IP112,IP113 PROCESS CTRL SS
IPV12 CVPCAG1
AH-A961G-M1
FICHE 1 OF 2
SEP 1982

IP112,IP113 PROCESS CTRL SS AH-A961G-M1 SEP 1982
IPV12 CVPCAG1 FICHE 2 OF 2 COPYRIGHT 77-82
MADE IN USA

.REM .

IDENTIFICATION

PRODUCT CODE: AC-A959G-M1

PRODUCT NAME: CVPCAG1 PROCESS CTRL SS

DATE: 23-JULY-82

MAINTENANCE: MDC DIAGNOSTIC GROUP

AUTHOR: H. SZEJNWALD

1.0 ABSTRACT

THIS PROGRAM IS A DIAGNOSTIC TOOL FOR TESTING THE ENTIRE FAMILY OF PCS MODULES. THE PROGRAM IS INTENDED TO BE USE BY FIELD ENGINEERING AND MANUFACTURING . BY USING THIS PROGRAM AN OPERATOR IS ABLE TO CHECK THE IOCM, AND ANY DIGITAL OR ANALOG MODULE. SINCE THE ASSUMPTION IS MADE THAT ALL OUTPUT MODULES ARE CONNECTED TO THE CUSTOMERS WIRING DURING THE TEST OF OUTPUT MODULES THE DISABLE BIT IS SET. THIS DIAGNOSTIC WILL NOT CHECK THE PART OF HARDWARE THAT IS INTERFACING DIGITAL OR ANALOG MODULES WITH CUSTOMERS WIRING (DRIVING TRANSISTOR, ISOLATING TRANSFORMERS, PHOTO-COUPLERS ECT). THIS DIAGNOSTIC DOES NOT REQUIRE ANY EXTERNAL DEVICES OR SPECIAL CONNECTIONS. NOTE: THERE ARE CERTAIN COMMANDS (F),(W) AND ANALOG CALIBRATION ROUTINES IN WHICH THE OUTPUT TERMINALS WILL BE ACTIVE.

2.0 HARDWARE REQUIREMENTS

1. ANY PDP11 CPU WITH 20K MEMORY, LINE CLOCK
2. CONSOLE TERMINAL (EXCEPT DPM50)
3. LOADING DEVICE
4. D-BUS INTERFACE

3.0 PROGRAM CONSIDERATION

3.1 THE PROGRAM CAN BE RUN UNDER XXDP, BY ITSELF OR UNDER THE DIAGNOSTIC MONITOR - CZKMP

3.1.1 XXDP

THIS PROGRAM CAN BE CHAINED BY XXDP & WILL NOT OVERLAY THE LOADER.
CHAIN MODE OPERATION
1. THE INPUT DIALOGUE WITH AN OPERATOR IS BYPASSED
2. THE SUBSYSTEM IS MAPPED IN MEMORY
3. THE SYSTEM TEST IS AUTOMATICALLY EXECUTED

3.1.2 NORMAL OPERATION

1. START AT LOC 200
2. SELECT TEST
3. PROGRAM RUNS AND GOES BACK TO MONITOR

3.1.3 CZKMP - DIAGNOSTIC MONITOR

THIS PROGRAM CAN RUN UNDER CZKMP IN HOST, COMMUNICATION AND LOCAL MODES. IN LOCAL MODE, CZKMP ACTS AS A LOAD MEDIA FOR THE DIAGNOSTIC. IN HOST AND COMMUNICATIONS MODE, A DBUS MAP IS DISPLAYED AT THE HOST TERMINAL AND THE SYSTEM TEST IS EXECUTED. INFORMATION ON CZKMP CAN BE FOUND IN THE DPM DIAGNOSTIC USERS GUIDE.

3.2 ACT/APT

THIS PROGRAM IS ACT COMPATIBLE TO THE EXTENT THAT APT HOOKS WILL BE IN THE PROGRAM & WILL WORK THRU THE "UPTON INTERFACE".

3.3 EXECUTION TIME

MOST INDIVIDUAL TESTS TAKE LESS THEN 1 SEC.THE ONESHOT MODULE TEST TAKES 15 SEC. THE SYSTEM TEST EXECUTION TIME DEPENDS ON THE NUMBER AND TYPE OF I/O MODULES. THE TEST OF ANALOG MODULES TAKES ALSO UP TO 1 MIN PER MODULE.

3.4 DEFAULT ADDRESSES & VECTORS

THE FOLLOWING IS A LIST OF THE DEFAULT ADDRESSES & VECTORS OF ALL HARDWARE TO BE USED AND THEIR CORRESPONDING MEMORY LOCATIONS. IN THE CASE THAT THE IOCM IS SET TO AN ADDRESS OTHER THAN 171000 THESE LOCATIONS MUST BE CHANGED BY USING THE B-COMMAND AS DETAILED IN 4.3.

```
BASE:   .WORD   171000  ;FIRST ADDRESS OF I/O ADDRESS BLOCK
CSR:    .WORD   171377  ;ADDRESS OF IOCM CSR REGISTER
IAR:    .WORD   171376  ;       "        IAR REGISTER
VECTO:  .WORD   234     ;INTERRUPT VECTOR OF IOCM
VECTOA: .WORD   236     ;INTERRUPT VECTOR+2 OF IOCM
```

4.0 CVPCA OPERATING PROCEDURE

4.1 CVPCA PROGRAM LOADING

THE PROGRAM CAN BE LOADED FROM ANY LOADING DEVICE USING STANDARD PROCEDURES.

4.2 PROGRAM STARTING

LOCATION 200 - STARTING ADDRESS TO RUN THE CVPCA DIAGNOSTICS MONITOR.

THE PROGRAM WILL TYPE "WHICH TEST OPTION (H=HELP) ?"

IF THE LETTER H IS ENTERED THE VALID LIST OF COMMANDS AND TEST DESCRIPTIONS WILL BE TYPED.

```
TEST OPTIONS:
S SYSTEM TEST
A MODULE TEST
B CHANGE BASE ADDRESS
M MAP OF DBUS INTERFACES
I IOCM TEST
T SET SWREG
F FIELD TEST
W LOOP TEST
L LOAD ITERATION COUNT
H HELP
```

******** NOTE ********
IF THE I/O MODULES ARE NOT PROPERLY GROUPED INTO TYPES ACCORDING TO THE GENERAL FUNCTION OF THE MODULE A WARNING MESSAGE WILL BE TYPED.

THE OPERATOR NOW SELECTS THE TEST TO BE RUN. A CONTROL C WILL

RETURN YOU TO THE MONITOR AND STEP 4.2.

4.3 B - CHANGE BASE ADDRESS

THIS OPTION ALLOWS THE USER TO CHANGE THE DEFAULT BASE AND VECTOR ADDRESSES VIA THE KEYBOARD.
THE PROGRAM WILL TYPE:

''BASE ADDRESS NOW ==> 171000''
''ENTER NEW BASE ADDRESS : ''
''VECTOR ADDRESS NOW ==> 234 ''
''ENTER NEW VECTOR ADDRESS : ''

THE NEW BASE AND VECTOR ADDRESS MUST BE ENTERED.

4.4 S-SYSTEM TEST

THIS OPTION OF THE PROGRAM WILL MAP ALL THE ANALOG AND DIGITAL MODULES CONNECTED TO THE IOCM AND BUILD A TABLE OF THEM IN MEMORY. THEN IT WILL PICK UP EACH ONE AND RUN THE TESTS THAT DO NOT REQUIRE OPERATOR INTERVENTION. IF SWITCH 14 IN THE SWITCH REGISTER IS SET ,IT WILL LOOP CURRENT TEST UNTIL OPERATOR TYPES CONTROL C. AFTER THE SELECTED # OF PASSES, THE PROGRAM WILL PRINT PASS COUNT UNLESS SWITCH 13 IS SET.

4.5 M - MAP OF DBUS

THIS OPTION ALLOWS AN OPERATOR TO CHECK WHICH I/O MODULES ARE CONNECTED TO THE IOCM. IT CHECKS ALL ADDRESSES BETWEEN THE BASE ADDRESS AND THE CSR ADDRESS. FOR ADDRESSES THAT ANSWER , IT CHECKS THE GENERIC CODE AND TYPES THE ADDRESS AND INTERFACE TYPE.

4.6 A - TEST SPECIFIED MODULE

THIS TEST WILL EXERCISE THE MODULE AT THE ADDRESS SPECIFIED BY THE OPERATOR.
EXAMPLE:

'' WHICH TEST OPTION (H=HELP) ? A ''
'' ADDRESS OF MUT: 171000 ''

THE MODULE AT ADDRESS 171000 WILL BE TESTED AND THE PROGRAM WILL RETURN TO THE MONITOR WHEN DONE.

4.7 I - TEST IOCM

THIS TEST WILL EXERCISE ALL THE FEATURES OF THE IOCM. THE FOLLOWING IS A DESCRIPTION OF ALL THE TESTS. THIS TEST IS RUN AS MANY TIMES AS SELECTED AND THE PASS COUNT IS TYPED

1. CHECKS IF EACH BIT OF THE IOCM IS CLEARED BY THE CBIT
2. CHECKS IF THE INTERRUPT ENABLE BIT (EBIT) CAN BE SET AND CLEARED
3. CHECKS IF THE MAINTENANCE BIT (MBIT) CAN BE SET AND CLEARED
4. CHECKS IF THE DISABLE BIT (DBIT) CAN BE SET AND CLEARED.

CHECKS IF DBIT GENERATES CLEAR

5. CHECKS IF THE TEST BIT (TBIT) CAN BE SET AND CLEARED

6. CHECKS IF THE GENERIC CODE BIT (GBIT) CAN BE SET AND CLEARED

7. CHECKS IF THE RIF BIT (RBIT) CAN BE SET AND CLEARED

8. CHECKS DBUS DATA PATHS IN A MAINTENANCE MODE. IF THE MBIT IS SET AND THE CPU ADDRESSES ANY LOCATION BETWEEN 171000 AND 171375 IT SHOULD READ BACK THE LOWER BYTE OF THE MODULE ADDRESS.

9. CHECKS MAINTENANCE INTERRUPT. IF THE MBIT & EBIT ARE SET, THE IOCM WILL GENERATE AN INTERRUPT WITH VECTOR 234. THE IAR (171376) HAS THE LOWER BYTE OF CSR ADDRESS (377).

4.8 W -WRAP AROUND TEST

THIS OPTION ALLOWS THE FIELD ENGINEER TO CONNECT MODULE M6010 TO EITHER M5010 OR M5011. THEN IT WRAPS AROUND A SET OF DATA PATTERNS (125,252,377,0). IF SWITCH 14 IN THE SOFTWARE SWITCH REGISTER IS SET ,IT WILL LOOP ON THIS TEST UNTIL THE OPERATOR TYPES CONTROL C. EVERY SELECTED # OF PASSES IT WILL PRINT PASS COUNT UNLESS SWITCH 13 IS SET. ERRORS WILL BE PRINTED INDICATING GOOD DATA,BAD DATA AND THE PASS # AT WHICH IT OCCURRED.

4.9 F -FIELD TEST

THE PURPOSE OF THIS TEST IS TO OUTPUT ANY SELECTED DATA PATTERN TO AN OUTPUT MODULE SPECIFIED BY THE OPERATOR AND TO MONITOR DATA FROM AN INPUT MODULE AS SPECIFIED BY THE OPERATOR.

WARNING:
THE FIELD ENGINEER MUST REALIZE THAT THIS IS THE ONLY TEST THAT PERMITS HIM TO OUTPUT DATA TO THE CUSTOMER'S WIRING. THERE WILL BE VOLTAGE APPLIED TO CUSTOMER EQUIPMENT CONNECTED TO THE OUTPUT MODULE UNDER TEST. THEREFORE PRECAUTIONS MUST BE TAKEN THAT AN ERRONEOUS OUTPUT WILL NOT CAUSE DAMAGE TO THE FIELD EQUIPMENT AND THAT ALL TESTING IS CONDUCTED WITH THE CUSTOMERS KNOWLEDGE

PROCEDURE:
USER SELECTS THE ADDRESS OF THE MODULE UNDER TEST. IF IT IS AN OUTPUT MODULE THEN: USER SELECTS AND OUTPUTS A DATA PATTERN,ONE BYTE AT A TIME, TO THE MODULE.TYPE CONTROL/C TO ABORT TEST. AFTER ALL BYTES OF MODULE HAVE BEEN SELECTED, USER MUST TYPE CONTROL/C TO GO BACK TO MONITOR.
EXAMPLE:

ADDRESS OF MUT: 171XXX
DATA PATTERN FOR FIRST BYTE: YYY
(WHERE YYY IS A 8 BIT DATA PATTERN TO OUTPUT. ALSO, THE REQUEST FOR DATA FOR OUTPUT WILL BE REPEATED UP TO 4 TIMES DEPENDENT UPON MODULE TYPE)

IF IT IS AN INPUT MODULE THEN:

THE TEST MONITORS AND PRINTS DATA FROM MODULE INPUTS. IT PRINTS EVERY BYTE ADDRESS OF THE MODULE WITH ITS CONTENT DURING THE FIRST PASS AND IT CONTINUES TO MONITOR THE DATA WITHOUT PRINTING IT UNLESS A CHANGE IN THE DATA OCCURED. TYPE CONTROL/C TO RETURN TO MONITOR.

EXAMPLE:

ADDRESS OF MUT: 171XXX
171XXX : YYY
(WHERE YYY IS THE 8 BITS OF DATA READ FROM THE MODULE. THE SECOND LINE OF THE MESSAGE WILL BE REPEATED UP TO 4 TIMES, WITH INCREMENTING ADDRESS, DEPENDING UPON MODULE TYPE.

4.10 L - SET PASS COUNT

TYPE THE OCTAL NUMBER OF ITERATIONS FOR ALL TESTS.
DEFAULT NUMBER IS 1.
MAXIMUM NUMBER IS 177777

4.11 T - SET SOFTWARE SWITCH REGISTER (CPU WITHOUT HARDWARE SWITCH REGISTER)

TYPE THE OCTAL NUMBER TO BE LOADED TO SWREG

100000 - HALT ON ERROR
40000 - LOOP ON TEST
20000 - INHIBIT ERROR AND END OF PASS PRINT
1000 - LOOP ON ERROR

4.12 CONTROL C

TYPING ^C CAUSES THE PRESENT TEST TO BE ABORTED AND PROGRAM RETURNS TO THE MONITOR. IN SOME CASES IT MAY TAKE UP TO 20 SECONDS FOR THIS TO HAPPEN.

4.13 CONTROL A

IF THE MODULE HAS AN INDIVIDUAL MONITOR (A014, A630, A631) TYPING ^A DURING THE TEST EXECUTION RETURNS THE PROGRAM TO THE BEGINNING OF THIS MONITOR.

4.14 DELETE KEY

DO NOT USE THE DELETE KEY. THE RESULTS WILL BE UNPREDICTABLE.

5.0 A630 DIGITAL TO ANALOG CONVERTER (DAC)

5.1 COMMANDS

IF THE ADDRESS OF THE MODULE UNDER TEST (MUT) IS FOR THE A630 THE PROGRAM ENTERS A SUBMONITOR FOR THE A630. THE FOLLOWING TESTS CAN BE SELECTED.
A - CALIBRATION TEST
T - INTERNAL COMPARATOR TEST
D - D BIT TEST (LOGIC TEST)

5.2 D-BIT TEST

THE D-BIT TEST CHECKS THE GENERIC CODE AND LOGIC PORTIONS OF THE MODULE ONLY. IT DOES NOT EXERCISE ANY PORTION OF THE ANALOG CIRCUITRY. THEREFORE, THE T-BIT TEST MUST BE RUN IN CONJUNCTION WITH THE D-BIT TEST TO INSURE PROPER MODULE OPERATION.

5.3 T - INTERNAL COMPARATOR TEST

****
CAUTION: THIS TEST CAUSES VOLTAGES TO APPEAR ON THE MODULE'S OUTPUTS WHICH MAY BE DANGEROUS TO THE CUSTOMER'S PROCESS. ALSO, OVERLOADS OR SHORT CIRCUITS IN THE FIELD WIRING WOULD CAUSE A FAILURE OF THIS TEST. THEREFORE, THE I/O CABLE MUST BE REMOVED FROM THE MODULE UNDER TEST BEFORE PROCEEDING.
****

THE INTERNAL STATUS BIT TEST ACTIVATES THE ANALOG PORTION OF THE MODULE AND UTILIZES ANALOG COMPARATORS TO MONITOR AND REPORT THE RELATIVE LEVELS OF THE VOLTAGE OUTPUTS. THIS TEST DOES NOT MONITOR THE CURRENT OUTPUTS AND PROVIDES NO INDICATION OF THEIR FUNCTIONALITY OR CALIBRATION. FAILURE TO PASS THIS TEST INDICATES THAT THE MODULE IS EITHER DEFECTIVE OR SEVERELY OUT OF CALIBRATION. PASSAGE OF THIS TEST DOES NOT INDICATE THAT THE MODULE'S VOLTAGE OUTPUTS ARE NECESSARILY WITHIN CALIBRATION.

WHEN AN ERROR IS DETECTED, THE EXPECTED AND ACTUAL STATUS BITS ARE PRINTED BY THE DIAGNOSTIC. THEY ARE PRESENTED IN RIGHT JUSTIFIED FORM I.E. 0017 INDICATES THAT ALL BITS ARE SET. A FAILURE IN ANY ONE CHANNEL WILL USUALLY CAUSE TWO STATUS BITS TO BE IN ERROR AT DIFFERENT TIMES IN THE DIAGNOSTIC ACCORDING TO THE TABLE BELOW.

FAULTED CHANNEL	STATUS BITS AFFECTED
CH 0	S1 AND/OR S3
CH 1	S1 AND/OR S2
CH 2	S2 AND/OR S4
CH 3	S3 AND/OR S4

5.4 A - TEST OR CALIBRATION PROCEDURE

NORMALLY THE DAC MODULE WILL BE CALIBRATED AND SEALED AT THE TIME OF MANUFACTURE. FIELD RECALIBRATION SHOULD ONLY BE ATTEMPTED WHEN THE CUSTOMER WISHES TO CHANGE THE CURRENT OUTPUT OPTION FROM THE 4 TO 20MA RANGE TO THE 0 TO 20MA RANGE OR A MALFUNCTION IS SUSPECTED. (THE MODULE IS SHIPPED WITH THE 4 TO 20MA RANGE SELECTED). BEFORE ATTEMPTING RECALIBRATION, IT IS IMPORTANT TO BECOME FAMILIAR WITH THE LOCATION OF THE VARIOUS ADJUSTMENTS AND POINTS WHERE THE TEST EQUIPMENT WILL BE ATTACHED. FOR THIS INFORMATION REFER TO THE TABLE BELOW AND DRAWING D-UA-A630-0-0 OR FIGURES 6 THROUGH 10 AND TABLES 4 AND 5 OF THE A630

HARDWARE MANUAL.

	CHANNEL 0	1	2	3
VOLTAGE OFFSET ADJUST	R35	R57	R79	R101
CURRENT GAIN ADJUST	R36	R58	R80	R102
CURRENT OFFSET ADJUST	R44	R66	R88	R110
CURRENT RANGE SWITCHES*	E78-3,4	E78-1,2	E80-3,4	E80-1,2
BERG PINS: VOLTAGE OUT	5	13	37	47
GND	6	14	38	48
CURRENT OUT	7	15	39	49
GND	8	16	40	50
SCREW TERMINALS: VOLTAGE OUT	5	13	21	31
GND	6	14	22	32
CURRENT OUT	7	15	23	33
GND	8	16	24	34

WHEN CALIBRATING THE 4-20MA RANGE, THESE SWITCHES MUST BE 'OFF' FOR THE PARTICULAR CHANNEL BEING CALIBRATED. FOR THE 0-20MA RANGE, THESE SWITCHES MUST BE 'ON' FOR THE PARTICULAR CHANNEL BEING CALIBRATED. SWITCH E79-5 SHOULD BE 'ON' AT ALL TIMES. THE FOLLOWING EQUIPMENT IS NEEDED:

DVM WESTON SHLUMBERGER MODEL 443 OR EQUIVALENT

EXTENDER MODULE W904B

RESISTOR 500OHM, 0.01%, 0.3 WATT
REQUIRED FOR CURRENT CALIBRATION
DEC PART NO. 13-09985-00

ACCESS TO THE ADJUSTMENTS AND SWITCHES IS PERMITTED BY PUTTING THE DAC ON AN EXTENDER MODULE. CAUTION: THE SYSTEM MUST BE POWERED DOWN BEFORE INSERTING OR REMOVING ANY MODULE. ALSO, THE CUSTOMER'S FIELD WIRING MUST BE REMOVED FROM THE ASSOCIATED SCREW TERMINAL ASSEMBLY.

THE 'A' TEST OR CALIBRATION ROUTINE OF THE DIAGNOSTIC PROGRAM WILL SYSTEMATICALLY AND INTERACTIVELY LEAD THE OPERATOR THROUGH THE REQUIRED STEPS. THE SPECIFIC COURSE OF ACTION AT EACH STEP IS DETAILED BELOW. FOR

EACH ADJUSTMENT IT SHOULD FIRST BE DETERMINED IF THAT PARTICULAR ADJUSTMENT NEEDS TO BE MADE BEFORE BREAKING THE SEAL ON THE ADJUSTMENT. THE A TEST AND CHANNEL # SHOULD NOW BE SELECTED. COMPLETE CALIBRATION OF THE SELECTED CHANNEL WILL BE COMPLETED BEFORE CONSIDERING ANOTHER CHANNEL.

VOLTAGE REFERENCE ADJUST

THIS ADJUSTMENT IS COMMON TO ALL FOUR CHANNELS. IF THE REFERENCE HAS BEEN SET AND IS IN SPECIFICATION TYPE 'N' TO BYPASS THIS SECTION; OTHERWISE TYPE 'Y' TO PROCEED.

STEP 1: CONNECT THE - LEAD OF THE VOLTMETER TO EITHER PIN 6 OF THE BERG CONNECTOR OR PIN 6 OF THE SCREW TERMINAL ASSEMBLY AND THE + LEAD TO E62, PIN 14. THE READING SHOULD BE +10.240V +-2MV IF THE READING IS WITHIN SPECIFICATION TYPE CARRIAGE RETURN (CR), OTHERWISE PROCEED AS FOLLOWS:

TRY TO BRING THE REFERENCE WITHIN SPECIFICATION BY ADJUSTING ONLY R135 WITHOUT ALTERING SWITCHES E78 1-4. NORMALLY ADJUSTING R135 SHOULD BE SUFFICIENT. IF THE REFERENCE CAN BE ADJUSTED TYPE CARRIAGE RETURN (CR) OTHERWISE PROCEED AS FOLLOWS.

SET R135 FULLY CLOCKWISE. USE SWITCHES E79-1 THRU 4 IN ANY COMBINATION TO OBTAIN A READING AS CLOSE TO BUT LESS THAN +10.240 V AS POSSIBLE. THE EFFECTS OF THE SWITCHES ARE BINARILY WEIGHTED WITH E79-4 BEING THE LEAST SIGNIFICANT. FINISH THE ADJUSTMENT BY ADJUSTING R135 TO ACHIEVE A READING OF +10.240 V. WHEN DONE TYPE CARRIAGE RETURN (CR).

STEP 2: REMOVE THE + LEAD OF THE VOLTMETER AND CONNECT TO E62 PIN 13. THE READING SHOULD BE +5.120 V +- 2MV. IF THE READING IS IN SPECIFICATION TYPE CARRIAGE RETURN (CR), OTHERWISE ADJUST R134 FOR THE PROPER READING. WHEN DONE TYPE CARRIAGE RETURN (CR).

CHANNEL VOLTAGE OUTPUT ADJUST

REMOVE THE + LEAD OF THE VOLTMETER AND CONNECT TO THE SPECIFIED VOLTAGE OUTPUT TERMINAL ON THE SCREW TERMINAL ASSEMBLY (SEE TABLE ABOVE). REMOVE THE - LEAD OF THE VOLTMETER AND CONNECT IT TO THE SCREW TERMINAL JUST BELOW THE + LEAD. BEFORE MAKING ANY ADJUSTMENTS LOCATE THE VARIOUS ADJUSTMENTS RELATED TO THE SELECTED CHANNEL. FOR THIS INFORMATION REFER TO THE TABLE ABOVE AND DRAWING UA-A630-0 OR FIGURE 7 OF THE A630 HARDWARE MANUAL.

STEP 1: THE VOLTMETER READING SHOULD BE 0.000V +-5MV. IF THE READING IS IN SPECIFICATION TYPE CARRIAGE RETURN (CR), OTHERWISE ADJUST THE PROPER VOLTAGE OFFSET POTENTIONETER. WHEN DONE TYPE CARRIAGE RETURN (CR).

STEP 2: THE READING SHOULD CHANGE TO +10.230V +- 40MV. IF READING IS OUT OF SPECIFICATION REPLACE THE MODULE. THERE IS NO ADJUSTMENT FOR THIS STEP. TYPE CARRIAGE RETURN (CR).

CHANNEL CURRENT OUTPUT ADJUST

THE DIAGNOSTIC WILL ASK IF THE CURRENT OUTPUT IS TO BE CALIBRATED. IF NOT, TYPE 'N' TO BYPASS THE REST OF THE PROCEDURE. IF IT IS TO BE

CALIBRATED, TYPE 'Y' TO PROCEED. THE DIAGNOSTIC WILL THEN ASK IF THE 4-20MA RANGE IS TO BE CALIBRATED. IF IT IS, TYPE 'Y', CHECK TO SEE THAT SWITCH E79-5 IS ON AND BOTH OF THE SPECIFIED CURRENT RANGE SWITCHES FOR THAT CHANNEL ARE OFF. (SEE TABLE ABOVE) A 'N' ANSWER WILL SELECT THE 0-20MA RANGE. IN THIS CASE, TURN ON BOTH OF THE SPECIFIED CURRENT RANGE SWITCHES FOR THAT CHANNEL AND SWITCH E79-5.

NOTE: THE FOLLOWING PROCEDURE IS USED FOR BOTH CURRENT RANGES.

THE ADJUSTMENT OF THE CURRENT OUTPUT REQUIRES THE USE OF EITHER A 500OHM, 0.01% PRECISION RESISTOR (DEC PART NO. 13-09985-00) OR ACCURATE CURRENT RANGES ON THE VOLTMETER. IF THE RESISTOR IS AVAILABLE, CONNECT IT BETWEEN THE SPECIFIED CURRENT OUTPUT TERMINAL AND THE GROUND TERMINAL JUST BELOW ON THE SCREW TERMINAL ASSEMBLY. (SEE TABLE ABOVE) CONNECT THE VOLTMETER LEADS DIRECTLY ACROSS THE RESISTOR LEADS WITH THE + LEAD OF THE VOLTMETER CONNECTED TO THE SPECIFIED CURRENT OUTPUT TERMINAL. IF CURRENT RANGES ON THE VOLTMETER ARE TO BE USED IN LIEU OF THE RESISTOR, CONNECT THE METER DIRECTLY TO THE TWO TERMINALS SPECIFIED.
STEP 1: THE VOLTMETER READING SHOULD BE +10.00V +- 5MV OR 20.000MA +- 10MA. IF THE READING IS IN SPECIFICATION TYPE CARRIAGE RETURN (CR), OTHERWISE ADJUST THE PROPER CURRENT GAIN POTENTIOMETER. WHEN DONE TYPE CARRIAGE RETURN (CR).

STEP 2: THE VOLTMETER READING SHOULD CHANGE TO +2.000V +- 5MV/4.000MA +-10UA IF IN THE 4-20MA RANGE OR 9.8MV +-5MV/19.5UA + 10UA IF IN THE 0-20 MA RANGE. IF THE READING IS IN SPECIFICATION TYPE CARRIAGE RETURN (CR), OTHERWISE ADJUST THE PROPER CURRENT OFFSET POTENTIOMETER. WHEN DONE TYPE CARRIAGE RETURN (CR).

THE DIAGNOSTIC WILL ASK IF THE CALIBRATION HAS BEEN REVERIFIED AND THE ADJUSTMENT OF EITHER POTENTIOMETER WILL AFFECT THE OTHER. THIS IS NECESSARY BECAUSE STEPS 1 AND 2 ABOVE ARE INTEREACTIVE TYPE 'Y' ONLY IF STEPS 1 AND 2 CAN BE COMPLETED SEQUENTIALLY AND IN THAT ORDER WITHOUT ADJUSTING EITHER POTENTIOMETER. IF EITHER ADJUSTMENT WAS MADE ON THIS PASS, TYPE 'N'. THE DIAGNOSTIC WILL GO BACK TO STEP 1 OF THE CURRENT OUTPUT ADJUSTMENT. WHEN A 'Y' IS TYPED, CALIBRATION OF THAT CHANNEL IS COMPLETE AND THE DIAGNOSTIC RETURN TO ITS STARTING POINT.

6.0 A014 -ANALOG TO DIGITAL CONVERTER

6.1 COMMANDS

IF THE MODULE UNDER TEST (MUT) ADDRESS IS FOR A A014 THE PROGRAM WILL JUMP TO THIS SUBMONITOR. THE PURPOSE OF THIS PROGRAM IS TO PERFORM THE TESTS OF THE A014 WITH OPERATOR INTERVENTION.

THE OPERATOR CAN CHOOSE ONE OF THE FOLLOWING TESTS:

D - LOGIC TEST
C - CALIBRATE AND VERIFY OFFSET AND GAIN OF A014, A156 OR A157
M - CHECK IF CONVERTER SKIPS OR MISSES ANY OUTPUT CODE
(MONOTONICITY TEST)
X - TEST IF A/D WORKS WITH A156 OR A157 MUX

6.2 OPERATING PROCEDURES:
WHEN ANALOG MODULE TEST IS SELECTED IT PRINTS:
WHICH TEST (D,C,M,X OR H FOR HELP)?
OPERATOR MUST RUN THE FOLLOWING TESTS IN THIS ORDER AND SHOULD TYPE:

6.3 D - A/D LOGIC TEST
THE LOGIC TEST VERIFIES THE FUNCTIONALITY OF THE A014 A/D CONVERTER CONSISTING OF A MOTHER AND DAUGHTER BOARD AS FOLLOWS:
1. VERIFY ADDRESS RESPONSE
2. VERIFY BIT FUNCTIONALITY (ERROR BIT,DONE BIT, GO BIT)
3. VERIFY SINGLE ENDED MODE FUNCTIONALITY
   CHANNEL SELECTION
   GAIN SELECTION
4. VERIFY DIFFERENTIAL MODE FUNCTIONALITY
5. CHECK IF CONVERSION GETS DONE
6. VERIFY FUNCTIONS USING T-BIT
   SET T-BIT AT IOCM
   SELECT CHANNEL FOR +,-,OR 0 REFERENCE MAINTENANCE VOLTAGE
   VERIFY INDIVIDUAL OUTPUTS
   SET MAINTENANCE RAMP VOLTAGE
   VERIFY THAT RAMP GOES FROM +10.240V TO -10.240V
5. SET D-BIT AT IOCM
6. VERIFY ZERO (4000) OUTPUT
8. CLEAR T-BIT AND D-BIT
9. RETURN TO MONITOR

6.4 C- CALIBRATE AND VERIFY A/D CONVERTER

TO ENSURE COMPLIANCE WITH SPECIFICATIONS, THE VOLTAGE SOURCE USED IN CALIBRATING THE A014 MUST HAVE BEEN ACCURATELY CALIBRATED WITHIN THE PREVIOUS SIX MONTHS. THE USE OF AN ACCURATE CUSTOMER PROVIDED VOLTAGE REFERENCE IS RECOMMENDED WHEN AVAILABLE, AS DIFFERENCES BETWEEN THE CUSTOMER'S REFERENCE AND THAT USED FOR CALIBRATION COULD INDUCE AN ERROR DURING CUSTOMER USE. IN ADDITION THE PERSONNEL DOING THE CALIBRATION SHOULD BE FAMILIAR WITH THE PROCEDURES FOR ALIGNMENT OF PRECISION ANALOG EQUIPMENT. IF YOU ARE NOT SURE OF THE FOREGOING, DO NOT ATTEMPT TO CALIBRATE THE A014; ITS FACTORY CALIBRATION IS PROBABLY BETTER THEN YOU WILL BE ABLE TO ACCOMPLISH.

IN THIS TEST OPERATOR SHOULD CONNECT HIS VOLTAGE SOURCE TO THE INPUT OF THE A014,A156 OR A157 MUX ON THE SELECTED CHANNEL. THE PROGRAM WILL MAKE 8 CONVERSIONS ,CALCULATE THE AVERAGE VALUE OF THIS CONVERSIONS AND PRINT THE RESULT IN OCTAL FORMAT AND IN MILLIVOLTS. THE RESULTS WILL BE PRINTED EVERY 2 - 4 SEC.TO EXIT THIS TEST TYPE CONTROL A OR CONTROL C. IN THE CASE OF THE A157 OPERATOR MUST ALSO SELECT THE GAIN.

R41 ADJUSTS THE ZERO OF THE A014. THIS IS BEST DONE AT A CODE TRANSITION POINT, SO THAT MORE RESOLUTION THEN THE CONVERTER QUANTIZATION LEVEL CAN BE OBTAINED.

R31 IS THE GAIN ADJUSTMENT POT; IT ADJUSTS BOTH PLUS AND MINUS FULL SCALE SYMMETRICALLY ABOUT ZERO. AGAIN, BEST ADJUSTMENT RESOLUTION CAN BE OBTAINED AT A CODE TRANSITION POINT.

R41 SHOULD ALWAYS BE ADJUSTED FIRST, USING AN INPUT VERY CLOSE TO ZERO. R31 CAN BE ADJUSTED WITH AN INPUT CLOSE TO EITHER FULL SCALE OR TO OPTIMIZE ACCURACY FOR A PARTICULAR APPLICATION, AT ANY POINT IN THE INPUT RANGE SUFFICIENTLY FAR FROM ZERO. FOR GREATEST AVERAGE ACCURACY OF AN EXPANDED SUBSYSTEM, THE A014 SHOULD BE CALIBRATED WITH AN INPUT APPLIED THROUGH THE ON-BOARD MULTIPLEXOR; IF IT IS DESIRED TO OPTIMIZE ACCURACY THROUGH AN EXPANDER MULTIPLEXOR AT A SPECIFIC GAIN, THE CALIBRATION INPUTS SHOULD BE APPLIED THROUGH THAT MULTIPLEXOR. THE USER SHOULD RECOGNIZE THAT THIS MAY PLACE THE A014 ALONE OUT OF SPEC.

ZERO ADJUSTMENT -

1. APPLY -2.5 MV TO THE SCREW TERMINALS OF THE SELECTED CHANNEL AND START THE CALIBRATION TEST.
2. VERIFY THAT THE OCTAL PRINTOUT VARIES BETWEEN 003777 AND 004000. A 50 - 50 DISTRIBUTION WOULD BE IDEAL, BUT THIS IS NOT USUALLY OBTAINABLE. IF NECESSARY, ADJUST R41(LOWER POT) UNTIL THE PRINTOUT VARIES PER THE ABOVE. THIS ADJUSTS THE TRANSITION POINT FROM 0.0 MV TO -5.0 MV.(ONE LSB.)
3. APPLY +2.5 MV TO THE SAME CHANNEL AND VERIFY THAT THE PRINTOUT IS 004000 OR 004001. VARIANCE IS NOT REQUIRED, BUT IS ACCEPTABLE BETWEEN THESE TWO VALUES.

GAIN ADJUSTMENT -

1. APPLY A VOLTAGE OF +10.2325V (FULL SCALE MINUS HALF LSB) TO THE SELECTED CHANNEL AND START THE CALIBRATION TEST.
2. VERIFY THAT THE PRINTOUT VARIES BETWEEN 007776 AND 007777. IF NECESSARY ADJUST THE GAIN POT R31 UNTIL THE PROPER PRINTOUT IS OBTAINED.
3. APPLY -10.2375V TO THE SELECTED CHANNEL (FULL SCALE MINUS HALF LSB) AND RESTART THE CALIBRATION TEST IF NECESSARY.
4. VERIFY THAT THE OCTAL PRINTOUT IS EITHER 000001 OR 000000 OR VARYING BETWEEN THE TWO VALUES.

6.7 M - MONOTONICITY TEST

THE OBJECTIVE OF THIS TEST IS TO READ EVERY POSSIBLE STATE THRU THE RAMP AT LEAST ONCE STARTING FROM THE BOTTOM (-10.240),GOING UP THE RAMP (+10.240) AND THEN DOWN TO (-10.240). BEFORE RUNNING THIS TEST, THE LOGIC TEST MUST BE RUN.

1. COMPARES LAST CONVERSION WITH THE PREVIOUS ONE AT ALL POINTS.
2. CONTINUES CONVERSION WITH NEXT POINT IN RAMP IF ANY ERROR IS FOUND.
3. PRINT ERRORS AT THE END OF THE RAMP, IF LESS THAN 20 ERROS ARE FOUND. THE ERRORS OCCUR IF THE PROGRAM DETECTS ANY STATE THAT IS MISSING OR ANY NOISE THAT IS MORE THEN 2 BITS.

