVE 8K OF MEMORY. ALSO KCM-11 MAY TALK TO A DMC-11. ALSO THE DMC-11 MAY TALK TO THIS OVERLAY. PARAM #1 MUST BE SET UP TO REFLECT WHICH VERSION OF MICRO-CODES YOU WISH TO RUN. SET PARAM #1 = AN OCTAL 1 FOR HIGH SPEED MICRO-CODE (USE WITH M8202 LINE UNIT). SET PARAM #1 - AN OCTAL 0 FOR LOW SPEED MICRO-CODE (USE WITH M8201 LINE UNIT).

2.0 REQUIREMENTS.

2.1 EQUIPMENT

- A. PDP-11 SYSTEM WITH 8K OF MEMORY.
- B. A KMC11 COMMUNICATION INTERFACE.

2.2 STORAGE.

4K OF CORE

3.0 LOADING PROCEDURE

THIS PROGRAM IS IN ABSOLUTE FORMAT.
THE ABS LOADER MUST BE USED TO LOAD THE PROGRAM.

4.0 OPERATING PROCEDURES.

- A. TWO METHODS OF ENTERING PARAMETERS ARE PROVIDED
 - 1. LOAD ADDRESS 200 AND START TO ENTER PARAMS FROM CONSOLE TTY, PROCEED TO SECTION B.
 - 2. LOAD ADDRESS 200 AND SET SWITCH REGISTER BIT 15 BEFORE STARTING TO ENTER PARAMS FROM CONSOLE SWITCHES, PROCEED TO SECTION C.
- *THE PROGRAM MAY BE RESTARTED AT LOC 204 (ONCE PARAMETERS HAVE ALREADY BEEN SELECTED)
- B. CONSOLE DIALOGUE PARAMETER INPUT (CURRENT VALUES FOR PARAMETERS ARE FOUND IN OVERLAY)
 - 1. THE PROGRAM WILL TYPEOUT THE NAME OF THE VARIABLE OVERLAY.
 - A. IF YOU WISH TO SETUP JUST THE INDICATED OVERLAY, TYPE A CARAGE RETURN
 - B. IF YOU WISH TO SETUP A DN11, TYPE IN DN.
 - C. IF YOU WISH TO SETUP A DM11BB, TYPE IN DMB.

IF DN OR DMB WAS TYPED IN STEP 1 ABOVE THEN THE BUS ADDRESS, VECTOR ETC. REFERED TO IN STEPS 2 THRU 7, PERTAIN TO THE DN11 OR DMBB.

- 2. THE PROGRAM WILL TYPE THE DEFAULT BUS ADDRESS OF THE INTERFACE UNDER TEST.

- A. TYPE A CAR. RETURN TO USE DEFAULT BUS ADDRESS
 - B. TYPEIN ACTUAL BUS ADDRESS
3. THE PROGRAM WILL TYPE OUT THE DEFAULT VECTOR ADDRESS
- A. TYPE A CAR. RETURN TO USE DEFAULT ADDRESS
 - B. TYPEIN ACTUAL VECTOR ADDRESS
4. THE PROGRAM WILL TYPE OUT THE DEFAULT INTERFACE PRIORITY
- NOTE: 200=PRIO 4, 240=PRIO 5, 300=PRIO 6, ETC.
- A. TYPE A CAR. RETURN TO USE DEFAULT VALUE
 - B. TYPEIN ACTUAL VALUE
5. THE PROGRAM WILL TYPEOUT THE DEFAULT VALUE OF PARAM#1
IF REQUIRED BY THE ISR. (SEE SECT. 10.0 IN OVERLAY LISTING FOR PARAMETER DESCRIPTION)
- A. TYPE A CAR. RETURN TO USE DEFAULT VALUE
 - B. TYPEIN ACTUAL VALUE
6. THE PROGRAM WILL TYPEOUT THE DEFAULT VALUE OF PARAM#2
IF REQUIRED BY THE ISR.
- A. TYPE A CAR. RETURN TO USE DEFAULT VALUE
 - B. ENTER ACTUAL VALUE
7. THE PROGRAM WILL TYPEOUT THE DEFAULT VALUE OF PARAM#3
IF REQUIRED BY THE OVERLAY.
- A. TYPE A CAR. RETURN TO USE DEFAULT VALUE
THE DN-11 WILL USE PARAM #3 AS THE # TO DIAL.
IF USING A MODEM WITHOUT AUTOMATIC HANDSHAKING,
THE NUMBER MUST TERMINATE WITH A
'END-OF-NUMBER' CHARACTER (:).
 - B. ENTER ACTUAL VALUE.
8. THE PROGRAM WILL RETURN TO STEP B1 IF THIS SETUP
WAS FOR DN11 OR DM11BB.
9. THE PROGRAM WILL REQUEST THAT SWITCH REGISTER BE SET.
- A. SETUP SWITCH REGISTER AS SPECIFIED IN STEP D.
AND TYPE A CAR. RETURN.

NOTE: IF ANY OF THE ABOVE ITEMS 2 THRU 7 WERE CHANGED BY ENTERING
NEW VALUES, THE NEW VALUE BECOMES THE DEFAULT VALUE FOR SUBSEQUENT
RESTARTS OF THE PROGRAM.

- C. MANUAL PARAMETER INPUT FROM SWITCH REGISTER
1. THE PROGRAM HALTS FOR ISR(INTERFACE SERVICE ROUTINE) SPECIFICATION
SWR14=SETUP DM-11B ISR
SWR13=SETUP DN-11 ISR
SWR=000000=SETUP VARIABLE ISR
 2. THE FOLLOWING HALTS ARE REPEATED FOR EACH ISR SPECIFIED.
SETUP SEQUENCE IS: DN11,DM11-BB THEN VARIABLE OVERLAY. (EACH ENTRY SET SWITCHES THEN HIT CONTINUE.)
 - A. HALT FOR BUS ADDRESS OF INTERFACE
 - B. HALT FOR VECTOR ADDRESS OF INTERFACE
 - C. HALT FOR PRIORITY OF INTERFACE
 - D. HALT FOR INTERFACE PARAM #1 (SEE SECT. 10.0 IN OVERLAY LISTING FOR PARAMETER DESCRIPTION)
 - E. HALT FOR INTERFACE PARAM #2 (DN11 AND DM11B PARAMETERS ARE DISCUSSED IN SECT. 10.0 OF THE MONITOR.
 - F. GO BACK TO STEP A IF THIS SETUP WAS FOR DN OR DM.
 3. HALT FOR OPERATIONAL SWITCH SETTINGS. (SEE STEP D.)
 - A. PRESS CONTINUE TO START TESTING

BEFORE ATTEMPTING TO RUN THIS PROGRAM, THE OPERATOR MUST ACCERTAIN THE COMPLETE COMMUNICATION LOOP AND PROCEEDURES TO BE USED, INCLUDING THE TYPE OF MODEMS, THE TYPE OF INTERFACE BEING USED AT THE OTHER CPU AND THE MODES OF OPERATION, DATA AND PARAMETERS TO BE USED AT EACH CPU.

THIS WILL REQUIRED VOCAL COMMUNICATION WITH THE OPERATOR AT THE OTHER CPU UNLESS ITS CONFIGURATION AND OPERATION ARE FIXED AS A TEST CENTER.

AFTER DETERMINING THAT THE EQUIPMENTS ARE COMPATIBLE AND AGREEING ON THE MODE AND VARIABLE PARAMETERS TO BE USED, THE SYSTEM WHICH IS TO RECEIVE DATA FIRST SHOULD BE LOADED AND STARTED. IF THE MODEM BEING USED ON THIS SYSTEM HAS AN AUTOMATIC ANSWER FEATURE, IT SHOULD BE ENABLED.

THE SYSTEM WHICH IS TO TRANSMIT FIRST SHOULD THEN BE LOADED AND STARTED AND THE CONNECTION ESTABLISHED EITHER MANUALLY OR AUTOMATICALLY (VIA DN-11).

D. OPERATIONAL SWITCH SETTINGS.

SW15=1 HALT ON ERROR
SW14=1 SINGLE PASS
SW14 HAS NO EFFECT IF SW04=0
SW13=1 INHIBIT ERROR TYPEOUTS
SW12=1 INHIBIT ALL TYPEOUTS EXCEPT ERRORS
IF SW12=0 AND SW04=1 END PASS IS TYPED
AND TRANSMITTED/RECEIVED DATA IS TYPED.
SW11=1 USE PREVIOUSLY SPECIFIED DATA
SW10=1 DATA SELECT (WITH SW09)
SW09=1 DATA SELECT (WITH SW10)
00=1 GET DATA FROM OPERATOR
01=1 TEST MESSAGE #1 (\$A QUICK BROWN FOX)
10=1 TEST MESSAGE #2 (\$B NUMERICS)
11=1 TEST MESSAGE #3 (\$C COMTEST/QUICK BROWN FOX/NUMERICS)
SW08=1 TRANSMIT RECEIVED DATA (INTERNAL LOOPBACK MODE)
SW07=1 DO NOT TEST RECEIVED DATA
SW06=1 MONITOR TRANSMITTED DATA ON CONSOLE TTY.*
SW05=1 MONITOR RECEIVED DATA ON CONSOLE TTY.*
* IN MANY CASES, NOT ALL DATA WILL APPEAR ON THE CONSOLE
TTY. THIS IS ESPECIALLY TRUE WHEN THE COMM INTERFACE IS
RUNNING AT A FASTER BAUD THAN THE CONSOLE, BUT EVEN AT EQUAL
OR SLOWER BAUDS, ALL CHARACTERS MAY NOT APPEAR ON THE CONSOLE.

SW04=1 RETURN TO MONITOR FOR END PASS
WHEN SW04=0 PROGRAM LOOPS IN THE OVERLAY NEVER RETURNING TO THE MONITOR.
SW03=1 INTERNAL LOOPBACK MODE
SW02=1 EXTERNAL LOOPBACK MODE
SW01=1 ONE-WAY-IN MODE
SW00=1 ONE-WAY-OUT MODE

THIS PROGRAM HAS BEEN MODIFIED TO RUN ON A PROCESSOR WITH OR WITHOUT A HARDWARE SWITCH REGISTER. WHEN FIRST EXECUTED THE PROGRAM TESTS THE EXISTENCE OF A HARDWARE SWITCH REGISTER. IF NOT FOUND A SOFTWARE SWITCH REGISTER LOCATION (SWREG=LOC. 176) IS DEFAULTED TO. IF THIS IS THE CASE, UPON EXECUTION THE CONTENTS OF THE SWREG ARE DUMPED IN OCTAL ON THE CONSOLE TTY AND ANY CHANGES ARE REQUESTED

(IE) SWR=XXXXXX NEW=

POSSIBLE RESPONSES ARE:

1. <CR> IF NO CHANGES ARE TO BE MADE
2. 6 DIGITS 0-7 TO REPRESENT IN OCTAL THE NEW SWITCH REGISTER VALUE ;LAST DIGIT FOLLOWED BY <CR>.
3. ^U TO ALLOW REENTERING VALUE IF ERROR IS COMMITTED KEYING IN SWREG VALUE.

BUILT INTO THE PROGRAM IS THE ABILITY TO DYNAMICALLY CHANGE THE CONTENTS OF SWREG DURING PROGRAM EXECUTION. BY STRIKING ^G (CNTRL G) ON CONSOLE TTY THE OPERATOR SETS A REQUEST FLAG TO CHANGE THE CONTENTS OF SWREG, WHICH IS PROCESSED IN KEY AREAS OF THE PROGRAM CODE (IE) ERROR ROUTINES, AFTER HALTS END OF PASS, AND OTHER APPLICABLE AREAS.

IF OPERATOR SPECIFIED DATA WAS INDICATED, THE PROGRAM WILL TYPE A REQUEST FOR THE DATA. DATA MAY BE ENTERED AS ASCII CHARACTERS OR OCTAL CODE. TYPE IN THE DATA TERMINATED WITH A CR. OCTAL CODE MAY BE ENTERED BY TYPING AN ^ (UP ARROW) FOLLOWED BY THE OCTAL CODE (IN THE RANGE 000 TO 377) SEPERATED BY SPACES AND TERMINATED BY ^ (UP ARROW).
I.E. ABCD^ 000 123 377^ EFG (CAR.RETURN)

A TYPICAL SWITCH SETTING FOR HALF-DUPLEX=003150 THIS SETTING USES INTERNAL LOOPBACK MODE, LOOPS IN OVERLAY, MONITORS TRANSMITTED AND RECEIVED DATA ON THE CONSOLE TTY, AND TESTS RECEIVED DATA USING TEST MESSAGE #3.

A TYPICAL SWITCH SETTING FOR FULL-DUPLEX=003144 THIS SETTING IS THE SAME AS ABOVE EXCEPT IT USES THE EXTERNAL LOOPBACK MODE.

ALL STANDARD MESSAGES (TEST MESSAGES 1-3) ARE PRECEDED BY 2 FILL CHARACTERS(177), AND ARE FOLLOWED BY A CR(015), LF(012), RECEIVE TERMINATING CHARACTER(001), 4 FILLS(177), AND A TRANSMIT TERMINATING CHARACTER(000). DURING TRANSMISSION, WHEN A 000 CHARACTER IS SEEN THE TRANSMISSION IS STOPPED. DURING RECEPTION, WHEN A 001 CHARACTER IS RECEIVED, THE RECEIVER IS SHUT OFF. IF THE MESSAGE WAS INPUTED BY THE OPERATER, THE TERMINATING CHARACTERS ARE ADDED.