EXAMPLE: OUT OF RANGE BY TWO BITS

GDDAT	BDDAT	RAMP(UP=0,DOWN=1)	
0	2	0	GOING UP THE RAMP
7776	7774	1	GOING DOWN THE RAMP

4. PRINTS A MESSAGE "TOO MANY ERRORS" FOR 20 ERRORS OR MORE AND RETURN TO MONITOR.
5. PRINTS A MESSAGE 'END OF RAMP' FOR NO ERRORS FOUND.
6. OPERATOR TYPES ^CONTROL 'A' TO TERMINATE ERRORS PRINTOUTS AND RETURN TO MONITOR.

6.8 X - MULTIPLEXER TEST (A156, A157)

THE OPERATOR IS ASKED FIRST WHICH MUX HE WANTS TO TEST. THEN THE PROGRAM DETERMINES FROM THE GENERIC CODE IF IT IS A156 SINGLE ENDED, A156 DIFF MODE OR A157 AND IT CHECKS THE LOGIC OF THE SELECTED MUX.
THE TEST INCLUDES:
1. CHANNEL SELECT REGISTER
2. GAIN SELECT REGISTER
3. ERROR BIT FUNCTIONALITY
4. RESULTS OF THE CONVERSION WITH T BIT SET (4000)

7.0 A020 HIGH COMMON MODE ANALOG/DIGITAL CONVERTER

7.1 GENERAL

THE FOLLOWING COMMANDS ARE PERFORMED BY THIS PROGRAM.
L-LOGIC TEST
C-CALIBRATION ROUTINE
H-HELP FILE

7.2 OPERATING PROCEDURES

AFTER THE DIAGNOSTIC IS LOADED THE MONITOR HEADER LINE IS PRINTED AND A PROMPT OF (#) IS PRINTED. CONTROL MAY BE RETURNED TO THE MONITOR AT ANY TIME BY TYPING CONTROL C. CONTROL A IS USED AFTER A TEST HAS BEEN ENTERED AND YOU FIND A NEED TO RESTART THE TEST WITHOUT RETURNING TO THE MONITOR. ONE SUCH USE OF CONTROL A IS DURING THE CALIBRATION ROUTINE.

7.3 (L)-LOGIC TEST

THE TEST WILL VERIFY MOST OF THE LOGIC ON THE MOTHER BOARD. THE DAUGHTER BOARD DOES NOT HAVE TO BE PRESENT FOR THIS TEST TO FUNCTION. MOUNTING THE DAUGHTER BOARD WILL NOT INTERFERE WITH THE TEST.

TST1-ALL FOUR ADDRESSES ARE READ TO CHECK THAT TIMEOUT DOES NOT OCCUR.

TST2-THE FIRST, SECOND, AND FOURTH ADDRESS ARE WRITTEN TO 1'S TO CHECK THAT TIMEOUT WILL OCCUR. THEN THE THIRD ADDRESS IS WRITTEN AND CHECKED THAT NO TIMEOUT OCCURS.

TST3-THE FOURTH ADDRESS IS CHECKED TO BE CLEARED BY THE C BIT. ALSO THE CSR IS CHECKED TO BE CLEARED.

TST4-THE G BIT IS SET AND EACH OF THE FOUR ADDRESSES ARE READ. THE GENERIC CODE OF 324 OR 304 IS EXPECTED FROM EACH ADDRESS.

TST5-DATA (NOT = 0) IS LOADED INTO THE CONVERTED CHANNEL BITS(4-7) OF THE THIRD ADDRESS. THIS SHOULD CAUSE THE ERROR BIT (BIT 2) OF THE FOURTH ADDRESS TO SET.

TST6-WITH THE ERROR BIT IS SET THE INTERRUPT SYSTEM IS TURNED ON. AN INTERRUPT SHOULD OCCUR, CAUSING THE IAR TO CONTAIN THE DEVICE ADDRESS. THE C BIT IS THEN SET AND THE ERROR AND THE INTERRUPTS ARE CHECKED TO BE CLEARED.

TST7-THE ERROR BIT IS SET, THEN THE R BIT IS SET TO CLEAR THE ERROR BIT. THE INTERRUPT SYSTEM IS THEN SET UP AND CLEARED BY SETTING THE R BIT.

TST10-THIS TEST CHECKS THAT THE CHANNEL NUMBER JUST WRITTEN CAN BE READ BACK IMMEDIATELY. THE CHANNEL IS LOADED INTO BITS 0-3 AND READ BACK FROM BITS 0-3 OF ADDRESS TWO.

TST11-THIS TEST OVERLOADS THE NUMBER OF CHANNELS INPUT AND CHECKS FOR THE ERROR BIT BEING SET. THE C BIT IS THEN SET, AND THE OVERLOAD CONDITION IS CHECKED TO BE CLEARED.

TST12-THIS TEST CHECKS THAT THE CHANNEL NUMBER WRITTEN CAN BE READ IN THE CONVERTED DATA CHANNEL BITS (4-7) OF ADDRESS THREE AFTER A CONVERSION USING DUMB MODE.

TST13-THIS TEST CHECKS THAT THE CHANNEL NUMBERS WRITTEN ARE READ BACK IN THE CONVERTED DATA CHANNEL BITS (4-7) AFTER A CONVERSION IS DONE USING SMART MODE. THIS TEST USES THE INTERRUPT SYSTEM TO GATHER THE DATA.

TST14-THIS TEST CHECKS FOR A DATA OVER RUN CONDITION BY LOADING TWO CHANNELS AND NOT READING THE FIRST CHANNEL LOADED UNTIL THE SECOND ONE HAS BEEN CONVERTED.

7.4 CALIBRATION ROUTINE

7.4.1 EQUIPMENT NEEDED

1. PRECISION VOLTAGE SOURCE, EDC(MODEL VS-11N OR EQUIVALENT)
2. EXTENDER MODULE W904B
3. I/O SUBSYSTEM USER'S GUIDE (EK-OPIOS-UG-00X)
4. A020 PRINT SET

7.4.2 MODULE SETUP

A. OBTAIN THE FOLLOWING FUNCTION REQUIREMENTS FROM THE CUSTOMER. REFERENCE USERS GUIDE FOR SWITCH LOCATIONS AND SETTINGS.
B. FUNCTIONS
   1. ADDRESS SELECT: E60-1 THRU E60-6
   2. 50/60 HERTZ: (E60-7 AND E60-8) OR (E60-9 AND E60-10)
   3. ATR16 POWER SELECT: E60-10
   4. 2 OR 3 WIRE MODE SELECT: E6-5
   5. SHEILD SELECT: E12-9
   6. INTERNAL FILTER SELECT: E12-10
   7. CHANNEL FULL SCALE RANGE AND BOUNDARY: E12-1 THRU E12-8, E13-1 THRU E13-10, E6-1 THRU E6-4

7.4.3 RESTRICTIONS

A. ALLOW ONE HALF HOUR FOR THE A020 AND EDC TO WARM UP TO OPERATING TEMPERATURE.
B. IF A PRECISION VOLTAGE SOURCE IS TO BE USED THROUGH TP1 AND TP2 REMOVE CUSTOMER CABLE.

7.4.4 OFFSET ADJUSTMENTS

A. SELECT THE CALIBRATION/MANUAL TEST FROM THE A020 DIAGNOSTIC UNDER CVPCAX.
B. SELECT TP1 AND TP2 USAGE.
C. USING A JUMPER CLIP, SHORT TP1 AND TP2 ON THE DAUGHTER BOARD (REFER TO THE USERS GUIDE FOR LOCATION).
D. SELECT A CHANNEL AT THE HIGHEST CUSTOMER SELECTED GAIN (LOWEST FULL SCALE RANGE).
E. ADJUST THE OFFSET POT (R17-LABELED EOS) UNTIL THE TERMINAL READOUT IS 000000 +/- THE FOLLOWING TOLERANCES:

FULL SCALE RANGE	TOLERANCE IN LSB'S
10MV	+/- 4 LSB
20MV	+/- 2 LSB
50MV AND UP	+/- 1 LSB

NOTE:LSB VALUES FOR EACH FULL SCALE RANGE ARE SHOWN IN TABLE3 IN THE PIOS USERS GUIDE.

7.4.5 GAIN ADJUSTMENTS

A. INSURE THAT THE A020 IS NOT DOING A CONVERSION BY TYPING CONTROL A IF ALREADY RUNNING THE CALIBRATION/MANUAL TEST.
B. CONNECT EDC OR CUSTOMERS VOLTAGE SOURCE TO TP1 AND TP2. CONNECT HIGH OR (+) TO TP1 AND LOW OR (-) TO TP2.
   NOTE:USE CUSTOMER PRECISION SOURCE WHENEVER POSSIBLE.
C. IF TWO GAINS ARE TO BE USED CALIBRATE A CHANNEL UNDER GAIN A (USING GAIN A POT R19), THEN A CHANNEL UNDER GAIN B (USING GAIN B POT R18). IF ONLY ONE GAIN IS TO BE USED, CALIBRATE ANY CHANNEL WHICH WILL BE UNDER GAIN B (USING GAIN B POT).

GAIN CALIBRATION SUMMARY:

1. SELECT THE A020 CALIBRATION/MANUAL TEST IF NOT ALREADY RUNNING.
2. SELECT TP1 AND TP2
3. DISCONNECT CUSTOMER CABLE
4. ENTER VALUE FROM PRINTED TABLE FOR RANGE REQUIRED
5. CONNECT VOLTAGE SOURCE TO MODULE
6. SET SOURCE TO VOLTAGE INDICATED ON TERMINAL
7. ADJUST APPROPRIATE POT WHILE OBSERVING THE CONSOLE PRINTOUT.
8. THE ACCEPTED TOLERANCES FOR EACH OF THE FULL SCALE RANGES IS AS FOLLOWS:
   10MV +/- 4 LSB
   20MV +/- 2 LSB
   50MV +/- 1 LSB
9. THE CONSOLE BELL WILL SOUND WHEN THE CONVERSION VALUE IS WITHIN THE WINDOW SPECIFIED BY THE TOLERANCES. THIS WILL ASSIST DOING THE CALIBRATION AT LOCATIONS AWAY FROM THE CONSOLE.

8.0 A631 12 BIT DIGITAL TO ANALOG CONVERTER

8.1 COMMANDS

IF THE ADDRESS OF THE MODULE UNDER TEST (MUT) IS FOR THE A631 THE PROGRAM ENTERS A SUBMONITOR FOR THE A631. AFTER THE HEADER IS PRINTED THE PROMPT "A631:" WILL BE DISPLAYED. THE FOLLOWING COMMANDS CAN BE ISSUED.

H - HELP
M - MANUAL OPERATION *
S - SCOPE ROUTINE *
C - CALIBRATION PROCEDURE *

* NOTE: VOLTAGES OR CURRENTS WILL APPEAR ON THE DAC'S OUTPUT TERMINALS. INSURE THAT THE CUSTOMER'S EQUIPMENT IS DISCONNECTED.

8.2 L - LOGIC TEST

THE LOGIC TEST CHECKS THAT THE 8 ADDRESSES CAN BE READ AND WRITTEN, THAT A DATA PATTERN CAN BE ROTATED THRU THE RAM MEMORY (WITHOUT ERROR) AND THAT THE CORRECT GENERIC CODE (262) CAN BE READ FROM THE 8 ADDRESSES. THERE IS NO CHECKING DONE ON THE ANALOG CIRCUITRY.

8.3 M - MANUAL OPERATION

THIS ROUTINE ALLOWS THE SETTING OF ANY DAC (0-3) TO ANY VALUE IN THE 0-7777 (OCTAL) RANGE.
A NEW VALUE CAN BE ENTERED OR A <CR> CAN BE ENTERED WHICH WILL LEAVE THE DAC AT ITS PRESENT VALUE AND STEP YOU TO THE NEXT DAC.

```
EXAMPLE:
DAC0 [OCTAL]:7777 0      SETS DAC0 = 0000
DAC1 [OCTAL]:7777 <CR>   LEAVES DAC1 = 7777
DAC2 [OCTAL]:7777 0001   SETS DAC2 = 0001
DAC3 [OCTAL]:7777 2000   SETS DAC3 = 2000
```

EXIT THIS ROUTINE WITH A CONTROL"A" OR CONTROL"C".

8.4 S - SCOPE ROUTINE

THIS ROUTINE WILL GENERATE A USER DEFINED STAIR-STEP PATTERN ON THE DAC'S OUTPUT. THE PARAMETERS WILL BE DEFINED BY A KEYBOARD DIALOG.

```
                                        -------UPPER LIMIT
                                               MAX=7777
                                 --------
                                  <== STEP SIZE (1-7776)
                          --------

                  --------

        -------- LOWER LIMIT (0-7776)
```

EXAMPLE:

```
DAC0                            DAC 0 (VOLTAGE MODE)
PARAMETERS[Y/N]:Y               (Y) WILL REQUEST PARAMETERS
                                (N) WILL LEAVE PARAMETERS
                                    UNCHANGED & SKIP TO NEXT
                                    DAC
LOWER LIMIT(OCTAL):3000         SET TO 3000 OR 3.84V
UPPER LIMIT(OCTAL):4000         SET TO 4000 OR 5.12V
STEP SIZE(OCTAL):200            EACH STEP = .32V

THE PROGRAM WILL ASK PARAMETERS FOR DAC(0-3) THEN PRINT

EXECUTE[Y/N] ? Y                (Y) WILL START THE PATTERNS
                                (N) WILL TAKE YOU BACK TO
                                    DAC(0) PARAMETERS
```

THE PROGRAM WILL CONTINUALLY LOAD THE DAC'S UNTIL CONTROL''A'' OR CONTROL ''C''. IF THE STEP SIZE IS SUFFICENTLY LARGE, THE ABOVE PATTERN CAN BE OBSERVED. OTHERWISE, A RAMP WILL BE SEEN WITH A SCOPE.

8.5 C - CALIBRATION PROCEDURE

BEFORE BEGINNING CALIBRATION OF THE A631, THE FOLLOWING EQUIPMENT IS NEEDED:

PRECISION VOLTAGE SOURCE EDC(MODEL VS-11N OR EQUIVALENT)
RESISTOR 500 OHM, 0.01%, 0.3 WATTS
REQUIRED FOR CURRENT CALIBRATION
DEC PART # 13-09985-00
A631 PRINT SET

THE PROGRAM WILL LEAD YOU THROUGH THE REQUIRED STEPS OF SETTING E66 (VOLTAGE OR CURRENT MODE), SCREW TERMINAL OR BERG CONNECTOR NUMBERS AND TRIMPOT NUMBER.
EXIT THE ROUTINE WITH CONTROL ''A'' OR CONTROL ''C''.

9.0 TEST DESCRIPTIONS

M5010:

1. SET D BIT
2. CHECK THAT ALL BITS ARE ZERO
3. SET T BIT
4. CHECK THAT ALL BITS ARE ONES
5. SET C BIT

M5011 & M5031:

* NOTE: IF THE INTERRUPT SWITCH IS OFF AN ERROR WILL BE RETURNED. IF RUNNING UNDER APT THE ERROR WILL BE RETURNED ON THE FIRST PASS ONLY.

1. SET DBIT AND TBIT

2. CHECK IF INPUTS ARE ALL ONES
3. CLEAR TBIT
4. CHECK IF INPUTS ARE ALL ZEROS
5. CLEAR ALL INTERRUPTS
6. SET TBIT
7. ENABLE INTERRUPT
8. CHECK IF INTERRUPT OCCURRED
   /AN ERROR WILL OCCUR HERE IF INTERRUPTS
   /ARE DISABLED BY A SWITCH SETTING ON THE MODULE
9. CHECK IF INPUTS ARE ALL ONES
10. CHECK IF COS REGISTERS ARE ALL ONES
11. SET RIF BIT
12. CLEAR ALL COS REGISTERS
13. CHECK IF THEY ARE CLEAR
14. CLEAR T BIT
15. CHECK IF INTERRUPT OCCURED
16. CHECK IF ALL INPUTS ARE ZERO
17. RUN STEP 10 TO 13
18. CLEAR ALL INTERRUPTS

M5012-M5012YA:

* NOTE: IF THE INTERRUPT SWITCH IS OFF AN ERROR WILL BE RETURNED.
  IF RUNNING UNDER APT THE ERROR WILL BE RETURNED ON THE
  FIRST PASS ONLY.

1. SET DBIT
2. CHECK IF ALL BITS ARE ZERO
3. SET TBIT
4. CHECK IF ALL BITS ARE ONE
5. CLEAR T BIT
6. CLEAR ALL INTERRUPTS
7. SET T BIT
8. ENABLE INTERRUPT
9. CHECK IF INTERRUPT OCCURRED
   /AN ERROR WILL OCCUR HERE IF INTERRUPTS
   /ARE DISABLED BY A SWITCH SETTING ON THE MODULE
10. SET RIF BIT
11. CLEAR INTERRUPT
12. CHECK IF INTERRUPTS ARE CLEAR
13. CLEAR T BIT
14. RUN STEP 9 TO 12
15. CLEAR ALL INTERRUPTS

M5013:

* NOTE: IF THE INTERRUPT SWITCH IS OFF AN ERROR WILL BE RETURNED.
  IF RUNNING UNDER APT THE ERROR WILL BE RETURNED ON THE
  FIRST PASS ONLY.

1. SET DBIT
2. CHECK IF ALL BITS ARE ZERO
3. SET TBIT

4. TEST IF ALL BITS ARE ONE
5. CLEAR T BIT
6. CLEAR ALL INTERRUPTS
7. SET T BIT
8. ENABLE INTERRUPT
9. CHECK IF INTERRUPT OCCURED
   /AN ERROR WILL OCCUR HERE IF INTERRUPTS
   /ARE DISABLED BY A SWITCH SETTING ON THE MODULE
10. SET RIF BIT
11. CLEAR INTERRUPT
12. CHECK IF INTERRUPT IS CLEAR
13. CLEAR T BIT
14. RUN STEP 9 TO 12
15. CLEAR ALL INTERRUPTS

M5014:

* NOTE: THE LINE CLOCK MUST BE ENABLED

1. CLEAR MODULE
2. CHECK THAT COUNTERS ARE CLEAR
3. CHECK READ/WRITE OF COUNTERS
4. CHECK THAT T BIT INITIALIZES MODULE
5. CHECK INTERRUPTS ON COUNTER A
6. CHECK THAT INTERRUPT CLEARS
7. CHECK THAT COUNTER HALTS AFTER INTERRUPT
8. DO 5 THRU 7 FOR COUNTER B
9. CHECK TRANSITION POINTS
10. CHECK THAT COUNTERS COUNT AT 2KHZ
11. CHECK THAT A 10 SECOND TIME BASE IS
    CHOSEN FOR INTERNAL TEST
12. CHECK THAT C BIT CLEARS MODULE .

M5016:

* NOTE: IF THE INTERRUPT SWITCH IS OFF AN ERROR WILL BE RETURNED.
  IF RUNNING UNDER APT THE ERROR WILL BE RETURNED ON THE
  FIRST PASS ONLY.

1. CLEAR MODULE WITH CBIT
2. CHECK THAT STATUS REGISTOR IS CLEAR
3. CHECK THAT ALL COUNTERS ARE CLEAR
4. TEST STATUS REGISTER LOAD AND READ
5. TEST THAT CLEAR ENABLE BIT SELF CLEARS
6. CHECK THAT STATUS REGISTER CLEAR BIT
   CLEARS EACH COUNTER INDIVIDUALLY

STORE DIP SWITCH SETTINGS FOR THE FOLLOWING
AND PRINT THE SETTINGS IF NOT IN APT MODE

7. CHECK THAT COUNTERS COUNT BINARILY UPWARD
8. CHECK OVERFLOW
9. DETERMINE IF LEGAL RADIX
10. CHECK OVERFLOW CLEARS
11. CHECK INDIVIDUAL COUNTER INTERRUPTS
12. CHECK THAT CBIT CLEARS MODULE

M5031: SEE M5011.

M6010-M6010YA:

1. SET D BIT
2. CLEAR ALL 4 BYTES OF I/O REGISTER
3. SET DATA PATTERN 125 IN FIRST BYTE
4. CHECK IF IT IS SET
5. CHECK IF OTHER BYTES ARE ZERO
6. DO THE SAME WITH DATA 252 , 377 , 000
7. DO THE SAME WITH OTHER BYTES

M6011:

* NOTE: THE LINE CLOCK MUST BE ENABLED

1. SET D BIT
2. SET DATA PATTERN 252 AT MUT
3. CHECK IF IT IS SET
4. WAIT UP TO 10 SEC
5. DATA SHOULD BE CLEAR
6. REPEAT STEP 2 TO 5 WITH DATA EQUAL TO 125

M6012:
M6013:

1. SET D BIT
2. SET OUTPUT REGISTER TO FOLLOWING DATA PATTERN
   0,377,252,125
3. CHECK IF IT IS SET

M6014:

* NOTE: THE LINE CLOCK MUST BE ENABLED

1. CLEAR MODULE WITH CBIT
2. CHECK THAT COUNTERS ARE CLEAR
3. CHECK READ/WRITE WITH PATTERNS
4. CHECK THAT T BIT INITIALIZES MODULE
5. CHECK THAT COUNTERS INTERRUPTS
6. CHECK THAT INTERRUPT CLEARS
7. CHECK TRANSITION POINTS
8. CHECK THAT COUNTERS COUNT A 2KHZ
9. CHECK THAT C BIT CLEARS MODULE

M6015:

1. SET D BIT
2. CLEAR THE TWO BYTE REGISTER
3. SET DATA PATTERN 125 IN FIRST BYTE
4. CHECK IF IT IS SET
5. CHECK IF OTHER BYTE IS ZERO
6. DO THE SAME WITH DATA 252 , 377 , 000
7. DO THE SAME WITH OTHER BYTE
8. VERIFY C-BIT IS CLEARING MODULE

```
                                        .SBTTL  DOCUMENT
                        123400          $SWR=123400
                        000001          $TN=1
                                        .SBTTL  TRAP CATCHER
                        000000                  .=0
                                        ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
                                        ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
                                        ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
                        000174                  .=174
                000174  000000          DISPREG: .WORD  0               ;;SOFTWARE DISPLAY REGISTER
                000176  000000          SWREG:   .WORD  0               ;;SOFTWARE SWITCH REGISTER
                                        .SBTTL  STARTING ADDRESS(ES)
                000200  000137  006360          JMP     @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
                        000100          .=100
                000100  015400          NOCLK
                        000102          .=102
                000102  000340          340
                        000042          .=42
                000042  000000                  HALT
                        000046          .-46
                000046  011722                  LOGIC
                        000204          .-204
                000204  000137  007002          JMP     SETCLK
                                        .SBTTL  BASIC DEFINITIONS
                                        ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
                        001100          STACK=  1100
                        104000          ERROR   =EMT
                        000004          SCOPE   =       IOT
                                        ;*MISCELLANEOUS DEFINITIONS
000011                  HT=     11              ;;CODE FOR HORIZONTAL TAB
000012                  LF=     12              ;;CODE FOR LINE FEED
000015                  CR=     15              ;;CODE FOR CARRIAGE RETURN
000200                  CRLF=   200             ;;CODE FOR CARRIAGE RETURN-LINE FEED
177776                  PS=     177776          ;;PROCESSOR STATUS WORD
177776                  PSW     =       PS
177774                  STKLMT= 177774          ;;STACK LIMIT REGISTER
177772                  PIRQ=   177772          ;;PROGRAM INTERRUPT REQUEST REGISTER
177570                  DSWR=   177570          ;;HARDWARE SWITCH REGISTER
177570                  DDISP=  177570          ;;HARDWARE DISPLAY REGISTER
                        ;*GENERAL PURPOSE REGISTER DEFINITIONS
000000                  R0=     %0              ;;GENERAL REGISTER
000001                  R1=     %1              ;;GENERAL REGISTER
000002                  R2=     %2              ;;GENERAL REGISTER
000003                  R3=     %3              ;;GENERAL REGISTER
000004                  R4=     %4              ;;GENERAL REGISTER
000005                  R5=     %5              ;;GENERAL REGISTER
000006                  R6=     %6              ;;GENERAL REGISTER
000007                  R7=     %7              ;;GENERAL REGISTER
000006                  SP=     %6              ;;STACK POINTER
000007                  PC=     %7              ;;PROGRAM COUNTER
                        ;*PRIORITY LEVEL DEFINITIONS
000000                  PR0=    0               ;;PRIORITY LEVEL 0
000040                  PR1=    40              ;;PRIORITY LEVEL 1
000100                  PR2=    100             ;;PRIORITY LEVEL 2
000140                  PR3=    140             ;;PRIORITY LEVEL 3
000200                  PR4=    200             ;;PRIORITY LEVEL 4
000240                  PR5=    240             ;;PRIORITY LEVEL 5
```

```
000300                  PR6=    300             ;;PRIORITY LEVEL 6
000340                  PR7=    340             ;;PRIORITY LEVEL 7
                        ;*''SWITCH REGISTER'' SWITCH DEFINITIONS
100000                  SW15=   100000
040000                  SW14=   40000
020000                  SW13=   20000
010000                  SW12=   10000
004000                  SW11=   4000
002000                  SW10=   2000
001000                  SW09=   1000
000400                  SW08=   400
000200                  SW07=   200
000100                  SW06=   100
000040                  SW05=   40
000020                  SW04=   20
000010                  SW03=   10
000004                  SW02=   4
000002                  SW01=   2
000001                  SW00=   1
001000                  SW9             SW09
000400                  SW8             SW08
000200                  SW7             SW07
000100                  SW6             SW06
000040                  SW5             SW05
000020                  SW4     =       SW04
000010                  SW3             SW03
000004                  SW2     =       SW02
000002                  SW1     =       SW01
000001                  SW0     -       SW00
                        ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
100000                  BIT15-  100000
040000                  BIT14=  40000
020000                  BIT13=  20000
010000                  BIT12=  10000
004000                  BIT11=  4000
002000                  BIT10=  2000
001000                  BIT09=  1000
000400                  BIT08=  400
000200                  BIT07=  200
000100                  BIT06=  100
000040                  BIT05=  40
000020                  BIT04=  20
000010                  BIT03=  10
000004                  BIT02=  4
000002                  BIT01=  2
000001                  BIT00-  1
001000                  BIT9    =       BIT09
000400                  BIT8    =       BIT08
000200                  BIT7    =       BIT07
000100                  BIT6    =       BIT06
000040                  BIT5    -       BIT05
000020                  BIT4    -       BIT04
000010                  BIT3    =       BIT03
000004                  BIT2    =       BIT02
000002                  BIT1    =       BIT01
000001                  BIT0    =       BIT00
                        ;*BASIC ''CPU'' TRAP VECTOR ADDRESSES
```

```
                000004                  ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
                000010                  RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
                000014                  TBITVEC=14              ;;"T" BIT
                000014                  TRTVEC= 14              ;;TRACE TRAP
                000014                  BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
                000020                  IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
                000024                  PWRVEC= 24              ;;POWER FAIL
                000030                  EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
                000034                  TRAPVEC=34              ;;"TRAP" TRAP
                000060                  TKVEC=  60              ;;TTY KEYBOARD VECTOR
                000064                  TPVEC=  64              ;;TTY PRINTER VECTOR
                000240                  PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR
   1175                                 .SBTTL  BIT DEFINITIONS
   1176         000200                  FBIT=BIT7                       ;FLAG BIT
   1177         000100                  EBIT=BIT6                       ;INTERRUPT ENABLE
   1178         000040                  MBIT=BIT5                       ;MAINTENANCE INTERRUPT
   1179         000020                  DBIT=BIT4                       ;I/O DISABLE BIT
   1180         000010                  TBIT=BIT3                       ;TEST BIT, INVERTS I/O BITS
   1181         000004                  GBIT=BIT2                       ;GENERIC CODE ENEBLE
   1182         000002                  CBIT=BIT1                       ;CLEAR I/O
   1183         000001                  RBIT=BIT0                       ;RIF BIT, CLEARS I/O INTERRJPT
   1184
   1191
   1199
   1206
   1214
   1220
   1248
   1253
   1257
   1267
   1281
   1294
   1304
   1305         001100                  .=1100
   1306                                 .SBTTL  APT PARAMETER BLOCK
                                        ;;*****************************************************************
                                        ;
                                        ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
                                        ;;*****************************************************************
                001100                          .$X=.   ;;SAVE CURRENT LOCATION
                000024                          .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
        000024  000200                          200     ;;FOR APT START UP
                000044                          .-44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
        000044  001100                          $APTHDR ;;POINT TO APT HEADER BLOCK
                001100                          . .$X   ;;RESET LOCATION COUNTER
                                        ;;*****************************************************************
                                        ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
                                        ;INTERFACE SPEC.
        001100                          $APTHD:
        001100  000000                  $HIBTS: .WORD   0       ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
        001102  001206                  $MBADR: .WORD   $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
        001104  000120                  $TSTM:  .WORD   120     ;;RUN TIM OF LONGEST TEST
        001106  000600                  $PASTM: .WORD   600     ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
        001110  000600                  $UNITM: .WORD   600     ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
        001112  000124                          .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```

```
1648

                                .SBTTL  COMMON TAGS
                                ;;**********************************************************************
                                ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
                                ;*USED IN THE PROGRAM.
                 001114                 .=1114
001114                          $CMTAG:                         ;;START OF COMMON TAGS
001114  000000                          .WORD   0
001116     000                  $TSTNM: .BYTE   0               ;;CONTAINS THE TEST NUMBER
001117     000                  $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
001120  000000                  $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
001122  000000                  $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP ADDRESS
001124  000000                  $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
001126  000000                  $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
001130     000                  $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
001131     001                  $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
001132  000000                  $ERRPC: .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001134  000000                  $GDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'GOOD' DATA
001136  000000                  $BDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
001140  000000                  $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
001142  000000                  $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
001144  000000                          .WORD   0               ;;RESERVED--NOT TO BE USED
001146  000000                          .WORD   0
001150     000                  $AUTOB: .BYTE   0               ;;AUTOMATIC MODE INDICATOR
001151     000                  $INTAG: .BYTE   0               ;;INTERRUPT MODE INDICATOR
001152  000000                          .WORD   0
001154  177570                  SWR:    .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
001156  177570                  DISPLAY: .WORD  DDISP           ;;ADDRESS OF DISPLAY REGISTER
001160  177560                  $TKS:   177560                  ;;TTY KBD STATUS
001162  177562                  $TKB:   177562                  ;;TTY KBD BUFFER
001164  177564                  $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
001166  177566                  $TPB:   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
001170     000                  $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
001171     002                  $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001172     012                  $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A "LINE FEED"
001173     000                  $TPFLG: .BYTE   0               ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0-YES)
001174  000000                  $ESCAPE:0                       ;;ESCAPE ON ERROR ADDRESS
001176     207      377     377 $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
001201     000
001202     077                  $QUES:  .ASCII  /?/             ;;QUESTION MARK
001203     015                  $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
001204     012      000         $LF:    .ASCIZ  <12>            ;;LINE FEED
                                ;;**********************************************************************
                                .SBTTL  APT MAILBOX-ETABLE
                                ;;**********************************************************************
                                .EVEN
001206                          $MAIL:                          ;;APT MAILBOX
001206  000000                  $MSGTY: .WORD   AMSGTY          ;;MESSAGE TYPE CODE
001210  000000                  $FATAL: .WORD   AFATAL          ;;FATAL ERROR NUMBER
001212  000000                  $TESTN: .WORD   ATESTN          ;;TEST NUMBER
001214  000000                  $PASS:  .WORD   APASS           ;;PASS COUNT
001216  000000                  $DEVCT: .WORD   ADEVCT          ;;DEVICE COUNT
001220  000000                  $UNIT:  .WORD   AUNIT           ;;I/O UNIT NUMBER
001222  000000                  $MSGAD: .WORD   AMSGAD          ;;MESSAGE ADDRESS
001224  000000                  $MSGLG. .WORD   AMSGLG          ;;MESSAGE LENGTH
001226                          $ETABLE:                        ;;APT ENVIRONMENT TABLE
001226     000                  $ENV:   .BYTE   AENV            ;;ENVIRONMENT BYTE
001227     000                  $ENVM:  .BYTE   AENVM           ;;ENVIRONMENT MODE BITS
```

APT MAILBOX-ETABLE

```
001230  000000                  $SWREG: .WORD   ASWREG  ;;APT SWITCH REGISTER
001232  000000                  $USWR:  .WORD   AUSWR   ;;USER SWITCHES
001234  000000                  $CPUOP: .WORD   ACPUOP  ;;CPU TYPE,OPTIONS
                                ;*                      BITS 15-11=CPU TYPE
                                ;*                              11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
                                ;*                              11/70=06,PDQ=07,Q=10
                                ;*                      BIT 10=REAL TIME CLOCK
                                ;*                      BIT  9=FLOATING POINT PROCESSOR
                                ;*                      BIT  8=MEMORY MANAGEMENT
001236                          $APTBUF: .BLKB 81.      ;;80 CHARACTER BUFFER FOR APTCOM
                                .EVEN
001360  000000                  $APTPTR: .WORD          ;; BUFFER POINTER FOR APTCOM
001362     000                  $MAMS1: .BYTE   AMAMS1  ;;HIGH ADDRESS,M.S. BYTE
001363     000                  $MTYP1: .BYTE   AMTYP1  ;;MEM. TYPE,BLK#1
                                ;*                      MEM.TYPE BYTE   --  (HIGH BYTE)
                                ;*                              900 NSEC CORE=001
                                ;*                              300 NSEC BIPOLAR=002
                                ;*                              500 NSEC MOS=003
001364  000000                  $MADR1: .WORD   AMADR1  ;;HIGH ADDRESS,BLK#1
                                ;*                        MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF ''TYPE'' ABOVE
001366     000                  $MAMS2: .BYTE   AMAMS2  ;;HIGH ADDRESS,M.S. BYTE
001367     000                  $MTYP2: .BYTE   AMTYP2  ;;MEM.TYPE,BLK#2
001370  000000                  $MADR2: .WORD   AMADR2  ;;MEM.LAST ADDRESS,BLK#2
001372     000                  $MAMS3: .BYTE   AMAMS3  ;;HIGH ADDRESS,M.S.BYTE
001373     000                  $MTYP3: .BYTE   AMTYP3  ;;MEM.TYPE,BLK#3
001374  000000                  $MADR3: .WORD   AMADR3  ;;MEM.LAST ADDRESS,BLK#3
001376     000                  $MAMS4: .BYTE   AMAMS4  ;;HIGH ADDRESS,M.S.BYTE
001377     000                  $MTYP4: .BYTE   AMTYP4  ;;MEM.TYPE,BLK#4
001400  000000                  $MADR4: .WORD   AMADR4  ;;MEM.LAST ADDRESS,BLK#4
001402  000000                  $VECT1: .WORD   AVECT1  ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
001404  000000                  $VECT2: .WORD   AVECT2  ;;INTERRUPT VECTOR#2BUS PRIORITY#2
001406  000000                  $BASE:  .WORD   ABASE   ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
001410  000000                  $DEVM:  .WORD   ADEVM   ;;DEVICE MAP
001412  000000                  $CDW1:  .WORD   ACDW1   ;;CONTROLLER DESCRIPTION WORD#1
001414  000000                  $CDW2:  .WORD   ACDW2   ;;CONTROLLER DESCRIPTION WORD#2
001416  000000                  $DDW0:  .WORD   ADDW0   ;;DEVICE DESCRIPTOR WORD#0
001420  000000                  $DDW1:  .WORD   ADDW1   ;;DEVICE DESCRIPTOR WORD#1
001422  000000                  $DDW2:  .WORD   ADDW2   ;;DEVICE DESCRIPTOR WORD#2
001424  000000                  $DDW3:  .WORD   ADDW3   ;;DEVICE DESCRIPTOR WORD#3
001426  000000                  $DDW4:  .WORD   ADDW4   ;;DEVICE DESCRIPTOR WORD#4
001430  000000                  $DDW5:  .WORD   ADDW5   ;;DEVICE DESCRIPTOR WORD#5
001432  000000                  $DDW6:  .WORD   ADDW6   ;;DEVICE DESCRIPTOR WORD#6
001434  000000                  $DDW7:  .WORD   ADDW7   ;;DEVICE DESCRIPTOR WORD#7
001436  000000                  $DDW8:  .WORD   ADDW8   ;;DEVICE DESCRIPTOR WORD#8
001440  000000                  $DDW9:  .WORD   ADDW9   ;;DEVICE DESCRIPTOR WORD#9
001442  000000                  $DDW10: .WORD   ADDW10  ;;DEVICE DESCRIPTOR WORD#10
001444  000000                  $DDW11: .WORD   ADDW11  ;;DEVICE DESCRIPTOR WORD#11
001446  000000                  $DDW12: .WORD   ADDW12  ;;DEVICE DESCRIPTOR WORD#12
001450  000000                  $DDW13: .WORD   ADDW13  ;;DEVICE DESCRIPTOR WORD#13
001452  000000                  $DDW14: .WORD   ADDW14  ;;DEVICE DESCRIPTOR WORD#14
001454  000000                  $DDW15: .WORD   ADDW15  ;;DEVICE DESCRIPTOR WORD#15
001456                          $ETEND:
001456                          ATABL:  .BLKW   80.             ;TABLE A=ADDRESSES OF I/O MODULES
001716                          BTABL:  .BLKW   80.             ;TABLE B=GENERIC CODES OF THESE MODULES

                                ;MESSAGE TABLE=-=THIS TABLE CONTAINS USER I.D. MESSAGE OF PCS MODULES
```

```
002156  034376                  MESTBL: MASS58                  ;M6015
002160  031415                          MASS3                   ;M5010
002162  031473                          MASS4                   ;M5011
002164  031555                          MASS5                   ;M5012
002166  031630                          MASS6                   ;M5013
002170  031671                          MASS7                   ;M6010
002172  031750                          MASS8                   ;M6011
002174  032020                          MASS9                   ;M6012
002176  032073                          MASS10                  ;M6013
002200  034172                          MASS48                  ;M5012-YA
002202  034240                          MASS49                  ;M6010-YA
002204  034401                          MASS57                  ;M5031
002206  034341                          MASS56                  ;M5014
002210  034341                          MASS56                  ;M5014
002212  034341                          MASS56                  ;M5014
002214  034341                          MASS56                  ;M5014
002216  034463                          MASS52                  ;M5016
002220  034530                          MASS53                  ;M6014
002222  032765                          MASS29                  ;A630
002224  034652                          MASS59                  ;A631
002226  033037                          MASS30                  ;A014 SE
002230  033115                          MASS31                  ;A014 DM
002232  034707                          MASS60                  ;A020 2 WIRE
002234  034755                          MASS61                  ;A020 3 WIRE
002236  000000                          0                       ;END OF TABLE

                        ;BYTE TABLE===THIS TABLE CONTAINS # OF BYTES PER MODULE

002240  000002          BYTE:   .WORD   2                       ;M6015
002242  000004                  .WORD   4                       ;M5010
002244  000004                  .WORD   4                       ;M5011
002246  000002                  .WORD   2                       ;M5012
002250  000001                  .WORD   1                       ;M5013
002252  000004                  .WORD   4                       ;M6010
002254  000002                  .WORD   2                       ;M6011
002256  000001                  .WORD   1                       ;M6012
002260  000001                  .WORD   1                       ;M6013
002262  000002                  .WORD   2                       ;M5012-YA
002264  000004                  .WORD   4                       ;M6010-YA
002266  000004                  .WORD   4                       ;M5031
002270  000004                  .WORD   4                       ;M5014
002272  000004                  .WORD   4                       ;M5014
002274  000004                  .WORD   4                       ;M5014
002276  000004                  .WORD   4                       ;M5014
002300  000002                  .WORD   2                       ;M5016
002302  000004                  .WORD   4                       ;M6014
002304  000010                  .WORD   8.                      ;A630
002306  000010                  .WORD   8.                      ;A631
002310  000004                  .WORD   4                       ;A014
002312  000004                  .WORD   4                       ;A014
002314  000004                  .WORD   4                       ;A020
002316  000004                  .WORD   4                       ;A020
002320  000000                  .WORD   0                       ;END OF TABLE

                        ;CATAGORY TABLE===THIS TABLE CONTAINS THE SOFTWARE CATAGORY # OF THE MODULE

002322  000004          CAT:    .WORD   4                       ;M6015
```

```
002324  000001                  .WORD   1               ,M5010
002326  000002                  .WORD   2               ;M5011
002330  000003                  .WORD   3               ;M5012
002332  000003                  .WORD   3               ;M5013
002334  000004                  .WORD   4               ;M6010
002336  000005                  .WORD   5               ;M6011
002340  000004                  .WORD   4               ;M6012
002342  000004                  .WORD   4               ;M6013
002344  000003                  .WORD   3               ;M5012-YA
002346  000004                  .WORD   4               ;M6010-YA
002350  000002                  .WORD   2               ;M5031
002352  000012                  .WORD   10.             ;M5014
002354  000012                  .WORD   10.             ;M5014
002356  000012                  .WORD   10.             ;M5014
002360  000012                  .WORD   10.             ;M5014
002362  00001'                  .WORD   9.              ;M5015
002364  00001?                  .WORD   11.             ;M6014
002366  000006                  .WORD   6               ,A630
002370  000006                  .WORD   6               ;A631
002372  000007                  .WORD   7               ;A014
002374  000007                  .WORD   7               ;A014
002376  000010                  .WORD   8.              ;A020
002400  000010                  .WORD   8.              ;A020
002402  000000                  .WORD   0               ;END OF TABLE

                        ;TABLE OF GENERIC CODES

002404  000022          GENER:  .WORD   22              ;M6015 RETENTIVE ISOLATED DC OUTPUT
002406  000141                  .WORD   141             ;GENERIC CODE FOR NONISOLATED DC 32 BITS IN
002410  000121                  .WORD   121             ;GENERIC CODE FOR NONISOLATED DC 16 BITS IN
002412  000122                  .WORD   122             ;GENERIC CODE FOR ISOLATED DC 16 BITS IN
002414  000101                  .WORD   101             ;GENERIC CODE FOR AC 8 BITS IN
002416  000041                  .WORD   41              ;GENERIC CODE FOR NONISOLATED DC 32 BITS OUT
002420  000021                  .WORD   21              ;GENERIC CODE FOR ONESHOT 16 BITS OUT
002422  000001                  .WORD   1               ;GENERIC CODE FOR ISOLATED DC 8 BITS OUT
002424  000002                  .WORD   2               ;GENERIC CODE FOR AC 8 BITS OUT
002426  000123                  .WORD   123             ;16 BIT TTL COMPATABLE ISOLATED INPUT
002430  000043                  .WORD   43              ;32 BIT TTL COMPATABLE NONISOLATED OUTPUT
002432  000124                  .WORD   124             ;16 BIT ISOLATED CHANGE OF STATE MODULE
002434  000144                  .WORD   144             ;M5014 INPUT COUNTER MODE 1
002436  000145                  .WORD   145             ;M5014 INPUT COUNTER MODE 2
002440  000146                  .WORD   146             ;M5014 INPUT COUNTER MODE 3
002442  000147                  .WORD   147             ;M5014 INPUT COUNTER MODE 4
002444  000142                  .WORD   142             ;M5016 QUAD 8 BIT COUNTER/PRESCALAR
002446  000044                  .WORD   44              ;M6014 DUAL 16 BIT OUTPUT COUNTER/PRESCALAR DAUGHTER
002450  000261                  .WORD   261             ;GENERIC CODE FOR FOUR CHANEL DAC
002452  000262                  .WORD   262             ;A631 12 BIT DAC
002454  000321                  .WORD   321             ;A/D SINGLE ENDED
002456  000301                  .WORD   301             ;A/D DIFFERENTIAL MODE
002460  000324                  .WORD   324             ;A020 A/D 2 WIRE
002462  000304                  .WORD   304             ;A020 A/D 3 WIRE
002464  000000                  .WORD   0               ;END OF TABLE FLAG

                        ;MODULE TYPE NUMBERS

002466  006015          MUT:    .WORD   6015            ;M6015
002470  005010                  .WORD   5010
```

```
002472  005011                  .WORD   5011
002474  005012                  .WORD   5012
002476  005013                  .WORD   5013
002500  006010                  .WORD   6010
002502  006011                  .WORD   6011
002504  006012                  .WORD   6012
002506  006013                  .WORD   6013
002510  005012                  .WORD   5012            ;M5012-YA
002512  006010                  .WORD   6010            ;M6010-YA
002514  005011                  .WORD   5011            ;M5031 USES M5011 ROUTINE
002516  005014                  .WORD   5014            ;M5014
002520  005014                  .WORD   5014            ;M5014
002522  005014                  .WORD   5014            ;M5014
002524  005014                  .WORD   5014            ;M5014
002526  005016                  .WORD   5016            ;M5016
002530  006014                  .WORD   6014            ;M6014
002532  000630                  .WORD   630
002534  000631                  .WORD   631
002536  000014                  .WORD   14
002540  000014                  .WORD   14
002542  000020                  .WORD   20              ;A020
002544  000020                  .WORD   20              ;A020

                        ;TABLE OF I/O SUBROUTINE ADDRESSES

002546  065204          MODUL:  .WORD   M6015           ;RETENTIVE ISOLATED DC OUTPUT
002550  040752                  .WORD   M5010
002552  042554                  .WORD   M5011
002554  043540                  .WORD   M5012
002556  041044                  .WORD   M5013
002560  041650                  .WORD   M6010
002562  042230                  .WORD   M6011
002564  042040                  .WORD   M6012
002566  042040                  .WORD   M6013
002570  043540                  .WORD   M5012           ;M5012-YA SAME A M5012
002572  041650                  .WORD   M6010           ;M6010-YA SAME AS M6010
002574  042554                  .WORD   M5011           ;M5031 USES M5011 ROUTINE
002576  044346                  .WORD   M5014           ;INPUT COUNTER
002600  044346                  .WORD   M5014           ;INPUT COUNTER
002602  044346                  .WORD   M5014           ;INPUT COUNTER
002604  044346                  .WORD   M5014           ;INPUT COUNTER
002606  052416                  .WORD   M5016           ;QUAD 8 BIT COUNTER/PRESCALAR
002610  047724                  .WORD   M6014           ;DUAL 16 BIT COUNTER DAUGHTER
002612  055144                  .WORD   DCAMON
002614  065672                  .WORD   A631            ; TEST FOR A631
002616  060260                  .WORD   ADTST
002620  060260                  .WORD   ADTST
002622  074646                  .WORD   AUTO20          ; TEST FOR A020
002624  074646                  .WORD   AUTO20          ; TEST FOR A020
002626  000000                  .WORD   0


002630      125         PATT:   .BYTE   125
002631      252                 .BYTE   252
002632      377                 .BYTE   377
002633      000                 .BYTE   0
002634      123                 .BYTE   123
```

```
                                .EVEN
002636  000000          K:      .WORD 0         ;TIMING CONSTANT FOR CPU
002640  000042          XXDP:   .WORD   42
002642  177546          CLKADR: .WORD   177546
002644  171377          CSR:    .WORD   171377          ;CONTROL STATUS REGISTER ADDRESS
002646  171376          IAR:    .WORD   171376          ;INTERRUPT ADDRESS REGISTER ADDRESS
002650  000234          VECTO:  .WORD   234             ;INTERRUPT VECTOR OF THE IOCM
002652  000236          VECTOA: .WORD   236             ;INTERRUPT VECTOR+2 OF THE IOCM
002654  000000          VECT1:  .WORD   0
002656  000000          VECT1A: .WORD   0
002660  171000          BASE:   .WORD   171000          ;FIRST I/O
002662  000001          PASCNT: .WORD   1
002664  000000          AFLAG:  .WORD    0
002666  000100          CLKVC:  .WORD   100
002670  000102          CLKVCA: .WORD   102
002672  015006          MOD:    .WORD   F6015           ;M6015
002674  014714                  .WORD   F5010
002676  014714                  .WORD   F5011
002700  014726                  .WORD   F5012
002702  014740                  .WORD   F5013
002704  014752                  .WORD   F6010
002706  015006                  .WORD   F6011
002710  014772                  .WORD   F6012
002712  014772                  .WORD   F6013
002714  014726                  .WORD   F5012           ;M5012-YA
002716  014752                  .WORD   F6010           ;M6010-YA
002720  014714                  .WORD   F5011           ;M5031
002722  000000                  .WORD   0
002724  000060          KBVEC:  .WORD   60
002726  171000          LDATA0: .WORD   171000
002730  171001          HDATA0: .WORD   171001
002732  171002          LDATA1: .WORD   171002
002734  171003          HDATA1: .WORD   171003
002736  171004          LDATA2: .WORD   171004
002740  171005          HDATA2: .WORD   171005
002742  171006          LDATA3: .WORD   171006
002744  171007          HDATA3: .WORD   171007
002746                  RAMP1:  .BLKW   40.
003066                  RAMP2:  .BLKW   40.
003206                  RAMP3:  .BLKW   40.
003326  177777          ROTFLG: .WORD   -1

                        ;TABLE OF A157 GAINS

003330  000001          GAINTB: .WORD   1       ;TABLE OF A157 GAINS
003332  000002                  .WORD   2
003334  000010                  .WORD   10
003336  000020                  .WORD   20
003340  000050                  .WORD   50
003342  000100                  .WORD   100
003344  000200                  .WORD   200
003346  001000                  .WORD   1000
003350  000000                  .WORD   0
003352  000000          GAIN:   .WORD   0               ;CORRESPONDING OCTAL VALUES TO BE
003354  000020                  .WORD   20              ;LOADED INTO THE GAIN REGISTER OF A157
003356  000100                  .WORD   100
```

```
003360  000120                          .WORD   120
003362  000200                          .WORD   200
003364  000220                          .WORD   220
003366  000300                          .WORD   300
003370  000320                          .WORD   320
003372  000000                          .WORD   0
003374     000      001      001  AVRTBL: .BYTE   0,1,1,2,2,3,3,4
003377     002      002      003
003402     003      004

                                .EVEN

003404                          VARSTART:                       ;VARIABLES SHOULD BE PLACED HERE
                                                                ;BEFORE EACH TEST THEY ARE RESET TO ZERO


003404  000000                  DUMMY:  .WORD 0         ;PSEUDO TTY LOCATION
003406  000000                  DUMMY1: .WORD 0         ;PSEUDO TTY LOCATION
003410  000000                  ATEMP:  .WORD 0         ;TEMP STORAGE
003412  000000                  BTEMP:  .WORD 0                 ;TEMP STORAGE
003414  000000                  CTEMP:  .WORD 0                 ;TEMP STORAGE
003416  000000                  AD0:    .WORD 0                 ;THESE 8 LOCATIONS ARE USED IN THE A020 AND
003420  000000                  AD1:    .WORD 0                 ;-A631 ROUTINES. DATA AND STATUS REGISTERS.
003422  000000                  AD2:    .WORD 0
003424  000000                  AD3:    .WORD 0
003426  000000                  AD4:    .WORD 0
003430  000000                  AD5:    .WORD 0
003432  000000                  AD6:    .WORD 0
003434  000000                  AD7:    .WORD 0
003436  000000                  CSS:    .WORD  0                ;CSS=1 IF WARNING MESSAGE HAS BEEN PRINTED
003440  000000                  GCOD:   .WORD  0                ;USED IN GROUP ROUTINE
003442  000000                  IFLAG:  .WORD  0
003444  000000                  YLOOP:  .WORD  0
003446  000000                  ZLOOP:  .WORD 0         ;USED IN DELAY MACRO
003450  000000                  MXNUM:  .WORD  0
003452  000000                  DBUFF:  .WORD  0
003454  000000                          .WORD  0
003456  000000                  $MUT:   .WORD  0
003460  000000                  BYTNUM: .WORD  0                ;NUMBER OF BITES OF I/O
003462  000000                  TADDR:  .WORD  0                ;ADDRESS OF MUT
003464  000000                  TADDR1: .WORD  0
003466  000000                  TBADDR: .WORD  0
003470  000000                  COSADR: .WORD  0                ;SECOND ADDRESS OF COS MODULE
003472  000000                  DMUT:   .WORD  0                ;DIGITAL MODULE UNDER TEST
003474  000000                  LONGIN: .WORD  0
003476  000000                  CLK:    .WORD  0
003500  000000                  NORAMP: .WORD  0
003502  000000                  TEMP:   .WORD  0
003504  000000                  RERROR: .WORD  0
003506  000000                  XXX:    .WORD  0
003510  000000                  TEMP1:  .WORD  0
003512  000000                  INTDAT: .WORD  0
003514  000000                  CHNUM:  .WORD  0
003516  000000                  CH0:    .WORD  0
003520  000000                  CH1:    .WORD  0
003522  000000                  CH2:    .WORD  0
003524  000000                  CH3:    .WORD  0
```

```
003526  000000                  CH4:     .WORD   0
003530  000000                  DCHAN:   .WORD   0
003532  000000                  XCHAN:   .WORD   0
003534  000000                  VCHAN:   .WORD   0
003536  000000                  ICHAN:   .WORD   0
003540  000000                  KOUNT:   .WORD   0
003542  000000                  ACOUNT:  .WORD   0       ;THESE WORDS A-F MUST BE CONTIGUOUS IN CORE
003544  000000                  BCOUNT:  .WORD   0
003546  000000                  CCOUNT:  .WORD   0
003550  000000                  DCOUNT:  .WORD   0
003552  000000                  ECOUNT:  .WORD   0
003554  000000                  FCOUNT:  .WORD  0
003556  000000                  BLAST:   .WORD   0
003560  000000                  LAST:    .WORD   0
003562  000000                  LASTCN:  .WORD   0
003564  000000                  ROTPAT:  .WORD   0
003566  000000                  ANSW:    .WORD   0
003570  000000                  CONV:    .WORD   0
003572  000000                  DATALO:  .WORD   0
003574  000000                  CONVCT:  .WORD   0
003576  000000                  TEMP2:   0
003600  000000                  TEMP3:   0
003602  000000                  TEMP4:   0
003604  000000                  TEMP5:   0
003606  000000                  GBITE:   0
003610  000000                  STAT1:   0
003612  000000                  STAT2:   0
003614  000000                  LBYTE:   0
003616  000000                  HBYTE:   0
003620  000000                  ST1SAV:  .WORD   0       ;STAT1 SAVE
003622  000000                  ST2SAV:  .WORD   0       ;STAT2 SAVE
003624  000000                  ACNTH:   0               ;HIGH BYTE ADDRESS OF COUNTER A
003626  000000                  ACNTL:   0               ;LOW BYTE ADDRESS OF COUNNTER A
003630  000000                  BCNTH:   0               ;HIGH BYTE ADDRESS OF COUNTER B
003632  000000                  BCNTL:   0               ;LOW BYTE ADDRESS OF COUNTER B
003634  000000                  CTRSH:   0               ;COUNTER UNDER TEST HIGH BYTE ADDRESS
003636  000000                  CTPSL:   0               ;COUNTER UNDER TEST LOW BYTE ADDRESS
003640  000000                  CNUM:    0               ;COUNTER NUMBER
003642  000000                  NUM:     0
003644  000000                  VARI:    0
003646  000000                  XLOOP:   0
003650  000000                  TEMPB:   0
003652  000000                  RETURN:  0
003654  000000                  AVRSAV:  0
003656                          CTABLE:   .BLKW   100.  ;TABLE C-CATAGORY NUMBERS OF CURRENT SYSTEM

004166  000000                  VAREND:  .WORD 0        ;END OF VARIABLES
```

```
                                .SBTTL  ERROR POINTER TABLE
                                ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
                                ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
                                ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
                                ;*NOTE1:        IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
                                ;*NOTE2:        EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
                                ;*      EM              ;;POINTS TO THE ERROR MESSAGE
                                ;*      DH              ;;POINTS TO THE DATA HEADER
                                ;*      DT              ;;POINTS TO THE DATA
                                ;*      DF              ;;POINTS TO THE DATA FORMAT
     004170                     $ERRTB:
1649                            ;;;ERROR 1
1650 004170  022626             EM1     ;.ASCIZ /CBIT NOT CLEARING IOCM/
1651 004172  027535             DH1     ;.ASCIZ /PC     TST#    GDDAT   BDDAT   PASS/
1652 004174  040464             DT1     ;.WORD  $ERRPC, $TESTN, $GDDAT, $BDDAT, $PASS,
1653 004176  000000             0
1654
1655                            ;;; ERROR 2
1656 004200  022655             EM2     ;.ASCII /IOCM TST/<15><12>
1657                            ;EM2Y:   .ASCII /CSR DATA ERROR WHEN TESTING /
1658                            ;EM2X:   .ASCIZ /  BIT  /<15><12>
1659 004202  027535             DH1     ;.ASCIZ /PC     TST#    GDDAT   BDDAT   PASS/
1660 004204  040464             DT1     ;.WORD  $ERRPC, $TESTN, $GDDAT, $BDDAT, $PASS,
1661 004206  000000             0
1662
1663                            ;;; ERROR 3
1664 004210  022736             EM3     ;.ASCIZ /IOCM IS NOT RESPONDING/<15><12>
1665 004212  000000             0
1666 004214  000000             0
1667 004216  000000             0
1668
1669 004220  022767             EM4     ;.ASCIZ /DBUS BIT IS FAILING/<15><12>
1670 004222  027535             DH1     ;.ASCIZ /PC     TST#    GDDAT   BDDAT   PASS/
1671 004224  040464             DT1     ;.WORD  $ERRPC, $TESTN, $GDDAT, $BDDAT, $PASS,
1672 004226  000000             0
1673
1674 004230  023015             EM5     ;.ASCIZ /NO INTERRUPT /<15><12>
1675 004232  027566             DH5
1676 004234  040500             DT5
1677 004236  000000             0
1678
1679 004240  023035             EM6
1680 004242  027535             DH1
1681 004244  040464             DT1
1682 004246  000000             0
1683
1684 004250  023104             EM7
1685 004252  027621             DH7
1686 004254  040444             DT7
1687 004256  000000             0
1688
1689 004260  023121             EM10
1690 004262  000000             0
1691 004264  000000             0
1692 004266  000000             0
1693
1694 004270  023154             EM11
```

ERROR POINTER TABLE

```
1695 004272  027621                  DH7
1696 004274  040444                  DT7
1697 004276  000000                  0
1698
1699 004300  023212                  EM12
1700 004302  027566                  DH5
1701 004304  040500                  DT5
1702 004306  000000                  0
1703
1704 004310  023242                  EM13
1705 004312  000000                  0
1706 004314  000000                  0
1707 004316  000000                  0
1708
1709 004320  023310                  EM14
1710 004322  000000                  0
1711 004324  000000                  0
1712 004326  000000                  0
1713
1714 004330  023353                  EM15
1715 004332  000000                  0
1716 004334  000000                  0
1717 004336  000000                  0
1718
1719 004340  023416                  EM16
1720 004342  027535                  DH1
1721 004344  040464                  DT1
1722 004346  000000                  0
1723
1724 004350  023104                  EM17
1725 004352  027621                  DH7
1726 004354  040374                  DT17
1727 004356  000000                  0
1728
1729 004360  023461                  EM20
1730 004362  027621                  DH7
1731 004364  040414                  DT20
1732 004366  000000                  0
1733
1734 004370  023513                  EM21
1735 004372  027671                  DH21
1736 004374  040514                  DT21
1737 004376  000000                  0
1738
1739 004400  023555                  EM22
1740 004402  000000                  0
1741 004404  000000                  0
1742 004406  000000                  0
1743
1744 004410  023631                  EM23
1745 004412  027715                  DH23
1746 004414  040434                  DT23
1747 004416  000000                  0
1748
1749 004420  023666                  EM24
1750 004422  000000                  0
1751 004424  000000                  0
```

ERROR POINTER TABLE

```
1752 004426  000000                  0
1753
1754 004430  023734                  EM25
1755 004432  000000                  0
1756 004434  000000                  0
1757 004436  000000                  0
1758
1759 004440  023775                  EM26
1760 004442  027566                  DH5
1761 004444  040500                  DT5
1762 004446  000000                  0
1763
1764 004450  024051                  EM27
1765 004452  027566                  DH5
1766 004454  040500                  DT5
1767 004456  000000                  0
1768
1769 004460  024115                  EM30
1770 004462  036217                  DH30
1771 004464  040526                  DT30
1772 004466  000000                  0
1773
1774 004470  024157                  EM31
1775 004472  040251                  DH31
1776 004474  040704                  DT31
1777 004476  000000                  0
1778
1779 004500  024232                  EM32
1780 004502  040251                  DH31
1781 004504  040704                  DT31
1782 004506  000000                  0
1783
1784 004510  024307                  EM33
1785 004512  036044                  DH33
1786 004514  040542                  DT33
1787 004516  000000                  0
1788
1789 004520  024343                  EM34
1790 004522  036113                  DH34
1791 004524  040564                  DT34
1792 004526  000000                  0
1793
1794 004530  024430                  EM35
1795 004532  027621                  DH7
1796 004534  040444                  DT7
1797 004536  000000                  0
1798
1799 004540  024501                  EM36
1800 004542  027566                  DH5
1801 004544  040500                  DT5
1802 004546  000000                  0
1803
1804 004550  024547                  EM37
1805 004552  027566                  DH5
1806 004554  040500                  DT5
1807 004556  000000                  0
1808
```

ERROR POINTER TABLE

```
1809 004560  024610                  EM40
1810 004562  027566                  DH5
1811 004564  040500                  DT5
1812 004566  000000                  0
1813
1814 004570  024653                  EM41
1815 004572  027566                  DH5
1816 004574  040500                  DT5
1817 004576  000000                  0
1818
1819 004600  024716                  EM42
1820 004602  027621                  DH7
1821 004604  040444                  DT7
1822 004606  000000                  0
1823
1824 004610  024757                  EM43
1825 004612  027566                  DH5
1826 004614  040500                  DT5
1827 004616  000000                  0
1828
1829 004620  025034                  EM44
1830 004622  027566                  DH5
1831 004624  040500                  DT5
1832 004626  000000                  0
1833
1834 004630  025073                  EM45
1835 004632  027566                  DH5
1836 004634  040500                  DT5
1837 004636  000000                  0
1838
1839 004640  025140                  EM46
1840 004642  027621                  DH7
1841 004644  040444                  DT7
1842 004646  000000                  0
1843
1844 004650  025212                  EM47
1845 004652  027566                  DH5
1846 004654  040500                  DT5
1847 004656  000000                  0
1848
1849 004660  025240                  EM50
1850 004662  027621                  DH7
1851 004664  040444                  DT7
1852 004666  000000                  0
1853
1854 004670  000000                  0
1855 004672  000000                  0
1856 004674  040606                  DT51
1857 004676  040740                  DF51
1858
1859 004700  000000                  0
1860 004702  000000                  0
1861 004704  040622                  DT52
1862 004706  040744                  DF52
1863
1864 004710  025302                  EM53    ;.ASCII <15><12>/LINEARITY TEST ERROR/<15><12>
1865                                         ;.ASCIZ /ERRPC  ADDRESS TSTPNT  MIN     MAX     WAS     (# OF CONVERS)/<15><12>
```

ERROR POINTER TABLE

```
1866 004712  000000                  0
1867 004714  000000                  0
1868 004716  000000                  0
1869
1870 004720  025414                  EM54
1871 004722  000000                  0
1872 004724  000000                  0
1873 004726  000000                  0
1874
1875 004730  025456                  EM55
1876 004732  027566                  DH5
1877 004734  040500                  DT5
1878 004736  000000                  0
1879
1880 004740  025531                  EM56
1881 004742  027566                  DH5
1882 004744  040500                  DT5
1883 004746  000000                  0
1884
1885
1886 004750  025572                  EM57
1887 004752  031242                  DH57
1888 004754  040640                  DT57
1889 004756  000000                  0
1890
1891 004760  025621                  EM60
1892 004762  031266                  DH61
1893 004764  040650                  DT61
1894 004766  000000                  0
1895
1896 004770  025655                  EM61
1897 004772  036171                  DH40
1898 004774  040434                  DT23
1899 004776  000000                  0
1900
1901 005000  024757                  EM43
1902 005002  031341                  DH62
1903 005004  040670                  DT62
1904 005006  000000                  0
1905
1906 005010  024716                  EM42
1907 005012  031266                  DH61
1908 005014  040650                  DT61
1909 005016  000000                  0
1910
1911 005020  024115                  EM30
1912 005022  031266                  DH61
1913 005024  040650                  DT61
1914 005026  000000                  0
1915
1916 005030  025765                  EM63
1917 005032  031341                  DH62
1918 005034  040670                  DT62
1919 005036  000000                  0
1920
1921 005040  025073                  EM45
1922 005042  031341                  DH62
```

ERROR POINTER TABLE

```
1923 005044  040670                DT62
1924 005046  000000                0
1925
1926 005050  025724                EM62
1927 005052  031341                DH62
1928 005054  040670                DT62
1929 005056  000000                0
1930
1931 005060  024610                EM40
1932 005062  031341                DH62
1933 005064  040670                DT62
1934 005066  000000                0
1935
1936 005070  024547                EM37
1937 005072  031341                DH62
1938 005074  040670                DT62
1939 005076  000000                0
1940
1941 005100  025240                EM50
1942 005102  031266                DH61
1943 005104  040650                DT61
1944 005106  000000                0
1945
1946 005110  026022                EM73
1947 005112  031341                DH62
1948 005114  040670                DT62
1949 005116  000000                0
1950
1951 005120  026074                EM74    ;.ASCIZ /INTERRUPT WON'T CLEAR/ <12><15>
1952 005122  000000                0
1953 005124  000000                0
1954 005126  000000                0
1955
1956 005130  026124                EM75    ;.ASCIZ /CBIT FAILS TO CLEAR MODULE/<12><15>
1957 005132  027434                DH100   ;.ASCIZ /TST#   PC      MODULE  ADDR    GDDAT   BDDAT   $PASS/
1958 005134  040354                DT100   ;.WORD  $TESTN, $ERRPC  $MUT,   TADDR,  $GDDAT, $BDDAT, $PASS,
1959 005136  040725                DF100   ;.BYTE  0,0,0,0,0,1
1960
1961 005140  026161                EM76    ;.ASCIZ %COUNTER R/W FAILURE%<12><15>
1962 005142  027434                DH100   ;.ASCIZ /TST#   PC      MODULE  ADDR    GDDAT   BDDAT   $PASS/
1963 005144  040354                DT100   ;.WORD  $TESTN, $ERRPC  $MUT,   TADDR,  $GDDAT, $BDDAT, $PASS,
1964 005146  040725                DF100   ;.BYTE  0,0,0,0,0,1
1965
1966 005150  026207                EM77    ;.ASCIZ /TBIT FAILS TO INITIALIZE MODULE/<12><15>
1967 005152  027434                DH100   ;.ASCIZ /TST#   PC      MODULE  ADDR    GDDAT   BDDAT   $PASS/
1968 005154  040354                DT100   ;.WORD  $TESTN, $ERRPC  $MUT,   TADDR,  $GDDAT, $BDDAT, $PASS,
1969 005156  040725                DF100   ;.BYTE  0,0,0,0,0,1
1970
1971 005160  026251                EM100   ;.ASCIZ /COUNTER FAILS TO HALT AFTER INTERRUPT/<12><15>
1972 005162  027434                DH100   ;.ASCIZ /TST#   PC      MODULE  ADDR    GDDAT   BDDAT   $PASS/
1973 005164  040354                DT100   ;.WORD  $TESTN, $ERRPC  $MUT,   TADDR,  $GDDAT, $BDDAT, $PASS,
1974 005166  040725                DF100   ;.BYTE  0,0,0,0,0,1
1975
1976 005170  026321                EM101   ;.ASCIZ /COUNTER FAILS TO COUNT/<12><15>
1977 005172  027434                DH100   ;.ASCIZ /TST#   PC      MODULE  ADDR    GDDAT   BDDAT   $PASS/
1978 005174  040354                DT100   ;.WORD  $TESTN, $ERRPC  $MUT,   TADDR,  $GDDAT, $BDDAT, $PASS,
1979 005176  040725                DF100   ;.BYTE  0,0,0,0,0,1
```

ERROR POINTER TABLE

```
1980
1981 005200  026352                EM102   ;.ASCIZ /WRONG COUNT IN COUNTER AFTER TRANSITION/<12><15>
1982 005202  027434                DH100   ;.ASCIZ /TST#   PC      MODULE  ADDR    GDDAT   BDDAT   $PASS/
1983 005204  040354                DT100   ;.WORD  $TESTN, $ERRPC  $MUT,   TADDR,  $GDDAT, $BDDAT, $PASS,
1984 005206  040725                DF100   ;.BYTE  0,0,0,0,0,1
1985
1986 005210  026424                EM103   ;.ASCIZ /NO INTERRUPT/<12><15>
1987 005212  027501                DH101   ;.ASCIZ /TST#   PC      MODULE  ADDRESS $PASS/
1988 005214  040340                DT101   ;.WORD  $TESTN, $ERRPC  $MUT,   TADDR,  $PASS,
1989 005216  040733                DF101   ;.BYTE  0,0,0,0,1
1990
1991 005220  026443                EM104   ;.ASCIZ /RIF BIT NOT CLEARING INTERRUPT/<12><15>
1992 005222  000000                0
1993 005224  000000                0
1994 005226  000000                0
1995
1996 005230  026504                EM105   ;.ASCIZ /NO CLOCK/<15><12>
1997 005232  027434                DH100   ;.ASCIZ /TST#   PC      MODULE  ADDR    GDDAT   BDDAT   $PASS/
1998 005234  040354                DT100   ;.WORD  $TESTN, $ERRPC  $MUT,   TADDR,  $GDDAT, $BDDAT, $PASS,
1999 005236  040725                DF100   ;.BYTE  0,0,0,0,0,1
2000
2001 005240  026531                EM106   ;.ASCIZ /COUNTER NOT AT INITIALIZED FREQ/<12><15>
2002 005242  027434                DH100   ;.ASCIZ /TST#   PC      MODULE  ADDR    GDDAT   BDDAT   $PASS/
2003 005244  040354                DT100   ;.WORD  $TESTN, $ERRPC  $MUT,   TADDR,  $GDDAT, $BDDAT, $PASS,
2004 005246  040725                DF100   ;.BYTE  0,0,0,0,0,1
2005
2006 005250  026573                EM107   ;.ASCIZ /TIME BASE ERROR/<15><12>
2007 005252  027434                DH100   ;.ASCIZ /TST#   PC      MODULE  ADDR    GDDAT   BDDAT   $PASS/
2008 005254  040354                DT100   ;.WORD  $TESTN, $ERRPC, $MUT,   TADDR,  $GDDAT, $BDDAT, $PASS,
2009 005256  040725                DF100   ;.BYTE  0,0,0,0,0,1
2010
2011 005260  026615                EM110   ;.ASCIZ /UNEXPECTED COUNTER INTERRUPT/<15><12>
2012 005262  035656                DH110   ;.ASCIZ /TST#   PC      ADDRESS COUNTER MODULE  ITER/
2013 005264  040302                DT110   ;.WORD  $TESTN, $ERRPC, TADDR,  CNUM,   $MUT,   ITER,
2014 005266  040725                DF100   ;.BYTE  0,0,0,0,0,1
2015
2016 005270  026654                EM111   ;.ASCIZ /MODULE READ-WRITE FAILURE/<12><15>
2017 005272  027434                DH100   ;.ASCIZ /TST#   PC      MODULE  ADDR    GDDAT   BDDAT   $PASS/
2018 005274  040354                DT100   ;.WORD  $TESTN, $ERRPC, $MUT,   TADDR,  $GDDAT, $BDDAT, $PASS,
2019 005276  040725                DF100   ;.BYTE  0,0,0,0,0,1
2020
2021 005300  026710                EM112   ;.ASCIZ /MODULE SELF-CLEAR ERROR/<12><15>
2022 005302  027434                DH100   ;.ASCIZ /TST#   PC      MODULE  ADDR    GDDAT   BDDAT   $PASS/
2023 005304  040354                DT100   ;.WORD  $TESTN, $ERRPC, $MUT,   TADDR,  $GDDAT, $BDDAT, $PASS,
2024 005306  040725                DF100   ;.BYTE  0,0,0,0,0,1
2025
2026 005310  026742                EM113   ;.ASCIZ /INCORRECT COUNT IN COUNTER/<12><15>
2027 005312  035722                DH102   ;.ASCIZ /TST#   PC      ADDRESS COUNTER GDDAT   BDDAT   ITER/
2028 005314  040320                DT102   ;.WORD  TNUMB,  $ERRPC, TADDR,  CNUM,   $GDDAT, $BDDAT, ITER,
2029 005316  040716                DF102   ;.BYTE  0,0,0,0,0,0,1
2030
2031 005320  026777                EM114   ;.ASCIZ /M5016 OVERFLOW FAILURE/<12><15>
2032 005322  035773                DH114   ;.ASCIZ /TST#   PC      ADDRESS COUNTER GDOVF   BDOVF   ITER/
2033 005324  040320                DT102   ;.WORD  TNUMB,  $ERRPC, TADDR,  CNUM,   $GDDAT, $BDDAT, ITER,
2034 005326  040716                DF102   ;.BYTE  0,0,0,0,0,0,1
2035
2036 005330  027030                EM115   ;.ASCIZ /OVERFLOW FAILS TO CLEAR/<12><15>
```

ERROR POINTER TABLE