TEST MODES

INTERNAL LOOPBACK MODE

1. THE OVERLAY WAITS TO RECEIVE A MESSAGE (TERMINATED BY <001>)
2. VERIFIES THE DATA AGAINST THE DATA SELECTED BY SW09 AND SW10 (SW7=0)
3. TRANSMIT THE DATA SELECTED BY SW09 AND SW10 (SW8=0) OR TRANSMIT THE RECEIVED DATA (SW8=1)
4. RETURNS TO MONITOR FOR 'END PASS' (SW4=1) OR GO TO STEP 1. (SW4=0)

EXTERNAL LOOPBACK MODE

1. THE OVERLAY SETS REQUEST TO SEND
2. WAIT FOR CLEAR TO SEND
3. TRANSMITS THE SELECTED DATA
4. RESETS REQUEST TO SEND
5. WAIT FOR MESSAGE TO BE RECEIVED
6. VERIFIES THE DATA (SW07=0)
7. RETURNS TO MONITOR FOR 'END PASS'. (SW04=1) OR GO TO STEP 1 (SW04=0)

ONE-WAY-IN MODE

1. THE OVERLAY WAITS FOR MESSAGE TO BE RECEIVED.
2. VERIFIES THE DATA (SW07=0)
3. RETURNS TO MONITOR FOR 'END PASS' (SW04=1) OR GO TO STEP 1 (SW04=0)

ONE-WAY-OUT MODE

1. THE OVERLAY SETS REQUEST TO SEND
2. WAITS FOR CLEAR TO SEND
3. TRANSMITS SELECTED DATA
4. RETURNS TO MONITOR FOR 'END PASS'. (SW04=1) OR GO TO STEP 1 (SW04=0)

- E. THE OVERLAY IS THEN ENTERED AND A CONNECTION ESTABLISHED EITHER MANUALLY OR AUTOMATICALLY.

IF ONE-WAY-IN OR INTERNAL LOOPBACK MODES ARE SELECTED.
THE OVERLAY WILL SET DATA TERMINAL READY AND WAIT FOR DATA.

IF ONE-WAY-OUT OR EXTERNAL LOOPBACK MODES WERE SELECTED.
THE OVERLAY WILL SET DATA TERMINAL READY AND REQUEST TO SEND.
THE OVERLAY WILL THEN WAIT FOR CLEAR TO SEND BEFORE ATTEMPTING TO TRANSMIT DATA.

THE PROGRAM WILL PRINTOUT A 'WAITING FOR CLEAR TO SEND'
MESSAGE AND THE CONTENTS OF THE XMIT CSR EVERY 60 SECS.
UNTIL CLEAR TO SEND IS ASSERTED.

F. IF SW04=0 THE OVERLAY WILL CONTINUE TO TRANSMIT/RECEIVE DATA.

IF SW04=1 THE OVERLAY WILL RETURN TO THE MONITOR AND TYPE 'END PASS'.

IF BOTH SW04=1 AND SW14=1, THE PROGRAM WILL REQUEST NEW INTERFACE PARAMS AFTER ONE PASS OF THE SELECTED TEST MODE.

TEST EXECUTION MAY BE INTERRUPTED BY TYPING THE FOLLOWING CHARACTERS ON THE CONSOLE TTY.

LINE FEED = RESTART PROGRAM AT LOCATION 200.

QUESTION MARK = PRINTOUT FIRST 8 WORDS OF INPUT BUFFER.(ASCII)

THEN TYPE EITHER:

*WXXXXXX TO PRINTOUT THE 8 WORDS
AT LOC XXXXXX.

*BXXXXXX TO PRINTOUT THE 16 BYTES
AFTER LOC XXXXXX.

*C TO CONTINUE

PROGRAM MUST BE RESTARTED AT 200 AFTER PRINTING.
CARRIAGE RETURN = RESTART AT REQUEST FOR NEW OPERATIONAL SWITCHES.

5.0 PROGRAM AND/OR OPERATOR ACTION

IF THE OPERATOR WISHES TO MANUALLY EXAMINE THE TRANSMIT OR RECEIVE BUFFERS, DO THE FOLLOWING; TO FIND THE STARTING ADDRESS OF THE RECEIVE BUFFER, LOAD ADDRESS 11020 AND EXAMINE. TO FIND THE STARTING ADDRESS OF THE TRANSMIT BUFFER, LOAD ADDRESS 11022 AND EXAMINE.

5.1 NORMAL HALTS SEE SECTION 4.

6.0 ERRORS

6.1 ERROR REPORTING

THE ONLY ERROR REPORT FROM THE CONTROL PROGRAM OCCURS IF THE INTERFACE SPECIFIED IS NOT LOADED.

IF DATA IS RECEIVED AND SWITCH 7 (NO DATA COMPARE) IS RESET, THE DATA WILL BE COMPARED AGAINST THE PRESELECTED DATA AFTER A LINE FEED CHARACTER IS RECEIVED. IF THERE IS A MISMATCH, THE FOLLOWING ERROR REPORT IS PRINTED:

RECEIVED DATA=RRRRRR
DATA SHOULD BE TTTTTT
DATA COMPARE ERROR; BAD DATA=BBB GOOD DATA=GGG

WHERE RRRRRR IS THE RECEIVE BUFFER (UP TO 512 CHARACTERS)

TTTTTT IS THE TRANSMIT BUFFER (UP TO 512 CHARACTERS)
BBB IS THE BAD DATA CHARACTER
GGG IS THE GOOD DATA CHARACTER

IF THE INTERFACE DETECTS A DATA ERROR, THE FOLLOWING
WILL BE PRINTED BEFORE THE DATA IS COMPARED:

THERE WAS A RECEIVER ERROR. RECEIVER DATA REGISTER =XXXXXX

WHERE XXXXXX IS THE CONTENTS OF THE RECEIVER DATA REGISTER
THE LOW BYTE IS THE DATA, AND THE HIGH BYTE IS THE ERROR BITS.

IF A RECEIVE TERMINATING CHARACTER<001> IS NOT DETECTED
WITHIN 512 CHARACTERS A 'BUFFER FULL' PRINTOUT WILL OCCUR.

7.0 RESTRICTIONS

THE OPERATION OF THIS PROGRAM REQUIRES COORDINATION BETWEEN
THE OPERATOR AND THE OPERATOR OF ANOTHER PDP-11 SYSTEM
UNLESS ONE OF THE SYSTEMS IS ALWAYS OPERATING IN A FIXED
MODE. THE FOLLOWING TABLE LISTS THE VALID COMBINATIONS:

CPU #1	CPU #2
ONE-WAY-OUT	ONE-WAY-IN
ONE-WAY-IN	ONE-WAY-OUT
EXTERNAL-LOOPBACK	INTERNAL-LOOPBACK
INTERNAL-LOOPBACK	EXTERNAL-LOOPBACK
EXTERNAL-LOOPBACK	EXTERNAL-LOOPBACK (FULL DUPLEX)

WHEN THE COMMUNICATION LINK INVOLVES MODEMS THE FOLLOWING
RESTRICTION APPLY:

IF RUNNING IN FULL DUPLEX MODE BOTH SYSTEMS
MUST BE IN EXTERNAL LOOP BACK MODE.

BOTH SYSTEMS SHOULD BE RUNNING IDENTICAL ROUTINES.

EXAMPLE:
SWITCHES 14,13,7,4 SHOULD BE THE SAME
ON BOTH CPU S

IF PROGRAM IS WAITING IN A SCAN ROUTINE AND TYPES OUT
A 'WAITING MESSAGE',IF AN INCOMING MESSAGE STARTS DURING
THE TYPE OUT, IT WILL BE LOST BECAUSE THE TYPEOUT PRIORITY
IS AT LEVEL 7. THIS WILL RESULT IN OVERRUN OR SILO OVER-
RUN ERRORS, DEPENDING ON THE DEVICE.TO AVOID THIS SITUATION
RUN WITH SWITCH 13 UP. IF OVERRUN DOES OCCURE DURING A
TYPEOUT THE PROGRAM SHOULD BE RESTARTED.

IF USING AN ASYNCRONOUS DEVICE, MODEMS AND THE
MAYNARD TEST STATION AND INITIALIZE DOES NOT CLEAR THE
CONNECTION (EXAMPLE THE DJ11) IF THE PROGRAM IS RESTARTED
IN THE MIDDLE OF A MESSAGE AT LOC 204 OR BY HITTING CR
AN IMMEDIATE ERROR MESSAGE FROM MAYNARD WILL BE RE-
CEIVED. THIS IS BECAUSE THE TEST STATION IS STILL LOOKING

FOR THE REST OF THE INTERRUPTED MESSAGE. TO AVOID THIS ERROR, RESTART PROGRAM ONLY AT THE END OF THE MESSAGE CURRENTLY BEING TRANSMITTED.

8.0 MISCELLANEOUS

ITEP WAS CHECKED OUT USING THE FOLLOWING BELL TELEPHONE MODEMS.
208A (SYNCHRONOUS 4800 BAUD)

9.0 PROGRAM DESCRIPTION

9.1 THE KMC11 INTERFACE SERVICE PARAMS ARE SETUP, AS SPECIFIED BY THE OPERATOR, BY THE ITEP CONTROL PROGRAM.

TIME: PROVIDES A MEANS OF MEASURING ELAPSED TIME. IT IS INCREMENTED EVERY SECOND BY A CLOCK INTERRUPT ROUTINE IN ITEP.

9.2 WHEN THE OVERLAY IS FIRST ENTERED BY ITEP AT LOCATION START:, THE CONTENTS OF THE SWITCH REGISTER ARE STORED IN REGISTER 0. THE MODE AND DATA SELECTIONS ARE FIXED AT THIS TIME AND CANNOT BE ALTERED WITHOUT RETURNING TO THE CONTROL PROGRAM. THE INTERRUPT VECTORS AND VARIABLES ARE THEN SETUP. THE SELECTED ROUTINE DETERMINED BY THE MODE IS THEN ENTERED

NEXT, THE MICRO-CODE TO LOAD INTO THE KMC-11 CRAM. AFTER LOAD, THE CRAM IS VERIFIED AGAINST THE MICRO-CODE IN PDP-11 MEMORY. IF A BAD LOAD, THE PROGRAM WILL REPORT AN ERROR AND CONTINUE. EITHER LOW SPEED MICRO-CODE OR HIGH SPEED MICRO-CODE WILL BE LOADED, DEPENDING ON THE CONTENTS OF PARAM #2.

9.3 THE OVERLAY THEN LOOPS IN ROUTINES: \$OWI, IF 'ONE WAY IN' MODE WAS SELECTED. \$OWO, IF 'ONE WAY OUT' MODE WAS SELECTED. \$ILB, IF 'INTERNAL LOOP BACK' MODE WAS SELECTED. \$XLB, IF 'EXTERNAL LOOP BACK' WAS SELECTED.

9.31 \$OWI: IN THIS ROUTINE THE RECEIVER IS INITIALIZED AND PROGRAM LOOPS WAITING FOR THE RECEIVER TO FINISH. IF NOTHING IS RECEIVED FOR 60 SECS A 'WAITING' MESSAGE IS TYPED. WHEN THE RECEIVER IS DONE, THE PROGRAM CHECKS DATA IF SWITCHES PERMIT, AND TYPES END PASS DEPENDING ON SWITCH SETTINGS.

9.32 \$OWO: THE TRANSMITTER IS INITIALIZED AND PROGRAM LOOPS WAITING FOR TRANSMITTER TO FINISH, A 'WAITING' MESSAGE IS TYPED EVERY 60 SECS IF THERE IS NO ACTION. WHEN THE TRANSMITTER IS DONE, THE PROGRAM EITHER LOOPS BACK TO \$OWO OR TYPES END PASS DEPENDING ON SWITCH SETTINGS.