```
2037 005332  035722                DH102   ;.ASCIZ /TST#   PC      ADDRESS COUNTER GDDAT   BDDAT   ITER/
2038 005334  040320                DT102   ;.WORD  TNUMB,  $ERRPC, TADDR,  CNUM,   $GDDAT, $BDDAT, ITER,
2039 005336  040716                DF102   ;.BYTE  0,0,0,0,0,0,1
2040
2041 005340  027062                EM116   ;.ASCIZ /CSR CBIT FAILS TO CLEAR COUNTER/<12><15>
2042 005342  027434                DH100   ;.ASCIZ /TST#   PC      MODULE  ADDR    GDDAT   BDDAT   $PASS/
2043 005344  040354                DT100   ;.WORD  $TESTN, $ERRPC, $MUT,   TADDR,  $GDDAT, $BDDAT, $PASS,
2044 005346  040725                DF100   ;.BYTE  0,0,0,0,0,1
2045
2046 005350  027124                EM117   ;.ASCIZ <12><15>/ILLEGAL RADIX/<12><15>
2047 005352  035656                DH110   ;.ASCIZ /TST#   PC      ADDRESS COUNTER MODULE  ITER/
2048 005354  040302                DT110   ;.WORD  $TESTN, $ERRPC, TADDR,  CNUM,   $MUT,   ITER,
2049 005356  040725                DF100   ;.BYTE  0,0,0,0,0,1
2050
2051 005360  112670                EM120   ;TIMEOUT OCCURED
2052 005362  121401                DH67    ;ERRPC  TSTNUM  BDADR
2053 005364  122340                DT67    ;ERRPC  TNUMB   $BDADR
2054 005366  122552                DF10
2055
2056 005370  112740                EM121   ;NO TIMEOUT WHEBN WRITTING ADDRESS
2057 005372  121401                DH67    ;ERRPC  TSTNUM  BDADR
2058 005374  122340                DT67    ;ERRPC  TNUMB   $BDADR
2059 005376  122552                DF10
2060
2061 005400  113016                EM122   ;FIRST LOAD STATUS ERROR
2062 005402  121443                DH71    ;ERRPC  TSTNUM  ADDR    GDDAT   BDDAT
2063 005404  122354                DT71    ;ERRPC  TNUMB   $BDADR  $GDDAT  $BDDAT
2064 005406  122552                DF10
2065
2066 005410  113071                EM123   ;FATAL ERROR - BAD GENERIC CODE
2067 005412  121443                DH71    ;ERRPC  TSTNUM  ADDR    GDDAT   BDDAT
2068 005414  122354                DT71    ;ERRPC  TNUMB   $BDADR  $GDDAT  $BDDAT
2069 005416  122552                DF10
2070
2071 005420  113246                EM124   ;SECOND LOAD STATUS ERROR
2072 005422  121443                DH71    ;ERRPC  TSTNUM  ADDR    GDDAT   BDDAT
2073 005424  122354                DT71    ;ERRPC  TNUMB   $BDADR  $GDDAT  $BDDAT
2074 005426  122552                DF10
2075
2076 005430  113317                EM125   ;NO INTERRUPT UPON ERROR CONDITION
2077 005432  121401                DH67    ;ERRPC  TSTNUM  BDADR
2078 005434  122340                DT67    ;ERRPC  TNUMB   $BDADR
2079 005436  122552                DF10
2080
2081 005440  113357                EM126   ;UNEXPECTED INTERRUPT
2082 005442  121554                DH75    ;ERRPC  ACTUAL PC
2083 005444  122406                DT75    ;ERRPC  $BDADR
2084 005446  122552                DF10
2085
2086 005450  113406                EM127   ;FLAG NOT SET UPON ERROR CONDITION
2087 005452  121443                DH71    ;ERRPC  TSTNUM  ADDR    GDDAT   BDDAT
2088 005454  122354                DT71    ;ERRPC  TNUMB   $BDADR  $GDDAT  $BDDAT
2089 005456  122552                DF10
2090
2091 005460  113521                EM130   ;THIRD LOAD STATUS ERROR
2092 005462  121443                DH71    ;ERRPC  TSTNUM  ADDR    GDDAT   BDDAT
2093 005464  122354                DT71    ;ERRPC  TNUMB   $BDADR  $GDDAT  $BDDAT
```

ERROR POINTER TABLE

```
2094 005466  122552                DF10
2095
2096 005470  113624                EM131   ;FOURTH LOAD STATUS ERROR
2097 005472  121443                DH71    ;ERRPC  TSTNUM  ADDR    GDDAT   BDDAT
2098 005474  122354                DT71    ;ERRPC  TNUMB   $BDADR  $GDDAT  $BDDAT
2099 005476  122552                DF10
2100
2101 005500  113730                EM132   ;C BIT DIDN'T CLEAR AFTER OVERLOAD
2102 005502  121401                DH67    ;ERRPC  TSTNUM  BDADR
2103 005504  122340                DT67    ;ERRPC  TNUMB   $BDADR
2104 005506  122552                DF10
2105
2106 005510  114031                EM133   ;NORMAL OPERATION NOT POSSIBLE
2107 005512  121443                DH71    ;ERRPC  TSTNUM  ADDR    GDDAT   BDDAT
2108 005514  122354                DT71    ;ERRPC  TNUMB   $BDADR  $GDDAT  $BDDAT
2109 005516  122552                DF10
2110
2111 005520  114143                EM134   ;.ASCIZ <15><12> /UNEXPECTED TIMEOUT, FATAL ERROR/
2112 005522  121125                DH15    ;.ASCIZ /MODULE TSTNUM  PC      ITERCNT/
2113 005524  122244                DT15    ;.WORD  MOD,    TNUMB,  $BDADR, ITERCT,
2114 005526  122552                DF10
2115
2116 005530  114205                EM135   ;STATUS REGISTOR ERROR
2117 005532  121443                DH71    ;ERRPC  TSTNUM  ADDR    GDDAT   BDDAT
2118 005534  122354                DT71    ;ERRPC  TNUMB   $BDADR  $GDDAT  $BDDAT
2119 005536  122552                DF10
2120
2121 005540  114317                EM136   ;STATUS REGISTOR ERROR
2122 005542  121443                DH71    ;ERRPC  TSTNUM  ADDR    GDDAT   BDDAT
2123 005544  122354                DT71    ;ERRPC  TNUMB   $BDADR  $GDDAT  $BDDAT
2124 005546  122552                DF10
2125
2126 005550  114421                EM137   ;FLAG NOT SET UPON ERROR CONDITION
2127 005552  121443                DH71    ;ERRPC  TSTNUM  ADDR    GDDAT   BDDAT
2128 005554  122354                DT71    ;ERRPC  TNUMB   $BDADR  $GDDAT  $BDDAT
2129 005556  122552                DF10
2130
2131 005560  114546                EM140   ;IAR HAS WRONG ADDRESS
2132 005562  121443                DH71    ;ERRPC  TSTNUM  ADDR    GDDAT   BDDAT
2133 005564  122354                DT71    ;ERRPC  TNUMB   $BDADR  $GDDAT  $BDDAT
2134 005566  122552                DF10
2135
2136 005570  114672                EM141   ;R BIT DIDN'T CLEAR ERROR BIT
2137 005572   21443                DH71    ;ERRPC  TSTNUM  ADDR    GDDAT   BDDAT
2138 005574  122354                DT71    ;ERRPC  TNUMB   $BDADR  $GDDAT  $BDDAT
2139 005576  122552                DF10
2140
2141 005600  114760                EM142   ;INTERRUPT OCCURED AFTER R BIT SET
2142 005602  121401                DH67    ;ERRPC  TSTNUM  BDADR
2143 005604  122340                DT67    ;ERRPC  TNUMB   $BDADR
2144 005606  122552                DF10
2145
2146 005610  115066                EM143   ;.ASCIZ <15><12> /CHANNEL WRITTEN WAS NOT SAME AS CHANNEL READ/
2147 005612  121443                DH71    ;ERRPC  TSTNUM  ADDR    GDDAT   BDDAT
2148 005614  122354                DT71    ;ERRPC  TNUMB   $BDADR  $GDDAT  $BDDAT
2149 005616  122552                DF10
2150
```

ERROR POINTER TABLE

```
2151 005620  115 45                EM144   ;C BIT DIDN'T CLEAR CONVERSION PROCESS
2152 005622  121443                DH71    ;ERRPC  TSTNUM  ADDR    GDDAT   BDDAT
2153 005624  122354                DT71    ;ERRPC  TNUMB   $BDADR  $GDDAT  $BDDAT
2154 005626  122552                DF10
2155
2156 005630  115246                EM145   ;ON FIRST LOAD STATUS REGISTOR ERROR
2157 005632  121443                DH71    ;ERRPC  TSTNUM  ADDR    GDDAT   BDDAT
2158 005634  122354                DT71    ;ERRPC  TNUMB   $BDADR  $GDDAT  $BDDAT
2159 005636  122552                DF10
2160
2161 005640  115373                EM146   ;DONE BIT NOT SET IN TIME
2162 005642  121443                DH71    ;ERRPC  TSTNUM  ADDR    GDDAT   BDDAT
2163 005644  122354                DT71    ;ERRPC  TNUMB   $BDADR  $GDDAT  $BDDAT
2164 005646  122552                DF10
2165
2166 005650  115473                EM147   ;AFTER CONVERSION DONE STATUS ERROR
2167 005652  121443                DH71    ;ERRPC  TSTNUM  ADDR    GDDAT   BDDAT
2168 005654  122354                DT71    ;ERRPC  TNUMB   $BDADR  $GDDAT  $BDDAT
2169 005656  122552                DF10
2170
2171 005660  115576                EM150   ;DATA READ NOT = DATA WRITTEN
2172 005662  121443                DH71    ;ERRPC  TSTNUM  ADDR    GDDAT   BDDAT
2173 005664  122354                DT71    ;ERRPC  TNUMB   $BDADR  $GDDAT  $BDDAT
2174 005666  122552                DF10
2175
2176 005670  115666                EM151   ;STATUS ERROR AFTER LOADING 2
2177 005672  121443                DH71    ;ERRPC  TSTNUM  ADDR    GDDAT   BDDAT
2178 005674  122354                DT71    ;ERRPC  TNUMB   $BDADR  $GDDAT  $BDDAT
2179 005676  122552                DF10
2180
2181 005700  116011                EM152   ;AFTER 1ST INT DATA ERROR
2182 005702  121443                DH71    ;ERRPC  TSTNUM  ADDR    GDDAT   BDDAT
2183 005704  122354                DT71    ;ERRPC  TNUMB   $BDADR  $GDDAT  $BDDAT
2184 005706  122552                DF10
2185
2186 005710  116133                EM153   ;STATUS ERROR AFTER LOADING 4
2187 005712  121443                DH71    ;ERRPC  TSTNUM  ADDR    GDDAT   BDDAT
2188 005714  122354                DT71    ;ERRPC  TNUMB   $BDADR  $GDDAT  $BDDAT
2189 005716  122552                DF10
2190
2191 005720  116232                EM154   ;AFTER 2CD INT DATA ERROR
2192 005722  121443                DH71    ;ERRPC  TSTNUM  ADDR    GDDAT   BDDAT
2193 005724  122354                DT71    ;ERRPC  TNUMB   $BDADR  $GDDAT  $BDDAT
2194 005726  122552                DF10
2195
2196 005730  116331                EM155   ;STATUS ERROR AFTER LOADING 5
2197 005732  121443                DH71    ;ERRPC  TSTNUM  ADDR    GDDAT   BDDAT
2198 005734  122354                DT71    ;ERRPC  TNUMB   $BDADR  $GDDAT  $BDDAT
2199 005736  122552                DF10
2200
2201 005740  116441                EM156   ;AFTER 3RD INT DATA ERROR
2202 005742  121443                DH71    ;ERRPC  TSTNUM  ADDR    GDDAT   BDDAT
2203 005744  122354                DT71    ;ERRPC  TNUMB   $BDADR  $GDDAT  $BDDAT
2204 005746  122552                DF10
2205
2206 005750  116537                EM157   ;STATUS ERROR AFTER READING THIRD
2207 005752  121443                DH71    ;ERRPC  TSTNUM  ADDR    GDDAT   BDDAT
```

ERROR POINTER TABLE

```
2208 005754  122354                DT71    ;ERRPC  TNUMB   $BDADR  $GDDAT  $BDDAT
2209 005756  122552                DF10
2210
2211 005760  116627                EM160   ;AFTER 4TH INT DATA ERROR
2212 005762  121443                DH71    ;ERRPC  TSTNUM  ADDR    GDDAT   BDDAT
2213 005764  122354                DT71    ;ERRPC  TNUMB   $BDADR  $GDDAT  $BDDAT
2214 005766  122552                DF10
2215
2216 005770  116726                EM161   ;STATUS ERROR AFTER READING FOURTH
2217 005772  121443                DH71    ;ERRPC  TSTNUM  ADDR    GDDAT   BDDAT
2218 005774  122354                DT71    ;ERRPC  TNUMB   $BDADR  $GDDAT  $BDDAT
2219 005776  122552                DF10
2220
2221 006000  117023                EM162   ;AFTER 5TH INT DATA ERROR
2222 006002  121443                DH71    ;ERRPC  TSTNUM  ADDR    GDDAT   BDDAT
2223 006004  122354                DT71    ;ERRPC  TNUMB   $BDADR  $GDDAT  $BDDAT
2224 006006  122552                DF10
2225
2226 006010  117121                EM163   ;STATUS ERROR AFTER 5TH READ
2227 006012  121443                DH71    ;ERRPC  TSTNUM  ADDR    GDDAT   BDDAT
2228 006014  122354                DT71    ;ERRPC  TNUMB   $BDADR  $GDDAT  $BDDAT
2229 006016  122552                DF10
2230
2231 006020  117255                EM164   ;DONE ADN ACTIVE NOT SET
2232 006022  121443                DH71    ;ERRPC  TSTNUM  ADDR    GDDAT   BDDAT
2233 006024  122354                DT71    ;ERRPC  TNUMB   $BDADR  $GDDAT  $BDDAT
2234 006026  122552                DF10
2235
2236 006030  117360                EM165   ;DONE ADN DATA OVERRUN NOT SET
2237 006032  121443                DH71    ;ERRPC  TSTNUM  ADDR    GDDAT   BDDAT
2238 006034  122354                DT71    ;ERRPC  TNUMB   $BDADR  $GDDAT  $BDDAT
2239 006036  122552                DF10
2240
2241 006040  117430                EM166   ;PREVIOUS CHANNEL WAS NOT INTACT
2242 006042  121443                DH71    ;ERRPC  TSTNUM  ADDR    GDDAT   BDDAT
2243 006044  122354                DT71    ;ERRPC  TNUMB   $BDADR  $GDDAT  $BDDAT
2244 006046  122552                DF10
2245
2246 006050  117547                EM167   ;R BIT DID NOT CLEAR STATUS REG
2247 006052  121443                DH71    ;ERRPC  TSTNUM  ADDR    GDDAT   BDDAT
2248 006054  122354                DT71    ;ERRPC  TNUMB   $BDADR  $GDDAT  $BDDAT
2249 006056  122552                DF10
2250
2251 006060  117656                EM170   ;R BIT DID NOT CLEAR STATUS REG
2252 006062  121443                DH71    ;ERRPC  TSTNUM  ADDR    GDDAT   BDDAT
2253 006064  122354                DT71    ;ERRPC  TNUMB   $BDADR  $GDDAT  $BDDAT
2254 006066  122552                DF10
2255
2256 006070  117732                EM171   ;CHANNEL VALUE TOO LOW
2257 006072  121655                DH140   ;ERRPC  GDDAT   BDDAT   CHNUM
2258 006074  122422                DT140   ;ERRPC  GDDAT   BDDAT   CHNUM
2259 006076  122552                DF10
2260
2261 006100  117774                EM172   ;D BIT SET, CHANNEL VALUE TOO HIGH
2262 006102  121655                DH140   ;ERRPC  GDDAT   BDDAT   CHNUM
2263 006104  122422                DT140   ;ERRPC  GDDAT   BDDAT   CHNUM
2264 006106  122552                DF10
```

ERROR POINTER TABLE

```
2265
2266 006110  120040                EM173   ;CHANNEL INDEPENDENCE ERROR, OVRNGE NOT SET
2267 006112  121703                DH142   ;ERRPC  MOD     ERRCH   TSTCH   $BDDAT
2268 006114  122432                DT142   ;$ERRPC MOD     ACOUNT  BCOUNT  CCOUNT
2269 006116  122552                DF10
2270
2271 006120  120123                EM174   ;CHANNEL INDEPENCE ERROR AVERAGE<>0
2272 006122  121752                DH143   ;ERRPC  MOD     GDDAT   BDDAT   ERRCH   TSTCH   OVR-RANGE
2273 006124  122450                DT143   ;$ERRPC MOD     $GDDAT  $BDDAT  ACOUNT  BCOUNT  CCOUNT
2274 006126  122552                DF10
2275
2276 006130  120225                EM175   ;CHANNEL INDEPENDENCE STATUS ERROR
2277 006132  122041                DH144   ;ERRPC  MODULE  ADDR    GDDAT   BDDAT
2278 006134  122472                DT144   ;$ERRPC MOD     $BDADDR $GDDAT  $BDDAT
2279 006136  122552                DF10
2280
2281 006140  000000                0
2282 006142  000000                0
2283 006144  000000                0
2284 006146  000000                0
2285
2286 006150  000000                0
2287 006152  000000                0
2288 006154  000000                0
2289 006156  000000                0
2290
2291 006160  120271                EM200   ;ACCURACY ERROR
2292 006162  122123                DH145   ;$ERRPC MOD $GDDAT $BDDAT ACOUNT
2293 006164  122514                DT145   ;$ERRPC MOD $GDDAT $BDDAT ACOUNT
2294 006166  122552                DF10
2295
2296 006170  120317                EM201   ;GAIN TEST ERROR
2297 006172  122170                DH150   ;ERRPC  GDDAT   BDDAT   STAT    GAIN    CHNUM
2298 006174  122532                DT150   ;ERRPC  GDDAT   BDDAT   STAT    GAIN    CHNUM
2299 006176  122552                DF10
2300
2301 006200  120341                EM202   ;IAR HAS WRONG ADDRESS
2302 006202  000000                0
2303 006204  000000                0
2304 006206  000000                0
2305
2306 006210  120410                EM203   ;MORE CONVERSIONS RETURNED THAN LOADED
2307 006212  000000                0
2308 006214  000000                0
2309 006216  000000                0
2310
2311 006220  120471                EM204   ;HCM ERROR
2312 006222  121240                DH65    ;ERRPC  MOD     LO-LIM  LO-VAL  HI-LIM  HI-VAL  GD-WIND BD-WINDOW
2313 006224  122300                DT65    ;$ERRPC MOD     ACOUNT, .. ,FCOUNT
2314 006226  122552                DF10
2315
2316 006230  120531                EM205   ;.ASCIZ <15><12>        /NO INTERRUPT OCCURED /
2317 006232  121337                DH66    ;.ASCIZ /MOD    TSTNUM  PC      INTRPT# ADDR    CSR/
2318 006234  122322                DT66    ;.WORD  MOD,    TNUMB, BCOUNT,  ACOUNT, $BDADR, $BDDAT,
2319 006236  122552                DF10
2320
2321 006240  120561                EM206   ;WHEN THE A/D WAS ALL DONE AND A DELAY, ACTIVE REAMINDED ON
```

ERROR POINTER TABLE

```
2322 006242  121443                DH71    ;ERRPC  TSTNUM  ADDR    GDDAT   BDDAT
2323 006244  122354                DT71    ;ERRPC  TNUMB   $BDADR  $GDDAT  $BDDAT
2324 006246  122552                DF10
2325
2326 006250  120663                EM207   ;HCM TEST LOW LIMIT OUT OF BOUNDS
2327 006252  000000                0
2328 006254  000000                0
2329 006256  000000                0
2330
2331 006260  120733                EM210   ;HCM TEST HIGH LIMIT OUT OF BOUNDS
2332 006262  000000                0
2333 006264  000000                0
2334 006266  000000                0
2335
2336 006270  121006                EM211   ;HCM TEST WINDOW OUT OF BOUNDS
2337 006272  000000                0
2338 006274  000000                0
2339 006276  000000                0
2340
2341 006300  121053                EM212   ;HCM TEST COMMON MODE ERROR
2342 006302  000000                0
2343 006304  000000                0
2344 006306  000000                0
2345 006310  027146                EM213   ;ILLEGAL CONFIGURATION
2346 006312  000000                0
2347 006314  000000                0
2348 006316  000000                0
2349 006320  112670                EM120 ;ERROR 214  TIMEOUT OCCURED WHEN READING ADDRESS
2350 006322  121401                DH67    ;PC  TST#  MODULE  ADDR  PASS #
2351 006324  122340                DT67    ;$ERRPC, $TESTN, $MUT, $BDADR, $PASS
2352 006326  067006                DF1
2353
2354 006330  066734                EM215   ;TIMEOUT WHEN WRITING ADDRESS
2355 006332  121401                DH67    ;PC  TST#  MODULE  ADDR  PASS
2356 006334  122340                DT67    ;$ERRPC  $TESTN  $MUT  $BDADR  $PASS
2357 006336  067006                DF1
2358
2359 006340  067152                EM216   ;RAM MEMORY FAILURE
2360 006342  121443                DH71    ;ERRPC  TST #  BDADR  GDDAT  BDDAT  PASS
2361 006344  122354                DT71    ;$ERRPC  $TESTN  $BDADR  $GDDAT  $BDDAT  $PASS
2362 006346  067006                DF1
2363
2364 006350  067552                EM217   ;GENERIC CODE INNCORRECT
2365 006352  121443                DH71    ;ERRPC  TST#  BDADR  GDDAT  BDDAT  PASS
2366 006354  122354                DT71    ;$ERRPC  $TESTN  $BDADR  $GDDAT  $BDDAT  $PASS
2367 006356  067006                DF1
```

ERROR POINTER TABLE

```
 2370 006360                                    START:
                                                .SBTTL  INITIALIZE THE COMMON TAGS
                                                ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
      006360  012706  001114                            MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
      006364  005026                                    CLR     (R6)+           ;;CLEAR MEMORY LOCATION
      006366  022706  001154                            CMP     #SWR,R6 ;;DONE?
      006372  001374                                    BNE     .-6             ;;LOOP BACK IF NO
      006374  012706  001100                            MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                                                ;;INITIALIZE A FEW VECTORS
      006400  012737  017410  000020                    MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
      006406  012737  000340  000022                    MOV     #340,@#IOTVEC+2 ;;LEVEL 7
      006414  012737  020506  000030                    MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
      006422  012737  000340  000032                    MOV     #340,@#EMTVEC+2 ;;LEVEL 7
      006430  012737  022366  000034                    MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
      006436  012737  000340  000036                    MOV     #340,@#TRAPVEC+2;LEVEL 7
      006444  013737  011752  011744                    MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
      006452  005037  001174                            CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
      006456  112737  000001  001131                    MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
      006464  012737  006464  001124                    MOV     #.,$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
                                                ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
                                                ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
      006472  013746  000004                            MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
      006476  012737  006532  000004                    MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
      006504  012737  177570  001154                    MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
      006512  012737  177570  001156                    MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
      006520  022777  177777  172426                    CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
      006526  001012                                    BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                                ;;AND  THE HARDWARE SWR IS NOT = -1
      006530  000403                                    BR      65$             ;;BRANCH IF NO TIMEOUT
      006532  012716  006540                    64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
      006536  000002                                    RTI
      006540  012737  000176  001154    65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
      006546  012737  000174  001156                    MOV     #DISPREG,DISPLAY
      006554  012637  000004            66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
      006560  005037  001214                            CLR     $PASS           ;;CLEAR PASS COUNT
      006564  132737  000200  001227                    BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
      006572  001403                                    BEQ     67$             ;;YES,USE NON-APT SWITCH
      006574  012737  001230  001154                    MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
      006602                                    67$:
 2371
 2372                                           ;; CHECK FOR CONSOLE TTY MODULE
 2373
 2374 006602  013746  000004                            MOV     ERRVEC,-(SP)    ;SAVE LOCATION 4
 2375 006606  013746  000006                            MOV     ERRVEC+2,-(SP)  ;SAVE LOCATION 6
 2376 006612  012737  006636  000004                    MOV     #1$,ERRVEC      ;NEW TIME-OUT ADDRESS
 2377 006620  105777  172334                            TSTB    @$TKS           ;TRAP TO 4 IF NO TTY MODULE
 2378 006624  012637  000006                            MOV     (SP)+,ERRVEC+2          ;RESTORE 6
 2379 006630  012637  000004                            MOV     (SP)+,ERRVEC            ;RESTORE 4
 2380 006634  000415                                    BR      2$                      ;BRANCH
 2381
 2382                                           ;; SET UP DUMMY ADDRESSES IF NO CONSOLE TTY MODULE
 2383 006636  012737  003404  001160    1$:     MOV     #DUMMY,$TKS
 2384 006644  012737  003404  001162            MOV     #DUMMY,$TKB
 2385 006652  012737  003406  001164            MOV     #DUMMY1,$TPS
 2386 006660  012737  003406  001166            MOV     #DUMMY1,$TPB
 2387 006666  000006                            RTT
 2388 006670  004737  007620            2$:     JSR     PC,CLEARVAR             ;INIT VARAIABLES
```

INITIALIZE THE COMMON TAGS

```
2389
```

```
2391                                            .SBTTL  TYPE PROGRAM NAME
                                                ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
     006674  005227  177777                     INC     #-1             ;;FIRST TIME?
     006700  001040                             BNE     68$             ;;BRANCH IF NO
     006702  022737  011766  000042             CMP     #$ENDAD,@#42    ;;ACT-11?
     006710  001434                             BEQ     68$             ;;BRANCH IF YES
     006712  104401  006720                     TYPE    ,69$            ;;TYPE ASCIZ STRING
     006716  000431                             BR      68$             ;;GET OVER THE ASCIZ
                                                ;;69$:  .ASCIZ  <CRLF>*CVPCAGO  MDC PROCESS CONTROL I/O SUBSYSTEM TEST*<CRLF>
     007002                                     68$:
2392 007002  012737  007020  000004             SETCLK: MOV     #1$,ERRVEC
2393 007010  052777  000100  173624                     BIS     #BIT6,@CLKADR   ;ENABLE LINE CLOCK INTERRUPT FOR UNIBUS COMPUTERS
2394 007016  000401                             BR      2$
2395 007020  022626                             1$:     CMP     (SP)+,(SP)+     ;TRAP IF LSI11
2396 007022  012737  007606  000004             2$:     MOV     #TMOVEC,ERRVEC          ;RESTORE ERROR VECTOR
2397 007030  012706  001100                             MOV     #1100,R6                ;SET UP STACK
2398 007034  004737  014454                             JSR     PC,KCLOCK       ;DETERMINE TIMING CONSTANT
2399 007040  005777  173574                             TST     @XXDP           ;ARE WE IN XXDP CHAIN MODE?
2400 007044  001404                             BEQ     6$              ;NO
2401 007046  005037  001214                     CLR     $PASS           ;CLEAR PASS COUNT
2402 007052  000137  007340                     JMP     S               ;XXDP CHAIN MODE--JUMP TO S
2403
2404 007056  132737  000001  001226             6$:     BITB    #BIT0,$ENV      ;ARE WE UNDER APT
2405 007064  001413                             BEQ     4$              ;NO
2406 007066  132737  000040  001227             BITB    #BIT5,$ENV+1    ;INHIBIT PRINT?
2407 007074  001403                             BEQ     3$              ;NO,BRANCH
2408 007076  052737  020000  000176             BIS     #BIT13,SWREG    ;SET SWR13=1
2409 007104  005037  001214                     3$:     CLR     $PASS           ;CLEAR PASS COUNT
2410 007110  000137  012472                     JMP     M               ;UNDER APT - JUMP TO MAPPER ROUTINE
2411                                            ;
2412                                            ;IN STANDALONE MODE
2413                                            ;
2414 007114                                     4$:
     007114  012777  015612  173602             MOV     #KBINT,@KBVEC
     007122  152777  000100  172030             BISB    #BIT6,@$TKS     ;ENABLE KB INTERRUPT
2415 007130  012746  000000                     MOV     #PR0,-(SP)      ;SET PSW TO PRIORITY 0
     007134  012746  007142                     MOV     #64$,-(SP)
     007140  000002                             RTI
     007142  000240                     64$:    NOP
2416 007144  000137  007240                     JMP     MONIT
2417
2418                                            ;TYPE THE TEST OPTIONS -- HELP FILE
2419
2420 007150  104401  030062             H:      TYPE    ,M12            ;TEST OPTIONS
2421                                                                    ;S - SYSTEM TEST
2422                                                                    ;M - MAP OF DBUS DEVICES
2423                                                                    ;I - IOCM TEST
2424                                                                    ;T - SET SWREG
2425                                                                    ;F - FIELD TEST
2426                                                                    ;W - WRAP AROUND TEST
2427                                                                    ;L - LOAD LOOP COUNT
2428                                                                    ;A - MODULE TEST
2429                                                                    ;B - BASE ADDRESS
2430 007154  000137  007240                     JMP     MONIT
```

```
                                        ;TABLE OF COMMAND ROUTINES
                                        ;
007160                          CMMAND:
007160  007340                          .WORD   S               ;SYSTEM TEST
007162  010332                          .WORD   A               ;MODULE TEST
007164  007404                          .WORD   B               ;CHANGE BASE ADDRESS
007166  012472                          .WORD   M               ;MAP OF DBUS DEVICES
007170  007150                          .WORD   H               ;TYPE OPERATOR OPTIONS
007172  013166                          .WORD   I               ;IOCM TEST
007174  010720                          .WORD   T               ;SET SWREG
007176  007356                          .WORD   L               ;LOAD ITERATION COUNT
007200  012256                          .WORD   F               ;FIELD TEST
007202  012002                          .WORD   W               ;WRAP AROUND TEST
007204  007374                          .WORD   CLEAR           ;CLEAR THE SYSTEM
007206  000000                          .WORD   0               ;END OF TABLE FLAG
                                        ;
                                        ;TABLE OF ASCII COMMANDS
                                        ;
007210                          CMDASC:
007210  000123                          .WORD   'S              ;SYSTEM TEST
007212  000101                          .WORD   'A              ; MODULE TEST
007214  000102                          .WORD   'B              ;CHANGE BASE ADDRESS
007216  000115                          .WORD   'M              ;MAP OF DBUS DEVICES
007220  000110                          .WORD   'H              ;TYPE OPERATOR OPTIONS
007222  000111                          .WORD   'I              ;IOCM TEST
007224  000124                          .WORD   'T              ;SET SWREG
007226  000114                          .WORD   'L              ;LOAD ITERATION COUNT
007230  000106                          .WORD   'F              ;FIELD TEST
007232  000127                          .WORD   'W              ;WRAP AROUND TEST
007234  000003                          .WORD   3               ;CONTROL C, CLEAR THE SYSTEM
007236  000000                          .WORD   0               ;END OF TABLE FLAG
                                        ;
007240  012706  001100          MONIT:  MOV     #1100,R6        ;SET THE STACK
007244  005037  002664                  CLR     AFLAG           ;INITIALIZE AUTO RUN FLAG
007250  012777  000340  173374          MOV     #340,@VECTOA    ;DISABLE INTERUPTS
                                        ;
007256                          1$:
007256  012701  177776                  MOV     #-2,R1          ;INITIALIZE TABLE POINTER
007262  104401  032170                  TYPE    ,MASS12         ;QUERY THE OPERATOR
007266  104406                          RDCHR                   ;READ THE KEYBORD INPUT
007270  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
007272  010037  003566                  MOV     R0,ANSW         ;PREPARE TO ECHO CARACTER
007276  104401  003566                  TYPE    ,ANSW           ;ECHO THE CHARACTER
007302                          2$:
007302  062701  000002                  ADD     #2,R1           ;ADVANCE TABLE POINTER
007306  020061  007210                  CMP     R0,CMDASC(R1)   ;IF TABLE AGREES WITH INPUT
007312  001406                          BEQ     3$              ;-THEN EXECUTE THE COMMAND
                                        ;
007314  005761  007210                  TST     CMDASC(R1)      ;-ELSE IF NOT AT END OF TABLE
007320  001370                          BNE     2$              ;--THEN LOOK FOR TABLE MATCH
                                        ;
007322  104401  001202                  TYPE    ,$QUES          ;--ELSE UNINTELIGIBLE ENTRY
007326  000753                          BR      1$              ;---TRY AGAIN
007330                          3$:
007330  104401  001203                  TYPE    ,$CRLF
007334  000171  007160                  JMP     @CMMAND(R1)     ;EXECUTE THE COMMAND
                                        ;SYSTEM TEST
```

```
2489 007340                                S:
2490 007340  012737  000001  002664                MOV     #1,AFLAG        ;SET MONITOR LOC FOR AUTO RUN
2491 007346  005037  001214                        CLR     $PASS           ;INITIALIZE PASS NUMBER
2492 007352  000137  010760                        JMP     AUTO
2493
2494
2495                                               ;LOAD LOOP COUNT
2496 007356                                L:
2497 007356  104401  030623                        TYPE    ,M27            ;QUERY FOR NUMBER OF ITERATIONS
2498 007362  104410                                RDOCT                   ;READ KEYBOARD
2499 007364  012637  002662                        MOV     (SP)+,PASCNT    ;;POP STACK INTO PASCNT
2500 007370  000137  007240                        JMP     MONIT           ;RETURN TO MONITOR
2501
2502
2503
2504
2505                                               ;CLEAR THE SYSTEM
2506 007374                                CLEAR:
2507 007374  004737  014310                        JSR     PC,KLEER        ;CLEAR THE SYSTEM
2508 007400  000137  007240                        JMP     MONIT           ;RETURN TO MONITOR
2509
2510
2511                                               ;CHANGE THE BASE ADDRESS
2512
2513 007404                                B:
2514 007404  012737  007574  000004                MOV     #5$,ERRVEC      ;TIME OUT ROUTINE
2515 007412  104401  035023                        TYPE    ,MASS62         ;BASE ADDRESS NOW=>
2516 007416  013746  002660                        MOV     BASE,-(SP)      ;CURRENT BASE ADDRESS
2517 007422  104402                                TYPOC
2518 007424  104401  035051                        TYPE    ,MASS63         ;ENTER NEW BASE ADDRESS
2519 007430  104410                                RDOCT                   ;GET INPUT
2520 007432  021627  160000                        CMP     (SP),#160000    ;ADDRESS OF I/O PAGE ?
2521 007436  103003                                BHIS    1$              ;YES,BRANCH
2522 007440  104401  001202                        TYPE    ,$QUES          ;TRY AGAIN
2523 007444  000757                                BR      B
2524
2525 007446  012637  002660          1$.           MOV     (SP)+,BASE      ;SET NEW BASE
2526 007452  013737  002660  002646                MOV     BASE,IAR        ;SET NEW IAR AND CSR
2527 007460  062737  000376  002646                ADD     #376,IAR
2528 007466  013737  002660  002644                MOV     BASE,CSR
2529 007474  062737  000377  002644                ADD     #377,CSR
2530 007502  117737  173136  001142                MOVB    @CSR,$BDDAT
2531
2532 007510  104401  035105                        TYPE    ,MASS64         ;CURRENT VECTOR ADDRESS
2533 007514  013746  002650                        MOV     VECTO,-(SP)
2534 007520  104402                                TYPOC                   ;TYPE ADDRESS
2535
2536 007522  104401  035137          2$:           TYPE    ,MASS65         ;NEW VECTOR ADDRESS
2537 007526  104410                                RDOCT                   ;GET INPUT
2538 007530  021627  001000                        CMP     (SP),#1000      ;LESS THAN LARGEST FLOATING ADDRESS
2539 007534  103401                                BLO     3$              ;YES,BRANCH
2540 007536  000771                                BR      2$              ;TRY AGAIN
2541
2542 007540  021627  000234          3$:           CMP     (SP),#234       ;GREATER THE ADDRESS 234
2543 007544  103001                                BHIS    4$              ;YES,BRANCH
2544 007546  000765                                BR      2$              ;TRY AGAIN
2545
```

TYPE PROGRAM NAME

```
2546 007550  012637  002650          4$:     MOV     (SP)+,VECTO     ;SET NEW VECTOR ADDRESS
2547 007554  013737  002650  002652          MOV     VECTO,VECTOA
2548 007562  062737  000002  002652          ADD     #2,VECTOA
2549
2550 007570  000137  007240                  JMP     MONIT           ;GO BACK TO MONITOR
2551
2552 007574                          5$:
     007574  104003                          EMT     3
2553 007576  104401  035175                  TYPE    ,MASS66         ;TRY AGAIN TURKEY
2554 007602  022626                          CMP     (SP)+,(SP)+     ;STORE STACK
2555 007604  000677                          BR      B
```

```
2557
2558
2559
2560 007606                          TMOVEC:
2561 007606  104401  034051                 TYPE    ,MASS45          ;UNEXPECTED TIMEOUT ERROR
2562 007612  011646                         MOV     (R6),-(SP)       ;;SAVE (R6) FOR TYPEOUT
     007614  104402                         TYPOC                    ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2563 007616  000000                         HALT
2564
2565
2566                                        ;THIS ROUTINE ZEROS VARIABLES FROM LOCATION
2567                                        ;VARSTART TO VAREND
2568
2569 007620                          CLEARVAR:
2570
2571 007620  010046                         MOV     R0,-(SP)                 ;SAVE R0
2572 007622  012700  003404                 MOV     #VARSTART,R0             ;START ADDRESS OF TABLE
2573 007626  005020                  1$:    CLR     (R0)+                    ;CLEAR MEMORY LOCATION
2574 007630  022700  004166'                CMP     #VAREND,R0               ;END OF TABLE ?
2575 007634  001374                         BNE     1$                       ;NO,BRANCH
2576 007636  012600                         MOV     (SP)+,R0                 ;RESTORE R0
2577 007640  000207                         RTS     PC                       ;RETURN
2578
```

```
2580
2581                                ;;;;; THIS MODULE CONSISTS OF TWO ROUTINES.
2582
2583                                ;;;;; MAKE ROUTINE === BUILDS THE CURRENT CATAGORY TABLE . CATAGORY
2584                                ;;;;;                  NUMBERS ARE STORED IN 'CAT' TABLE.
2585                                ;;;;; CHECK ROUTINE === INSURES HARDWARE MODULES ARE GROUPED ACCORDING
2586                                ;;;;;                  TO THEIR CATAGORY NUMBER
2587
2588
2589
2590                                ;; BASE === LOWEST ADDRESS OF D-BUS
2591                                ;; IAR === BASE + 376
2592                                ;; BYTE[X] === TABLE OF NUMBER OF BYTES PER MODULE
2593                                ;; CTABLE === CURRENT SYSTEM MODULE CATAGORY TABLE
2594                                ;; TADDR === D-BUS ADDRESS
2595                                ;; GCOD  === GENERIC CODE OF HARDWARE MODULE
2596                                ;; GENER[X] === TABLE OF ALL VALID GENERIC CODES
2597                                ;; CAT[X] === TABLE OF VALID MODULE CATAGORIES
2598                                ;; R0 === INDEX FOR CTABLE  I.E. CTABLE[R0]
2599                                ;; R1 === INDEX FOR CAT AND GENERIC TABLE I.E. CAT [R1],GENER[R1]
2600                                ;; TEMP=== SET TO -1 TO INDICATE A TIMEOUT HAS OCCURRED
2601
2602
2603                                .REM%
2604                                GROUP
2605                                        MAKE ROUTINE
2606                                        :  SET UP TIMEOUT VECTOR ADDRESS
2607                                        :  CLEAR R0,R1
2608                                        :  SET G-BIT
2609                                        :  SET TADDR= BASE ADDRESS
2610                                210$:   ;  REPEAT
2611                                        :  :  CLEAR R1
2612                                220$:   ;  :  READ GCOD AT TADDR (TRAP TO 4 IF NO MODULE AT THIS ADDRESS)
2613                                        :  :  CLEAR HIGH BYTE OF GCOD
2614                                        :  :  CLEAR TEMP
2615                                230$:   ;  :  :  REPEAT
2616                                        :  :  :  :  IF GENER[R1] = GCOD
2617                                        :  :  :  :  :  THEN
2618                                        :  :  :  :  :  :  CTABLE[R0] = CAT[R1]
2619                                        :  :  :  :  :  :  R0 = R0 + 2
2620                                        :  :  :  :  :  :  TADDR = TADDR + BYTE [R1]
2621                                240$:   ;  :  :  :  :  ELSE
2622                                        :  :  :  :  :  :  R1 = R1 + 2
2623                                250$:   ;  :  :  UNTIL (GCOD = GENER[R1]) OR (GENER[R1]) = 0
2624                                        :  :  IF GENER[R1]=0
2625                                        :  :  :  THEN
2626                                        :  :  :  :  PRINT "UNKNOWN GENERIC CODE"
2627                                        :  :  :  :  TADDR = TADDR + 1
2628                                270$:   ;  UNTIL TADDR = IAR
2629                                        END MAKE
```

```
                                .SBTTL CHECK GROUPING
                                 CHECK CONFIGURATION ROUTINE
                        275$:    :  CLEAR R0,R1
                        280$:    ;  REPEAT
                                 :  :  WHILE CTABLE[R0] = CTABLE[R0+2] DO
                                 :  :  :  R0=R0+2
                                 :  :  END WHILE
                                 :  :  R1 = R0 + 4
                        290$:    ;  :  REPEAT
                                 :  :  :  IF CTABLE[R0] = CTABLE[R1]
                                 :  :  :  :  THEN
                                 :  :  :  :  :  PRINT ''CONFIGURATION PROBLEM''
                        300$:    ;  :  :  R1 = R1 + 2
                                 :  :  UNTIL (CTABLE[R1+2]) = 0
                                 :  :  R0 = R0 + 2
                                 :  :  IF TABLE[R0]=-1
                                 :  :  :  THEN
                                 :  :  :  :  R0=R0+2
                        305$:    :  UNTIL CTABLE[R0+2] = 0
                        310$:    :  CLEAR R0
                        311$:    :  WHILE (CTABLE[R0]<>0) OR (CTABLE[R0] <>8) DO
                                 :  :  R0=R0+2
                        312$:    :  IF CTABLE[R0]=8
                                 :  :  THEN
                                 :  :  :  REPEAT
                                 :  :  :  :  IF CTABLE[R0]=7
                                 :  :  :  :  :  THEN
                                 :  :  :  :  :  :  PRINT ''CONFIGURATION PROBLEM''
                        313$:    :  :  :  :  R0=R0+2
                                 :  :  :  UNTIL CTABLE[R0]=0
                        319$:    END CHECK

                                 TRAP ROUTINE
                        320$:    ;  TADDR = TADDR + 1
                                 :  IF TEMP = 0
                                 :  :  THEN
                                 :  :  :  TEMP = -1
                                 :  :  :  CTABLE[R0] = -1
                                 :  :  :  R0 =R0 + 2
                                 :  UPDATE RETURN ADDRESS
                                 END TRAP
                                 %
007642  012737  010252  000004  GROUP:  MOV     #320$,ERRVEC
007650  005037  000006                 CLR     ERRVEC+2                ;CLEAR PSW
007654  005000                         CLR     R0
007656  005001                         CLR     R1
007660  152777  000004  172756         BISB    #GBIT,@CSR              ;ENABLE G-BIT
007666  013737  002660  003462         MOV     BASE,TADDR              ;STARTING ADDRESS

007674  005001                 210$:   CLR     R1                      ;ZERO REG 1
007676  117737  173560  003440 220$:   MOVB    @TADDR,GCOD             ;READ GCOD
007704  042737  177400  003440         BIC     #177400,GCOD            ;CLEAR HI BYTE
007712  005037  003502                 CLR     TEMP


                                 ;***REPEAT UNTIL GENER(R1)=GCOD OR END OF GENER TABLE
007716                           230$:
```

```
TYPE PROGRAM NAME

2688 007716  026137  002404  003440          CMP     GENER(R1),GCOD          ;GOT A MATCH ?
2689 007724  001011                          BNE     240$                    ;NO,BRANCH
2690 007726  016160  002322  003656          MOV     CAT(R1),CTABLE(R0)      ;SAVE CATAGORY OF MODULE
2691 007734  062700  000002                  ADD     #2,R0                   ;BUMP CTABLE INDEX
2692 007740  066137  002240  003462          ADD     BYTE(R1),TADDR          ;BUMP ADDRESS BY # OF BYTES OF MODULE
2693 007746  000405                          BR      250$                    ;BRANCH
2694 007750  062701  000002          240$:   ADD     #2,R1                   ;BUMP GENER TABLE INDEX
2695 007754  005761  002404                  TST     GENER(R1)               ;END OF TABLE ?
2696 007760  001356                          BNE     230$                    ;NO,BRANCH
2697                                ;;;END OF REPEAT
2698
2699 007762  026137  002404  003440  250$:   CMP     GENER(R1),GCOD          ;HAD A MATCH ?
2700 007770  001414                          BEQ     270$                    ;YES, BRANCH
2701 007772  132737  000004  003440          BITB    #BIT2,GCOD              ;IF CSS MODULE
2702 010000  001404                          BEQ     255$                    ;-BIT2 AND
2703 010002  132737  000010  003440          BITB    #BIT3,GCOD              ;--BIT3 WILL
2704 010010  001001                          BNE     260$                    ;---BE = 1
2705 010012                          255$:
     010012  104057                          EMT     57
2706 010014  062737  000001  003462  260$:   ADD     #1,TADDR                ;BUMP D-BUS TEST ADDRESS
2707
2708 010022  023737  003462  002646  270$:   CMP     TADDR,IAR               ;LAST D-BUS ADDRESS ?
2709 010030  001321                          BNE     210$                    ;NO,BRANCH
2710
2711
2712
2713                                ;*** CHECK CONFIGURATION
2714 010032  005000                  275$:   CLR     R0                      ;CLEAR REGS
2715 010034  005001                          CLR     R1
2716 010036  010001                  280$:   MOV     R0,R1                   ;
2717 010040  062701  000002                  ADD     #2,R1                   ;R1=R0+2
2718 010044  026027  003656  177777          CMP     CTABLE(R0),#-1          ;MODULE ?
2719 010052  001404                          BEQ     282$                    ;NO,BRANCH
2720 010054  026061  003656  003656          CMP     CTABLE(R0),CTABLE(R1)   ;SAME CATAGORY ?
2721 010062  001003                          BNE     281$                    ;NO,BRANCH
2722 010064  062700  000002          282$:   ADD     #2,R0                   ;BUMP TABLE
2723 010070  000762                          BR      280$                    ;GO COMPARE NEXT TWO IN TABLE
2724 010072  062701  000002          281$:   ADD     #2,R1                   ;CHECK REMAINDER OF TABLE
2725 010076  026061  003656  003656  290$:   CMP     CTABLE(R0),CTABLE(R1)   ;THERE SHOULD BE NO MATCHES
2726 010104  001002                          BNE     300$                    ;NO, BRANCH
2727 010106  104213                          EMT     213
2728 010110  000451                          BR      319$                    ;REPORT ONLY ONE ERROR
2729
2730 010112  062701  000002          300$:   ADD     #2,R1                   ;BUMP INDEX TO TABLE
2731 010116  005761  003656                  TST     CTABLE(R1)              ;END OF TABLE ?
2732 010122  001401                          BEQ     305$                    ;YES,BRANCH
2733 010124  000764                          BR      290$                    ;YES,END
2734 010126  062700  000004          305$:   ADD     #4,R0                   ;BUMP INDEX
2735 010132  005760  003656                  TST     CTABLE(R0)              ;END OF TABLE ?
2736 010136  001411                          BEQ     307$                    ;YES,BRANCH
2737 010140  162700  000002                  SUB     #2,R0                   ;ADJUST INDEX
2738 010144  026027  003656  177777          CMP     CTABLE(R0),#-1          ;BLANK SPACE
2739 010152  001002                          BNE     306$                    ;NO,BRANCH
2740 010154  062700  000002                  ADD     #2,R0                   ;SKIP
2741 010160  000726                  306$:   BR      280$
2742
2743 010162  005000                  307$:   CLR     R0
```

```
2744 010164  005760  003656          310$:   TST     CTABLE(R0)              ;END OF TABLE ?
2745 010170  001401                          BEQ     319$                    ;YES,BRANCH
2746 010172  022760  000010  003656          CMP     #8.,CTABLE(R0)          ;A020 CATAGORY MODULE ?
2747 010200  001403                          BEQ     313$                    ;YES,BRANCH
2748 010202  062700  000002                  ADD     #2,R0                   ;BUMP INDEX
2749 010206  000766                          BR      310$                    ;BRANCH
2750
2751
2752 010210  022760  000007  003656  313$:   CMP     #7,CTABLE(R0)           ;A014 CATAGORY MODULE ?
2753 010216  001001                          BNE     314$                    ;NO,BRANCH
2754 010220  104213                          EMT     213
2755
2756 010222  062700  000002          314$:   ADD     #2,R0                   ;BUMP INDEX
2757 010226  005760  003656                  TST     CTABLE(R0)              ;END OF TABLE ?
2758 010232  001366                          BNE     313$                    ;NO,BRANCH
2759 010234  142777  000004  172402  319$:   BICB    #GBIT,@CSR              ;CLEAR G-GIT
2760 010242  012737  007606  000004          MOV     #TMOVEC,ERRVEC          ;RESTORE VECTOR
2761 010250  000207                          RTS     PC                      ;RETURN
2762
2763
2764                                  ;*** TIMEOUT ROUTINE
2765 010252  062737  000001  003462  320$:   ADD     #1,TADDR                ;BUMP TEST ADDRESS
2766 010260  023737  003462  002646          CMP     TADDR,IAR               ;END OF D-BUS ADDRESSES
2767 010266  001416                          BEQ     330$                    ;YES,BRANCH
2768 010270  005737  003502                  TST     TEMP                    ;MULTIPLE TRAPS ?
2769 010274  001010                          BNE     322$                    ;YES,BRANCH
2770 010276  012737  177777  003502          MOV     #-1,TEMP                ;SET 'BEEN HERE BEFORE FLAG'
2771 010304  012760  177777  003656          MOV     #-1,CTABLE(R0)          ;SET BOUNDARY IN TABLE
2772 010312  062700  000002                  ADD     #2,R0                   ;BUMP INDEX
2773 010316  012716  007676          322$:   MOV     #220$,(SP)              ;FUDGE RETURN ADDRESS
2774 010322  000006                          RTT
2775
2776 010324  012716  010032          330$:   MOV     #275$,(SP)              ;ADDRESS TO CHECK ROUTINE
2777 010330  000006                          RTT
```

```
                                        .SBTTL  DIGITAL MODULE MONITOR
010332  004737  007620          A:      JSR     PC,CLEARVAR     ;CLEAR VARIABLES
010336  012777  015612  172360          MOV     #KBINT,@KBVEC
010344  152777  000100  170606          BISB    #BIT6,@$TKS     ;ENABLE KB INTERRUPT
010352  104401  032230                  TYPE    ,MASS13         ;TYPE ADDRESS OF MUT
010356  104410                          RDOCT
010360  012637  003462                  MOV     (SP)+,TADDR     ;;POP STACK INTO TADDR
010364  023737  003462  002660          CMP     TADDR,BASE
010372  103757                          BLO     A
010374  013746  000004                  MOV     ERRVEC,-(SP)    ;SAVE ERROR VECTOR
010400  012737  010704  000004          MOV     #5$,ERRVEC      ;SET LOC 4
010406  004737  014310                  JSR     PC,KLEER        ;CLEAR THE IOCM
010412  105777  172226                  TSTB    @CSR            ;MAKE SURE IOCM IS OK
010416  001404                          BEQ     17$
010420  104010                          EMT     10
010422  012637  000004                  MOV     (SP)+,ERRVEC    ;;POP STACK INTO ERRVEC
010426  000207                          RTS     PC
010430  112777  000004  172206  17$:    MOVB    #GBIT,@CSR      ;SET GENERIC BIT
010436  117700  173020                  MOVB    @TADDR,R0       ;R0=GENERIC CODE OF MUT
010442  042700  177400                  BIC     #177400,R0
010446  012701  002546                  MOV     #MODUL,R1       ;ADDRESS OT TEST
010452  012703  002466                  MOV     #MUT,R3
010456  012702  002404                  MOV     #GENER,R2       ;TABLE OF GENERIC CODES
010462  020022                  3$:     CMP     R0,(R2)+
010464  001054                          BNE     1$
010466  011337  003456                  MOV     (R3),$MUT
010472  004737  014310                  JSR     PC,KLEER
010476  011137  003472                  MOV     (R1),DMUT
010502  005037  001214          9$:     CLR     $PASS
010506  004777  172760          4$:     JSR     PC,@DMUT        ;TEST MODULE
010512  004737  015526                  JSR     PC,CNTRC        ;CONTROL C?
010516  005737  002662                  TST     PASCNT          ;IF ZERO LOOP ON TEST
010522  001771                          BEQ     4$
010524  005237  001214                  INC     $PASS
010530  023737  002662  001214          CMP     PASCNT,$PASS
010536  001363                          BNE     4$
010540  032777  020000  170406          BIT     #BIT13,@SWR     ;INHIBIT PRINT ?
010546  001011                          BNE     7$
010550  104401  001203                  TYPE    ,$CRLF
010554  013746  002662                  MOV     PASCNT,-(SP)    ;;SAVE PASCNT FOR TYPEOUT
010560  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
010562  104401  030040                  TYPE    ,M11            ;# OF  PASSES
010566  104401  001203                  TYPE    ,$CRLF
010572  032777  040000  170354  7$:     BIT     #BIT14,@SWR     ;LOOP ?
010600  001340                          BNE     9$
010602  104401  032300                  TYPE    ,MASS15         ;END OF MODULE TEST
010606  004737  014310                  JSR     PC,KLEER
010612  000137  007240                  JMP     MONIT
010616  062701  000002          1$:     ADD     #2,R1
010622  062703  000002                  ADD     #2,R3
010626  005712                          TST     (R2)            ;END OF TABLE OF GENERIC CODES
010630  001314                          BNE     3$
010632  132700  000004                  BITB    #BIT2,R0        ;IF CSS MODULE
010636  001406                          BEQ     13$             ;- BIT2
010640  132700  000010                  BITB    #BIT3,R0        ;--AND BIT3
010644  001403                          BEQ     13$             ;--WILL BE SET
010646  104401  027731                  TYPE    ,MESCSS         ;PRINT CSS WARNING MESSAGE
```

```
2834 010652  000406                      BR      14$
2835 010654  104401  C32135        13$:   TYPE    ,MASS11          ;UNKNOWN GENERIC CODE
2836 010660  010046                      MOV     R0,-(SP)         ;;SAVE R0 FOR TYPEOUT
     010662  104402                      TYPOC                    ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2837 010664  104401  031400              TYPE    ,MASS0           ;CR,LF
2838 010670  012637  000004        14$:   MOV     (SP)+,ERRVEC     ;RESTORE ERRVEC
2839 010674  004737  014310              JSR     PC,KLEER
2840 010700  000137  007240              JMP     MONIT
2841 010704  022626                5$:   CMP     (SP)+,(SP)+      ;RESTORE STACK POINTER
2842 010706  104024                      EMT     24
2843 010710  012637  000004              MOV     (SP)+,ERRVEC
2844 010714  000137  007240              JMP     MONIT
```

```
                                        .SBTTL  SET SOFTWARE SWITCH REGISTER

010720  023727  001154  177570  T:      CMP     SWR,#DSWR       ;IS THERE A HARDWARE SWR ?
010726  001003                          BNE     1$              ;NO
010730  104401  030564                  TYPE    ,M23            ;''USE HARDWARE SWR''
010734  000407                          BR      2$              ;RETURN
010736  104401  030403          1$:     TYPE    ,M22            ;SWICHES OPTIONS  SW REG =
010742  104410                          RDOCT
010744  012677  170204                  MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
010750  104401  031400                  TYPE    ,MASSO          ;CR,LF
010754  000137  007240          2$:     JMP     MONIT
```

```
                                        .SBTTL  AUTO TEST MONITOR


                                                ;THIS PART OF A PROGRAM IS A MONITOR FOR AUTOMATIC TEST.
                                                ;FIRST IT TESTS THE IOCM, THEN IT GENERATES A TABLE AND B TABLE.
                                                ;THEN IT TESTS EACH INDIVIDUAL I/O MODULE CONNECTED TO DBUS.
                                                ;REGISTERS R0 & R1 SERVE AS THE POINTERS FOR MUT

010760                                  AUTO:
010760  012737  010776  000004                  MOV     #40$,ERRVEC
010766  052777  000100  171646                  BIS     #BIT6,@CLKADR   ;ENABLE LINE CLOCK INTERRUPT (UNIBUS COMPUTERS)
010774  000401                                  BR      41$
010776  022626                          40$:    CMP     (SP)+,(SP)+     ;TRAP HERE IF LSI11
011000  012737  007606  000004          41$:    MOV     #TMOVEC,ERRVEC  ;RESTORE ERROR VECTOR
011006  004737  013230                          JSR     PC,TIOCM
011012  004737  014310                          JSR     PC,KLEER        ;CLEAR THE IOCM
011016  105777  171622                          TSTB    @CSR            ;MAKE SURE IOCM IS OK
011022  001402                                  BEQ     7$
011024  104010                                  EMT     10
011026  000000                                  HALT                    ;STOP HERE ! BIG PROBLEMS
011030  013746  000004                  7$:     MOV     @#ERRVEC,-(SP)  ;SAVE ERROR VECTOR
011034  012737  011206  000004                  MOV     #1$,ERRVEC      ;SET ERROR VECTOR
011042  013700  002660                          MOV     BASE,R0         ;R0=171000
011046  112777  000004  171570                  MOVB    #GBIT,@CSR      ;SET GENERIC CODE ENABLE
011054  012702  001456                          MOV     #ATABL,R2
011060  012705  001716                          MOV     #BTABL,R5
011064  111001                          5$:     MOVB    (R0),R1         ;R1=GENERIC CODE,READ GENERIC CODE
011066  005037  003436                          CLR     CSS             ;CLEAR CSS FLAG
011072  042701  177400                          BIC     #177400,R1
011076  004737  015230                          JSR     PC,TABLE        ;FIND WHAT I/O IT IS
011102  005764  002240                          TST     BYTE(R4)        ;VALID MODULE ?
011106  001015                                  BNE     3$              ;YES,BRANCH
011110  005200                                  INC     R0              ;ADD 1 TO ADDRESS
011112  132701  000004                          BITB    #BIT2,R1        ;IF CSS MODULE
011116  001413                                  BEQ     13$             ;- BIT2
011120  132701  000010                          BITB    #BIT3,R1        ;--AND BIT3
011124  001410                                  BEQ     13$             ;---WILL BE SET
011126  105737  003436                          TSTB    CSS             ;PRINT CSS WARNING MESSAGE ?
011132  001005                                  BNE     13$             ;NO,BRANCH
011134  104401  027731                          TYPE    ,MESCSS         ;TYPE MESSAGE
011140  000402                                  BR      13$             ;BRANCH
011142  066400  002240                  3$:     ADD     BYTE(R4),R0     ;ADD # OF BYTES TO ADDRESS
011146  020037  002646                  13$:    CMP     R0,IAR
011152  103744                                  BLO     5$
011154  005012                                  CLR     (R2)
011156  022702  001456                          CMP     #ATABL,R2
011162  001017                                  BNE     6$
011164  012637  000004                          MOV     (SP)+,ERRVEC    ;;POP STACK INTO ERRVEC
011170  142777  000004  171446                  BICB    #GBIT,@CSR
011176  104401  034307                          TYPE    ,MASS50         ;NO I/O
011202  000137  011526                          JMP     26$
011206  022626                          1$:     CMP     (SP)+,(SP)+
011210  005200                                  INC     R0
011212  020037  002646                          CMP     R0,IAR
011216  103722                                  BLO     5$
011220  005015                                  CLR     (R5)
011222  012637  000004                  6$:     MOV     (SP)+,@#ERRVEC  ;START TESTING INDIVIDUAL I/O
```

```
2921 011226  142777  000004  171410          BICB    #GBIT,@CSR      ;CLEAR G BIT
2922 011234  004737  013230                  JSR     PC,TIOCM
2923 011240  005000                          CLR     R0
2924 011242  005001                  22$:     CLR     R1
2925 011244  005737  001716          23$:     TST     BTABL                   ;I/O MODULES ON D-BUS ?
2926 011250  001526                          BEQ     26$                     ;NO,BRANCH
2927 011252  132760  000004  001716          BITB    #BIT2,BTABL(R0)         ;IF CSS MODULE
2928 011260  001406                          BEQ     2$                      ;-BIT2 AND
2929 011262  132760  000010  001716          BITB    #BIT3,BTABL(R0)         ;--BIT3
2930 011270  001402                          BEQ     2$                      ;--WILL BE SET
2931 011272  000137  011510                  JMP     30$
2932 011276  026061  001716  002404  2$:     CMP     BTABL(R0),GENER(R1)     ;GENERIC CODE MATCH ?
2933 011304  001415                          BEQ     21$                     ;YES,BRANCH
2934 011306  062701  000002                  ADD     #2,R1                   ;BUMP INDEX
2935 011312  005761  002404                  TST     GENER(R1)               ;END OF VALID GENERIC CODE TABLE?
2936 011316  001352                          BNE     23$                     ;NO,BRANCH
2937 011320  016037  001716  003576          MOV     BTABL(R0),TEMP2         ;BAD GENERIC CODE OF MODULE
2938 011326  016037  001456  003600          MOV     ATABL(R0),TEMP3         ;ADDRESS OF OFFENDING MODULE
2939 011334  104057                          EMT     57
2940 011336  000464                          BR      30$                     ;BRANCH
2941 011340  004737  007620          21$:     JSR     PC,CLEARVAR     ;CLEAR VARIABLES BEFORE TESTING
2942 011344  016137  002546  003472          MOV     MODUL(R1),DMUT
2943 011352  016037  001456  003462          MOV     ATABL(R0),TADDR
2944 011360  016137  002466  003456          MOV     MUT(R1),$MUT
2945 011366  010146                          MOV     R1 -(SP)        ;;PUSH R1 ON STACK
2946 011370  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
2947 011372  004777  172074                  JSR     PC,@DMUT        ;TEST THIS MODULE
2948 011376  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
2949 011400  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
2950 011402  022761  000014  002466          CMP     #14,MUT(R1)             ;WAS IT A014
2951 011410  001037                          BNE     30$
2952 011412  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
2953 011414  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
2954 011416  013746  000004                  MOV     ERRVEC,-(SP)    ;;PUSH ERRVEC ON STACK
2955 011422  012737  011674  000004          MOV     #31$,ERRVEC
2956 011430  012737  000040  003450          MOV     #40,MXNUM               ;SET FIRST MUX NUMBER
2957 011436  152777  000001  171200  32$:     BISB    #RBIT,@CSR
2958 011444  113777  003450  172136          MOVB    MXNUM,@STAT1
2959 011452  105777  172132                  TSTB    @STAT1                  ;CHECK IF RESPONDS
2960 011456  004737  064312                  JSR     PC,MUX1                 ;TEST MUX
2961 011462  062737  000040  003450  33$:     ADD     #40,MXNUM               ;DO NEXT MUX
2962 011470  022737  000400  003450          CMP     #400,MXNUM              ;LAST ONE?
2963 011476  001357                          BNE     32$
2964 011500  012637  000004                  MOV     (SP)+,ERRVEC    ;;POP STACK INTO ERRVEC
2965 011504  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
2966 011506  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
2967 011510  062700  000002          30$:     ADD     #2,R0
2968 011514  005760  001716                  TST     BTABL(R0)
2969 011520  001250                          BNE     22$
2970 011522  004737  015526                  JSR     PC,CNTRC        ;CONTROL C ?
2971 011526  004737  014310          26$:     JSR     PC,KLEER
2972 011532  132737  000001  001226          BITB    #BIT0,$ENV      ;RUNNING UNDER APT?
2973 011540  001070                          BNE     LOGIC           ;YES
2974 011542  005737  002662                  TST     PASCNT          ;PASCNT = 0 ?
2975 011546  001632                          BEQ     25$             ;YES, LOOP FOREVER
2976 011550  005237  001214                  INC     $PASS
2977 011554  005777  171060                  TST     @XXDP           ;UNDER XXDP ?
```

```
2978 011560  001004                          BNE     50$             ;YES
2979 011562  023737  001214  002662          CMP     $PASS,PASCNT    ;IS IT LAST $PASS ?
2980 011570  001221                          BNE     25$             ;NO
2981 011572  032777  020000  167354  50$:    BIT     #BIT13,@SWR     ;YES, INHIBIT PRINT ?
2982 011600  001007                          BNE     20$             ;YES
2983 011602  013746  001214                  MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
     011606  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2984 011610  104401  030040                  TYPE    ,M11            ;# OF  PASSES COMPLETED
2985 011614  104401  001203                  TYPE    ,$CRLF
2986 011620  032777  040000  167326  20$:    BIT     #BIT14,@SWR     ;DO WE LOOP ON TESTS
2987 011626  001404                          BEQ     24$             ;NO
2988 011630  005037  001214                  CLR     $PASS           ;YES, CLEAR $PASS
2989 011634  000137  011234                  JMP     25$             ;LOOP
2990 011640  004737  014310          24$:    JSR     PC,KLEER        ;CLEAR IOCM
2991 011644  005777  170770                  TST     @XXDP           ;UNDER XXDP ?
2992 011650  001403                          BEQ     51$             ;NO
2993 011652  005337  001214                  DEC     $PASS           ;YES, FIX $PASS
2994 011656  000421                          BR      LOGIC           ;GO TO $EOP
2995 011660  104401  032327          51$:    TYPE    ,MASS17         ;END OF SYSTEM TEST
2996 011664  005037  002664                  CLR     AFLAG
2997 011670  000137  007240                  JMP     MONIT
2998
2999
3000                                         ;TIMEOUT TRAPS HERE WITH PC & PSW ON STACK
3001 011674  152777  000001  170742  31$:    BISB    #RBIT,@CSR
3002 011702  013716  003610                  MOV     STAT1,(SP)      ;SAVE ADDRESS
3003 011706  042716  000003                  BIC     #3,(SP)         ;MAKE MODULE BASE ADDRESS
3004 011712  005776  000000                  TST     @(SP)           ;CLEAR THE MODULE
3005 011716  022626                          CMP     (SP)+,(SP)+     ;CLEAN STACK
3006 011720  000660                          BR      33$
```

```
   3008 011722                          LOGIC:
                                        .SBTTL  END OF PASS ROUTINE
                                        ;;*****************************************************************
                                        ;*INCREMENT THE PASS NUMBER ($PASS)
                                        ;*IF THERES A MONITOR GO TO IT
                                        ;*IF THERE ISN'T JUMP TO AUTO
        011722                          $EOP:
        011722  000004                          SCOPE
        011724  005037  001116                  CLR     $TSTNM          ;;ZERO THE TEST NUMBER
        011730  005237  001214                  INC     $PASS           ;;INCREMENT THE PASS NUMBER
        011734  042737  100000  001214          BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
        011742  005327                          DEC     (PC)+           ;;LOOP?
        011744  000001                  $EOPCT: .WORD   1
        011746  003013                          BGT     $DOAGN          ;;YES
        011750  012737                          MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
        011752  000001                  $ENDCT: .WORD   1
        011754  011744                          $EOPCT
        011756  013700  000042          $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
        011762  001405                          BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
        011764  000005                          RESET                   ;;CLEAR THE WORLD
        011766  004710                  $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
        011770  000240                          NOP                     ;;SAVE ROOM
        011772  000240                          NOP                     ;;FOR
        011774  000240                          NOP                     ;;ACT11
        011776                          $DOAGN:
        011776  000137                          JMP     @(PC)+          ;;RETURN
        012000  010760                  $RTNAD: .WORD   AUTO
```

```
                                        .SBTTL  LOOP TEST
                                        ;THIS TEST ENABLE FIELD ENGINEER TO CONNECT OUTPUT MODULE TO INPUT MODULE
                                        ;OUTPUT TEST PATTERNS AND READ THEM BACK
                                        ;
                                        ;
012002  104401  035306          W:      TYPE    ,MASS70         ;TYPE WARNING MESSAGE
012006  132737  000001  001226          BITB    #BIT0,$ENV      ;UNDER APT ?
012014  001404                          BEQ     7$              ;NO,BRANCH
012016  104401  035615                  TYPE    ,LOCAL          ;''LOCALMODE ONLY''
012022  000137  007240                  JMP     MONIT           ;RETURN TO MONITOR
012026  104401  032543          7$:     TYPE    ,MASS23         ;CONNECT OUTPUT TO INPUT MODULE
012032  012777  015612  170664          MOV     #KBINT,@KBVEC
012040  152777  000100  167112          BISB    #BIT6,@$TKS     ;ENABLE KB INTERRUPT
012046  104401  032464                  TYPE    ,MASS21         ;TYPE ADDRESS OF OUTPUT MUT
012052  104410                          RDOCT
012054  012637  003462                  MOV     (SP)+,TADDR     ;;POP STACK INTO TADDR
012060  104401  032514                  TYPE    ,MASS22         ;TYPE ADDRESS OF INPUT MUT
012064  104410                          RDOCT
012066  012637  003510                  MOV     (SP)+,TEMP1     ;;POP STACK INTO TEMP1
012072  112777  000004  170544          MOVB    #GBIT,@CSR      ;SET GENERIC BIT
012100  117700  171356                  MOVB    @TADDR,R0       ;GENERIC CODE OF OUTPUT MUT
012104  020027  000041                  CMP     R0,#41          ;IS IT M6010?
012110  001051                          BNE     1$              ;NOT M6010, ERROR
012112  117701  171372                  MOVB    @TEMP1,R1       ;GENERIC CODE OF INPUT MUT
012116  020127  000141                  CMP     R1,#141         ;IS IT M5010?
012122  001403                          BEQ     2$
012124  020127  000121                  CMP     R1,#121         ;IS IT M5011?
012130  001044                          BNE     6$
012132  005037  001214          2$:     CLR     $PASS
012136  004737  016162          3$:     JSR     PC,LOPTST       ;TEST SUBROUTINE
012142  004737  015410                  JSR     PC,CLRINT       ;CLEAR ALL INTERRUPTS
012146  004737  015526                  JSR     PC,CNTRC        ;CONTROL-C?
012152  005237  001214                  INC     $PASS
012156  023737  002662  001214          CMP     PASCNT,$PASS    ;
012164  001364                          BNE     3$
012166  032777  020000  166760          BIT     #BIT13,@SWR     ;INHIBIT PRINTOUT
012174  001005                          BNE     5$
012176  013746  002662                  MOV     PASCNT,-(SP)    ;;SAVE PASCNT FOR TYPEOUT
012202  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
012204  104401  030040                  TYPE    ,M11            ;PASSES COMPLETED
012210  032777  040000  166736  5$:     BIT     #BIT14,@SWR     ;LOOP
012216  001345                          BNE     2$
012220  104401  032606                  TYPE    ,MASS25         ;END OF LOOP TEST
012224  004737  014310                  JSR     PC,KLEER
012230  000137  007240                  JMP     MONIT
012234  104401  032633          1$:     TYPE    ,MASS26         ;WRONG MODULE -OUTPUT MUST BE M6010
012240  000402                          BR      4$
012242  104401  032702          6$:     TYPE    ,MASS28         ;WRONG MODULE- INPUT MUST BE EITHER M5010 OR M5011
012246  004737  014310          4$:     JSR     PC,KLEER
012252  000137  007240                  JMP     MONIT
```

```
                                        .SBTTL  MODULE TEST
                                        .
                                                ;THIS TEST ENABLE FIELD ENGINEER TO SELECT AND OUTPUT ANY DATA
                                                ;PATTERN TO OUTPUT MODULES,MONITOR AND PRINT DATA FROM INPUT MODULES
                                                ;
                                                ;.
012256  132737  000001  001227  F:    BITB    #BIT0,$ENVM     ;UNDER APT ?
012264  001404                      BEQ     6$              ;NO,BRANCH
012266  104401  035615              TYPE    ,LOCAL          ;''LOCAL MODE ONLY''
012272  000137  007240              JMP     MONIT           ;RETURN TO MONITOR
012276  104401  035306          6$: TYPE    ,MASS70         ;TYPE WARNING MESSAGE
012302  104401  032230              TYPE    ,MASS13         ;TYPE ADDRESS OF MUT
012306  012777  015612  170410      MOV     #KBINT,@KBVEC
012314  152777  000100  166636      BISB    #BIT6,@$TKS     ;ENABLE KB INTERRUPT
012322  104410                      RDOCT
012324  012637  003462              MOV     (SP)+,TADDR     ;;POP STACK INTO TADDR
012330  013746  000004              MOV     ERRVEC, -(SP)   ;SAVE ERROR VECTOR
012334  012737  012456  000004      MOV     #5$,ERRVEC      ;SET LOC 4
012342  112777  000004  170274      MOVB    #GBIT,@CSR      ;SET GENERIC BIT
012350  117700  171106              MOVB    @TADDR,R0       ;GENERIC CODE OF MUT
012354  042700  177400              BIC     #177400,R0
012360  012701  002672              MOV     #MOD,R1         ;ADDRESS OF TEST
012364  012702  002404              MOV     #GENER,R2       ;TABLE OF GENERIC CODES
012370  020022                  3$: CMP     R0,(R2)+
012372  001013                      BNE     2$
012374  004737  014310              JSR     PC,KLEER
012400  011137  003472              MOV     (R1),DMUT
012404  004777  171062              JSR     PC,@DMUT        ;TEST MODULE
012410  104401  032423              TYPE    ,MASS20         ;TYPE CONTROL-C TO RETURN TO MONITOR
012414  004737  015526          1$: JSR     PC,CNTRC        ;CONTROL-C ?
012420  000775                      BR      1$
012422  062701  000002          2$: ADD     #2,R1
012426  062703  000002              ADD     #2,R3
012432  005711                      TST     (R1)
012434  001355                      BNE     3$
012436  104401  035250              TYPE    ,MASS67         ;DIGITAL I/O ONLY
012442  012637  000004              MOV     (SP)+,ERRVEC    ;RESTORE ERRVEC
012446  004737  014310              JSR     PC,KLEER
012452  000137  007240              JMP     MONIT
012456  022626                  5$: CMP     (SP)+,(SP)+     ;RESTORE STACK POINTER
012460  104024                      EMT     24
012462  012637  000004              MOV     (SP)+,ERRVEC
012466  000137  006360              JMP     START
```