9.33 \$ILB: THE RECEIVER IS INITIALIZED AND PROGRAM LOOPS WAITING FOR RECEIVER TO FINISH, A 'WAITING' MESSAGE IS TYPED EVERY 60 SEC IF NO ACTION. WHEN RECEIVER IS DONE PROGRAM CHECKS DATA IF SWITCH SETTINGS PERMIT, AND END PASS IS TYPED IF SWITCH SETTINGS PERMIT. THEN THE TRANSMITTER IS INITIALIZED, A 'WAITING' MESSAGE IS TYPED EVERY 60 SEC IF NO ACTION. WHEN TRANSMITTER IS DONE PROGRAM RETURNS TO START OF ROUTINE. (\$ILB)

- 9.34 \$XLB: IF IN HALF DUPLEX THE TRANSMITTER IS INITIALIZED,
A 'WAITING MESSAGE IS TYPED EVERY 60 SEC IF THERE IS NO ACTION
WHEN THE TRANSMITTER IS DONE THE RECEIVER IS INITIALIZED
,A 'WAITING' MESSAGE IS TYPED EVERY 60 SEC IF THERE IS NO ACTION.
WHEN THE RECEIVER IS DONE,DATA IS CHECKED IF SWITCH SETTINGS
PERMIT AND END PASS IS TYPED IF SWITCHES ALLOW.THE PROGRAM NOW
REPEATS CYCLE STARTING AT \$XLB.
IF IN FULL DUPLEX THE RECEIVER AND TRANSMITTER ARE INITIALIZED
, A 'WAITING' MESSAGE IS TYPED EVERY 60 SEC IF THERE IS NO
ACTION. WHEN BOTH THE RECEIVER AND TRANSMITTER ARE DONE,DATA IS
CHECKED, END PASS IS TYPED AND PROGRAM LOOPS TO \$XLB DEPENDING
ON THE SWITCH SETTINGS.
- 9.4 THE RETURN TO MONITOR ROUTINE FOR END PASS AT EOP:
LOCKS OUT INTERRUPTS AND SAVES THE TRANSMITTER INTERRUPT ENABLE
BIT AND ALL GENERAL REGISTERS. IT THEN RETURNS TO THE MONITOR
TO TYPE 'END PASS'. THE MONITOR CHECKS SW14 IF UP IT RETURNS
TO ENTER:, OTHERWISE IT RESTARTS THE PROGRAM.
- 9.5 ENTER: IS ENTERED FROM THE MONITOR AFTER TYPEING 'END PASS',
IT RESTORES THE GENERAL REGISTERS AND THE TRANSMITTER CSR
AS SAVED IN EOP. THE DELAY FLAG IS SET AND PROGRAM RETURNS TO
THE SCAN ROUTINE(OWO,OWI,ILB,XLB) WHERE IT CAME FROM.
- 9.6 THE INITIALIZE TRANSMIT SUBROUTINE AT STARTX:
SETS UP THE INTERFACE AND POINTERS NECESSARY TO
INITIATE A TRANSMIT OPERATION.
AFTER SETTING 'DATA TERMINAL READY' AND 'REQUEST TO SEND' A CHECK
IS MADE ON PARAM2 TO DETERMINE IF HALF DUPLEX OPERATION
WAS SELECTED BY THE OPERATOR. IF IT WAS, THE
SUBROUTINE WAITS FOR CLEAR TO SEND.
A 'WAITING FOR CLEAR TO SEND' PRINTOUT OCCURS
EVERY 30 SECONDS UNTIL CLEAR TO SEND IS ASSERTED.
- 9.7 THE INITIALIZE RECEIVED SUBROUTINE AT STARTR:
SETS UP THE INTERFACE AND POINTERS NECESSARY TO
RECEIVE A MESSAGE.
- 9.8 THE TRANSMIT INTERRUPT SERVICE ROUTINE,
AT XISR:, IS ENTERED VIA TRANSMIT INTERRUPTS
FROM THE INTERFACE.
A TEST IS MADE TO SEE IF THE LAST CHARACTER
TRANSMITTED WAS A NULL (ALL ZEROS) CHARACTER.
IF IT WAS; THE TRANSMIT LOGIC IN THE INTERFACE
IS RESET AND THE TRANSMIT COMPLETE FLAG IS SET.
AT XISR1: THE NEXT CHARACTER IS TRANSMITTED
AND PRINTED ON THE TTY IF THE MONITOR TRANSMIT
SWITCH IS SET.
- 9.9 THE RECEIVE INTERRUPT SERVICE ROUTINE
,AT RISR:, IS ENTERED VIA RECEIVER INTERRUPTS
FROM THE INTERFACE.
THE RECEIVED CHARACTER IS STORED IN
THE INPUT BUFFER AND PRINTED ON THE TTY IF
THE MONITOR RECEIVER SWITCH IS SET.

IF THE INPUT BUFFER IS FULL, A 'BUFFER FULL' PRINTOUT WILL OCCUR. THIS INDICATES THAT A LINE FEED CHARACTER WAS NOT RECOGNIZED IN THE RECEIVED DATA (WITHIN 1000 CHARACTERS). IF THE RECEIVED CHARACTER IS A LINE FEED, THE RECEIVED LOGIC IS RESET AND THE RECEIVE COMPLETE FLAG IS SET. IF A 'RECEIVE ERROR' IS DETECTED AT RISR:, THE CSR AND DBR WILL BE SAVED AND PRINTED OUT AFTER THE COMPLETE MESSAGE HAS BEEN RECEIVED.

- 9.10 THE DATA TEST SUBROUTINE AT TESTD: IS ENTERED AFTER A COMPLETE MESSAGE HAS BEEN RECEIVED. IF A 'RECEIVE ERROR' HAD BEEN DETECTED, THE CONTENTS OF THE 'RECEIVE BUFFER' AT THE TIME THE ERROR OCCURRED WILL BE PRINTED. THE DATA IS COMPARED UNTIL A 'ALL ZEROS' CHARACTER IS RECOGNIZED. 'FILL' (ALL ONES) CHARACTERS ARE IGNORED. IF A MISMATCH IS DETECTED, THE COMPLETE CONTENTS OF THE INPUT BUFFER AND GOOD DATA IS PRINTED.

10.0 PARAMETERS FOR THE KMC11

DZDMOA PROVIDES THREE TESTS FOR THE KMC-11, SELECTABLE BY THE PARAMETER LOCATIONS PROVIDED IN ITEP. THE THREE TESTS ARE:

10.1 1.) LINK TEST

NORMAL ITEP OPERATION, THE ONLY RESTRICTION IS IT MUST BE KMC TO KMC OR DMC. IT IS NORMAL TO GET SOFT ERRORS DURING THE LINK TEST. THE PARAMETERS FOR THE LINK TEST ARE AS FOLLOWS:

PARAM#1

BIT1 = 0	LOW SPEED MICRO-CODE (M8201)
BIT1 = 1	HIGH SPEED MICRO-CODE (M8202)
BIT15 = 0	KMC-11 (DEFAULT)
BIT15 = 1	DO NOT VERIFY MICRO-CODE LOAD

PARAM#2 FULL/HALF DUPLEX SELECTION

BIT0 = 0	HALF DUPLEX
= 1	FULL DUPLEX (DEFAULT)

PARAM#3 IS NOT USED (177777)

10.2 2.) SECONDARY MODE TEST

THIS TEST CHECKS THE SECONDARY STATION DELAY.

IF RUNNING THIS TEST, EXTERNAL LOOP BACK IS THE ONLY LEGAL MODE OF OPERATION. ALSO BOTH KMC-11'S MUST HAVE HALF-DUPLEX SELECTED AND THE SECONDARY MODE BIT SET IN THE PARAMETER WORD. ADDITIONALLY ONE KMC IS SET TO BE THE SECONDARY STATION AND THE OTHER THE PRIMARY STATION. AGAIN IT IS NORMAL TO GET SOFT ERRORS DURING THE SECONDARY MODE TEST AS IN THE LINK TEST. THE PARAMETERS FOR THE SECONDARY MODE TEST ARE AS FOLLOWS:

PARAM#1

BIT1 = 0 LOW SPEED MICRO-CODE (M8201)
BIT1 = 1 HIGH SPEED MICRO-CODE (M8202)
BIT15 = 0 KMC-11 (DEFAULT)
BIT15 = 1 DO NOT VERIFY MICRO-CODE LOAD

PARAM#2 SECONDARY MODE TEST SELECTION

BIT0 = 0 HALF DUPLEX(MUST BE 0 FOR THIS TEST)
BIT1 = 1 SECONDARY MODE TEST(MUST BE 1)
BIT2 = 0 PRIMARY STATION
 = 1 SECONDARY STATION

PARAM#3 IS NOT USED (177777)

11.0 KMC11 RESTRICTIONS

- 11.1 THE KMC11 IS A DMA DEVICE AND THEREFORE THE TRANSMITTED AND RECEIVED DATA CAN NOT BE MONITORED ON A PER CHARACTER BASIS BY THE CONSOLE TTY. BECAUSE OF THIS, SW05 AND SW06 HAVE NO EFFECT.
- 11.2 KMC ITEP IS MEANT TO BE AN ON LINE LINK TEST FOR TWO KMC11S. KMC ITEP WILL NOT WORK WITH ANY OTHER DEVICE RUNNING ITEP EXCEPT ANOTHER KMC11 OR SLAVE DMC11.
- 11.3 BECAUSE THE KMC11 SUPPORTS DDCMP OPERATION IN THE FIRMWARE, THE PDP-11 PROGRAM (ITEP) IS UNABLE TO CONTROL OR KNOW EXACTLY WHAT IS BEING TRANSMITTED AT ANY GIVEN TIME. ALL DATA MESSAGES ARE ENCLOSED IN A DDCMP ENVELOPE AND THERE MAY ALSO BE CONTROL MESSAGES (AKS NAKS ETC) BEING TRANSMITTED. BECAUSE OF THIS PLEASE BEWARE IF YOU ARE SCOPING DATA.

647
648
649
650
651 011000 011000
652 011000 046513 000103
653 011004 160010
654 011006 000300
655 011010 000240
656 011012 000000
657 011014 000001
658 011016 177777
659 011020 000000
660 011022 000000
661 011024 000000
662 011026 000000
663 011030 000000
664 011032 000000
665 011034 000000
666 011036 011106
667 011040
668 011040 000
669 011041
670 011041 001
671 011042 000000
672 011044 177570
673 011046 177570
674
675
676
677
678 000000
679 100000
680 040000
681 020000
682 020000
683
684 011050 000000
685 011052 000000
686 011054 000000
687 011056 000000
688 011060 000000
689
690 011062 000000
691 011064 000000
692 011066 000000
693 011070 000000
694 011072 000000
695 011074 000000
696
697 011076 177560
698 011100 177562
699 011102 177564
700 011104 177566
701
702 000001

: KMC11 INTERFACE SERVICE PARAMS

KMC11: .=11000
BA: .ASCIZ /KMC/ :ISR NAME
RIV: 300 :BUS ADDRESS
PRIOR: 240 :VECTOR ADDRESS
PARAM1: 0 :PRIORITY
PARAM2: 1 :PARAM #1
PARAM3: -1 :PARAM #2
IRDA: .WORD 0 :PARAM #3
IXDA: .WORD 0 :INITIAL READ DATA ADDRESS
SETTLE: .WORD 0 :INITIAL XMIT DATA ADDRESS
B2016: .WORD 0 :LINE SETTLE DELAY FLAG
TIME: .WORD 0 :
TX.TERM: .WORD START :ADDR OF BIN TO OCT TYPE ROUTINE
RX.TERM: .BYTE 000 :TIMER
FLAG: .BYTE 001 :RECEIVER TERMINATING CHAR.
SWR: 177570
DISPLAY:177570

: CONSTANTS + WORKING STORAGE

STAT=R0
XFLG=100000 :XMIT COMPLETE FLAG
RFLG=40000 :RCV COMPLETE FLAG
DSFLG=20000 :DATA SET STATUS CHANGE FLAG
BIT13=20000 :INHIBIT PRINTOUTS
SXCSR: 0 :SAVED XMIT CSR
SRCSR: 0 :SAVED RCV CSR
ERCSR: 0 :RCV CSR SAVED ON ERROR
ERDBR: 0 :RCV DATA REG SAVED ON ERROR
DSSTAT: 0 :RCV CSR SAVED ON DS CHANGE
TXWC: 0
RXWC: 0
XCC: 0 :XMIT CHAR COUNT
RCC: 0 :RCV CHAR COUNT
RDA: 0 :RCV DATA ADDR.
XDA: 0 :XMIT DATA ADDR.
TKS: 177560
TKB: 177562
TPS: 177564
TPB: 177566
FULL.DUPLEX=000001

703
704
705
706 011106 000240
707 011110 017700 177730
708 011114 042700 177400
709 011120 013702 011006
710 011124 012722 014036
711 011130 013722 011010
712 011134 012722 013550
713 011140 013722 011010
714 011144 013704 011004
715 011150 013702 011006
716 011154 012712 013550
717 011160 012762 014036 000004
718 011166 032737 000400 011012
719 011174 001405
720 011176 012712 015132
721 011202 012762 015316 000004
722 011210 012714 040000 3\$:
723 011214 005014
724 011216 010401
725 011220 004737 017446
726 011224 012714 100000
727 011230 013702 011022
728 011234 005003
729 011236 005203 1\$:
730 011240 123722 011040
731 011244 001374
732 011246 010337 011062
733 011252 062703 000010
734 011256 010337 011064
735 011262 005714 2\$:
736 011264 100376
737 011266 005037 017014
738 011272 005037 017024
739 011276 005037 017022
740 011302 005714 4\$:
741 011304 100376
742 011306 052764 000100 000002
743 011314 052714 000143
744 011320 005037 011054
745 011324 005037 011056
746 011330 005037 017036
747 011334 005037 017040
748 011340 005037 017042
749 011344 005037 017044
750 011350 105037 017051
751 011354 005037 017052
752 011360 005037 017054
753 011364 005037 017056
754 011370 105037 017060
755 011374 005037 017020
756 011400 005037 017012
757 011404
758

```
*****  
: KMC11-X INTERFACE SERVICE ROUTINE  
*****  
START: NOP  
MOV @SWR, R0 ;SETUP MODE IN R0  
E:IC #177400, R0 ;STRIP JUNK  
MOV RIV, R2 ;SETUP  
MOV #RISR, (R2)+ ;INTERRUPT  
MOV PRIOR, (R2)+ ;VECTORS  
MOV #XISR, (R2)+  
MOV PRIOR, (R2)+  
MOV BA, R4 ;SETUP BUS ADDR INDEX  
MOV RIV, R2 ;ADJUST VECTORS FOR  
MOV #XISR, (R2) ;INPUT SERVICE ROUTINE  
MOV #RISR, 4(R2) ;OUTPUT SERVICE ROUTINE  
BIT #BIT8, PARAP11 ;BOOT MODE?  
BEQ 3$ ;BR IF NO  
MOV #IISR, (R2) ;LOAD BOOT VECTORS  
MOV #OISR, 4(R2)  
3$: MOV #BIT14, (R4) ;MASTER CLEAR KMC11  
CLR (R4)  
MOV R4, R1  
JSR PC, LDRVRF  
MOV #BIT15, (R4)  
MOV IXDA, R2 ;CALCULATE WORD COUNTS  
CLR R3 ;CLEAR COUNT  
1$: INC R3 ;INC COUNT  
CMPB TX.TERM, (R2)+ ;LAST CHARACTER?  
BNE 1$ ;BR IF NO  
MOV R3, TXWC ;STORE XMIT COUNT  
ADD #10, R3 ;ADD 10 TO IT  
MOV R3, RXWC ;STORE REC COUNT  
2$: TST (R4) ;WAIT FOR RUN  
BPL 2$ ;BR IF RUN NOT SET  
CLR BASEFG ;CLEAR BASE LOAD FLAG  
CLR RFLAG ;CLEAR RECEIVE FLAG  
CLR XFLAG ;CLEAR XMIT FLAG  
4$: TST (R4) ;RUN SET?  
BPL 4$ ;BR IF NO  
BIS #100, 2(R4) ;SET OIE  
BIS #143, (R4) ;ASK FOR BASE TRANSFER  
CLR ERCSR ;CLEAR ERROR LOCATIONS  
CLR ERDBR  
CLR ERRCNT ;CLEAR SOFT ERROR STORAGE  
CLR ERRCNT+2 ;LOCATIONS  
CLR ERRCNT+4  
CLR ERRCNT+6  
CLRB BASE+3 ;CLEAR BASE ERROR COUNT LOCATIONS  
CLR BASE+4  
CLR BASE+6  
CLR BASE+10  
CLRB BASE+12  
CLR TEMP2  
5$: CLR RESUME ;CLEAR RESUME FLAG
```

```

759 011404 017700 177434      MOV    @SWR,R0      ;REREAD SWR
760
761
762
763
764
765
766
767 011410 005037 011032      GO:    CLR    TIME
768 011414 005037 013402      CLR    DELAY
769 011420 005037 013406      CLR    STOP
770 011424 032700 000001      BIT    #OWO,MODE
771 011430 001402              BEQ    1$
772 011432 000137 011612      JMP    $OWO
773 011436 032700 000002      1$:   BIT    #OWI,MODE
774 011442 001402              BEQ    2$
775 011444 000137 011500      JMP    $OWI
776 011450 032700 000010      2$:   BIT    #ILB,MODE
777 011454 001402              BEQ    3$
778 011456 000137 011714      JMP    $ILB
779 011462 032700 000004      3$:   BIT    #XLB,MODE
780 011466 001402              BEQ    4$
781 011470 000137 012150      JMP    $XLB
782 011474 000000              4$:   HALT
783 011476 000776              BR     .-2
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798 011500 104416              $OWI:  KBDIN
799 011502 004737 013500      JSR    PC,STARTR
800 011506 032700 040000      1$:   BIT    #RFLG,STAT
801 011512 001013              BNE    2$
802 011514 023727 011032 000100  CMP    TIME,#100
803 011522 103771              BLO    1$
804 011524 011402              MOV    @RCSR,R2
805 011526 016403 000002      MOV    XCSR(R4),R3
806 011532 104001              HLT    1
807 011534 005037 011032      CLR    TIME
808 011540 000762              BR     1$
809
810 011542 032777 000200 177274 2$:   BIT    #NODAT,@SWR
811 011550 001002              BNE    3$
812 011552 004737 012626      JSR    PC,TESTD
813 011556 042700 040000      3$:   BIC    #RFLG,STAT
814 011562 004737 016242      JSR    PC,CKBASE      ;CHECK KMC SOFT ERROR COUNTERS
  
```

```

:*****
:  ROUTINE USED TO GOTO
:  SUBROUTINE DEPENDENT
:  ON MODE SELECTED.
:*****
  
```

```

:*****
:  ROUTINE USED IF 'ONE WAY IN' MODE WAS SELECTED.
:  NOTE THAT WHEN IN THIS MODE HALF DUPLEX IS THE
:  ONLY MODE AVAILABLE.
:  'ONE WAY IN' MEANS THAT ONLY THE RECEIVER IS
:  ENABLED. THE TRANSMITTER IS NEVER 'TURNED ON'.
:*****
  
```

```

815 011566 032777 000020 177250      BIT    #LOOP,@SWR
816 011574 001405                      BEQ    4$
817 011576 012737 011610 013404      MOV    #4$,BACK
818 011604 000137 012466                      JMP    EOP
819 011610 000733                      4$:   BR    $OWI
820
821
822
823
824
825
826
827
828
829
830 011612 104416                      $OWO: KBDIN
831 011614 004737 013410                      JSR    PC,STARTX
832 011620 005037 011032                      CLR    TIME
833 011624 032700 100000                      1$:   BIT    #XFLG,STAT
834 011630 001013                      BNE    2$
835 011632 023727 011032 000100      CMP    TIME,#100
836 011640 103771                      BLO    1$
837 011642 011402                      MOV    @RCSR,R2
838 011644 016403 000002                      MOV    XCSR(R4),R3
839 011650 104001                      HLT    1
840 011652 005037 011032                      CLR    TIME
841 011656 000762                      BR     1$
842 011660 042700 100000                      2$:   BIC    #XFLG,STAT
843 011664 004737 016242                      JSR    PC,CKBASE
844 011670 032777 000020 177146      BIT    #LOOP,@SWR
845 011676 001405                      BEQ    3$
846 011700 012737 011712 013404      MOV    #3$,BACK
847 011706 000137 012466                      JMP    EOP
848 011712 000737                      3$:   BR    $OWO
849
850
851
  
```

```

:*****
:   ROUTINE USED IF 'ONE WAY OUT' WAS SELECTED.
:   NOTE THAT WHEN IN THIS MODE HALF DUPLEX IS THE ONLY
:   MODE AVAILABLE.
:   'ONE WAY OUT' MEANS THAT ONLY THE TRANSMITTER IS
:   ENABLED. THE RECEIVER IS NEVER 'TURNED ON.'
:*****
  
```

;CHECK KMC SOFT ERROR COUNTERS

852
853
854
855
856
857
858
859
860
861
862
863 011714 104416
864 011716 004737 013500
865 011722 005037 011032
866 011726 032700 040000
867 011732 001013
868 011734 023727 011032 000100
869 011742 103771
870 011744 011402
871 011746 016403 000002
872 011752 104001
873 011754 005037 011032
874 011760 000762
875 011762 032777 000200 177054
876 011770 001002
877 011772 004737 012626
878 011776 042700 040000
879 012002 004737 016242
880 012006 032777 000020 177030
881 012014 001405
882 012016 012737 012030 013404
883 012024 000137 012466
884 012030 032777 000400 177006
885 012036 001416
886 012040 013702 011020
887 012044 013703 011022
888 012050 010337 011074
889 012054 112223
890 012056 001376
891 012060 112743 000177
892 012064 005203
893 012066 112723 000177
894 012072 105023
895 012074 005037 011032
896 012100 004737 013410
897 012104 032700 100000
898 012110 001013
899 012112 023727 011032 000100
900 012120 103771
901 012122 011402
902 012124 016403 000002
903 012130 104001
904 012132 005037 011032
905 012136 000762
906 012140 042700 100000
907 012144 000137 011714

```
*****
: ROUTINE USED IF INTERNAL LOOP BACK'' WAS SELECTED.
: NOTE THAT WHEN IN THIS MODE; HALF DUPLEX IS THE
: ONLY MODE AVAILABLE.
: ''INTERNAL LOOP BACK'' MEANS THAT THE RECEIVER IS ''TURNED ON''
: AND A COMLPETE MESSAGE IS RECEIVED. IF DATA IS TO BE CHECKED
: IT IS; IF ''END PASS'' IS DESIRED; IT IS GIVEN.
: THEN THE TRANSMITTER IS ENABLED. AFTER THE WHOLE MESSAGE
: IS TRANSMITTED; THE CYCLE IS REPETED AS ABOVE.
*****
$ILB: KBDIN
      JSR PC,STARTR
      CLR TIME
1$:   BIT #RFLG,STAT
      BNE 2$
      CMP TIME,#100
      BLO 1$
      MOV @RCSR,R2
      MOV XCSR(R4),R3
      HLT 1
      CLR TIME
874:  BR 1$
875:  BIT #NODAT,@SWR
      BNE 3$
      JSR PC,TESTD
878:  BIC #RFLG,STAT
      JSR PC,CKBASE ;CHECK KMC SOFT ERROR COUNTERS
      BIT #LOOP,@SWR
      BEQ 4$
      MOV #4$,BACK
884:  JMP EOP
885:  BIT #400, @SWR ;USE EXTERNAL DATA?
      BEQ 7$ ;BR IF NO
      MOV IRDA, R2 ;SET POINTER
      MOV IXDA, R3 ;SET POINTER
      MOV R3, XDA ;SETUP XMIT DATA ADDR
      MOVB (R2)+, (R3)+ ;MOVE INPUT TO OUTPUT
      BNE -2 ;LOOP IF NOT ZERO CHAR
      MOVB #177, -(R3) ;INSERT A FILL CHAR
      INC R3 ;BUMP ADDRESS
      MOVB #177, (R3)+ ;INSERT ANOTHER FILL
      CLRB (R3)+ ;INSERT ZERO CHAR
7$:   CLR TIME
896:  JSR PC,STARTX
897:  BIT #XFLG,STAT
      BNE 6$
      CMP TIME,#100
      BLO 5$
      MOV @RCSR,R2
      MOV XCSR(R4),R3
      HLT 1
      CLR TIME
905:  BR 5$
906:  BIC #XFLG,STAT
907:  JMP $ILB
```

908
909
910
911
912
913
914
915
916
917
918
919
920
921 012150 104416
922 012152 032737 000002 011014
923 012160 001004
924 012162 032737 000001 011014
925 012170 001417
926 012172 004737 013500
927 012176 032737 000004 011014
928 012204 001411
929 012206 005737 017020
930 012212 001403
931 012214 032700 040000
932 012220 001774
933 012222 012737 177777 017020
934 012230 004737 013410
935 012234 005037 011032
936 012240 032700 100000
937 012244 001016
938 012246 032700 040000
939 012252 001030
940 012254 023727 011032 000100
941 012262 103766
942 012264 011402
943 012266 016403 000002
944 012272 104001
945 012274 005037 011032
946 012300 000757
947 012302 032737 000002 011014
948 012310 001356
949 012312 032737 000001 011014
950 012320 001352
951 012322 042700 100000
952 012326 004737 013500
953 012332 000742
954 012334 032737 000002 011014
955 012342 001004
956 012344 032737 000001 011014
957 012352 001420
958 012354 032700 100000
959 012360 001013
960 012362 023727 011032 000100
961 012370 103761
962 012372 011402
963 012374 016403 000002

```
*****  
: ROUTINE USED IF 'EXTERNAL LOOP BACK' WAS SELECTED.  
: EITHER HALF OR FULL DUPLEX MAY BE SELECTED IN THIS MODE.  
: 'EXTERNAL LOOP BACK' MEANS THAT THE TRANSMITTER IS FIRST  
: TURNED ON (IF HALF DUPLEX) AND THE WHOLE MESSAGE IS TRANSMITTED;  
: THEN THE RECEIVER IS ENABLED. AFTER THE WHOLE MESSAGE IS RECEIVED  
: DATA WILL THEN BE CHECKED IF DESIRED AND END PASS WILL  
: BE GIVEN IF DESIRED. THEN THE CYCLE IS REPEATED  
: AS ABOVE. IF RUNNING IN FULL DUPLEX THE PROGRAM  
: WAITS FOR BOTH THE RECEIVER AND TRANSMITTER TO  
: FINISH THEN RESTARTS THE RECEIVER AND TRANSMITTER.  
*****
```

```
$XLB: KBDIN  
BIT #BIT1,PARAM2 ;SECONDARY MODE?  
BNE 9$ ;BR IF YES  
BIT #FULL.DUPLEX,PARAM2  
BEQ 1$  
9$: JSR PC,STARTR  
BIT #BIT2,PARAM2 ;SECONDARY STATION?  
BEQ 1$ ;BR IF NO  
TST TEMP2 ;FIRST TIME HERE?  
BEQ 11$ ;BR IF YES  
BIT #RFLG,STAT ;WAIT FOR RECEIVE BEFORE  
BEQ -6$ ;TRANSMITTING  
11$: MOV #-1,TEMP2 ;SET FIRST TIME FLAG  
1$: JSR PC,STARTX  
CLR TIME  
2$: BIT #XFLG,STAT  
BNE 3$  
7$: BIT #RFLG,STAT  
BNE 4$  
CMP TIME,#100  
BLO 2$  
MOV @RCSR,R2  
MOV XCSR(R4),R3  
HLT 1  
CLR TIME  
BR 2$  
3$: BIT #BIT1,PARAM2 ;SECONDARY MODE?  
BNE 7$ ;BR IF YES  
BIT #FULL.DUPLEX,PARAM2  
BNE 7$  
BIC #XFLG,STAT  
JSR PC,STARTR  
BR 2$  
4$: BIT #BIT1,PARAM2 ;SECONDARY MODE?  
BNE 10$ ;BR IF YES  
BIT #FULL.DUPLEX,PARAM2  
BEQ 8$  
10$: BIT #XFLG,STAT  
BNE 6$  
CMP TIME,#100  
BLO 4$  
MOV @RCSR,R2  
MOV XCSR(R4),R3
```

964	012400	104001			HLT	1	
965	012402	005037	011032		CLR	TIME	
966	012406	000752			BR	4\$	
967	012410	042700	100000	6\$:	BIC	#XFLG,STAT	
968	012414	042700	040000	8\$:	BIC	#RFLG,STAT	
969	012420	005037	011032		CLR	TIME	
970	012424	032777	000200	176412	BIT	#NODAT,@SWR	
971	012432	001002			BNE	5\$	
972	012434	004737	012626		JSR	PC,TESTD	
973	012440	004737	016242	5\$:	JSR	PC,CKBASE	:CHECK KMC SOFT ERROR COUNTERS
974	012444	032777	000020	176372	BIT	#LOOP,@SWR	
975	012452	001636			BEQ	\$XLB	
976	012454	012737	012150	013404	MOV	#\$XLB,BACK	
977	012462	000137	012466		JMP	EOP	

978
979
980
981
982
983
984 012466
985 012466 104414 000340
986 012472 016437 000002 012624
987 012500 042737 177677 012624
988 012506 042764 000100 000002
989 012514 012766 012554 000002
990 012522 010037 013366
991 012526 010137 013370
992 012532 010237 013372
993 012536 010337 013374
994 012542 010437 013376
995 012546 010537 013400
996 012552 000207
997
998 012554
999 012554 013700 013366
1000 012560 013701 013370
1001 012564 013702 013372
1002 012570 013703 013374
1003 012574 013704 013376
1004 012600 013705 013400
1005 012604 012737 177777 013402
1006 012612 053764 012624 000002
1007 012620 000177 000560
1008 012624 000000
1009
1010
1011
1012
1013
1014
1015
1016 012626 013746 011056
1017 012632 001413
1018 012634 032777 020000 176202
1019 012642 001007
1020 012644 104400 013026
1021 012650 004077 176154
1022 012654 005746
1023 012656 104400 013107
1024 012662 013701 011022
1025 012666 013702 011020
1026 012672 122122
1027 012674 001776
1028 012676 123741 011040
1029 012702 001447
1030 012704 122742 000002
1031 012710 001005
1032 012712 010237 012720
1033 012716 104400

: ROUTINE TO RETURN
: TO MONITOR FOR
: END PASS.