```
3103                                    .SBTTL  MAP OF DBUS
3104
3105                                            ;THIS MAPPER WILL PRINT I.D. MESSAGES OF PCS MODULES
3106                                            ;-CONNECTED TO THE D-BUS. A SPECIAL CSS MESSAGE WILL
3107                                            ;--BE PRINTED INFORMING THE USER THAT THE MODULE IS
3108                                            ;---NOT TESTED BY THIS SOFTWARE.
3109
3110
3111
3112
3113
3114
3115
3116 012472  013746  000004          M:      MOV     ERRVEC,-(SP)    ;SAVE ERROR VECTOR
3117 012476  012737  013136  000004          MOV     #130$,ERRVEC    ;SET ERROR VECTOR
3118 012504  012777  015612  170212          MOV     #KBINT,@KBVEC
     012512  152777  000100  166440          BISB    #BIT6,@$TKS     ;ENABLE KB INTERRUPT
3119 012520  013700  002660                  MOV     BASE,R0         ;BASE ADDRESS IOCM
3120 012524  004737  014310                  JSR     PC,KLEER        ;CLEAR EVERYTHING FOR SURE
3121 012530  004737  015410                  JSR     PC,CLRINT       ;CLEAR ALL INTERRUPTS
3122 012534  112777  000004  170102          MOVB    #GBIT,@CSR      ;SET GENERIC CODE ENABLE
3123 012542  104401  001203                  TYPE    ,$CRLF
3124
3125 012546  111001                  5$:     MOVB    (R0),R1         ;READ GENERIC CODE
3126 012550  104401  031403                  TYPE    ,MASS2          ;ADDRESS
3127 012554  010046                          MOV     R0,-(SP)        ;;SAVE R0 FOR TYPEOUT
     012556  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
3128 012560  042701  177400                  BIC     #177400,R1      ;CLEAR HIGH BYTE
3129 012564  005004                          CLR     R4              ;R4 WILL BE THE INDEX
3130 012566  012703  002404                  MOV     #GENER,R3       ;STARTING ADDRESS GENERIC CODE TABLE
3131 012572  022301                  10$:    CMP     (R3)+,R1        ;GENERIC CODE MATCH ?
3132 012574  001431                          BEQ     20$             ;YES,BRANCH
3133 012576  062704  000002                  ADD     #2,R4           ;BUMP INDEX
3134 012602  005713                          TST     (R3)            ;END OF TABLE ?
3135 012604  001372                          BNE     10$             ;NO,BRANCH
3136 012606  132701  000004                  BITB    #BIT2,R1        ;IF CSS MODULE
3137 012612  001413                          BEQ     15$             ;-BIT2 AND
3138 012614  132701  000010                  BITB    #BIT3,R1        ;--BIT3
3139 012620  001410                          BEQ     15$             ;---WILL BE SET
3140 012622  105737  003436                  TSTB    CSS             ;PRINT MESSAGE ?
3141 012626  001114                          BNE     40$             ;NO,BRANCH
3142 012630  104401  027731                  TYPE    ,MESCSS         ;CSS MODULE I.D. MESSAGE
3143 012634  005237  003436                  INC     CSS             ;SET PRINT ONLY ONCE FLAG
3144 012640  000507                          BR      40$             ;BUMP D-BUS BY 1
3145
3146 012642  104401  032135          15$:    TYPE    ,MASS11         ;UNKNOWN GENERIC CODE
3147 012646  010146                          MOV     R1,-(SP)        ;GENERIC CODE OF MODULE
3148 012650  104402                          TYPOC
3149 012652  104401  001203                  TYPE    ,$CRLF
3150 012656  000500                          BR      40$             ;BUMP D-BUS ADDRESS BY 1
3151
3152 012660  016437  002156  012670  20$:    MOV     MESTBL(R4),25$  ;MODULE I.D. MESSAGE
3153 012666  104401                          TYPE                    ;PRINT IT
3154 012670  000000                  25$:    .WORD 0
3155 012672  005037  003436                  CLR     CSS             ;CLEAR CSS FLAG
3156 012676  022701  000321                  CMP     #321,R1         ;A014 SINGLE ENDED ?
3157 012702  001404                          BEQ     30$             ;YES,BRANCH
```

MAP OF DBUS

```
3158 012704  022701  000301                CMP     #301,R1         ;A014 DIFFERENTIAL MODE ?
3159 012710  001401                        BEQ     30$             ;YES,BRANCH
3160 012712  000464                        BR      45$             ;BRANCH TO ADDRESS BUMPER
3161
3162
3163                                       ;A156 AND A157 MUX MAPPER
3164 012714  012737  013142  000004  30$:  MOV     #140$,ERRVEC    ;MUX TIME-OUT ROUTINE
3165 012722  005005                        CLR     R5              ;MUX NUMBER
3166 012724  005003                        CLR     R3              ;
3167 012726  062700  000002                ADD     #2,R0           ;MUX I/O REGISTER ADDRESS
3168 012732  062703  000040          31$:  ADD     #40,R3          ;MUX NUMBER TO MAP
3169 012736  005205                        INC     R5              ;MUX NUMBER (1-7) TO PRINT OUT
3170 012740  032705  000010                BIT     #BIT3,R5        ;LAST MUX ?
3171 012744  001037                        BNE     37$             ;YES, BRANCH
3172 012746  110310                        MOVB    R3,(R0)         ;SELECT MUX
3173 012750  111001                        MOVB    (R0),R1         ;READ GENERIC CODE OF MUX
3174                                                               ;-WILL TRAP TO 4 IF NO MUX
3175 012752  042701  177400                BIC     #177400,R1      ;CLEAR HIGH BYTE
3176 012756  104401  033201                TYPE    ,MASS32         ;MUX # MESSAGE
3177 012762  010546                        MOV     R5,-(SP)        ;MUX #
3178 012764  104402                        TYPOC
3179 012766  022701  000342                CMP     #342,R1         ;A156 SINGLE ENDED ?
3180 012772  001003                        BNE     32$             ;NO,BRANCH
3181 012774  104401  033213                TYPE    ,MASS33         ;PRINT A156
3182 013000  000754                        BR      31$             ;GET NEXT MUX
3183 013002  022701  000322          32$:  CMP     #322,R1         ;A156 DIFFERENTIAL ?
3184 013006  001003                        BNE     34$             ;NO,BRANCH
3185 013010  104401  033242                TYPE    ,MASS34         ;PRINT A156 DIFF
3186 013014  000746                        BR      31$             ;GET NEXT MUX
3187
3188 013016  022701  000323          34$:  CMP     #323,R1         ;A157 ?
3189 013022  001003                        BNE     36$             ;NO,BRANCH
3190 013024  104401  033276                TYPE    ,MASS35
3191 013030  000740                        BR      31$             ;GET NEXT MUX
3192
3193 013032  010146                36$:    MOV     R1,-(SP)
3194 013034  104402                        TYPOC
3195 013036  104401  032135                TYPE    ,MASS11
3196 013042  000733                        BR      31$             ;GET NEXT MUX
3197 013044  012737  013136  000004  37$:  MOV     #130$,ERRVEC    ;RESTORE TIME VECTOR
3198 013052  062700  000002                ADD     #2,R0           ;BUMP ADDRESS OVER MUXS
3199 013056  000404                        BR      47$             ;BRANCH
3200 013060  005200                40$:    INC     R0              ;BUMP ADDRESS BY 1
3201 013062  000402                        BR      47$
3202 013064  066400  002240          45$:  ADD     BYTE(R4),R0     ;BUMP D-BUS ADDRESS BY CORRECT # OF BYTES
3203 013070  020037  002646          47$:  CMP     R0,IAR          ;END OF D-BUS ?
3204 013074  103002                        BHIS    50$             ;YES,BRANCH
3205 013076  000137  012546                JMP     5$              ;GET NEXT D-BUS ADDRESS
3206
3207 013102  012637  000004          50$:  MOV     (SP)+,ERRVEC    ;RESTORE ORGINAL VECTOR
3208 013106  004737  007642                JSR     PC,GROUP        ;CHECK GROUPINGS
3209 013112  004737  014310                JSR     PC,KLEER        ;CLEAR D-BUS
3210 013116  132737  000001  001226        BITB    #BIT0,$ENV      ;UNDER APT ?
3211 013124  001402                        BEQ     51$             ;NO,BRANCH
3212 013126  000137  007340                JMP     S               ;GO DO AUTO TEST FOR HOST MODE
3213 013132  000137  007240          51$:  JMP     MONIT           ;GOTO BEGINNING
3214
```

```
3215                                    ;;;;TIME-OUT ROUTINE FOR D-BUS MODULES
3216 013136  022626                     130$:   CMP     (SP)+,(SP)+     ;ADJUST STACK
3217 013140  000747                             BR      40$             ;BRANCH
3218
3219
3220                                    ;;;TIME-OUT ROUTINE FOR MUXS
3221                                    ;;;THE A014 MUST BE READ TO CLEAR THE MODULE
3222
3223 013142  152777  000001  167474     140$:   BISB    #RBIT,@CSR      ;SET RIF BIT
3224 013150  010016                             MOV     R0,(SP)         ;ADDRESS OF MUX NOT RESPONDING
3225 013152  042716  000003                     BIC     #3,(SP)         ;SUBTRACT 3 FROM MUX ADDRESS
3226 013156  005776  000000                     TST     @(SP)           ;READ A014 TO CLEAR MODULE
3227 013162  022626                             CMP     (SP)+,(SP)+     ;RESTORE STACK
3228 013164  000662                             BR      31$             ;READ ANOTHER MUX
```

```
                                .SBTTL  IOCM TEST


                                        ;THIS PART OF DIAGNOSTIC TEST THE IOCM
                                        ;IT HAS 9 TESTS. IT CHECKS IF ALL THE BITS OF THE CSR
                                        ;CAN BE SET AND CLEAR , CHECKS MAINTENANCE
                                        ;INTERRUPT AND CHECKS ALL ADDRESSES IN MAINTENANCE MODE

013166                          I:
013166  013746  000004                  MOV     ERRVEC,-(SP)    ;;PUSH ERRVEC ON STACK
013172  012777  015612  167524          MOV     #KBINT,@KBVEC
013200  152777  000100  165752          BISB    #BIT6,@$TKS     ;ENABLE KB INTERRUPT
013206  004737  017306                  JSR     PC,SWLOOP
013212  013230                          TIOCM                   ;DO IOCM TEST
013214  012637  000004                  MOV     (SP)+,ERRVEC    ;;POP STACK INTO ERRVEC
013220  004737  014310                  JSR     PC,KLEER
013224  000137  007240                  JMP     MONIT

                                        ;THIS TEST CHECKS IF EACH BIT OF CSR
                                        ;IS CLEAREDBY THE C BIT.  IT SETS THE CSR TO 74, SETS THE
                                        ;C BIT AND VERIFIES THE CSR IS NOW SET TO 0
                                        ;
013230                          TIOCM:
                                ;;*****************************************************************
013230  000004                  TST1:   SCOPE
013232  112737  000001  001212          MOVB    #1,$TESTN
013240  012737  013310  000004          MOV     #10$,ERRVEC
013246  112777  000074  167370          MOVB    #74,@CSR        ;SET ALL BITS
013254  004737  014310                  JSR     PC,KLEER        ;SET CBIT
013260  005037  001142                  CLR     $BDDAT
013264  005037  001140                  CLR     $GDDAT
013270  117737  167350  001142          MOVB    @CSR,$BDDAT     ;STORE CONTENTS OF CSR IN BDDAT
013276  105737  001142                  TSTB    $BDDAT          ;IS CSR CLEAR
013302  001405                          BEQ     2$              ;YES
013304  104001                          EMT     1
013306  000207                          RTS     PC              ;FATAL ERROR RETURN TO MONITOR
013310                          10$:
013310  104003                          EMT     3
013312  022626                          CMP     (SP)+,(SP)+     ;RESTORE STACK
013314  000207                          RTS     PC
013316                          2$:
```

```
                                                ;THIS TEST CHECKS E BIT
                                        ;;*****************************************************************
013316  000004                        TST2:   SCOPE
3269 013320  112737  000002  001212           MOVB    #2,$TESTN
3270 013326  012737  000100  001140           MOV     #EBIT,$GDDAT    ;LOAD $GDDAT WITH BIT TO BE SET IN CSR
3271 013334  113737  030034  022724           MOVB    M7,EM2X         ;SET EBIT IN ERROR MESSAGE
3272 013342  004737  015260                    JSR     PC,BITSET       ;GO LOAD CSR AND CHK FOR BITS SET
3273 013346  104002                           EMT     2
3274 013350  005037  001140                    CLR     $GDDAT          ;RETURN HERE IF NO ERROR
3275 013354  004737  015260                    JSR     PC,BITSET
3276 013360  104002                           EMT     2
3277
3278                                            ;THIS TEST CHECKS M BIT
3279                                    ;;*****************************************************************
013362  000004                        TST3:   SCOPE
3280                                            ;SETTING THE M BIT CAUSES THE F BIT TO BE SER
3281
3282 013364  112737  000003  001212           MOVB    #3,$TESTN
3283 013372  012737  000240  001140           MOV     #MBIT!FBIT,$GDDAT;LOAD WITH BITS TO BE SET IN CSR
3284 013400  113737  030032  022724           MOVB    M6,EM2X         ;SET ERROR MESSAGE
3285 013406  004737  015260                    JSR     PC,BITSET       ;LOAD CSR AND VERIFY RESULTS
3286 013412  104002                           EMT     2
3287 013414  005037  001140                    CLR     $GDDAT          ;RETURN HERE IF NO ERROR AND CLR DATA TO BE SET
3288 013420  004737  015260                    JSR     PC,BITSET       ;RESET CSR TO 0 AND VERIFY RESULTS
3289 013424  104002                           EMT     2
3290
3291
3292                                            ;THIS TEST CHECKS THE D BIT
3293                                    ;;*****************************************************************
013426  000004                        TST4:   SCOPE
3294                                            ;CLEARING THE DBIT CAUSES A CLEAR WHICH COULD CAUSE THE
3295                                            ;F BIT TO BE SET
3296 013430  112737  000004  001212           MOVB    #4,$TESTN
3297 013436  012737  000020  001140           MOV     #DBIT,$GDDAT
3298 013444  113737  030030  022724           MOVB    M5,EM2X         ;FIX ERROR MESSAGE
3299 013452  004737  015260                    JSR     PC,BITSET       ;SET D BIT
3300 013456  104002                           EMT     2
3301 013460  005037  001140                    CLR     $GDDAT
3302 013464  004737  015260                    JSR     PC,BITSET
3303 013470  104002                           EMT     2
3304 013472  005037  001140                    CLR     $GDDAT          ;INITIALIZE EXPECTED DATA.
3305 013476  112777  000074  167140           MOVB    #74,@CSR        ;SET FEW BITS AT CSR
3306 013504  142777  000020  167132           BICB    #DBIT,@CSR      ;CLEAR DBIT
3307 013512  013737  002636  003446           MOV     K,ZLOOP
013520  005037  003444          65$:    CLR     YLOOP           ;WAIT
013524  005237  003444          64$:    INC     YLOOP
013530  023727  003444  000007          CMP     YLOOP,#7
013536  001372                          BNE     64$
013540  005337  003446                  DEC     ZLOOP
013544  001365                          BNE     65$
3308 013546  117737  167072  001142           MOVB    @CSR,$BDDAT
3309 013554  132737  000177  001142           BITB    #177,$BDDAT     ;CSR = 0 ?
3310 013562  001401                          BEQ     1$              ;BR IF YES - ALL OK
3311 013564  104016                          EMT     16
3312 013566  004737  014310          1$:     JSR     PC,KLEER        ;SET CLEAR IN THE CSR
```

```
                                        ;THIS TEST CHECKS T BIT
                                ;;*****************************************************************
013572  000004                TST5:   SCOPE
3316 013574  112737  000005  001212          MOVB    #5,$TESTN
3317 013602  012737  000010  001140          MOV     #TBIT,$GDDAT    ;LOAD $GDDAT WITH DATA TO BE SET IN CSR
3318 013610  113737  030026  022724          MOVB    M4,EM2X         ;SET ERROR MESSAGE
3319 013616  004737  015260                  JSR     PC,BITSET       ;SET AND CHECK T BIT
3320 013622  104002                         EMT     2
3321 013624  005037  001140                  CLR     $GDDAT          ;LOAD $GDDAT WITH 0 TO BE SET IN CSR
3322 013630  004737  015260                  JSR     PC,BITSET       ;ZERO CSR AND VERIFY RESULTS
3323 013634  104002                         EMT     2
3324 013636  004737  015410                  JSR     PC,CLRINT       ;CLEAR ALL INTERRUPT CAUSED BY T BIT
3325
3326                                    ;THIS TEST CHECKS G BIT
3327                            ;;*****************************************************************
013642  000004                TST6:   SCOPE
3328 013644  112737  000006  001212          MOVB    #6,$TESTN
3329 013652  012737  000004  001140          MOV     #GBIT,$GDDAT    ;LOAD $GDDAT WITH DATA TO BE SET IN CSR
3330 013660  113737  030024  022724          MOVB    M3,EM2X         ;FIX ERROR MESSAGE
3331 013666  004737  015260                  JSR     PC,BITSET       ;SET AND CHECK G BIT
3332 013672  104002                         EMT     2
3333 013674  005037  001140                  CLR     $GDDAT          ;LOAD $GDDAT WITH 0 TO BE SET IN CSR
3334 013700  004737  015260                  JSR     PC,BITSET       ;ZERO CSR AND VERIFY RESULTS
3335 013704  104002                         EMT     2
3336
3337
3338                                    ;THIS TEST CHECKS RBIT
3339                            ;;*****************************************************************
013706  000004                TST7:   SCOPE
3340 013710  112737  000007  001212          MOVB    #7,$TESTN
3341 013716  012737  000001  001140          MOV     #RBIT,$GDDAT    ;LOAD $GDDAT WITH DATA TO BE SET IN CSR
3342 013724  113737  030022  022724          MOVB    M2,EM2X         ;SET ERROR MESSAGE
3343 013732  004737  015260                  JSR     PC,BITSET       ;SET BIT IN CSR AND VERIFY RESULTS
3344 013736  104002                         EMT     2
3345 013740  005037  001140                  CLR     $GDDAT          ;LOAD $GDDAT WITH 0 TO BE LOADED INTO CSR
3346 013744  004737  015260                  JSR     PC,BITSET       ;ZERO CSR AND VERIFY RESULTS
3347 013750  104002                         EMT     2
```

```
                                        ;THIS TEST CHECKS ALL BITS OF DBUS IN A MAINTENANCE MODE.
                                        ;IF MBIT IS SET AND CPU ADDRESSES ANY LOCATION BETWEEN
                                        ;171000 AND 171375 IT SHOULD READ BACK A LOWER
                                        ;BYTE OF AN ADDRESS.


                                ;;*****************************************************************
                                ;;
013752  000004                  TST10:  SCOPE
3356 013754  112737  000010  001212    MOVB    #10,$TESTN
3357 013762  112777  000040  166654    MOVB    #MBIT,@CSR      ;SET MAINTENANCE MODE
3358 013770  013700  002660            MOV     BASE,R0
3359 013774  005001                    CLR     R1
3360 013776  005037  001140            CLR     $GDDAT
3361 014002  005037  001142            CLR     $BDDAT
3362 014006  111001              1$:   MOVB    (R0),R1         ;READ FIRST ADDRESS
3363 014010  042701  177400            BIC     #177400,R1
3364 014014  120001                    CMPB    R0,R1           ;CHECK IF DBUS=ADDRESS
3365 014016  001405                    BEQ     2$
3366 014020  110037  001140            MOVB    R0,$GDDAT
3367 014024  110137  001142            MOVB    R1,$BDDAT
3368 014030  104004                    EMT     4
3369 014032  005200              2$:   INC     R0
3370 014034  122700  000376            CMPB    #376,R0         ;IS IT THE LAST ADDRESS
3371 014040  001362                    BNE     1$
3372 014042  004737  014310            JSR     PC,KLEER                ;CLEAR IOCM
```

```
                                        ;THIS TEST WILL CHECK MAINTENANCE INTERRUPT
                                        ;IF MBIT & EBIT ARE SET IOCM WILL GENERATE AN INTERRUPT
                                        ;AT LOCATION 234 AND IAR WILL HAVE LOWER BYTE
                                        ;OF CSR ADDRESS


                                ;;*****************************************************************
                                ;
                                TST11:  SCOPE
014046  000004
3381 014050  112737  000011  001212    MOVB    #11,$TESTN
3382 014056  012777  014166  166564    MOV     #5$,@VECTO      ;SET VECTOR ADDRESS
3383 014064  012777  000340  166560    MOV     #PR7,@VECTOA    ;SET VECTOR+2 ADDRESS
3384 014072  013746  000004            MOV     ERRVEC,-(SP)    ;;PUSH ERRVEC ON STACK
3385 014076  012737  014302  000004    MOV     #8$,ERRVEC
3386 014104  012746  000000            MOV     #PR0,-(SP)      ;SET PSW TO PRIORITY 0
     014110  012746  014116            MOV     #64$,-(SP)
     014114  000002                    RTI
     014116  000240                64$:NOP
3387 014120  112777  000140  166516 1$:MOVB    #EBIT!MBIT,@CSR;START INTERRUPT
3388 014126  013737  002636  003446    MOV     K,ZLOOP
     014134  005037  003444        66$:CLR     YLOOP           ;WAIT
     014140  005237  003444        65$:INC     YLOOP
     014144  023727  003444  177777    CMP     YLOOP,#-1
     014152  001372                    BNE     65$
     014154  005337  003446            DEC     ZLOOP
     014160  001365                    BNE     66$
3389 014162  104014                    EMT     14
3390 014164  000433                    BR      6$
3391 014166  022626                5$: CMP     (SP)+,(SP)+     ;INTERRUPT WORKED
3392 014170  117737  166452  003502    MOVB    @IAR,TEMP       ;READ ADDRESS OF INTERRUPTING MODULE
3393 014176  122737  000377  003502    CMPB    #377,TEMP       ;IAR SHOULD HAVE ALL ONES
3394 014204  001411                    BEQ     7$
3395 014206  012737  000377  001140    MOV     #377,$GDDAT
3396 014214  005037  001142            CLR     $BDDAT          ;ASSURE HIGH BYTE IS CLEAR
3397 014220  113737  003502  001142    MOVB    TEMP,$BDDAT     ;PREPARE FOR ERROR TYPEOUT
3398 014226  104006                    EMT     6
3399 014230  152777  000001  166406 7$:BISB    #RBIT,@CSR
3400 014236  105777  166402            TSTB    @CSR            ;CLEAR INTERRUPT
3401 014242  132777  000200  166374    BITB    #EBIT,@CSR      ;CHECK IF INTERRUPT CLEAR
3402 014250  001401                    BEQ     6$
3403 014252  104015                    EMT     15
3404 014254  004737  014310        6$: JSR     PC,KLEER                ;CLEAR CSR
3405 014260  012637  000004            MOV     (SP)+,ERRVEC    ;;POP STACK INTO ERRVEC
3406 014264  012746  000000            MOV     #PR0,-(SP)      ;SET PSW TO PRIORITY 0
     014270  012746  014276            MOV     #67$,-(SP)
     014274  000002                    RTI
     014276  000240                67$:NOP
3407 014300  000207                    RTS     PC
3408
3409 014302  022626                8$: CMP     (SP)+,(SP)+
3410 014304  104022                    EMT     22
3411 014306  000762                    BR      6$
```

```
3413 014310                            KLEER:

                                               ;NOTE: IN CLEARING THE CSR IT IS NECESSARY TO DO IT 2 TIMES SINCE
                                               ;THE FIRST CLEAR CAN CAUSE A CHANGE OF STATE IN CERTAIN MODULES
                                               ,WHICH WILL SET THE FBIT.  THE 2ND CLEAR WOULD RESET THE F BIT
                                               ;IN THE EVENT THAT IT DID GET SET

     014310  152777  000002  166326                BISB    #CBIT,@CSR      ;SET C BIT
     014316  013737  002636  003446                MOV     K,ZLOOP
     014324  005037  003444          65$:          CLR     YLOOP           ;WAIT
     014330  005237  003444          64$:          INC     YLOOP
     014334  023727  003444  000007                CMP     YLOOP,#7
     014342  001372                                BNE     64$
     014344  005337  003446                        DEC     ZLOOP
     014350  001365                                BNE     65$
     014352  152777  000002  166264                BISB    #CBIT,@CSR      ;DO IT AGAIN
     014360  013737  002636  003446                MOV     K,ZLOOP
     014366  005037  003444          67$:          CLR     YLOOP           ;WAIT
     014372  005237  003444          66$:          INC     YLOOP
     014376  023727  003444  000007                CMP     YLOOP,#7
     014404  001372                                BNE     66$
     014406  005337  003446                        DEC     ZLOOP
     014412  001365                                BNE     67$
3414 014414  000207                                RTS     PC              ;AND RETURN
3415 014416                            DALLY:
3416 014416  013737  002636  003446                MOV     K,ZLOOP
     014424  005037  003444          65$:          CLR     YLOOP           ;WAIT
     014430  005237  003444          64$:          INC     YLOOP
     014434  023727  003444  000200                CMP     YLOOP,#200
     014442  001372                                BNE     64$
     014444  005337  003446                        DEC     ZLOOP
     014450  001365                                BNE     65$
3417 014452  000207                                RTS     PC
3418
3419
3420                                           ;*****************************************************************
3421                                           ; THIS ROUTINE IS USED TO DETERMINE THE VALUE OF K.
3422                                           ; THIS CONSTANT K WILL BE USED IN THE DELAY MACRO.
3423                                           ; THE INNER LOOP OF THE DELAY MACRO WILL BE DONE K TIMES.
3424                                           ; EACH INNER LOOP IN THE DELAY MACRO WILL TAKE ~ 32 MICROSEC.
3425                                           ;*****************************************************************
3426
3427 014454  017746  166206          KCLOCK:  MOV     @CLKVC,-(SP)    ;SAVE OLD VECTOR
3428 014460  017746  166204                   MOV     @CLKVCA,-(SP)   ;SAVE OF PSW
3429 014464  012777  015402  166174            MOV     #COUNT,@CLKVC   ;NEW INTERRUPT VECTOR
3430 014472  012777  000300  166170            MOV     #300,@CLKVCA    ;NEW PSW
3431 014500  005037  003476                   CLR     CLK
3432 014504  005037  002636                   CLR     K
3433 014510  012746  000000                   MOV     #PRO,-(SP)      ;NEW PSW
3434 014514  012746  014522                   MOV     #6$,-(SP)       ;NEW PC
3435 014520  000002                           RTI
3436 014522  012737  000005  003446  6$:      MOV     #5,ZLOOP
3437 014530  005237  003646        7$:       INC     XLOOP
3438 014534  001375                          BNE     7$
3439 014536  005337  003446                   DEC     ZLOOP
3440 014542  001372                          BNE     7$
3441 014544  005737  003476                   TST     CLK
```

```
   3442 014550  001020                        BNE     1$
   3443 014552  104401  014560                TYPE    ,65$            ;;TYPE ASCIZ STRING
        014556  000414                        BR      64$             ;;GET OVER THE ASCIZ
                                       ;;65$:  .ASCIZ  <CR><LF>/TURN ON LINE CLOCK./<CR><LF>
        014610                         64$:
   3444 014610  000744                        BR      6$
   3445
   3446 014612  005037  003476          1$:   CLR     CLK
   3447 014616  022737  000001  003476  8$:   CMP     #1,CLK          ;WAIT FOR FIRST CLOCK INTERRUPT
   3448 014624  001374                        BNE     8$              ;
   3449 014626  005037  003476                CLR     CLK             ;
   3450 014632  005237  002636          2$:   INC     K               ;STAY IN THIS LOOP FOR 16.3 MSEC
   3451 014636  022737  000001  003476        CMP     #1,CLK          ;
   3452 014644  001372                        BNE     2$
   3453 014646  005000                        CLR     R0
   3454 014650  012700  000011                MOV     #9.,R0          ;SET UP FOR 9 SHIFTS RIGHT
   3455 014654  006237  002636          3$:   ASR     K               ;DIVIDE K BY 1000
   3456 014660  005300                        DEC     R0
   3457 014662  005700                        TST     R0
   3458 014664  001373                        BNE     3$
   3459 014666  005737  002636          4$:   TST     K               ;DON'T LET K BE = 0 OR LESS THAN 0
   3460 014672  003003                        BGT     5$
   3461 014674  012737  000001  002636        MOV     #1,K
   3462 014702  012677  165762          5$:   MOV     (SP)+,@CLKVCA
   3463 014706  012677  165754                MOV     (SP)+,@CLKVC
   3464 014712  000207                        RTS     PC
```

```
3466 014714                          F5010:
3467 014714                          F5011:
3468 014714  012737  000004  003460          MOV     #4,BYTNUM       ;# OF BYTES PER MODULE
3469 014722  004737  016044                  JSR     PC,MONDAT       ;SUBROUTINE TO MONITOR DATA CONTINUOUSLY
3470
3471 014726                          F5012:
3472 014726  012737  000002  ) 3460          MOV     #2,BYTNUM       ;# OF BYTES PER MODULE
3473 014734  004737  016044                  JSR     PC,MONDAT
3474
3475 014740                          F5013:
3476 014740  012737  000001  003460          MOV     #1,BYTNUM       ;# OF BYTES PER MODULE
3477 014746  004737  016044                  JSR     PC,MONDAT
3478
3479 014752                          F6010:
3480 014752  012737  000004  003460          MOV     #4,BYTNUM;MODULE IS 4 BYTE LONG
3481 014760  012700  000004                  MOV     #4,R0   ;SET A BYTE COUNTER
3482 014764  004737  015742                  JSR     PC,SETPTN
3483 014770  000207                          RTS     PC
3484
3485 014772                          F6012:
3486 014772                          F6013:
3487 014772  012737  000001  003460          MOV     #1,BYTNUM
3488 015000  004737  015742                  JSR     PC,SETPTN       ;ROUTINE TO OUTPUT ANY PATTERN TOOUT MODULE
3489 015004  000207                          RTS     PC
3490
3491 015006                          F6015:
3492 015006                          F6011:
3493 015006  012737  000002  003460          MOV     #2,BYTNUM       ;MODULE IS 2 BYTE LONG
3494 015014  004737  015742                  JSR     PC,SETPTN
3495 015020  000207                          RTS     PC
3496
```

```
                                        ;THIS SUBROUTINE IS USED TO CHECK IF
                                        ;A BITE OF DATA IN $GDDAT=$BDDAT.
                                        ;IT CHECKS AS MANY BYTES AS SPECIFIED
                                        ;BY     BYTNUM

                              TSTBYT:
015022  013746  003460                  MOV     BYTNUM,-(SP)    ;;PUSH BYTNUM ON STACK
015026  013746  003462                  MOV     TADDR,-(SP)     ;;PUSH TADDR ON STACK
015032  117737  166424  001142  2$:     MOVB    @TADDR,$BDDAT   ;MOV DATA TO BDDAT
015040  123737  001140  001142          CMPB    $GDDAT,$BDDAT   ;IS DATA OK
015046  001404                          BEQ     1$
015050  013737  003462  003466          MOV     TADDR,TBADDR
015056  104007                          EMT     7
015060  005237  003462          1$:     INC     TADDR           ;NEXT BYTE
015064  005337  003460                  DEC     BYTNUM
015070  001360                          BNE     2$
015072  012637  003462                  MOV     (SP)+,TADDR     ;;POP STACK INTO TADDR
015076  012637  003460                  MOV     (SP)+,BYTNUM    ;;POP STACK INTO BYTNUM
015102  000207                          RTS     PC

015104                        TSTONE:
015104  013746  003460                  MOV     BYTNUM,-(SP)    ;;PUSH BYTNUM ON STACK
015110  013746  003462                  MOV     TADDR,-(SP)     ;;PUSH TADDR ON STACK
015114  117737  166342  001142  2$:     MOVB    @TADDR,$BDDAT   ;MOV DATA TO BDDAT
015122  123737  001140  001142          CMPB    $GDDAT,$BDDAT   ;IS DATA OK
015130  001404                          BEQ     1$
015132  013737  003462  003466          MOV     TADDR,TBADDR
015140  104011                          EMT     11
015142  005237  003462          1$:     INC     TADDR           ;NEXT BYTE
015146  005337  003460                  DEC     BYTNUM
015152  001360                          BNE     2$
015154  012637  003462                  MOV     (SP)+,TADDR     ;;POP STACK INTO TADDR
015160  012637  003460                  MOV     (SP)+,BYTNUM    ;;POP STACK INTO BYTNUM
015164  000207                          RTS     PC
```

```
                                        ;THIS SUBROUTINE IS USED TO SET AS MANY
                                        ;BYTES AS SPECIFIED BY BYTNUM IN MUT

015166                          SETBYT:
015166  013746  003460                  MOV     BYTNUM,-(SP)    ;;PUSH BYTNUM ON STACK
015172  013746  003462                  MOV     TADDR,-(SP)     ;;PUSH TADDR ON STACK
015176  113777  001140  166256  1$:     MOVB    $GDDAT,@TADDR   ;SET DATA IN MUT
015204  005237  003462                  INC     TADDR
015210  005337  003460                  DEC     BYTNUM
015214  001370                          BNE     1$
015216  012637  003462                  MOV     (SP)+,TADDR     ;;POP STACK INTO TADDR
015222  012637  003460                  MOV     (SP)+,BYTNUM    ;;POP STACK INTO BYTNUM
015226  000207                          RTS     PC
```

```
   3545
   3546                                 ;;THIS ROUTINE IS USED IN S-COMMAND
   3547 015230  005004                  TABLE:  CLR     R4
   3548 015232  012703  002404                  MOV     #GENER,R3
   3549 015236  022301                  3$:     CMP     (R3)+,R1
   3550 015240  001404                          BEQ     1$
   3551 015242  062704  000002                  ADD     #2,R4
   3552 015246  005713                          TST     (R3)
   3553 015250  001372                          BNE     3$
   3554 015252  010022                  1$:     MOV     R0,(R2)+
   3555 015254  010125                          MOV     R1,(R5)+
   3556 015256  000207                          RTS     PC
```

```
                                        ;THIS SUBROUTINE LOADS THE CSR AND VERIFIES THAT THE VALUE SET
                                        ;GETS LOADED PROPERLY.  IF THE TEST CALLING THIS ROUTINE IS
                                        ;TEST 4 OR TEST 5, THE ROUTINE CLEARS THE FLAG BIT BEFORE CHECKING
                                        ;TAKES PLACE
                                        ;
                                        ;PRELOADS:
                                        ;       $GDDAT  <== VALUE TO BE SET IN CSR
                                        ;       $TESTN  <== THE TEST NUMBER
                                        ;
                                        ;RETURNS:
                                        ;       $BDDAT          DATA READ FROM CSR
                                        ;       IF ERROR -      THE ROUTINE RETURNS
                                        ;       IF NO ERROR     THE ROUTINE RETURNS+2
                                        ;
                                        ;

015260  113777  001140  165356  BITSET: MOVB    $GDDAT,@CSR     ;LOAD TESTED BIT
015266  005037  001142                  CLR     $BDDAT
015272  013737  002636  003446          MOV     K,ZLOOP
015300  005037  003444          65$:    CLR     YLOOP           ;WAIT
015304  005237  003444          64$:    INC     YLOOP
015310  023727  003444  000007          CMP     YLOOP,#7
015316  001372                          BNE     64$
015320  005337  003446                  DEC     ZLOOP
015324  001365                          BNE     65$
015326  117737  165312  001142          MOVB    @CSR,$BDDAT     ;READ CSR
015334  023727  001212  000005          CMP     $TESTN,#5       ;IS THIS TEST 5
015342  001403                          BEQ     3$              ;BRANCH IF YES
015344  023727  001212  000004          CMP     $TESTN,#4       ;IS THIS TEST 4
015352  001003                  3$:     BNE     2$              ;BR IF NO
015354  142737  000200  001142          BICB    #FBIT,$BDDAT    ;EXECUTE IF TEST 4 OR TEST 5
015362  123737  001140  001142  2$:     CMPB    $GDDAT,$BDDAT   ;COMPARE THEM
015370  001002                          BNE     1$
015372  062716  000002                  ADD     #2,(SP)         ;NO ERROR
015376  000207                  1$:     RTS     PC