EOP: STPS,PRTY7 ;SET PS PRIORITY TO 7
MOV XCSR(R4),QTPIE ;SAVE TX CSR
BIC #^C<TIE>,QTPIE ;CLEAR ALL BUT TX IE.
BIC #TIE,XCSR(R4) ;CLEAR TX IE (EVEN IF IT WASN'T SET)
MOV #ENTER,2(SP) ;SET FOR RETURN IF SW 14=1
MOV R0,SAVR0 ;SAVE REGISTER 0
MOV R1,SAVR1 ;SAVE REGISTER 1
MOV R2,SAVR2 ;SAVE REGISTER 2
MOV R3,SAVR3 ;SAVE REGISTER 3
MOV R4,SAVR4 ;SAVE REGISTER 4
MOV R5,SAVR5 ;SAVE REGISTER 5
RTS PC ;RETURN TO CONTROL PROGRAM

ENTER: MOV SAVR0,R0 ;RESTORE R0
MOV SAVR1,R1 ;RESTORE R1
MOV SAVR2,R2 ;RESTORE R2
MOV SAVR3,R3 ;RESTORE R3
MOV SAVR4,R4 ;RESTORE R4
MOV SAVR5,R5 ;RESTORE R5
MOV #-1,DELAY ;IF ORGINALLY SET; SET TX IE
BIS QTPIE,XCSR(R4)
JMP @BACK
QTPIE: 000000

: SUBROUTINE TO CHECK
: RECEIVER DATA.

TESTD: MOV ERDBR, -(SP) ;WAS THERE A RECEIVE ERROR?
BEQ TSTDAT ;BR IF NO
BIT #BIT13,@SWR ;INHIBIT PRINTOUTS?
BNE TSTDAT ;BR IF YES
TYPE ,MSG0 ;<15><12>THERE WAS A RECEIVE ERROR. RBUF=
JSR R0,@B2016 ;PRINT CONTENTS OF RBUF
TST -(SP)
TYPE ,MSG1 ;<15><12>
TSTDAT: MOV IXDA, R1 ;SETUP XMIT DATA ADDR
MOV IRDA, R2 ;SETUP RCV DATA ADDR
SCAN4: CMPB (R1)+, (R2)+ ;DATA OK ?
BEQ SCAN4 ;BR IF OK
CMPB TX.TERM,-(R1) ;IS IT END OF DATA
BEQ TESTDX ;BR IF YES
CMPB #002,-(R2)
BNE 2\$
MOV R2,1\$
TYPE

```
1034 012720 000000 1$: .WORD 0
1035 012722 000437 BR TESTDX
1036 012724 2$:
1037 012724 105712 TSTB (R2) ;
1038 012726 001435 BEQ TESTDX ;BR IF YES
1039 012730 122721 000177 CMPB #177, (R1)+ ;IS IT FILL CHAR?
1040 012734 001756 BEQ SCAN4 ;BR IF YES
1041 012736 005301 DEC R1 ;BACKUP
1042 012740 122722 000177 CMPB #177, (R2)+ ;IS IT FILL?
1043 012744 001752 BEQ SCAN4 ;BR IF YES
1044 012746 000240 SCANS: NOP ;DATA ERROR
1045 012750 032777 020000 176066 BIT #BIT13,@SWR ;INHIBIT PRINTOUTS
1046 012756 001016 BNE DERR ;BR IF YES
1047 012760 104400 013112 TYPE ,MSG2 ;<15><12>RECEIVED DATA = <15><12>
1048 012764 013737 011020 012774 MOV IRDA, RDAX ;SETUP DATA ADRESS
1049 012772 104400 TYPE ;PRINT RECEIVED DATA
1050 012774 000000 RDAX: 0 ;RECEIVED DATA ADDR.
1051 012776 104400 013137 TYPE ,MSG3 ;<15><12>DATA SHOULD BE<15><12>
1052 013002 013737 011022 013012 MOV IXDA, .+10 ;SETUP ADDR.
1053 013010 104400 TYPE ;PRINT GOOD DATA
1054 013012 011022 IXDA
1055 013014 111103 DERR: MOVB (R1),R3 ;SETUP XMIT DATA
1056 013016 114202 MOVB -(R2),R2 ;SETUP RCV DATA
1057 013020 104007 HLT+7 ;DATA ERROR HALT
1058 013022 005726 TESTDX: TST (SP)+ ;POP STACK
1059 013024 000207 RTS PC ;RETURN FROM SUB/ROUT
1060
1061 013026 005015 044124 051105 MSG0: .ASCIZ <15><12>/THERE WAS A RECEIVER ERROR. REGISTER (SEL 2) =/
(1) 013107 015 000012 MSG1: .ASCIZ <15><12>
(1) 013112 005015 042522 042503 MSG2: .ASCIZ <15><12>/RECEIVED DATA = /<15><12>
(1) 013137 015 042012 052101 MSG3: .ASCIZ <15><12>/DATA SHOULD BE/<15><12>
(1) 013162 005015 046120 040505 MSG4: .ASCII <15><12>/PLEASE MAKE CONNECTION (DIAL NUMBER)./
(1) 013231 015 053412 042510 .ASCIZ <15><12>/WHEN CONNECTION COMPLETE; HIT CONTINUE SWITCH./<15><12>
(1) 013314 005015 046120 040505 MSG5: .ASCIZ <15><12>/PLEASE MAKE CONNECTION (DIAL NUMBER)./<15><12>
(1) .EVEN
(1) 013366 000000 SAVR0: 0
1062 013370 000000 SAVR1: 0
1063 013372 000000 SAVR2: 0
1064 013374 000000 SAVR3: 0
1065 013376 000000 SAVR4: 0
1066 013400 000000 SAVR5: 0
1067 013402 000000 DELAY: 0
1068 013404 000000 BACK: 0
1069 013406 000000 STOP: 0
1070
```

```
1071 :*****
1072 : TRANSMIT INIT ROUTINE
1073 :*****
1074
1075 013410 005737 013406 STARTX: TST STOP ;FIRST TIME HERE?
1076 013414 001005 BNE 1$ ;BR IF NOT
1077 013416 104400 013162 TYPE ,MSG4 ;TYPE CONNECT MESS
1078 013422 000000 HALT
1079 013424 005137 013406 COM STOP ;SET FIRST TIME FLAG
1080 013430 005737 017014 1$: TST BASEFG ;BASE AND CNTL IN ALL DONE?
1081 013434 001775 BEQ 1$ ;BR IF NO
1082 013436 005037 017014 CLR BASEFG ;CLEAR FLAG
1083 013442 005037 017016 CLR TEMP1 ;GET SET TO DELAY
1084 013446 012737 000025 017020 MOV #25,TEMP2 ;
1085 013454 062737 000001 017016 ADD #1,TEMP1 ;INC DELAY
1086 013462 001374 BNE -6 ;
1087 013464 005337 017020 DEC TEMP2 ;DEC DELAY COUNTER
1088 013470 001371 BNE -14 ;BR IF NOT DONE
1089 013472 152714 000140 BISB #140,(R4) ;ASK FOR XMIT BUFFER
1090 013476 000207 RTS PC ;RETURN
1091
1092 :*****
1093 : RECEIVE INIT ROUTINE
1094 :*****
1095
1096 013500 005737 013406 STARTR: TST STOP ;FIRST TIME HERE?
1097 013504 001004 BNE 1$ ;BR IF NOT
1098 013506 104400 013314 TYPE ,MSG5 ;TYPE CONNECT MESS
1099 013512 005137 013406 COM STOP ;SET FIRST TIME FLAG
1100 013516 005737 017024 1$: TST RFLAG ;HAS AN REC BUFFER ALREADY BEEN GIVEN
1101 013522 001007 BNE 2$ ;BR IF YES
1102 013524 005737 017014 TST BASEFG ;BASE AND CNTL IN ALL DONE?
1103 013530 001772 BEQ 1$ ;BR IF NO
1104 013532 005037 017014 CLR BASEFG ;CLEAR FLAG
1105 013536 152714 000144 BISB #144,(R4) ;ASK FOR REC BUFFER
1106 013542 005037 017024 2$: CLR RFLAG ;CLEAR FLAG
1107 013546 000207 RTS PC ;RETURN
1108
1109 :*****
1110 : INPUT SERVICE ROUTINE
1111 :*****
1112
1113 013550 032714 000002 XISR: BIT #BIT1,(R4) ;BASE REQUEST?
1114 013554 001426 BEQ 1$ ;BR IF NO
1115 013556 032714 000004 BIT #BIT2,(R4) ;IS IT REALLY A SHUT DOWN?
1116 013562 001403 BEQ 9$ ;BR IF NO
1117 013564 004737 014024 JSR PC,4$ ;YES, CLEAR RQI
1118 013570 000002 RTI ;AND RETURN
1119 013572 012764 017046 000004 9$: MOV #BASE,4(R4) ;LOAD BASE ADDRESS
1120 013600 005064 000006 CLR 6(R4) ;CLEAR SEL 6
1121 013604 005737 017012 TST RESUME ;RESUME FLAG SET?
1122 013610 001403 BEQ 8$ ;BR IF NOT
1123 013612 012764 010000 000006 MOV #BIT12,6(R4) ;OTHERWISE SET RESUME BIT
1124 013620 004737 014024 8$: JSR PC,4$ ;CLEAR RQI
1125 013624 152714 000141 BISB #141,(R4) ;ASK FOR CNTL I
1126 013630 000002 RTI ;RETURN
```

```

1127 013632 032714 000001      1$:   BIT    #BIT0,(R4)      ;CNTL I REQUEST?
1128 013636 001430              BEQ    2$              ;BR IF NO
1129 013640 032737 000001 011014  BIT    #FULL.DUPLEX,PARAM2;FULL OR HALF?
1130 013646 001403              BEQ    5$              ;BR IF HALF
1131 013650 005064 000006      CLR    6(R4)           ;SELECT FULL DUPLEX
1132 013654 000413              BR     6$              ;CONTINUE
1133 013656 032737 000004 011014  5$:   BIT    #BIT2,PARAM2   ;SECONDARY STATION?
1134 013664 001404              BEQ    7$              ;BR IF NO
1135 013666 012764 006000 000006      MOV    #BIT10!BIT11,6(R4);SET SEC MODE AND HALF DUPLEX
1136 013674 000403              BR     6$              ;CONTINUE
1137 013676 012764 002000 000006  7$:   MOV    #BIT10,6(R4)   ;SELECT HALF DUPLEX
1138 013704 004737 014024      6$:   JSR    PC,4$          ;CLEAR RQI
1139 013710 012737 177777 017014  MOV    #-1,BASEFG     ;SET BASE LOADED FLAG
1140 013716 000002              RTI                   ;RETURN
1141 013720 032714 000004      2$:   BIT    #BIT2,(R4)    ;RECEIVE BUFFER REQUEST?
1142 013724 001416              BEQ    3$              ;BR IF NO
1143 013726 042700 040000      BIC    #RFLG,STAT     ;CLEAR RECEIVE FLAG BIT
1144 013732 013764 011020 000004      MOV    IRDA,4(R4)    ;LOAD REC BUFFER ADDRESS
1145 013740 013764 011064 000006      MOV    RXWC,6(R4)   ;LOAD REC BYTE COUNT
1146 013746 004737 014024      JSR    PC,4$          ;CLEAR RQI
1147 013752 012737 177777 017014  MOV    #-1,BASEFG     ;SET FLAG
1148 013760 000002              RTI                   ;RETURN
1149 013762 042700 100000      3$:   BIC    #XFLG,STAT     ;CLEAR XMIT FLAG BIT
1150 013766 012737 177777 017022  MOV    #-1,XFLAG     ;SET XMIT FLAG(FOR TIMEOUT ERROR)
1151 013774 013764 011022 000004      MOV    IXDA,4(R4)   ;LOAD XMIT BUFFER ADDRESS
1152 014002 013764 011062 000006      MOV    TXWC,6(R4)   ;LOAD XMIT BYTE COUNT
1153 014010 004737 014024      JSR    PC,4$          ;CLEAR RQI
1154 014014 012737 177777 017014  MOV    #-1,BASEFG     ;SET FLAG
1155 014022 000002              RTI                   ;RETURN
1156 014024 142714 000040      4$:   BICB   #40,(R4)      ;CLEAR RQI
1157 014030 105714              TSTB  (R4)            ;WAIT FOR RDI TO DROP
1158 014032 100776              BMI   -2              ;BR IF STILL SET
1159 014034 000207              RTS    PC              ;RETURN