                                        ;THIS IS INTERRUPT RUTINE FOR LINE CLOCK
                                        ;WHEN DIAGNOSTIC DOES NOT USE IT

015400                          NOCLK:
015400  000002                          RTI

                                        ;THIS IS INTERRUPT RUTINE FOR CLOCK TO COUNT TICKS

015402  005237  003476          COUNT:  INC     CLK
015406  000002                          RTI
```

```
3600                                                    ;THIS SUBROUTINE CLEARS UNEXPECTED INTERRUPTS FROM DBUS
3601
3602 015410  013737  002660  003600  CLRINT: MOV     BASE,TEMP3      ;INITIALIZE BASE ADDRESS
3603 015416  005037  003602                  CLR     TEMP4
3604 015422                          2$:
      015422  013737  002636  003446          MOV     K,ZLOOP
      015430  005037  003444          65$:    CLR     YLOOP           ;WAIT
      015434  005237  003444          64$:    INC     YLOOP
      015440  023727  003444  000001          CMP     YLOOP,#1
      015446  001372                          BNE     64$
      015450  005337  003446                  DEC     ZLOOP
      015454  001365                          BNE     65$
3605 015456  132777  000200  165160          BITB    #FBIT,@CSR      ;IS INTERRUPT PENDING?
3606 015464  001417                          BEQ     1$              ;NO
3607 015466  117737  165154  003600          MOVB    @IAR,TEMP3
3608 015474  152777  000001  165142          BISB    #RBIT,@CSR
3609 015502  105777  166072                  TSTB    @TEMP3
3610 015506  005237  003602                  INC     TEMP4
3611 015512  032737  000400  003602          BIT     #BIT8,TEMP4
3612 015520  001740                          BEQ     2$
3613 015522  104023                          EMT     23
3614 015524  000207                  1$:     RTS     PC
3615
3616
3617                                                    ;THIS SUBROUTINE CHECKS FOR CONTROL C
3618
3619
3620 015526  132737  000001  001226  CNTRC:  BITB    #BIT0,$ENV      ;ARE WE UNDER APT ?
3621 015534  001025                          BNE     3$              ;YES,BRANCH
3622 015536  117746  163420                  MOVB    @$TKB,-(SP)
3623 015542  042716  000200                  BIC     #BIT7,(SP)      ;CLEAR PARITY BIT
3624 015546  122716  000003                  CMPB    #3,(SP)         ;CONTROL C?
3625 015552  001007                          BNE     1$              ;NO
3626 015554  004737  014310                  JSR     PC,KLEER
3627 015560  012737  007606  000004          MOV     #TMOVEC,ERRVEC
3628 015566  000137  007240                  JMP     MONIT
3629 015572  122716  000001          1$.     CMPB    #1,(SP)
3630 015576  001003                          BNE     2$
3631 015600  022626                          CMP     (SP)+,(SP)+
3632 015602  000177  166044                  JMP     @RETURN
3633 015606  005726                  2$:     TST     (SP)+
3634 015610  000207                  3$:     RTS     PC
3635
3636
3637 015612  004737  015526          KBINT:  JSR     PC,CNTRC
3638 015616  000002                          RTI
```

```
                                        ;THIS SUBROUTINE IS USED IN THE LOOP TEST
                                        ;AND IT IS SIMILAR TO THE TSTBYT ROUTINE

015620                          CHKBYT:
015620  013746  003510                  MOV     TEMP1,-(SP)     ;;PUSH TEMP1 ON STACK
015624  013746  003460                  MOV     BYTNUM,-(SP)    ;;PUSH BYTNUM ON STACK
015630  117737  165654  001142  2$:     MOVB    @TEMP1,$BDDAT   ;READ DATA FROM INPUT MODULE
015636  123737  001140  001142          CMPB    $GDDAT,$BDDAT
015644  001410                          BEQ     1$
015646  013737  003462  003466          MOV     TADDR,TBADDR
015654  012637  003460                  MOV     (SP)+,BYTNUM    ;;POP STACK INTO BYTNUM
015660  012637  003510                  MOV     (SP)+,TEMP1     ;;POP STACK INTO TEMP1
015664  000207                          RTS     PC              ;ERROR
015666  005237  003510          1$:     INC     TEMP1           ;GO TO NEXT BYTE
015672  005337  003460                  DEC     BYTNUM
015676  001354                          BNE     2$
015700  012637  003460                  MOV     (SP)+,BYTNUM    ;;POP STACK INTO BYTNUM
015704  012637  003510                  MOV     (SP)+,TEMP1     ;;POP STACK INTO TEMP1
015710  062716  000002                  ADD     #2,(R6)         ;BYPASS ERROR IN MAINLINE CODE
015714  000207                          RTS     PC


                                        ;THIS SUBROUTINE STARTS CONVERSION
                                        ;THIS SUBROUTINE WAITS FOR A <CR> AND
                                        ;MONITORS CNTRLC
015716  104406                  CRTST:  RDCHR
015720  012637  003566                  MOV (SP)+,ANSW          ;SAVE ANSWER
015724  004737  015526                  JSR     PC,CNTRC                ;CONTROL C ?
015730  122737  000015  003566          CMPB    #15,ANSW        ;A <CR> ?
015736  001367                          BNE     CRTST           ;WRONG CHARACTER, JUST WAIT
015740  000207                          RTS PC
```

```
3673                                            ;THIS SUBROUTINE ALLOWS FIELD ENGINEER TO SELECT ANY
3674                                            ;DATA PATTERN FOR OUTPUT MODULES
3675
3676
3677 015742  142777  000020  164674  SETPTN:   BICB    #DBIT,@CSR
3678 015750  142777  000004  164666            BICB    #GBIT,@CSR
3679 015756  104401  032363                    TYPE    ,MASS19           ;SELECT A DATA PATTERN
3680 015762  104410                            RDOCT
3681 015764  012637  001140                    MOV     (SP)+,$GDDAT      ;;POP STACK INTO $GDDAT
3682 015770  113777  001140  165464            MOVB    $GDDAT,@TADDR     ;OUTPUT PATTERN TO MODULE
3683 015776  005237  003462                    INC     TADDR
3684 016002  005337  003460                    DEC     BYTNUM
3685 016006  001415                            BEQ     2$
3686 016010  104401  035560          1$:       TYPE    ,MASS71
3687 016014  104410                            RDOCT
3688 016016  012637  001140                    MOV     (SP)+,$GDDAT      ;;POP STACK INTO $GDDAT
3689 016022  113777  001140  165432            MOVB    $GDDAT,@TADDR     ;OUTPUT PATTERN TO MODULE
3690 016030  005237  003462                    INC     TADDR
3691 016034  005337  003460                    DEC     BYTNUM
3692 016040  001363                            BNE     1$
3693 016042  000207                  2$:       RTS     PC
3694
3695                                            ;THIS SUBROUTINE IS USED IN FIELD TEST TO CONTINUOUSLY
3696                                            ;MONITOR DATA FROM INPUT MODULE.DATA IS PRINTED ONLY
3697                                            ;DURING FIRST PASS UNLESS THERE IS A CHANGE IN THE DATA
3698 016044  005001                  MONDAT:   CLR     R1                ;CLEAR PASS REGISTER
3699 016046  013700  003460          6$:       MOV     BYTNUM,R0         ;# OF BYTES PER MODULE
3700 016052  005300                  5$:       DEC     R0
3701 016054  117737  165402  001140  1$:       MOVB    @TADDR,$GDDAT     ;STORE DATA
3702 016062  005701                            TST     R1                ;IS IT FIRST PASS?
3703 016064  001016                            BNE     4$                ;NO
3704 016066  117760  165370  003452  2$:       MOVB    @TADDR,DBUFF(R0)          ;STORE DATA IN TABLE
3705 016074  013746  003462                    MOV     TADDR,-(SP)       ;;SAVE TADDR FOR TYPEOUT
     016100  104402                            TYPOC                     ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
3706 016102  104401  032677                    TYPE    ,MASS27           ;COLON & TAB
3707 016106  013746  001140                    MOV     $GDDAT,-(SP)      ;;SAVE $GDDAT FOR TYPEOUT
     016112  104402                            TYPOC                     ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
3708 016114  104401  031400                    TYPE    ,MASS0            ;CR & LF
3709 016120  000404                            BR      3$
3710 016122  123760  001140  003452  4$:       CMPB    $GDDAT,DBUFF(R0)          ;ANY CHANGE IN DATA ?
3711 016130  001356                            BNE     2$                ;YES
3712 016132  005237  003462          3$:       INC     TADDR             ;GO TO NEXT BYTE IN MODULE
3713 016136  005700                            TST     R0                ;LAST BYTE IN MODULE ?
3714 016140  001344                            BNE     5$                ;NO
3715 016142  163737  003460  003462            SUB     BYTNUM,TADDR      ;RESTORE ADDRESS OF MUT
3716 016150  004737  015526                    JSR     PC,CNTRC          ;TYPE CONTROL C TO RETURN TO MONITOR
3717 016154  012701  000001                    MOV     #1,R1             ;NEW PASS
3718 016160  000732                            BR      6$
```

```
3720
3721                                    ;THIS SUBROUTINE IS USED TO WRAP AROUND DATA "LOOP TEST"
3722
3723
3724 016162  004737  014310          LOPTST: JSR     PC,KLEER
3725 016166  020127  000141                  CMP     R1,#141 ;IS IT M5010?
3726 016172  001161                          BNE     1$              ;YES
3727 016174  012737  000125  001140          MOV     #125,$GDDAT
     016202  012737  000004  003460          MOV     #4,BYTNUM
     016210  004737  015166                  JSR     PC,SETBYT       ;OUTPUT DATAT PATTERN
     016214  013737  002636  003446          MOV     K,ZLOOP
     016222  005037  003444          65$:    CLR     YLOOP           ;WAIT
     016226  005237  003444          64$:    INC     YLOOP
     016232  023727  003444  001777          CMP     YLOOP,#1777
     016240  001372                          BNE     64$
     016242  005337  003446                  DEC     ZLOOP
     016246  001365                          BNE     65$
     016250  012737  000004  003460          MOV     #4,BYTNUM
     016256  004737  015620                  JSR     PC,CHKBYT       ;COMPARE DATA
     016262  104021                          EMT     21
3728 016264  012737  000252  001140          MOV     #252,$GDDAT
     016272  012737  000004  003460          MOV     #4,BYTNUM
     016300  004737  015166                  JSR     PC,SETBYT       ;OUTPUT DATAT PATTERN
     016304  013737  002636  003446          MOV     K,ZLOOP
     016312  005037  003444          67$:    CLR     YLOOP           ;WAIT
     016316  005237  003444          66$:    INC     YLOOP
     016322  023727  003444  001777          CMP     YLOOP,#1777
     016330  001372                          BNE     66$
     016332  005337  003446                  DEC     ZLOOP
     016336  001365                          BNE     67$
     016340  012737  000004  003460          MOV     #4,BYTNUM
     016346  004737  015620                  JSR     PC,CHKBYT       ;COMPARE DATA
     016352  104021                          EMT     21
3729 016354  012737  000377  001140          MOV     #377,$GDDAT
     016362  012737  000004  003460          MOV     #4,BYTNUM
     016370  004737  015166                  JSR     PC,SETBYT       ;OUTPUT DATAT PATTERN
     016374  013737  002636  003446          MOV     K,ZLOOP
     016402  005037  003444          69$:    CLR     YLOOP           ;WAIT
     016406  005237  003444          68$:    INC     YLOOP
     016412  023727  003444  001777          CMP     YLOOP,#1777
     016420  001372                          BNE     68$
     016422  005337  003446                  DEC     ZLOOP
     016426  001365                          BNE     69$
     016430  012737  000004  003460          MOV     #4,BYTNUM
     016436  004737  015620                  JSR     PC,CHKBYT       ;COMPARE DATA
     016442  104021                          EMT     21
3730 016444  012737  000000  001140          MOV     #0,$GDDAT
     016452  012737  000004  003460          MOV     #4,BYTNUM
     016460  004737  015166                  JSR     PC,SETBYT       ;OUTPUT DATAT PATTERN
     016464  013737  002636  003446          MOV     K,ZLOOP
     016472  005037  003444          71$:    CLR     YLOOP           ;WAIT
     016476  005237  003444          70$:    INC     YLOOP
     016502  023727  003444  001777          CMP     YLOOP,#1777
     016510  001372                          BNE     70$
     016512  005337  003446                  DEC     ZLOOP
     016516  001365                          BNE     71$
     016520  012737  000004  003460          MOV     #4,BYTNUM
```

```
                016526  004737  015620                  JSR     PC,CHKBYT       ;COMPARE DATA
                016532  104021                          EMT     21
        3731    016534  000560                          BR      2$
        3732    016536                          1$:
                016536  012737  000125  001140          MOV     #125,$GDDAT
                016544  012737  000002  003460          MOV     #2,BYTNUM
                016552  004737  015166                  JSR     PC,SETBYT       ;OUTPUT DATAT PATTERN
                016556  013737  002636  003446          MOV     K,ZLOOP
                016564  005037  003444          73$:    CLR     YLOOP           ;WAIT
                016570  005237  003444          72$:    INC     YLOOP
                016574  023727  003444  001777          CMP     YLOOP,#1777
                016602  001372                          BNE     72$
                016604  005337  003446                  DEC     ZLOOP
                016610  001365                          BNE     73$
                016612  012737  000002  003460          MOV     #2,BYTNUM
                016620  004737  015620                  JSR     PC,CHKBYT       ;COMPARE DATA
                016624  104021                          EMT     21
        3733    016626  012737  000252  001140          MOV     #252,$GDDAT
                016634  012737  000002  003460          MOV     #2,BYTNUM
                016642  004737  015166                  JSR     PC,SETBYT       ;OUTPUT DATAT PATTERN
                016646  013737  002636  003446          MOV     K,ZLOOP
                016654  005037  003444          75$:    CLR     YLOOP           ;WAIT
                016660  005237  003444          74$:    INC     YLOOP
                016664  023727  003444  001777          CMP     YLOOP,#1777
                016672  001372                          BNE     74$
                016674  005337  003446                  DEC     ZLOOP
                016700  001365                          BNE     75$
                016702  012737  000002  003460          MOV     #2,BYTNUM
                016710  004737  015620                  JSR     PC,CHKBYT       ;COMPARE DATA
                016714  104021                          EMT     21
        3734    016716  012737  000377  001140          MOV     #377,$GDDAT
                016724  012737  000002  003460          MOV     #2,BYTNUM
                016732  004737  015166                  JSR     PC,SETBYT       ;OUTPUT DATAT PATTERN
                016736  013737  002636  003446          MOV     K,ZLOOP
                016744  005037  003444          77$:    CLR     YLOOP           ;WAIT
                016750  005237  003444          76$:    INC     YLOOP
                016754  023727  003444  001777          CMP     YLOOP,#1777
                016762  001372                          BNE     76$
                016764  005337  003446                  DEC     ZLOOP
                016770  001365                          BNE     77$
                016772  012737  000002  003460          MOV     #2,BYTNUM
                017000  004737  015620                  JSR     PC,CHKBYT       ;COMPARE DATA
                017004  104021                          EMT     21
        3735    017006  012737  000000  001140          MOV     #0,$GDDAT
                017014  012737  000002  003460          MOV     #2,BYTNUM
                017022  004737  015166                  JSR     PC,SETBYT       ;OUTPUT DATAT PATTERN
                017026  013737  002636  003446          MOV     K,ZLOOP
                017034  005037  003444          79$:    CLR     YLOOP           ;WAIT
                017040  005237  003444          78$:    INC     YLOOP
                017044  023727  003444  001777          CMP     YLOOP,#1777
                017052  001372                          BNE     78$
                017054  005337  003446                  DEC     ZLOOP
                017060  001365                          BNE     79$
                017062  012737  000002  003460          MOV     #2,BYTNUM
                017070  004737  015620                  JSR     PC,CHKBYT       ;COMPARE DATA
                017074  104021                          EMT     21
        3736    017076  000207                  2$:     RTS     PC
```

```
3737
3738
```

```
                                        ;THIS SUBROUTINE SETS UP ADDRESSES FOR A/D A014

017100  013737  003462  003614  ADADDR: MOV     TADDR,LBYTE
017106  013746  003462                  MOV     TADDR,-(SP)     ;;PUSH TADDR ON STACK
017112  013737  003462  003616          MOV     TADDR,HBYTE
017120  005237  003616                  INC     HBYTE
017124  013737  003616  003610          MOV     HBYTE,STAT1
017132  005237  003610                  INC     STAT1
017136  013737  003610  003612          MOV     STAT1,STAT2
017144  005237  003612                  INC     STAT2
017150  013737  003462  003466          MOV     TADDR,TBADDR
017156  013746  000004                  MOV     ERRVEC,-(SP)    ;;PUSH ERRVEC ON STACK
017162  012737  017236  000004          MOV     #1$,ERRVEC
017170  105777  164420                  TSTB    @LBYTE
017174  005237  003462                  INC     TADDR
017200  105777  164412                  TSTB    @HBYTE
017204  005237  003462                  INC     TADDR
017210  105777  164374                  TSTB    @STAT1
017214  005237  003462                  INC     TADDR
017220  105777  164366                  TSTB    @STAT2
017224                          2$:
017224  012637  000004                  MOV     (SP)+,ERRVEC    ;;POP STACK INTO ERRVEC
017230  012637  003462                  MOV     (SP)+,TADDR     ;;POP STACK INTO TADDR
017234  000207                          RTS     PC

017236  022626                  1$:     CMP     (SP)+,(SP)+
017240  104026                          EMT     26
017242  012737  007606  000004          MOV     #TMOVEC,ERRVEC
017250  005205                          INC     R5
017252  000764                          BR      2$


017254  152777  000004  163362  GCODE:  BISB    #GBIT,@CSR              ;LOAD GBITE WITH GENERIC CODE
017262  117737  164322  003606          MOVB    @STAT1,GBITE
017270  042737  177400  003606          BIC     #177400,GBITE
017276  142777  000004  163340          BICB    #GBIT,@CSR
017304  000207                          RTS     PC
                                        ;THIS SUBROUTINE RUNS A TEST AND CHECKS SW13 AND SW14
017306  005037  001214          SWLOOP: CLR $PASS
017312  011637  003506                  MOV     (SP),XXX                ;GET SUBR ADDR
017316  017737  164164  003506          MOV     @XXX,XXX                ;ANOTHER LEVEL OF DEFERRED
017324  004777  164156          1$:     JSR     PC,@XXX
017330  005237  001214                  INC     $PASS
017334  023737  001214  002662          CMP     $PASS,PASCNT
017342  001370                          BNE     1$
017344  032777  020000  161602          BIT     #BIT13,@SWR
017352  001007                          BNE     2$
017354  104401  001203                  TYPE    ,$CRLF
017360  013746  002662                  MOV     PASCNT,-(SP)    ;;SAVE PASCNT FOR TYPEOUT
017364  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
017366  104401  030040                  TYPE    ,M11
017372  032777  040000  161554  2$:     BIT     #BIT14,@SWR
017400  001342                          BNE     SWLOOP
017402  062716  000002                  ADD     #2,(R6)         ;ADVANCE RETURN PC
017406  000207                          RTS     PC
```

```
 3795                                   .SBTTL  SCOPE HANDLER ROUTINE
                                        ;;**********************************************************************
                                        ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
                                        ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
                                        ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
                                        ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                        ;*SW09=1         LOOP ON ERROR
                                        ;*SW08=1         LOOP ON TEST IN SWR<7:0>
                                        ;*CALL
                                        ;*      SCOPE             ;;SCOPE=IOT
 017410                                 $SCOPE:
                                        ;#####START OF CODE FOR THE XOR TESTER#####
 017410  000416                         $XTSTR: BR      6$                ;;IF RUNNING ON THE ''XOR'' TESTER CHANGE
                                                                          ;;THIS INSTRUCTION TO A ''NOP'' (NOP=240)
 017412  013746  000004                         MOV     @#ERRVEC,-(SP)    ;;SAVE THE CONTENTS OF THE ERROR VECTOR
 017416  012737  017436  000004                 MOV     #5$,@#ERRVEC      ;;SET FOR TIMEOUT
 017424  005737  177060                         TST     @#177060          ;;TIME OUT ON XOR?
 017430  012637  000004                         MOV     (SP)+,@#ERRVEC    ;;RESTORE THE ERROR VECTOR
 017434  000446                               BR      $SVLAD            ;;GO TO THE NEXT TEST
 017436  022626                         5$:   CMP     (SP)+,(SP)+       ;;CLEAR THE STACK AFTER A TIME OUT
 017440  012637  000004                         MOV     (SP)+,@#ERRVEC    ;;RESTORE THE ERROR VECTOR
 017444  000434                               BR      7$                ;;LOOP ON THE PRESENT TEST
 017446                                 6$:;#####END OF CODE FOR THE XOR TESTER#####
 017446  032777  000400  161500               BIT     #BIT08,@SWR       ;;LOOP ON SPEC. TEST?
 017454  001421                               BEQ     2$                ;;BR IF NO
 017456  005046                               CLR     -(SP)             ;;CLEAR A TEMP. LOCATION
 017460  117716  161470                       MOVB    @SWR,(SP)         ;;PICKUP THE DESIRED TEST NUMBER
 017464  001414                               BEQ     8$                ;;BRANCH IF BAD TEST NUMBER IN SWR
 017466  022716  000011                       CMP     #11,(SP)          ;;CHECK THE NUMBER IN THE SWR
 017472  002411                               BLT     8$                ;;BRANCH IF TEST NUMBER IS OUT OF RANGE
 017474  011637  001116                       MOV     (SP),$TSTNM       ;;UPDATE THE TEST NUMBER
 017500  005316                               DEC     (SP)              ;;BACKUP BY ONE
 017502  006316                               ASL     (SP)              ;;SCALE THE TEST NUMBER AS AN INDEX
 017504  062716  017622                       ADD     #$SW08TBL,(SP)    ;;FORM THE ADDRESS OF TEST POINTER
 017510  013637  001122                       MOV     @(SP)+,$LPADR     ;;SET LOOP ADDRESS TO DESIRED TEST
 017514  000434                               BR      $OVER             ;;GO LOOP ON THE TEST
 017516  005726                         8$.   TST     (SP)+             ;;CLEAN THE BAD TEST NUMBER OFF OF THE STACK
 017520  105737  001117                 2$:   TSTB    $ERFLG            ;;HAS AN ERROR OCCURRED?
 017524  001412                               BEQ     $SVLAD            ;;BR IF NO
 017526  032777  001000  161420               BIT     #BIT09,@SWR       ;;LOOP ON ERROR?
 017534  001404                               BEQ     4$                ;;BR IF NO
 017536  013737  001124  001122         7$:   MOV     $LPERR,$LPADR     ;;SET LOOP ADDRESS TO LAST SCOPE
 017544  000420                               BR      $OVER
^17546  105037  001117                 4$:   CLRB    $ERFLG            ;;ZERO THE ERROR FLAG
 017552  105237  001116                 $SVLAD: INCB  $TSTNM            ;;COUNT TEST NUMBERS
 017556  113737  001116  001212               MOVB    $TSTNM,$TESTN     ;;SET TEST NUMBER IN APT MAILBOX
 017564  011637  001122                       MOV     (SP),$LPADR       ;;SAVE SCOPE LOOP ADDRESS
 017570  011637  001124                       MOV     (SP),$LPERR       ;;SAVE ERROR LOOP ADDRESS
 017574  005037  001174                       CLR     $ESCAPE           ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
 017600  112737  000001  001131               MOVB    #1,$ERMAX         ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
 017606  013777  001116  161342         $OVER:  MOV   $TSTNM,@DISPLAY   ;;DISPLAY TEST NUMBER
 017614  013716  001122                       MOV     $LPADR,(SP)       ;;FUDGE RETURN ADDRESS
 017620  000002                               RTI                       ;;FIXES PS
 017622                                 $SW08TBL:
         000011                         .REPT   $TN-1
 017622  013232                               .WORD   TST1+2            ;;STARTING ADDRESS OF TEST 1
 017624  013320                               .WORD   TST2+2            ;;STARTING ADDRESS OF TEST 2
```

```
017626  013564                          .WORD   TST3+2          ;;STARTING ADDRESS OF TEST 3
017630  013430                          .WORD   TST4+2          ;;STARTING ADDRESS OF TEST 4
017632  013574                          .WORD   TST5+2          ;;STARTING ADDRESS OF TEST 5
017634  013644                          .WORD   TST6+2          ;;STARTING ADDRESS OF TEST 6
017636  013710                          .WORD   TST7+2          ;;STARTING ADDRESS OF TEST 7
017640  013754                          .WORD   TST10+2         ;;STARTING ADDRESS OF TEST 10
017642  014050                          .WORD   TST11+2         ;;STARTING ADDRESS OF TEST 11
 3796                           .SBTTL  INTEGER MULTIPLY ROUTINE
                                ;;**************************************************************
                                ;*CALL
                                ;*      MOV     MULTIPLER,-(SP)
                                ;*      MOV     MULTIPLICAND,-(SP)
                                ;*      JSR     PC,@#$MULT
                                ;*      RETURN  ;;PRODUCT IS ON THE STACK
                                ;*
                                ;*      STACK   PRODUCT
                                ;*      -----   -------
                                ;*      TOP     LSB'S
                                ;*      +2      MSB'S
017644                          $MULT:
017644  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
017646  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
017650  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
017652  005046                          CLR     -(SP)           ;;CLEAR THE SIGN KEY
017654  016601  000012                  MOV     12(SP),R1       ;;GET THE MULTIPLICAND
017660  100002                          BPL     1$              ;;BR IF PLUS
017662  005216                          INC     (SP)            ;;SET THE SIGN KEY
017664  005401                          NEG     R1              ;;MAKE THE MULTIPLICAND POSTIVE
017666  016602  000014          1$:     MOV     14(SP),R2       ;;GET THE MULTIPLIER
017672  100002                          BPL     2$              ;;BR IF PLUS
017674  005316                          DEC     (SP)            ;;UPDATE THE SIGN KEY
017676  005402                          NEG     R2              ;;MAKE THE MULTIPLIER POSTIVE
017700  012746  000021          2$:     MOV     #17.,-(SP)      ;;SET THE LOOP COUNT
017704  005000                          CLR     R0              ;;SETUP FOR THE MULTIPLY LOOP
017706  103001                  3$:     BCC     4$              ;;DON'T ADD IF MULTIPLICAND = 0
017710  060200                          ADD     R2,R0
017712  006000                  4$      ROR     R0              ;;POSITION THE PARITIAL PRODUCT AND
017714  006001                          ROR     R1              ;;THE MULTIPLICAND
017716  005316                          DEC     (SP)            ;;HAS ALL BITS OF THE MULTIPLICAND BEEN DONE?
017720  001372                          BNE     3$              ;;BR IF NO
017722  022616                          CMP     (SP)+,(SP)      ;;SHOULD PRODUCT BE NEGATIVE?
017724  001403                          BEQ     5$              ;;GO TO EXIT IF NO
017726  005400                          NEG     R0              ;;YES--SO MAKE IT SO
017730  005401                          NEG     R1
017732  005600                          SBC     R0
017734  005726                  5$:     TST     (SP)+           ;;CLEAR SIGN INFO. OFF OF STACK
017736  010066  000012                  MOV     R0,12(SP)       ;;PUT THE PRODUCT ON THE STACK (MSB'S)
017742  010166  000010                  MOV     R1,10(SP)       ;;LSB'S
017746  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
017750  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
017752  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
017754  000207                          RTS     PC
 3797                           .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
                                ;
017756  011646                  $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
017760  016666  000004  000002          MOV     4(SP),2(SP)     ;;SAVE THE PS
017766  132737  000100  001227          BITB    #APTSPOOL,$ENVM ;;UNDER APT COMM MODE?
017774  001422                          BEQ     1$              ;;NO,BRANCH
017776  105777  161356                  TSTB    @$APTPTR        ;;CHAR IN BUFFER ?
020002  001011                          BNE     5$              ;;YES,BRANCH
020004  005737  001206          7$:     TST     $MSGTYP         ;;IS APT STILL BUSY ?
020010  001375                          BNE     7$              ;;YES,BRANCH
020012  012737  000010  001206          MOV     #10,$MSGTYP     ;;ASK APT FOR A RESPONSE
020020  005737  001206          6$:     TST     $MSGTYP         ;;HAS APT ANSWERED ?
020024  001375                          BNE     6$              ;;NO,BRANCH
020026  117766  161326  000004  5$:     MOVB    @$APTPTR,4(SP)  ;;GET A CHARACTER
020034  005237  001360                  INC     $APTPTR         ;;BUMP BUFFER POINTER
020040  000430                          BR      3$              ;;BRANCH AND FINISH UP
020042  105777  161112          1$:     TSTB    @$TKS           ;;WAIT FOR
020046  100375                          BPL     1$              ;;A CHARACTER
020050  117766  161106  000004          MOVB    @$TKB,4(SP)     ;;READ THE TTY
020056  042766  177600  000004          BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
020064  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
020072  001013                          BNE     3$              ;;BRANCH IF NO
020074  105777  161060          2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
020100  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
020102  117746  161054                  MOVB    @$TKB,-(SP)     ;;GET CHARACTER
020106  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
020112  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
020116  001366                          BNE     2$              ;;IF NOT DISCARD IT
020120  000750                          BR      1$              ;;YES, RESUME
020122  026627  000004  000140  3$:     CMP     4(SP),#140      ;;IS IT UPPER CASE?
020130  002407                          BLT     4$              ;;BRANCH IF YES
020132  026627  000004  000175          CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
020140  003003                          BGT     4$              ;;BRANCH IF YES
020142  042766  000040  000004          BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
020150  000002                  4$:     RTI                     ;;GO BACK TO USER
                                ;;******************************************************************
                                ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
                                ;*CALL:
                                ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
                                ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
                                ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
020152  010346                  $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
020154  012703  001236          1$:     MOV     #$APTBUF,R3     ;;STARTING ADDRESS OF BUFFER
020160  005023                  11$:    CLR     (R3)+           ;;CLEAR LOCATION
020162  022703  001306                  CMP     #$APTBUF+40.,R3 ;;END OF BUFFER ?
020166  101374                          BHI     11$             ;;NO,BRANCH
020170  012737  001236  001360          MOV     #$APTBUF,$APTPTR;;INIT BUFFER POINTER
020176  012703  020302                  MOV     #$TTYIN,R3      ;;GET ADDRESS
020202  022703  020312          2$:     CMP     #$TTYIN+8.,R3   ;;BUFFER FULL?
020206  101405                          BLOS    4$              ;;BR IF YES
020210  104406                          RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
020212  112613                          MOVB    (SP)+,(R3)      ;;GET CHARACTER
020214  122713  000177          10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
```

```
020220  001003                          BNE     3$              ;;SKIP IF NOT
020222  104401  001202          4$:     TYPE    ,$QUES          ;;TYPE A '?'
020226  000752                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
020230  111337  020300          3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
020234  104401  020300                  TYPE    ,9$
020240  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
020244  001356                          BNE     2$              ;;LOOP IF NOT RETURN
020246  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
020252  104401  001204                  TYPE    ,$LF            ;;TYPE A LINE FEED
020256  012603                          MOV     (SP)+,R3        ;;RESTORE R3
020260  011646                          MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
020262  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
020270  012766  020302  000004          MOV     #$TTYIN,4(SP)
020276  000002                          RTI                     ;;RETURN
020300     000                  9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
020301     000                          .BYTE   0               ;;TERMINATOR
020302                          $TTYIN: .BLKB   8.              ;;RESERVE 8 BYTES FOR TTY INPUT
020312     136     125     015  $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL ''U''
020315     012     000
020317     136     107     015  $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL ''G''
020322     012     000
020324     015     012     123  $MSWR:  .ASCIZ  <15><12>/SWR = /
020327     127     122     040
020332     075     040     000
020335     040     040     116  $MNEW:  .ASCIZ  /  NEW = /
020340     105     127     040
020343     075     040     000

3798                                    .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
                                        ;**********************************************************************
                                        ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
                                        ;*CHANGE IT TO BINARY.
                                        ;*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
                                        ;*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A ''?'' WILL BE TYPED
                                        ;*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
                                        ;*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
                                        ;*CALL:
                                        ;*      RDOCT                   ;;READ AN OCTAL NUMBER
                                        ;*      RETURN HERE             ;;LOW ORDER BITS ARE ON TOP OF THE STACK
                                        ;*                              ;;HIGH ORDER BITS ARE IN $HIOCT
020346  011646                  $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
020350  016666  000004  000002          MOV     4(SP),2(SP)     ;;INPUT NUMBER
020356  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
020360  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
020362  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
020364  104407                  1$:     RDLIN                   ;;READ AN ASCIZ LINE
020366  012600                          MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
020370  010037  020474                  MOV     R0,5$           ;;AND SAVE IT
020374  005001                          CLR     R1              ;;CLEAR DATA WORD
020376  005002                          CLR     R2
020400  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
020402  001420                          BEQ     3$              ;;IF ZERO GET OUT
020404  122716  000060                  CMPB    #'0,(SP)        ;;MAKE SURE THIS CHARACTER
020410  003026                          BGT     4$              ;;IS AN OCTAL DIGIT
020412  122716  000067                  CMPB    #'7,(SP)
020416  002423                          BLT     4$
020420  006301                          ASL     R1              ;;*2
020422  006102                          ROL     R2
```

```
020424  006301                          ASL     R1              ;;*4
020426  006102                          ROL     R2
020430  006301                          ASL     R1              ;;*8
020432  006102                          ROL     R2
020434  042716  177770                  BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
020440  062601                          ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
020442  000756                          BR      2$              ;;LOOP
020444  005726                  3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
020446  010166  000012                  MOV     R1,12(SP)       ;;SAVE THE RESULT
020452  010237  020504                  MOV     R2,$HIOCT
020456  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
020460  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
020462  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
020464  000002                          RTI                     ;;RETURN
020466  005726                  4$:     TST     (SP)+           ;;CLEAN PARTIAL FROM STACK
020470  105010                          CLRB    (R0)            ;;SET A TERMINATOR
020472  104401                          TYPE                    ;;TYPE UP THRU THE BAD CHAR.
020474  000000                  5$:     .WORD   0
020476  104401  001202                  TYPE    ,$QUES          ;;''?'' ''CR'' & ''LF''
020502  000730                          BR      1$              ;;TRY AGAIN
020504  000000                  $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
3799                                    .SBTTL  ERROR HANDLER ROUTINE
                                        ;;**************************************************************
                                        ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
                                        ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
                                        ;*AND GO TO $ERRTYP ON ERROR
                                        ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                        ;*SW15=1         HALT ON ERROR
                                        ;*SW13=1         INHIBIT ERROR TYPEOUTS
                                        ;*SW10=1         BELL ON ERROR
                                        ;*SW09=1         LOOP ON ERROR
                                        ;*CALL
                                        ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
020506                                  $ERROR:
020506  105237  001117                  7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
020512  001775                          BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
020514  013777  001116  160434          MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
020522  032777  002000  160424          BIT     #BIT10,@SWR     ;;BELL ON ERROR?
020530  001402                          BEQ     1$              ;;NO - SKIP
020532  104401  001176                  TYPE    ,$BELL          ;;RING BELL
020536  005237  001126                  1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
020542  011637  001132                  MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
020546  162737  000002  001132          SUB     #2,$ERRPC
020554  117737  160352  001130          MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
020562  032777  020000  160364          BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
020570  001004                          BNE     20$             ;;SKIP TYPEOUTS
020572  004737  020716                  JSR     PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
020576  104401  001203                  TYPE    ,$CRLF
020602                                  20$:
020602  122737  000001  001226          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
020610  001015                          BNE     2$              ;;NO,SKIP APT ERROR REPORT
020612  132737  000100  001227          BITB    #APTSPOOL,$ENVM ;;SPOOL ENTIRE ERROR TO HOST?
020620  001007                          BNE     10$             ;;YES
020622  113737  001130  020634          MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
020630  004737  021576                  JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
020634    000                   21$:    .BYTE   0
020635    000                           .BYTE   0
```

```
020636  000777                  22$:    BR      22$             ;;APT ERROR LOOP
020640  004737  020716          10$:    JSR     PC,$ERRTYP      ;;GO SPOOL ERROR TO HOST
020644  005777  160304          2$:     TST     @SWR            ;;HALT ON ERROR
020650  100001                          BPL     3$              ;;SKIP IF CONTINUE
020652  000000                          HALT                    ;;HALT ON ERROR!
020654  032777  001000  160272  3$:     BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
020662  001402                          BEQ     4$              ;;BR IF NO
020664  013716  001124                  MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
020670  005737  001174          4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
020674  001402                          BEQ     5$              ;;BR IF NONE
020676  013716  001174                  MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
020702                          5$:
020702  022737  011766  000042          CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
020710  001001                          BNE     6$              ;;BRANCH IF NO
020712  000000                          HALT                    ;;YES
020714                          6$:
020714  000002                          RTI                     ;;RETURN
   3800                         .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
                                ;;**************************************************************
                                ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
                                ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
                                ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
020716                          $ERRTYP:
020716  104401  001203                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
020722  010046                          MOV     R0,-(SP)        ;;SAVE R0
020724  005000                          CLR     R0              ;;PICKUP THE ITEM INDEX
020726  153700  001130                  BISB    @#$ITEMB,R0
020732  001004                          BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
                                                                ;;TYPE THE PC OF THE ERROR
020734  013746  001132                  MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
                                                                ;;ERROR ADDRESS
020740  104402                          TYPOC                   ;;GO TYPF--OCTAL ASCII(ALL DIGITS)
020742  000426                          BR      6$              ;;GET OUT
020744  005300                  1$:     DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
020746  006300                          ASL     R0              ;;      WORK FOR THE ERROR TABLE
020750  006300                          ASL     R0
020752  006300                          ASL     R0
020754  062700  004170                  ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
020760  012037  020770                  MOV     (R0)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
020764  001404                          BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
020766  104401                          TYPE                    ;;TYPE THE "ERROR MESSAGE"
020770  000000                  2$:     .WORD   0               ;;"ERROR MESSAGE" POINTER GOES HERE
020772  104401  001203                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
020776  012037  021006          3$:     MOV     (R0)+,4$        ;;PICKUP "DATA HEADER" POINTER
021002  001404                          BEQ     5$              ;;SKIP TYPEOUT IF 0
021004  104401                          TYPE                    ;;TYPE THE "DATA HEADER"
021006  000000                  4$:     .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
021010  104401  001203                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
021014  011000                  5$:     MOV     (R0),R0         ;;PICKUP "DATA TABLE" POINTER
021016  001004                          BNE     7$              ;;GO TYPE THE DATA
021020  012600                  6$:     MOV     (SP)+,R0        ;;RESTORE R0
021022  104401  001203                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
021026  000207                          RTS     PC              ;;RETURN
021030                          7$:
021030  013046                          MOV     @(R0)+,-(SP)    ;;SAVE @(R0)+ FOR TYPEOUT
021032  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
021034  005710                          TST     (R0)            ;;IS THERE ANOTHER NUMBER?
```