```

```

:*****
:      OUTPUT SERVICE ROUTINE
:*****

```

```

1165 014036 032764 000001 000002  RISR:  BIT    #BIT0,2(R4)   ;ERROR?
1166 014044 001463              BEQ    1$              ;BR IF NO
1167 014046 005737 017012      TST    RESUME         ;RESUME FLAG SET?
1168 014052 001413              BEQ    5$              ;BR IF NOT
1169 014054 022764 001000 000006      CMP    #BIT9,6(R4)   ;IS PROC. ERROR BIT SET?
1170 014062 001007              BNE    5$              ;BR IF NOT
1171 014064 142764 000207 000002  BICB   #207,2(R4)    ;CLEAR DONE
1172 014072 012737 177777 017014  MOV    #-1,BASEFG     ;SET BASEFG
1173 014100 000002              RTI                   ;RETURN
1174 014102 022764 000004 000006  5$:   CMP    #4,6(R4)      ;OVERUN ERROR?
1175 014110 001003              BNE    3$              ;BR IF NO
1176 014112 152714 000144      BISB   #144,(R4)     ;REQUEUE XMIT BUFFER
1177 014116 000432              BR     4$              ;
1178 014120 016437 000004 011054  3$:   MOV    4(R4),ERCSR   ;SAVE SEL4
1179 014126 016437 000006 011056  MOV    6(R4),ERDBR   ;SAVE SEL6 (ERROR BITS)
1180 014134 104400 016436      TYPE   ,KMCER        ;ERROR MESSAGE
1181 014140 013746 011054      MOV    ERCSR,-(SP)   ;PUSH SEL4 ON STACK FOR TYPEOUT
1182 014144 004037 016034      JSR    R0,$B20CT     ;TYPE IT OUT

```

```
1183 014150 006 .BYTE 6
1184 014151 001 .BYTE 1
1185 014152 104400 016632 TYPE ,SPACE3 ;INSERT 3 SPACES
1186 014156 013746 011056 MOV ERDBR,-(SP) ;SPUSH SEL6 ON STACK FOR TYPEOUT
1187 014162 004037 016034 JSR R0,$B20CT ;TYPE IT OUT
1188 014166 006 .BYTE 6
1189 014167 001 .BYTE 1
1190 014170 005777 174650 TST @SWR ;CHECK BIT 15
1191 014174 100001 BPL .+4 ;SKIP HALT IF = 0
1192 014176 000000 HALT ;HALT IF SWR15 = 1
1193 014200 005037 011056 CLR ERDBR ;CLR ERDBR LOCATION
1194 014204 142764 000207 000002 4$: BICB #207,2(R4) ;CLEAR DONE
1195 014212 000002 RTI ;RETURN
1196 014214 032764 000004 000002 1$: BIT #BIT2,2(R4) ;REC DONE?
1197 014222 001406 BEQ 2$ ;BR IF NO
1198 014224 052700 040000 BIS #RFLG,STAT ;SET REC DONE FLAG
1199 014230 142764 000207 000002 BICB #207,2(R4) ;CLEAR DONE
1200 014236 000002 RTI ;RETURN
1201 014240 052700 100000 2$: BIS #XFLG,STAT ;SET XMIT DONE FLAG
1202 014244 005037 017022 CLR XFLAG ;CLEAR XFLAG
1203 014250 142764 000207 000002 BICB #207,2(R4) ;CLEAR DONE
1204 014256 000002 RTI ;RETURN
1205
1206
1207 ;*****
1208 ; ENTER HERE IF BOOT MODE WAS SELECTED
1209 ;*****
1210
1211 014260 032737 003000 011012 BOOT: BIT #BIT10!BIT9,PARAM1;DETERMINE WHICH BOOT MODE
1212 014266 001413 BEQ AUTORG ;BR IF AUTO MODE, ORIGINATING STATION
1213 014270 032737 002000 011012 BIT #BIT10,PARAM1
1214 014276 001522 BEQ AUTBOO ;BR IF AUTO MODE, BOOT STATION
1215 014300 032737 001000 011012 BIT #BIT9,PARAM1
1216 014306 001540 BEQ MANORG ;BR IF MANUAL MODE, ORIGINATING STATION
1217 014310 104400 016636 TYPE ,BOOMSG ;MANUAL MODE, BOOT STATION
1218 014314 000777 BR . ;WAIT FOR MANUAL BOOT
1219
1220 ;*****
1221 ; AUTOMATIC MODE ORIGINATING STATION
1222 ;*****
1223
1224 014316 012701 015607 AUTORG: MOV #MOP1+1,R1 ;LOAD MOP1 MESSAGE WITH
1225 014322 113721 011012 MOVB PARAM1,(R1)+ ;PASSWORD FOUR TIMES
1226 014326 113721 011012 MOVB PARAM1,(R1)+
1227 014332 113721 011012 MOVB PARAM1,(R1)+
1228 014336 113721 011012 MOVB PARAM1,(R1)+
1229 014342 005737 017014 TST BASEFG ;IS BASE REQUEST DONE?
1230 014346 001763 BEQ AUTORG ;BR IF NO
1231 014350 005037 017014 CLR BASEFG ;CLEAR LOCK FLAG
1232 014354 012737 002400 017034 MOV #2400,SEL6 ;MAINT. MODE BIT(MOP)
1233 014362 052714 000141 BIS #141,(R4) ;ASK FOR CNTL IN
1234 014366 005737 017014 1$: TST BASEFG ;IS CNTL IN DONE?
1235 014372 001775 BEQ 1$ ;BR IF NO
1236 014374 005037 017014 CLR BASEFG ;CLEAR LOCK FLAG
1237 014400 005037 017024 CLR RFLAG ;CLEAR RECEIVE FLAG
1238 014404 052714 000144 BIS #144,(R4) ;ASK FOR REC BA/CC
```

```
1239 014410 005737 017014      2$:  TST  BASEFG      ;IS REQUEST DONE?
1240 014414 001775              BEQ  2$           ;BR IF NO
1241 014416 005037 017014      CLR  BASEFG      ;CLEAR LOCK FLAG
1242 014422 005037 017022      CLR  XFLAG       ;CLEAR XMIT FLAG
1243 014426 012737 015606 017032  MOV  #MOP1,SEL4  ;SET FOR KMC LOAD
1244 014434 012737 000005 017034  MOV  #5,SEL6     ;SET FOR KMC LOAD
1245 014442 052714 000140              BIS  #140,(R4)   ;ASK FOR XMIT BA/CC
1246 014446 012737 000005 017016  MOV  #5,TEMP1    ;SET UP DELAY COUNT
1247 014454 005037 013402      CLR  DELAY       ;
1248 014460 005737 017024      3$:  TST  RFLAG     ;RECEIVED ANYTHING?
1249 014464 001012              BNE  4$          ;BR IF YES
1250 014466 005337 013402      DEC  DELAY       ;DEC DELAY COUNTER
1251 014472 001372              BNE  3$          ;WAIT TO RECEIVE
1252 014474 005337 017016      DEC  TEMP1       ;DEC SECOND COUNT
1253 014500 001367              BNE  3$          ;BR IF NOT DONE DELAY
1254 014502 005737 017022      TST  XFLAG       ;WAS XMIT COMPLETED ?
1255 014506 001340              BNE  2$          ;BR IF YES, SEND MOP1 AGAIN
1256 014510 000000              HALT             ;ERROR, MOP1 WAS NEVER SENT OUT
```

;CHECK TO SEE IF RECEIVED MESSAGE IS MOP2

```
1257
1258
1259
1260 014512 012701 015614      4$:  MOV  #MOP2,R1  ;STARTING ADDRESS OF MOP2
1261 014516 013702 011020      MOV  IRDA,R2     ;RECEIVE BUFFER ADDRESS
1262 014522 005003              CLR  R3          ;CLEAR COUNT
1263 014524 122122              5$:  CMPB (R1)+,(R2)+ ;COMPARE DATA
1264 014526 001317              BNE  1$          ;IF NOT MOP2 TRY AGAIN
1265 014530 005203              INC  R3          ;DATA OK, BUMP COUNTER
1266 014532 022703 000004      CMP  #4,R3       ;DONE YET?
1267 014536 001372              BNE  5$          ;BR IF NO
1268 014540 004737 015512      JSR  PC,MP3      ;IT WAS MOP2, SO SEND MOP3
```

: AUTOMATIC MODE BOOT STATION
:*****

```
1269
1270
1271
1272
1273
1274 014544 005737 017014      AUTBOO: TST  BASEFG      ;BASE COMPLETED?
1275 014550 001775              BEQ  AUTBOO      ;BR IF NO
1276 014552 005037 017014      CLR  BASEFG      ;CLEAR LOCK FLAG
1277 014556 005037 017030      CLR  MFLAG       ;CLEAR MAINT FLAG
1278 014562 005037 017024      CLR  RFLAG       ;CLEAR REC FLAG
1279 014566 052714 000144      BIS  #144,(R4)   ;ASK FOR REC BA/CC
1280 014572 005737 017030      1$:  TST  MFLAG     ;ARE WE IN MAINT MODE?
1281 014576 001362              BNE  AUTBOO      ;BR IF YES
1282 014600 005737 017024      2$:  TST  RFLAG     ;DID WE RECEIVE ANYTHING?
1283 014604 001357              BNE  AUTBOO      ;YES REQUEUE RECEIVE BUFFER
1284 014606 000771              BR   1$          ;KEEP ON TRUCKIN'
```

: MANUAL MODE ORGINATING STATION
:*****

```
1285
1286
1287
1288
1289
1290 014610 005037 017030      MANORG: CLR  MFLAG     ;CLEAR MAINT FLAG
1291 014614 005037 017026      CLR  SFLAG       ;CLEAR DDCMP START RECEIVED ERROR FLAG
1292 014620 005737 017014      7$:  TST  BASEFG     ;BASE LOADED?
1293 014624 001775              BEQ  7$          ;BR IF NO
1294 014626 005037 017014      CLR  BASEFG      ;RESET FLAG
```

1295	014632	005037	017034		CLR	SEL6	:LOAD SEL6 FOR FULL-DUPLEX
1296	014636	052714	000141		BIS	#141,(R4)	:ASK FOR CNTLI
1297	014642	005737	017014	8\$:	TST	BASEFG	:CNTLI DONE?
1298	014646	001775			BEQ	8\$:BR IF NOT
1299	014650	005037	017014		CLR	BASEFG	:RESET FLAG
1300	014654	005037	017024		CLR	RFLAG	:CLEAR RECEIVE FLAG
1301	014660	052714	000144		BIS	#144,(R4)	:ASK FOR REC BA/CC
1302	014664	005737	017026	1\$:	TST	SFLAG	:DDCMP START ERROR?
1303	014670	001437			BEQ	4\$:BR IF NO
1304	014672	012714	040000		MOV	#BIT14,(R4)	:INITIALIZE KMC
1305	014676	005714		2\$:	TST	(R4)	:RUN SET?
1306	014700	100376			BPL	2\$:BR IF NO
1307	014702	005037	017026		CLR	SFLAG	:CLEAR FLAG
1308	014706	005037	017014		CLR	BASEFG	:CLEAR LOCK FLAG
1309	014712	052764	000100	000002	BIS	#100,2(R4)	:SET INT ENABLE
1310	014720	052714	000143		BIS	#143,(R4)	:ASK FOR BASE
1311	014724	005737	017014	3\$:	TST	BASEFG	:BASE DONE?
1312	014730	001775			BEQ	3\$:BR IF NO
1313	014732	005037	017014		CLR	BASEFG	:CLEAR FLAG
1314	014736	005037	017034		CLR	SEL6	:SET UP FOR FULL-DUPLEX
1315	014742	052714	000141		BIS	#141,(R4)	:ASK FOR CNTLI
1316	014746	005737	017014	9\$:	TST	BASEFG	:CNTLI FINISHED?
1317	014752	001775			BEQ	9\$:BR IF NOT
1318	014754	005037	017014		CLR	BASEFG	:CLEAR FLAG
1319	014760	005037	017024		CLR	RFLAG	:CLEAR RECEIVER FLAG
1320	014764	052714	000144		BIS	#144,(R4)	:ASK FOR REC BA/CC
1321	014770	005737	017030	4\$:	TST	MFLAG	:ARE WE IN MAINT MODE?
1322	014774	001733			BEQ	1\$:BR IF NO
1323	014776	012714	040000		MOV	#BIT14,(R4)	:INITIALIZE KMC
1324	015002	005714		10\$:	TST	(R4)	:RUN SET?
1325	015004	100376			BPL	10\$:BR IF NO
1326	015006	005037	017014		CLR	BASEFG	:CLEAR LOCK FLAG
1327	015012	052764	000100	000002	BIS	#100,2(R4)	:SET INT ENABLE
1328	015020	052714	000143		BIS	#143,(R4)	:ASK FOR BASE
1329	015024	005737	017014	11\$:	TST	BASEFG	:BASE DONE?
1330	015030	001775			BEQ	11\$:BR IF NO
1331	015032	005037	017014		CLR	BASEFG	:CLEAR FLAG
1332	015036	012737	002400	017034	MOV	#2400,SEL6	:MAINT. MODE (MOP)
1333	015044	052714	000141		BIS	#141,(R4)	:ASK FOR CNTLI
1334	015050	005737	017014	12\$:	TST	BASEFG	:CNTLI FINISHED?
1335	015054	001775			BEQ	12\$:BR IF NOT
1336	015056	005037	017014		CLR	BASEFG	:CLEAR FLAG
1337	015062	005037	017024		CLR	RFLAG	:CLEAR RECEIVER FLAG
1338	015066	052714	000144		BIS	#144,(R4)	:ASK FOR REC BA/CC
1339	015072	005737	017024	5\$:	TST	RFLAG	:HAVE WE RECEIVED ANYTHING?
1340	015076	001775			BEQ	5\$:BR IF NO
1341							
1342							
1343							:CHECK TO SEE IF RECEIVED MESSAGE IS MOP2
1344	015100	012701	015614		MOV	#MOP2,R1	:MOP2 STARTING ADDRESS
1345	015104	013702	011020		MOV	IRDA,R2	:RECEIVE BUFFER ADDRESS
1346	015110	005003			CLR	R3	:CLEAR COUNT
1347	015112	122122		6\$:	CMPB	(R1)+,(R2)+	:COMPARE DATA
1348	015114	001343			BNE	11\$:IT ISN'T MOP2, TRY AGAIN
1349	015116	005203			INC	R3	:DATA OK, BUMP COUNT
1350	015120	022703	000004		CMP	#4,R3	:DONE YET?

```

1351 015124 001372          BNE      6$          ;BR IF NO
1352 015126 004737 015512  JSR      PC,MP3      ;IT WAS MOP2, SO SEND OUT MOP3
1353
1354
1355 :*****
1356 :      INPUT INTERRUPT SERVICE ROUTINE (BOOT MODE)
1357 :*****
1358 015132 032714 000002  IISR:  BIT      #BIT1,(R4)  ;IS IT A BASE REQUEST?
1359 015136 001413          BEQ      1$          ;NO
1360 015140 012764 017046 000004  MOV      #BASE,4(R4)  ;YES, LOAD BASE ADDRESS
1361 015146 005064 000006          CLR      6(R4)      ;CLEAR SEL6
1362 015152 004737 015304          JSR      PC,4$      ;CLEAR RQI
1363 015156 012737 177777 017014  MOV      #-1,BASEFG  ;SET FLAG
1364 015164 000002          RTI          ;RETURN
1365 015166 032714 000001  1$:   BIT      #BIT0,(R4)  ;IS IT CNTL IN?
1366 015172 001411          BEQ      2$          ;BR IF NO
1367 015174 013764 017034 000006  MOV      SEL6,6(R4)  ;LOAD SEL6 FOR CNTLI
1368 015202 004737 015304          JSR      PC,4$      ;CLEAR RQI
1369 015206 012737 177777 017014  MOV      #-1,BASEFG  ;SET FLAG
1370 015214 000002          RTI          ;RETURN
1371 015216 032714 000004  2$:   BIT      #BIT2,(R4)  ;IS IT RECEIVE REQUEST?
1372 015222 001414          BEQ      3$          ;NO
1373 015224 013764 011020 000004  MOV      IRDA,4(R4)  ;YES, LOAD REC BA
1374 015232 012764 000010 000006  MOV      #10,6(R4)  ;CC
1375 015240 004737 015304          JSR      PC,4$      ;CLEAR RQI
1376 015244 012737 177777 017014  MOV      #-1,BASEFG  ;SET FLAG
1377 015252 000002          RTI          ;RETURN
1378 015254 013764 017032 000004  3$:   MOV      SEL4,4(R4)  ;XMIT REQUEST, LOAD XMIT BA
1379 015262 013764 017034 000006  MOV      SEL6,6(R4)  ;XMIT CC
1380 015270 004737 015304          JSR      PC,4$      ;CLEAR RQI
1381 015274 012737 177777 017014  MOV      #-1,BASEFG  ;SET FLAG
1382 015302 000002          RTI          ;RETURN
1383 015304 142714 000040  4$:   BICB     #40,(R4)    ;CLEAR RQI
1384 015310 105714          TSTB     (R4)       ;RDI CLEAR?
1385 015312 100776          BMI      -2         ;NO
1386 015314 000207          RTS      PC         ;RETURN
1387
1388 :*****
1389 :      OUTPUT INTERRUPT SERVICE ROUTINE (BOOT MODE)
1390 :*****
1391
1392 015316 032764 000001 000002  OISR:  BIT      #BIT0,2(R4) ;ERROR?
1393 015324 001453          BEQ      3$          ;NO
1394 015326 022764 000010 000006  CMP      #10,6(R4)  ;YES, MAINT MODE ENTERED?
1395 015334 001004          BNE      1$          ;NO
1396 015336 012737 177777 017030  MOV      #-1,MFLAG  ;YES, SET MFLAG
1397 015344 000456          BR       5$          ;RETURN
1398 015346 022764 000200 000006  1$:   CMP      #200,6(R4) ;DDCMP START RECEIVED ERROR?
1399 015354 001004          BNE      2$          ;NO
1400 015356 012737 177777 017026  MOV      #-1,SFLAG  ;YES, SET SFLAG
1401 015364 000446          BR       5$          ;RETURN
1402 015366 016437 000004 011054  2$:   MOV      4(R4),ERCSR ;SAVE SEL4
1403 015374 016437 000006 011056  MOV      6(R4),ERDBR ;SAVE SEL6 (ERROR BITS)
1404 015402 104400 016436          TYPE     ,KMCER     ;ERROR MESSAGE
1405 015406 013746 011054          MOV      ERCSR,-(SP) ;PUSH SEL4 ON STACK FOR TYPEOUT
1406 015412 004037 016034          JSR      RO,$B2OCT  ;TYPE IT OUT

```

```

1407 015416 006 .BYTE 6
1408 015417 001 .BYTE 1
1409 015420 104400 016632 TYPE ,SPACE3 ;INSERT 3 SPACES
1410 015424 013746 011056 MOV ERDBR,-(SP) ;SPUSH SEL6 ON STACK FOR TYPEOUT
1411 015430 004037 016034 JSR R0,$B2OCT ;TYPE IT OUT
1412 015434 006 .BYTE 6
1413 015435 001 .BYTE 1
1414 015436 005777 173402 TST @SWR ;CHECK BIT 15
1415 015442 100001 BPL .+4 ;SKIP HALT IF = 0
1416 015444 000000 HALT ;HALT IF SWR15 = 1
1417 015446 005037 011056 CLR ERDBR ;CLR ERDBR LOCATION
1418 015452 000413 BR 5$ ;RETURN
1419 015454 032764 000004 000002 3$: BIT #BIT2,2(R4) ;RECEIVE DONE?
1420 015462 001404 BEQ 4$ ;BR IF NO
1421 015464 012737 177777 017024 MOV #-1,RFLAG ;SET RECEIVE FLAG
1422 015472 000403 BR 5$ ;RETURN
1423 015474 012737 177777 017022 4$: MOV #-1,XFLAG ;XMIT DONE, SET XMIT FLAG
1424 015502 142764 000207 000002 5$: BICB #207,2(R4) ;CLEAR DONE
1425 015510 000002 RTI ;RETURN
1426
1427 ;*****
1428 ; SUBROUTINE TO SEND MOP3 MESSAGE
1429 ;*****
1430
1431 015512 005737 017014 MP3: TST BASEFG ;IS IT OK TO REQUEST
1432 015516 001775 BEQ MP3 ;BR IF NO
1433 015520 005037 017014 CLR BASEFG ;CLEAR LOCK FLAG
1434 015524 005037 017022 CLR XFLAG ;CLEAR XMIT FLAG
1435 015530 012737 015620 017032 MOV #MOP3,SEL4 ;MOP3 ADDRESS
1436 015536 012737 000154 017034 MOV #MOP3ED-MOP3,SEL6 ;MOP3 COUNT
1437 015544 052714 000140 BIS #140,(R4) ;ASK FOR XMIT BA/CC
1438 015550 005037 013402 CLR DELAY ;START DELAY COUNT
1439 015554 005737 017022 1$: TST XFLAG ;XMIT DONE?
1440 015560 001004 BNE 2$ ;BR IF YES
1441 015562 005337 013402 DEC DELAY ;DEC DELAY COUNT
1442 015566 001372 BNE 1$ ;BR IF NO DONE
1443 015570 000000 HALT ;ERROR, MOP3 SEND NOT DONE
1444 015572 104400 016705 2$: TYPE ,ORGOK ;OK MOP3 SEND DONE
1445 015576 005726 TST (SP)+ ;PCP STACK (ENTERED BY JSR)
1446 015600 000000 HALT ;ALL DONE HIT CONT TO DO IT AGAIN
1447 015602 000137 011106 JMP START
1448
1449 015606 006 000 000 MOP1: .BYTE 6,0,0,0,0
1450 015611 000 000 .EVEN
1451 015614 010 014 001 MOP2: .BYTE 10,12.,1,0
1452 015617 000 006 MOP3: .BYTE 0,0,6,0,6,0
1453 015620 000 006 000
1454 015623 000 006 000
1455 015623 000 006 000
1456 ;IMAGE OF PROGRAM TO BE DOWN LINE LOADED
1457
1458 015626 005037 000006 CLR @#6 ;SET UP TIMEOUT VECTOR TO HALT
1459 015632 000005 RESET ;CLEAR ALL!!
1460 015634 012706 001000 MOV #1000,SP ;SET UP STACK
1461 015640 012701 177560 MOV #177560,R1 ;SET TTY CSR
1462 015644 010700 MOV PC,R0 ;MAKE ADDRESS PIC
    
```

```

1463 015646 062700 000034          ADD    #<MSG-.>,R0      ;ADDRESS OF MESSAGE
1464 015652 105761 000004          TSTB   4(R1)           ;READY SET?
1465 015656 100375                BPL    1$              ;BR IF NO
1466 015660 112061 000006          MOVB   (R0)+,6(R1)    ;TYPE A CHARACTER
1467 015664 001372                BNE    1$              ;KEEP TYPING IF NOT ZERO
1468 015666 012737 000026 000024  MOV    #26,@#24       ;SET UP POWER FAIL VECTOR
1469 015674 005037 000026          CLR    @#26           ;MAKE SURE T BIT CLEAR
1470 015700 000777                BR     ;BR
1471 015702 006412 047502 052117  MSG:   .ASCIZ <12><15>/BOOT MESSAGE WAS RECEIVED SUCCESSFULLY - END OF TEST!!/
1472 015710 046440 051505 040523
1473 015716 042507 053440 051501
1474 015724 051040 041505 044505
1475 015732 042526 020104 052523
1476 015740 041503 051505 043123
1477 015746 046125 054514 026440
1478 015754 042440 042116 047440
1479 015762 020106 042524 052123
1480 015770 020441      000
1481 015773      006
1482 015774                MOP3ED: .BYTE 6
1483                .EVEN
1484
1485
1486 015774 006412 051105 047522  MLDER: .ASCIZ <12><15>'ERROR IN LOADING MICRO-CODE'<15><12>
1487 016002 020122 047111 046040
1488 016010 040517 044504 043516
1489 016016 046440 041511 047522
1490 016024 041455 042117 006505
1491 016032 000012
1492
1493                .EVEN
1494                ;*****
1495                ;BINARY TO OCTAL (ASCII) AND TYPE
1496                ;$B2OCT---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
1497                ;CALL:
1498                ;       MOV    NUM,-(SP)           ;NUMBER TO BE TYPED
1499                ;       JSR    R0,$B2OCT         ;CALL FOR TYPEOUT
1500                ;       .BYTE  N               ;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
1501                ;       .BYTE  M               ;M=1 OR 0
1502                ;                               ;1=TYPE LEADING ZEROS
1503                ;                               ;0=SUPPRESS LEADING ZEROS
1504                ;$B201----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST $BSOCT OR $B2016
1505                ;CALL:
1506                ;       MOV    NUM,-(SP)
1507                ;       JSR    R0,$B201
1508                ;
1509                ;$B2016---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
1510                ;CALL:
1511                ;       MOV    NUM,-(SP)
1512                ;       JSR    R0,$B2016
1513
1514 016034 112037 016241  $B2OCT: MOVB   (R0)+,$SOMODE+1 ;PICKUP THE NUMBER OF DIGITS TO TYPE
1515 016040 112037 016237  MOVB   (R0)+,$OFILL      ;GET THE ZERO FILL SWITCH
1516 016044 000406          BR     $B201
1517 016046 112737 000001 016237 $B2016: MOVB   #1,$OFILL    ;SET THE ZERO FILL SWITCH
1518 016054 112737 000006 016241  MOVB   #6,$SOMODE+1    ;SET FOR SIX(6) DIGITS
    
```

```

1519 016062 112737 000005 016236 $B201:  MOVB   #5,$SOCNT      ;SET THE ITERATION COUNT
1520 016070 010346          MOV    R3,-(SF)      ;SAVE R3
1521 016072 010446          MOV    R4,-(SP)      ;SAVE R4
1522 016074 010546          MOV    R5,-(SP)      ;SAVE R5
1523 016076 113704 016241  MOVB   $SOMODE+1,R4  ;GET THE NUMBER OF DIGITS TO TYPE
1524 016102 005404          NEG    R4
1525 016104 062704 000006  ADD    #6,R4        ;SUBTRACT IT FOR MAX. ALLOWED
1526 016110 110437 016240  MOVB   R4,$SOMODE    ;SAVE IT FOR USE
1527 016114 113704 016237  MOVB   $OFILL,R4     ;GET THE ZERO FILL SWITCH
1528 016120 016605 000010  MOV    10(SP),R5     ;PICKUP THE INPUT NUMBER
1529 016124 005003          CLR    R3           ;CLEAR THE OUTPUT WORD
1530 016126 006105          1$:   ROL    R5       ;ROTATE MSB INTO 'C'
1531 016130 000404          BR     3$          ;GO DO MSB
1532 016132 006105          2$:   ROL    R5       ;FORM THIS DIGIT
1533 016134 006105          ROL    R5
1534 016136 006105          ROL    R5
1535 016140 010503          MOV    R5,R3
1536 016142 006103          3$:   ROL    R3       ;GET LSB OF THIS DIGIT
1537 016144 105337 016240  DECB   $SOMODE      ;TYPE THIS DIGIT?
1538 016150 100016          BPL    7$          ;BR IF NO
1539 016152 042703 177770  BIC    #177770,R3   ;GET RID OF JUNK
1540 016156 001002          BNE    4$          ;TEST FOR 0
1541 016160 005704          TST   R4           ;SUPPRESS THIS 0?
1542 016162 001403          BEQ   5$          ;BR IF YES
1543 016164 005204          4$:   INC    R4       ;DON'T SUPPRESS ANYMORE 0'S
1544 016166 052703 000060  BIS    #'0,R3      ;MAKE THIS DIGIT ASCII
1545 016172 052703 000040  5$:   BIS    #' ,R3   ;MAKE ASCII IF NOT ALREADY
1546 016176 110337 016234  MOVB   R3,8$       ;SAVE FOR TYPING
1547 016202 104400 016234  TYPE   ,8$        ;GO TYPE THIS DIGIT
1548 016206 105337 016236  7$:   DECB   $SOCNT   ;COUNT BY 1
1549 016212 003347          BGT   2$          ;BR IF MORE TO DO
1550 016214 002402          BLT   6$          ;BR IF DONE
1551 016216 005204          INC   R4          ;INSURE LAST DIGIT ISN'T A BLANK
1552 016220 000744          BR    2$          ;GO DO THE LAST DIGIT
1553 016222 012605          6$:   MOV    (SP)+,R5 ;RESTORE R5
1554 016224 012604          MOV    (SP)+,R4   ;RESTORE R4
1555 016226 012603          MOV    (SP)+,R3   ;RESTORE R3
1556 016230 012616          MOV    (SP)+,(SP) ;SET THE STACK FOR RETURNING
1557 016232 000200          RTS   R0          ;RETURN
1558 016234 000          8$:   .BYTE 0       ;STORAGE FOR ASCII DIGIT
1559 016235 000          .BYTE 0       ;TERMINATOR FOR TYPE ROUTINE
1560 016236 000          $SOCNT: .BYTE 0 ;OCTAL DIGIT COUNTER
1561 016237 000          $OFILL: .BYTE 0 ;ZERO FILL SWITCH
1562 016240 000000          $SOMODE: 0     ;NUMBER OF DIGITS TO TYPE
1563
1564
1565 016242 012737 177777 017012  CKBASE: MOV    #-1,RESUME ;SET RESUME FLAG
1566 016250 005037 017014          CLR    BASEFG     ;CLEAR BASEFG
1567 016254 052714 000146          BIS    #146,(R4)  ;SHUT DOWN KMC TO UPDATE BASE TABLE
1568 016260 005737 017014          1$:   TST   BASEFG   ;SHUT DOWN DONE?
1569 016264 001775          BEQ   1$          ;BR IF NO
1570 016266 012714 040000  MOV    #BIT14,(R4) ;MASTER CLEAR KMC
1571 016272 012714 100000  MOV    #BIT15,(R4)
1572 016276 005714          2$:   TST   (R4)    ;RUN SET?
1573 016300 100376          BPL   2$          ;BR IF NO
1574 016302 005037 017014          CLR    BASEFG     ;CLEAR BASEFG

```

```

1575 016306 052714 000143      BIS      #143,(R4)      ;ASK FOR BASE REQUEST
1576 016312 005737 017014      3$:     TST      BASEFG      ;BASE LOADED?
1577 016316 001775          BEQ      3$          ;BR IF NO
1578 016320 012764 000100 000002      MOV      #100,2(R4)  ;SET INTERRUPT ENABLE
1579 016326 005037 017012          CLR      RESUME      ;CLEAR RESUME FLAG
1580 016332 012702 000003      MOV      #3,R2      ;LOAD BASE OFFSET TO ERROR COUNTS
1581 016336 005001          CLR      R1         ;R1 IS OFFSET INTO SOFTWARE TABLE
1582 016340 126261 017046 017036 5$:     CMPB     BASE(R2),ERRCNT(R1);ANY ERRORS THIS PASS?
1583 016346 001005          BNE     6$          ;BR IF YES
1584 016350 122122          CMPB     (R1)+,(R2)+ ;INC INDEXS
1585 016352 022702 000013      CMP      #13,R2     ;DONE?
1586 016356 001370          BNE     5$          ;BR IF NO
1587 016360 000207          RTS      PC         ;RETURN
1588 016362 104400 016507      6$:     TYPE     ,SOFT      ;TYPE SOFT ERROR MESSAGE
1589 016366 012702 000003      MOV      #3,R2      ;LOAD BASE OFFSET TO ERROR COUNTS
1590 016372 005001          CLR      R1         ;SOFTWARE TABLE OFFSET
1591 016374 116261 017046 017036 7$:     MOVB     BASE(R2),ERRCNT(R1);SAVE ERROR COUNTS
1592 016402 116246 017046      MOVB     BASE(R2),-(SP) ;PUSH ON STACK FOR TYPEOUT
1593 016406 042716 177400      BIC      #^C<377>,(SP) ;CLEAR HI-BYTE
1594 016412 004037 016034      JSR      R0,$B20CT  ;TYPE IT OUT
1595 016416          .BYTE   3
1596 016417          .BYTE   1
1597 016420 104400 016632      TYPE     ,SPACE3    ;INSERT 3 SPACES
1598 016424 122221          CMPB     (R2)+,(R1)+ ;INC INDEXS
1599 016426 022702 000013      CMP      #13,R2     ;DONE?
1600 016432 001360          BNE     7$          ;BR IF NO
1601 016434 000207          RTS      PC         ;RETURN
1602
1603
1604
1605          .NLIST   BEX
1606          KMCER: .ASCII <15><12><12>/CONTROL OUT ERROR/
1607          .ASCII <15><12>/ SEL4 SEL6/
1608          .ASCII <15><12>/ /
1609          SOFT: .ASCII <15><12><12>/SOFT ERROR - DDCMP ERROR COUNTS ARE NON ZERO/
1610          .ASCII <15><12>/ BASE+3 THRU BASE+12/<15><12><0>
1611          SPACE3: .ASCII / /
1612          BOOMSG: .ASCII <15><12>/MANUALLY BOOT KMC NOW (VIA M9301-YJ)/
1613          ORGOK: .ASCII <15><12>/ORIGINATING STATION HAS COMPLETED BOOT SUCCESSFULLY - END OF TE
1614          .EVEN
1615          .LIST   BEX
1616
1617          RESUME: 0
1618          BASEFG: 0 ;BASE LOAD FLAG
1619          TEMP1: 0
1620          TEMP2: 0
1621          XFLAG: 0
1622          RFLAG: 0
1623          SFLAG: 0
1624          MFLAG: 0
1625          SEL4: 0
1626          SEL6: 0
1627
1628          ERRCNT: .BLKW 4
1629
1630          BASE: .+.256.

```

1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670

```

:+
: THIS ROUTINE IS DESIGNED TO LOAD MICRO CODE INTO THE
: MICRO-CPU AND VERIFY THAT IT WAS LOADED PROPERLY.
:CALL = JSR PC,LDRVRF
:IF A LOAD ERROR IS DETECTED,AN ERROR WILL BE TYPED OUT.
:ROUTINE 'SETMAP' IS CALLED IN ORDER TO DETERMINE WHICH
:VERSION OF MICRO-CODE TO LOAD.
:-

```

```

LDRVRF: MOV #BIT14,(R1) ;MASTER CLEAR KMC-11.
        BIC #BIT15!BIT14,(R1) ;AND SHUT IT DOWN.
        CLR R0 ;CLEAR UPC POINTER.
        JSR PC,SETMAP ;SET MICRO-CODE POINTER IN R2.
3$: CLR (R1) ;START WITH THE CLEAN WORLD.
    MOV R0,4(R1) ;LOAD CRAM ADDRESS.
    MOV (R2)+,6(R1) ;LOAD INSTRUCTION WORD.
    MOV #BIT10,(R1) ;SET ROM 0.
    MOV #BIT13!BIT10,(R1) ;WRITE IT...
    INC R0 ;UPDATE UPC POINTER.
    CMP #2000,R0 ;OVER FLOW?
    BGT 3$ ;BR IF NO.
VERFY: JSR PC,SETMAP ;SET MICRO-CODE POINTER IN R2.
        TST PARAM1 ;VERIFY MICRO-CODE
        BMI 10$ ;=1 NO.
        CLR R0 ;SET UPC POINTER.
6$: CLR (R1) ;START WITH THE CLEAN WORLD.
    MOV R0,4(R1) ;LOAD CRAM ADDRESS.
    MOV #BIT10,(R1) ;SET ROM 0.
    CMP 6(R1),(R2)+ ;CHECK IF RIGHT?
    BEQ 9$ ;BR IF GOOD.
    TYPE ,MLDER ;LOADING ERROR.
9$: INC R0 ;BUMP UPC POINTER.
    CMP #2000,R0 ;IS IT DONE?
    BGT 6$ ;BR IF NO.
10$: MOV #BIT14,(R1) ;MASTER CLEAR KMC-11.
     RTS PC ;RETURN.

SETMAP: MOV #LOMAP,R2 ;LOAD ADDRESS OF LOW SPEED.
        BIT #BIT1,PARAM1 ;IS IT HIGH SPEED?
        BEQ 3$ ;BR IF NO.
        MOV #HIMAP,R2 ;LOAD HIGH SPEED ADDRESS.
3$: RTS PC ;RETURN TO CALLER.

```

AUTBOO 014544	EOP 012466	MSG4 013162	R6 =%000006	TIME 011032
AUTORG 014316	ERCSR 011054	MSG5 013314	R7 =%000007	TKB 011100
BA 011004	ERDBR 011056	NODAT = 000200	SAVR0 013366	TKS 011076
BACK 013404	ERRCNT 017036	OISR 015316	SAVR1 013370	TPB 011104
BASE 017046	FLAG 011042	ORGOK 016705	SAVR2 013372	TPS 011102
BASEFG 017014	FULL.D= 000001	OWI = 000002	SAVR3 013374	TSTDAT 012662
BIT0 = 000001	GO 011410	OWO = 000001	SAVR4 013376	TXWC 011062
BIT1 = 000002	HIMAP 027654	PARAM1 011012	SAVR5 013400	TX.TER 011040
BIT10 = 002000	IISR 015132	PARAM2 011014	SCAN4 012672	TYPE = 104400
BIT11 = 004000	ILB = 000010	PARAM3 011016	SCAN5 012746	VERFY 017516
BIT12 = 010000	IRDA 011020	PRIOR 011010	SEL4 017032	XCC 011066
BIT13 = 020000	IXDA 011022	PRTY0 = 000000	SEL6 017034	XCSR = 000002
BIT14 = 040000	KBDIN = 104416	PRTY1 = 000040	SEMAP 017574	XDA 011074
BIT15 = 100000	KMCER 016436	PRTY2 = 000100	SETTLE 011024	XFLAG 017022
BIT2 = 000004	KMC11 011000	PRTY3 = 000140	SFLAG 017026	XFLG = 100000
BIT3 = 000010	LDRVRF 017446	PRTY4 = 000200	SOFT 016507	XISR 013550
BIT4 = 000020	LOMAP 023650	PRTY5 = 000240	SPACE3 016632	XLB = 000004
BIT5 = 000040	LOOP = 000020	PRTY6 = 000300	SRCR 011052	XWAIT = 104412
BIT6 = 000100	MANORG 014610	PRTY7 = 000340	START 011106	\$B2OCT 016034
BIT7 = 000200	MFLAG 017030	QTPIE 012624	STARTR 013500	\$B201 016062
BIT8 = 000400	MLDER 015774	RCC 011070	STARTX 013410	\$B2016 016046
BIT9 = 001000	MODE =%000000	RCSR =%000004	STAT =%000000	\$ILB 011714
BOOMSG 016636	MOP1 015606	RDA 011072	STOP 013406	\$OCNT 016236
BOOT 014260	MOP2 015614	RDAX 012774	STPS = 104414	\$OMODE 016240
B2016 011030	MOP3 015620	RESUME 017012	SWR 011044	\$OWI 011500
CKBASE 016242	MOP3ED 015774	RFLAG 017024	SW12 = 010000	\$OWO 011612
DELAY 013402	MP3 015512	RFLG = 040000	SXCSR 011050	\$XLB 012150
DERR 013014	MSG 015702	RISR 014036	TEMP1 017016	\$OFILL 016237
DISPLA 011046	MSG0 013026	RIV 011006	TEMP2 017020	. = 033654
DSFLG = 020000	MSG1 013107	RWAIT = 104410	TESTD 012626	
DSSTAT 011060	MSG2 013112	RXWC 011064	TESTDX 013022	
ENTER 012554	MSG3 013137	RX.TER 011041	TIE = 000100	

. ABS. 033654 000

ERRORS DETECTED: 0

DSKZ:CZKMOA,DSKZ:CZKMOA/SOL=CZKMOA.MAC,CZKMOA.P11
RUN-TIME: 6 8 .1 SECONDS
RUN-TIME RATIO: 217/14=14.6
CORE USED: 17K (33 PAGES)