```
021036  001?70                          BEQ     6$              ;;BR IF NO
021040  104401  C?1046                  TYPE    ,8$             ;;TYPE TWO(2) SPACES
021044  000771                          BR      7$              ;;LOOP
021046     040     040     000  8$:     .ASCIZ  /  /            ;;TWO(2) SPACES
                                        .EVEN
 3801                                .SBTTL  TYPE ROUTINE
                                    ;;**********************************************************************
                                    ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
                                    ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
                                    ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
                                    ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
                                    ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
                                    ;*
                                    ;*CALL:
                                    ;*1) USING A TRAP INSTRUCTION
                                    ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
                                    ;*OR
                                    ;*      TYPE
                                    ;*      MESADR
                                    ;*
021052  105737  001173          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
021056  100002                          BPL     1$              ;;BR IF YES
021060  000000                          HALT                    ;;HALT HERE IF NO TERMINAL
021062  000430                          BR      3$              ;;LEAVE
021064  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
021066  017600  000002                  MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
021072  122737  000001  001226          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
021100  001011                          BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
021102  132737  000100  001227          BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
021110  001405                          BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
021112  010037  021122                  MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
021116  004737  021566                  JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
021122  000000                  61$:    .WORD   0               ;;MESSAGE ADDRESS
021124  132737  000040  001227  62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
021132  001003                          BNE     60$             ;;YES,SKIP TYPE OUT
021134  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
021136  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
021140  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
021142  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
021144  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
021150  000002                          RTI                     ;;RETURN
021152  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
021156  001430                          BEQ     8$
021160  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
021164  001006                          BNE     5$
021166  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
021170  104401                          TYPE                    ;;TYPE A CR AND LF
021172  001203                          $CRLF
021174  105037  021330                  CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
021200  000755                          BR      2$              ;;GET NEXT CHARACTER
021202  004737  021264          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
021206  123726  001172          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
021212  001350                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
021214  013746  001170                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                                ;;AND THE NULL CHAR.
021220  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
021224  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
```

TYPE ROUTINE

```
021226  004737  021264                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
021232  105337  021330                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
021236  000770                          BR      7$              ;;LOOP
                                ;HORIZONTAL TAB PROCESSOR
021240  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
021244  004737  021264          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
021250  132737  000007  021330          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
021256  001372                          BNE     9$              ;;TAB STOP
021260  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
021262  000724                          BR      2$              ;;GET NEXT CHARACTER
021264  105777  157674          $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
021270  100375                          BPL     $TYPEC
021272  116677  000002  157666          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
021300  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
021306  001003                          BNE     1$              ;;BRANCH IF NO
021310  105037  021330                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
021314  000406                          BR      $TYPEX          ;;EXIT
021316  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
021324  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
021326  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
021330  000000                  $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
021332  000207                  $TYPEX: RTS     PC
   3802                                 .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
                                        ;;**************************************************************
                                        ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
                                        ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
                                        ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
                                        ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
                                        ;*REPLACED WITH SPACES.
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
                                        ;*      TYPDS                   ;;GO TO THE ROUTINE
021334                          $TYPDS:
021334  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
021336  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
021340  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
021342  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
021344  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
021346  012746  020200                  MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
021352  016605  000020                  MOV     20(SP),R5       ;;GET THE INPUT NUMBER
021356  100004                          BPL     1$              ;;BR IF INPUT IS POS.
021360  005405                          NEG     R5              ;;MAKE THE BINARY NUMBER POS.
021362  112766  000055  000001          MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
021370  005000                  1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
021372  012703  021550                  MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
021376  112723  000040                  MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
021402  005002                  2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
021404  016001  021540                  MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
021410  160105                  3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
021412  002402                          BLT     4$              ;;BR IF DONE
021414  005202                          INC     R2              ;;INCREASE THE BCD DIGIT BY 1
021416  000774                          BR      3$
021420  060105                  4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
021422  005702                          TST     R2              ;;CHECK IF BCD DIGIT=0
021424  001002                          BNE     5$              ;;FALL THROUGH IF 0
021426  105716                          TSTB    (SP)            ;;STILL DOING LEADING 0'S?
021430  100407                          BMI     7$              ;;BR IF YES
```

```
021432  106316                  5$:     ASLB    (SP)            ;;MSD?
021434  103003                          BCC     6$              ;;BR IF NO
021436  116663  000001  177777          MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
021444  052702  000060          6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
021450  052702  000040          7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
021454  110223                          MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
021456  005720                          TST     (R0)+           ;;JUST INCREMENTING
021460  020027  000010                  CMP     R0,#10          ;;CHECK THE TABLE INDEX
021464  002746                          BLT     7$              ;;GO DO THE NEXT DIGIT
021466  003002                          BGT     8$              ;;GO TO EXIT
021470  010502                          MOV     R5,R2           ;;GET THE LSD
021472  000764                          BR      6$              ;;GO CHANGE TO ASCII
021474  105726                  8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
021476  100003                          BPL     9$              ;;BR IF NO
021500  116663  177777  177776          MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
021506  105013                  9$:     CLRB    (R3)            ;;SET THE TERMINATOR
021510  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
021512  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
021514  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
021516  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
021520  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
021522  104401  021550                  TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
021526  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
021534  012616                          MOV     (SP)+,(SP)
021536  000002                          RTI                     ;;RETURN TO USER
021540  023420                  $DTBL:  10000.
021542  001750                          1000.
021544  000144                          100.
021546  000012                          10.
021550                          $DBLK:  .BLKW   4
3804                                    .SBTTL  APT COMMUNICATIONS ROUTINE
                                        ;;**************************************************************
021560  112737  000001  022024  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
021566  112737  000001  022022  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
021574  000403                          BR      $ATYC
021576  112737  000001  022024  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
021604                          $ATYC:
021604  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
021606  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
021610  105737  022022                  TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
021614  001450                          BEQ     5$              ;;IF NOT:  BR
021616  122737  000001  001226          CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
021624  001031                          BNE     3$              ;;IF NOT:  BR
021626  132737  000100  001227          BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
021634  001425                          BEQ     3$              ;;IF NOT:  BR
021636  017600  000004                  MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
021642  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
021650  005737  001206          1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
021654  001375                          BNE     1$              ;;IF NOT:  WAIT
021656  010037  001222                  MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
021662  105720                  2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
021664  001376                          BNE     2$
021666  163700  001222                  SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
021672  006200                          ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
021674  010037  001224                  MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
021700  012737  000004  001206          MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
021706  000413                          BR      5$
```

```
021710  017637  000004  021734  3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
021716  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
021724  013746  177776                  MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
021730  004737  021052                  JSR     PC,$TYPE        ;;CALL TYPE MACRO
021734  000000                  4$:     .WORD   0
021736                          5$:
021736  105737  022024          10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
021742  001416                  BEQ     12$             ;;IF NOT:  BR
021744  005737  001226                  TST     $ENV            ;;RUNNING UNDER APT?
021750  001413                  BEQ     12$             ;;IF NOT:  BR
021752  005737  001206          11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
021756  001375                  BNE     11$             ;;IF NOT:  WAIT
021760  017637  000004  001210          MOV     @4(SP),$FATAL   ;;GET ERROR #
021766  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
021774  005237  001206                  INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
022000  105037  022024          12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
022004  105037  022023                  CLRB    $LFLG           ;;CLEAR LOG FLAG
022010  105037  022022                  CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
022014  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
022016  012600                  MOV     (SP)+,R0        ;;POP STACK INTO R0
022020  000207                  RTS     PC              ;;RETURN
022022     000          $MFLG:  .BYTE   0               ;;MESSG. FLAG
022023     000          $LFLG:  .BYTE   0               ;;LOG FLAG
022024     000          $FFLG:  .BYTE   0               ;;FATAL FLAG
                                .EVEN
            000200          APTSIZE=200
            000001          APTENV=001
            000100          APTSPOOL=100
            000040          APTCSUP=040
3805                        .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
                        ;;**********************************************************************
                        ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
                        ;*OCTAL (ASCII) NUMBER AND TYPE IT.
                        ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
                        ;*CALL:
                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                        ;*      TYPOS                   ;;CALL FOR TYPEOUT
                        ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
                        ;*      .BYTE   M               ;;M=1 OR 0
                        ;*                                      ;;1=TYPE LEADING ZEROS
                        ;*                                      ;;0=SUPPRESS LEADING ZEROS
                        ;*
                        ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
                        ;*$TYPOS OR $TYPOC
                        ;*CALL:
                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                        ;*      TYPON                   ;;CALL FOR TYPEOUT
                        ;*
                        ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
                        ;*CALL:
                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                        ;*      TYPOC                   ;;CALL FOR TYPEOUT
022026  017646  000000  $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
022032  116637  000001  022267          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
022040  112637  022271          MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
022044  062716  000002          ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
022050  000406                  BR      $TYPON
```

```
022052  112737  000001  022267  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
022060  112737  000006  022271          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
022066  112737  000005  022266  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
022074  010346                          MOV     R3,-(SP)        ;;SAVE R3
022076  010446                          MOV     R4,-(SP)        ;;SAVE R4
022100  010546                          MOV     R5,-(SP)        ;;SAVE R5
022102  113704  022271                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
022106  005404                          NEG     R4
022110  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
022114  110437  022270                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
022120  113704  022267                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
022124  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
022130  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
022132  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
022134  000404                          BR      3$              ;;GO DO MSB
022136  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
022140  006105                          ROL     R5
022142  006105                          ROL     R5
022144  010503                          MOV     R5,R3
022146  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
022150  105337  022270                  DECB    $OMODE          ;;TYPE THIS DIGIT?
022154  100025                          BPL     7$              ;;BR IF NO
022156  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
022162  001002                          BNE     4$              ;;TEST FOR 0
022164  005704                          TST     R4              ;;SUPPRESS THIS 0?
022166  001403                          BEQ     5$              ;;BR IF YES
022170  005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
022172  052703  000060                  BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
022176  052703  000040          5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
022202  132737  000001  001226          BITB    #APTENV,$ENV    ;;OPERATING UNDER APT?
022210  001403                          BEQ     9$              ;;NO,BRANCH
022212  005737  001206          10$:    TST     $MSGTYPE        ;;APT READY FOR ANOTHER DIGIT ?
022216  001375                          BNE     10$             ;;NO,BRANCH
022220  110337  022264          9$:     MOVB    R3,8$           ;;SAVE FOR TYPING
022224  104401  022264                  TYPE    ,8$             ;;GO TYPE THIS DIGIT
022230  105337  022266          7$:     DECB    $OCNT           ;;COUNT BY 1
022234  003340                          BGT     2$              ;;BR IF MORE TO DO
022236  002402                          BLT     6$              ;;BR IF DONE
022240  005204                          INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
022242  000735                          BR      2$              ;;GO DO THE LAST DIGIT
022244  012605                  6$:     MOV     (SP)+,R5        ;;RESTORE R5
022246  012604                          MOV     (SP)+,R4        ;;RESTORE R4
022250  012603                          MOV     (SP)+,R3        ;;RESTORE R3
022252  016666  000002  000004          MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
022260  012616                          MOV     (SP)+,(SP)
022262  000002                          RTI                     ;;RETURN
022264     000                  8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
022265     000                          .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
022266     000                  $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
022267     000                  $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
022270  000000                  $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
3806                                    .SBTTL  SAVE AND RESTORE R0-R5 ROUTINES
                                        ;;******************************************************************
                                        ;*SAVE R0-R5
                                        ;*CALL:
                                        ;*      SAVREG
                                        ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
```

```
                                ;*
                                ;*TOP---(+16)
                                ;*  +2---(+18)
                                ;*  +4---R5
                                ;*  +6---R4
                                ;*  +8---R3
                                ;*+10---R2
                                ;*+12---R1
                                ;*+14---R0
022272                          $SAVREG:
022272  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
022274  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
022276  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
022300  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
022302  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
022304  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
022306  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
022312  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
022316  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF CALL
022322  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
022326  000002                          RTI
                                ;*RESTORE R0-R5
                                ;*CALL:
                                ;*      RESREG
022330                          $RESREG:
022330  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
022334  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
022340  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
022344  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
022350  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
022352  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
022354  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
022356  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
022360  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
022362  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
022364  000002                          RTI
3807                            .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.
022366  010046                  $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
022370  016600  000002                  MOV     2(SP),R0        ;;GET TRAP ADDRESS
022374  005740                          TST     -(R0)           ;;BACKUP BY 2
022376  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
022400  006300                          ASL     R0              ;;POSITION FOR INDEXING
022402  016000  022422                  MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
022406  000200                          RTS     R0              ;;GO TO ROUTINE
                                ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
022410  011646                  $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
022412  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
022420  000002                          RTI                     ;;RESTORE THE PSW
                                .SBTTL  TRAP TABLE
                                ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
                                ;*BY THE "TRAP" INSTRUCTION.
                                ;       ROUTINE
```

```
                                        .       -------
022422  022410                  $TRPAD: .WORD   $TRAP2
022424  021052                          $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
022426  022052                          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
022430  022026                          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
022432  022066                          $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
022434  021334                          $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
022436  017756                          $RDCHR  ;;CALL=RDCHR    TRAP+6(104406)  TTY TYPEIN CHARACTER ROUTINE
022440  020152                          $RDLIN  ;;CALL=RDLIN    TRAP+7(104407)  TTY TYPEIN STRING ROUTINE
022442  020346                          $RDOCT  ;;CALL=RDOCT    TRAP+10(104410) READ AN OCTAL NUMBER FROM TTY
022444  022272                          $SAVREG ;;CALL=SAVREG   TRAP+11(104411) SAVE R0-R5 ROUTINE
022446  022330                          $RESREG ;;CALL=RESREG   TRAP+12(104412) RESTORE R0-R5 ROUTINE
  3808                          .SBTTL  POWER DOWN AND UP ROUTINES
                                ;;**************************************************************
                                ;POWER DOWN ROUTINE
022450  012737  022610  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
022456  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
022464  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
022466  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
022470  010246                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
022472  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
022474  010446                        MOV     R4,-(SP)        ;;PUSH R4 ON STACK
022476  010546                        MOV     R5,-(SP)        ;;PUSH R5 ON STACK
022500  017746  156450                MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
022504  010637  022614                MOV     SP,$SAVR6       ;;SAVE SP
022510  012737  022522  000024        MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
022516  000000                        HALT
022520  000776                        BR      .-2             ;;HANG UP
                                ;;**************************************************************
                                ;POWER UP ROUTINE
022522  012737  022610  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
022530  013706  022614                MOV     $SAVR6,SP       ;;GET SP
022534  005037  022614                CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
022540  005237  022614          1$:   INC     $SAVR6          ;;WAIT FOR THE INC
022544  001375                        BNE     1$              ;;OF  WORD
022546  012677  156402                MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
022552  012605                        MOV     (SP)+,R5        ;;POP STACK INTO R5
022554  012604                        MOV     (SP)+,R4        ;;POP STACK INTO R4
022556  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
022560  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
022562  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
022564  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
022566  012737  022450  000024        MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
022574  012737  000340  000026        MOV     #340,@#PWRVEC+2 ;;PRIO:7
022602  104401                        TYPE                    ;REPORT THE POWER FAILURE
022604  022616                  $PWRMG: .WORD   $POWER          ;;POWER FAIL MESSAGE POINTER
022606  000002                        RTI
022610  000000                  $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
022612  000776                        BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
022614  000000                  $SAVR6: 0                       ;;PUT THE SP HERE
022616     015     012     120  $POWER: .ASCIZ  <15><12>"POWER"
022621     117     127     105
022624     122     000
                                        .EVEN
```

```
3811                                    .NLIST BEX
3812 022626  103  102  111  EM1:    .ASCIZ  /CBIT NOT CLEARING IOCM/
3813 022655  111  117  103  EM2:    .ASCII  /IOCM TEST/<15><12>
3814 022670  103  123  122  EM2Y:   .ASCII  /CSR DATA ERROR WHEN TESTING /
3815 022724  040  040  102  EM2X:   .ASCIZ  /  BIT  /<15><12>
3816 022736  111  117  103  EM3:    .ASCIZ  /IOCM IS NOT RESPONDING/<15><12>
3817 022767  104  102  125  EM4:    .ASCIZ  /DBUS BIT IS FAILING/<15><12>
3818 023015  116  117  040  EM5:    .ASCIZ  /NO INTERRUPT /<15><12>
3819 023035  127  122  117  EM6:    .ASCIZ  /WRONG PATTERN IN IAR AFTER INTERRUPT/<15><12>
3820 023104                  EM17:   ; ** NOTE ** THIS ERROR MESSAGE IS THE SAME AS EM7
3821 023104  104  101  124  EM7:    .ASCIZ  /DATA ERROR/<15><12>
3822 023121  124  105  123  EM10:   .ASCIZ  /TEST ABORTED, IOCM ERROR/<15><12>
3823 023154  104  101  124  EM11:   .ASCIZ  /DATA NOT CLEAR AFTER 10 SEC/<15><12>
3824 023212  111  116  124  EM12:   .ASCIZ  /INTERRUPT TOO LATE   /<15><12>
3825 023242  122  111  106  EM13:   .ASCIZ  /RIF BIT NOT CLEARING COS REGISTERS /<15><12>
3826 023310  116  117  040  EM14:   .ASCIZ  /NO INTERRUPT IN MAINTENANCE MODE/<15><12>
3827 023353  122  111  106  EM15:   .ASCIZ  /RIF BIT ISN''T CLEARING INTERRUPT/<15><12>
3828 023416  103  114  105  EM16:   .ASCIZ  /CLEARING DBIT ISN''T CLEARING CSR/<15><12>
3829 023461  103  117  123  EM20:   .ASCIZ  /COS REGISTER DATA ERROR/<15><12>
3830 023513  114  117  117  EM21:   .ASCIZ  /LOOP ERROR ON 6010-5010 OR 5011/<15><12>
3831 023555  101  104  104  EM22:   .ASCIZ  /ADDRESS TEST WITH MBIT SET ISN'T WORKING /<15><12>
3832 023631  125  116  101  EM23:   .ASCIZ  /UNABLE TO CLEAR INTERRUPT /<15><12>
3833 023666  115  117  104  EM24:   .ASCIZ  /MODULE UNDER TEST IS NOT RESPONDING/<15><12>
3834 023734  114  111  116  EM25:   .ASCIZ  /LINE CLOCK IS NOT INTERRUPTING/<15><12>
3835 023775  015  012  101  EM26:   .ASCIZ  <15><12>/A REGISTER DID NOT RESPOND WHEN ADDRESSED/
3836 024051  015  012  101  EM27:   .ASCIZ  <15><12>/A REGISTER DID NOT CLEAR ON INIT /
3837 024115  015  012  104  EM30:   .ASCIZ  <15><12>/DATA ERROR DURING REGISTER TEST/
3838 024157  015  012  101  EM31:   .ASCIZ  <15><12>/ALL COMPARATOR STATUS BITS SHOULD BE SET/
3839 024232  015  012  101  EM32:   .ASCIZ  <15><12>/ALL COMPARATOR STATUS BITS SHOULD BE CLEAR/
3840 024307  015  012  040  EM33:   .ASCIZ  <15><12>/ CHANNEL COMPARATOR ERROR/
3841 024343  015  012  115  EM34:   .ASCIZ  <15><12>/MORE THEN ONE CHANNEL FAILS DURING COMPARATOR TEST/
3842 024430  123  124  101  EM35:   .ASCIZ      /STATUS REGISTER ERROR AFTER INITIALIZE/<15><12>
3843 024501  102  125  123  EM36:   .ASCIZ      /BUSY BIT NOT CLEAR AFTER CONVERSION/<15><12>
3844 024547  105  122  122  EM37:   .ASCIZ      /ERROR BIT SET AFTER CONVERSION/<15><12>
3845 024610  103  117  116  EM40:   .ASCIZ  /CONVERSION NOT DONE AFTER .5MSEC/<15><12>
3846 024653  116  117  040  EM41:   .ASCIZ  /NO INTERRUPT AFTER CONVERSION DONE/
3847 024716  103  110  101  EM42:   .ASCIZ  /CHANNEL SELECTION REGISTER ERROR/
3848 024757  105  122  122  EM43:   .ASCIZ  /ERROR BIT NOT SET AFTER SETTING WRONG GAIN/<15><12>
3849 025034  105  122  122  EM44:   .ASCIZ  /ERROR BIT WILL NOT INTERRUPT/<15><12>
3850 025073  105  122  122  EM45:   .ASCIZ  /ERROR BIT NOT SET AFTER WRONG GAIN/<15><12>
3851 025140  115  101  111  EM46:   .ASCIZ      /MAINTANCE REFERENCE VOLTAGE TEST FAILED/<15><12>
3852 025212  122  101  115  EM47:   .ASCIZ  /RAMP IS NOT WORKING/<15><12>
3853 025240  104  101  124  EM50:   .ASCIZ  /DATA IS NOT ZERO WITH D BIT SET/<15><12>
3854 025302  015  012  114  EM53:   .ASCII  <15><12>/LINEARITY TEST ERROR/<15><12>
3855 025332  105  122  122          .ASCIZ  /ERRPC  ADDRESS TSTPNT  MIN     MAX     WAS (# OF CONVERS)/<15><12>
3856 025414  015  012  124  EM54:   .ASCIZ  <15><12>/TOO MANY ERRORS, TEST ABORTED/<15><12>
3857 025456  015  012  105  EM55:   .ASCIZ  <15><12>/ERROR BIT SET AFTER SETTING CHANNEL REG./
3858 025531  015  012  124  EM56:   .ASCIZ  <15><12>%TEST RAMP IN A/D ISN'T WORKING%
3859 025572  015  012  125  EM57:   .ASCIZ  <15><12>/UNKNOWN GENERIC CODE/
3860 025621  015  012  115  EM60:   .ASCIZ  <15><12>/MUX SELECT REGISTER ERROR/
3861 025655  015  012  127  EM61:   .ASCIZ  <15><12>/WRONG GENERIC CODE WITH MUX SELECTED/
3862 025724  015  012  122  EM62:   .ASCIZ  <15><12>/RIF BIT NOT CLEARING ERROR BIT/
3863 025765  015  012  123  EM63:   .ASCIZ  <15><12>/SELECTED MUX DID'T RESPOND/
3864 026022  015  012  115  EM73:   .ASCIZ  <15><12>/MUX REGISTER NOT CLEAR AFTER CONVERSION/
3865 026074  111  116  124  EM74:   .ASCIZ  /INTERRUPT WON'T CLEAR/<12><15>
3866 026124  103  102  111  EM75:   .ASCIZ  /CBIT FAILS TO CLEAR MODULE/<12><15>
3867 026161  103  117  125  EM76:   .ASCIZ  %COUNTER R/W FAILURE%<12><15>
```

```
3868 026207   24   102   111  EM77:   .ASCIZ  /TBIT FAILS TO INITIALIZE MODULE/<12><15>
3869 026251  103   117   125  EM100:  .ASCIZ  /COUNTER FAILS TO HALT AFTER INTERRUPT/<12><15>
3870 026321  103   117   125  EM101:  .ASCIZ  /COUNTER FAILS TO COUNT/<12><15>
3871 026352  127   122   117  EM102:  .ASCIZ  /WRONG COUNT IN COUNTER AFTER TRANSITION/<12><15>
3872 026424  116   117   040  EM103:  .ASCIZ  /NO INTERRUPT/<12><15>
3873 026443  122   111   106  EM104:  .ASCIZ  /RIF BIT NOT CLEARING INTERRUPT/<12><15>
3874 026504  105   122   122  EM105:  .ASCIZ  /ERROR MESSAGE STUB/<15><12>
3875 026531  103   117   125  EM106:  .ASCIZ  /COUNTER NOT AT INITIALIZED FREQ/<12><15>
3876 026573  124   111   115  EM107:  .ASCIZ  /TIME BASE ERROR/<15><12>
3877 026615  125   116   105  EM110:  .ASCIZ /UNEXPECTED COUNTER INTERRUPT/ <15><12>
3878 026654  115   117   104  EM111:  .ASCIZ  /MODULE READ-WRITE FAILURE/<12><15>
3879 026710  115   117   104  EM112:  .ASCIZ  /MODULE SELF-CLEAR ERROR/<12><15>
3880 026742  111   116   103  EM113:  .ASCIZ  /INCORRECT COUNT IN COUNTER/<12><15>
3881 026777  115   065   060  EM114:  .ASCIZ  /M5016 OVERFLOW FAILURE/<12><15>
3882 027030  117   126   105  EM115:  .ASCIZ  /OVERFLOW FAILS TO CLEAR/<12><15>
3883 027062  103   123   122  EM116:  .ASCIZ  /CSR CBIT FAILS TO CLEAR COUNTER/<12><15>
3884 027124  012   015   111  EM117:  .ASCIZ  <12><15>/ILLEGAL RADIX/<12><15>
3885 027146  012   015   040  EM213:  .ASCII <12><15>\ ILLEGAL "DENSE SET" CONFIGURATION. MODULES MUST BE\
3886 027233  012   015   040          .ASCII <12><15>\ GROUPED BY TYPE FOR RSX-11 SERIAL BUS DRIVER.\
3887 027313  012   015   040          .ASCII <12><15>\ SEE THE I/O SUBSYSTEM SOFTWARE MANUAL FOR DETAILS.\
3888 027400  012   015   012          .ASCIZ <12><15><12><15>\ CVPCA IS NOT AFFECTED.\
```

```
3890 027434  124  123  124  DH100:  .ASCIZ  /TST#   PC      MODULE  ADDR    GDDAT   BDDAT   ITER/
3891 027501  124  123  124  DH101:  .ASCIZ  /TST#   PC      MODULE  ADDRESS ITER/
3892 027535  120  103  011  DH1:    .ASCIZ  /PC     TST#    GDDAT   BDDAT   PASS/
3893 027566  120  103  011  DH5:    .ASCIZ  /PC     TST#    MODULE   ADDR   PASS/
3894 027621  120  103  011  DH7:    .ASCIZ  /PC     MODULE  ADDRESS TST#    GDDAT   BDDAT   PASS/
3895 027671  120  103  011  DH21:   .ASCIZ  /PC     GDDAT   BDDAT   PASS/
3896 027715  120  103  011  DH23:   .ASCIZ  /PC     TST#    IAR/
3897 027731  040  040  123  MESCSS: .ASCIZ /  SPECIAL CSS MODULE - USE SEPARATE DIAGNOSTIC/<CR><LF>
3898 030012  116  015  012  M1:     .ASCIZ  /N/<15><12>
3899 030016  131  015  012  M0:     .ASCIZ  /Y/<15><12>
3900 030022  122  000       M2:     .ASCIZ  /R/
3901 030024  107  000       M3:     .ASCIZ  /G/
3902 030026  124  000       M4:     .ASCIZ  /T/
3903 030030  104  000       M5:     .ASCIZ  /D/
3904 030032  115  000       M6:     .ASCIZ  /M/
3905 030034  105  000       M7:     .ASCIZ  /E/
3906 030036  011  000       M10:    .ASCIZ  /       /
3907 030040  040  120  101  M11:    .ASCIZ  / PASSES COMPLETED/
3908
3909 030062  015  012  124  M12:    .ASCII  <15><12>/TEST OPTIONS:/<15><12>
3910 030103  123  040  040       .ASCII  /S  SYSTEM TEST/<15><12>
3911 030123  101  040  040       .ASCII  /A  MODULE TEST/<15><12>
3912 030143  102  040  040       .ASCII  /B  CHANGE BASE ADDRESS/<15><12>
3913 030173  115  040  040       .ASCII  /M  MAP OF DBUS INTERFACES/<15><12>
3914 030226  111  040  040       .ASCII  /I  IOCM TEST/<15><12>
3915 030244  124  040  040       .ASCII  /T  SET SOFTWARE SWREG/<15><12>
3916 030273  106  040  040       .ASCII  /F  FIELD TEST/<15><12>
3917 030312  127  040  040       .ASCII  /W  LOOP TEST/<15><12>
3918 030330  114  040  040       .ASCII  /L  LOAD ITERATION COUNT/<15><12>
3919 030361  110  040  040       .ASCIZ  /H  HELP /<15><12>
3920 030374  040  040  040  M20:    .ASCIZ  /   ?/<15><12>
3921 030403  061  060  060  M22:    .ASCII  /100000  HALT ON ERROR/<15><12>
3922 030432  064  060  060       .ASCII  /40000   LOOP ON TEST/<15><12>
3923 030460  062  060  060       .ASCII  /20000   INHIBIT ERROR & EOP PRINT/<15><12>
3924 030523  061  060  060       .ASCII  /1000    LOOP ON ERROR/<15><12>
3925 030552  015  012  123       .ASCIZ  <15><12>/SWREG =/
3926 030564  125  123  105  M23:    .ASCIZ  /USE HARDWARE SWITCH REGISTER/<12><15>
3927 030623  116  125  115  M27:    .ASCIZ  /NUMBER OF ITERATIONS(OCT) = /
3928 030660  015  012  102  M30:    .ASCII  <15><12>/BEFORE RUNNING THIS TEST MAKE SURE CUSTOMER WIRES ARE DISCONNECTED/
3929 030764  015  012  127       .ASCIZ  <15><12>/WHEN READY TYPE CR/
3930 031011  015  012  123  M84:    .ASCIZ <15><12> /SELECTED MUX DIDN'T RESPOND/
3931 031047  015  012  115  MASS55: .ASCII  <15><12>/MONOTONICITY ERROR BY TWO BITS OR MORE/<15><12>
3932 031121  105  122  122       .ASCIZ  /ERRPC  ADDRESS GDDAT   BDDAT   RAMP    (UP=0,DOWN=1)/<15><12>
3933 031200  126  105  122  MASS51: .ASCIZ  /VERIFY IT BY RUNNING LOGIC TEST/<15><12>
3934 031242  105  122  122  DH57:   .ASCIZ/ERRPC    ADDRESS GCODE/
3935 031266  105  122  122  DH61:   .ASCIZ/ERRPC    MODULE  ADDRESS MUX#    TST#    GDDAT   BDDAT/
3936 031341  105  122  122  DH62:   .ASCIZ/ERRPC    MODULE  ADDRESS MUX#    TST#/
3937 031400  015  012  000  MASS0:  .ASCIZ  <15><12>
3938 031403  101  104  104  MASS2:  .ASCIZ  /ADDRESS  /
3939 031415  040  040  115  MASS3:  .ASCIZ /  M5010 TYPE 1  NONISOLATED 32 BIT DC INPUT/<15><12>
3940 031473  040  040  115  MASS4:  .ASCIZ /  M5011 TYPE 2  NONISOLATED 16 BIT COS DC INPUT/<15><12>
3941 031555  040  040  115  MASS5:  .ASCIZ /  M5012 TYPE 3  ISOLATED 16 BIT DC INPUT/<15><12>
3942 031630  040  040  115  MASS6:  .ASCIZ /  M5013 TYPE 3  8 BIT AC INPUT/<15><12>
3943 031671  040  040  115  MASS7:  .ASCIZ /  M6010 TYPE 4  NONISOLATED 32 BIT DC OUTPUT/<15><12>
3944 031750  040  040  115  MASS8:  .ASCIZ /  M6011 TYPE 5  16 BIT ONESHOT OUTPUT/<15><12>
3945 032020  040  040  115  MASS9:  .ASCIZ /  M6012 TYPE 4  ISOLATED 8 BIT DC OUTPUT/<15><12>
3946 032073  040  040  115  MASS10: .ASCIZ /  M6013 TYPE 4  8 BIT AC OUTPUT/<15><12>
```

```
3947 032135  040  040  125  MASS11: .ASCIZ /  UNKNOWN GENERIC CODE =  /
3948 032170  015  012  127  MASS12: .ASCIZ  <15><12>/WHICH TEST OPTION (H=HELP)?  /
3949 032230  015  012  101  MASS13: .ASCIZ  <15><12>/ADDRESS OF MUT: /
3950 032253  111  116  124  MASS14: .ASCIZ  /INTERRUPT TOO LATE/<15><12>
3951 032300  015  012  105  MASS15: .ASCIZ  <15><12>/END OF MODULE TEST/<15><12>
3952 032327  015  012  105  MASS17: .ASCIZ  <15><12> /END OF  AUTO TEST/<15><12>
3953 032355  040  040  040  MASS18: .ASCIZ  /   /<15><12>
3954 032363  117  116  105  MASS19: .ASCIZ  /ONE BYTE DATA PATTERN (OCTAL): /
3955 032423  103  117  116  MASS20: .ASCIZ  /CONTROL-C TO RETURN TO MONITOR/<15><12>
3956 032464  101  104  104  MASS21: .ASCIZ  /ADDRESS OF OUTPUT MUT: /
3957 032514  101  104  104  MASS22: .ASCIZ  /ADDRESS OF INPUT MUT: /
3958 032543  015  012  103  MASS23: .ASCIZ  <15><12>/CONNECT OUTPUT TO INPUT MODULE/<15><12>
3959 032606  015  012  105  MASS25: .ASCIZ  <15><12>/END OF LOOP TEST/<15><12>
3960 032633  127  122  117  MASS26: .ASCIZ  /WRONG MODULE-OUTPUT MUST BE M6010/<15><12>
3961 032677  072  011  000  MASS27: .ASCIZ  /:      /
3962 032702  127  122  117  MASS28: .ASCIZ  /WRONG MODULE-INPUT MUST BE EITHER M5010 OR M5011/<15><12>
3963 032765  040  040  101  MASS29: .ASCIZ /  A630  TYPE 6  FOUR CHANNEL DAC MODULE/<15><12>
3964 033037  040  040  101  MASS30: .ASCIZ %  A014  TYPE 7  A/D CONVERTER -SINGLE ENDED%<15><12>
3965 033115  040  040  101  MASS31: .ASCIZ %  A014  TYPE 7  A/D CONVERTER - DIFFFRENTIAL MODE%<15><12>
3966 033201  040  040  040  MASS32: .ASCIZ /    MUX# /
3967 033213  011  101  061  MASS33: .ASCIZ /        A156 - SINGLE-ENDED/<15><12>
3968 033242  011  101  061  MASS34: .ASCIZ /        A156 - DIFFERENTIAL MODE/<15><12>
3969 033276  011  101  061  MASS35: .ASCIZ /        A157/<15><12>
3970 033306  015  012  104  MASS36: .ASCII  <15><12> /D - LOGIC TEST/<15><12>
3971 033330  103  040  055          .ASCII  /C - CALIBRATION/<15><12>
3972 033351  115  040  055          .ASCII  /M - MONOTONICITY TEST/<15><12>
3973 033400  130  040  055          .ASCIZ  /X - MUX TEST/<15><12>
3974 033417  015  012  127  MASS37: .ASCIZ  <15><12>        /WHICH MUX # DO YOU WANT TO TEST? /
3975 033463  015  012  127  MASS38: .ASCIZ  <15><12>        %WHICH A/D CHANNEL YOU WANT TO CALIBRATE? %
3976 033537  015  012  124  MASS39: .ASCIZ  <15><12>        /TOO HIGH CHANNEL FOR DIFF MODE/<15><12>
3977 033602  015  012  124  MASS40: .ASCIZ  <15><12>        /TOO HIGH CHANNEL FOR SINGLE-ENDED MODE/<15><12>
3978 033655  015  012  127  MASS41: .ASCIZ  <15><12>/WHICH GAIN (1,10,20,50,100,200,1000) ?/
3979 033726  127  122  117  MASS42: .ASCIZ  /WRONG GAIN SELECTED/<15><12>
3980 033754  103  117  116  MASS43: .ASCIZ  /CONNECT VOLTAGE SOURCE  (CR)/
3981 034011  015  012  117  MASS44: .ASCIZ  <15><12>  /OCTAL   AVERAGE VOLTAGE(MV)/<15><12>
3982 034051  125  116  105  MASS45: .ASCIZ  /UNEXPECTED TIME-OUT TRAP. LAST PC BEFORE TRAP = /
3983 034132  015  012  127  MASS46: .ASCIZ  <15><12>        /WHICH TEST (D,C,M,X,H=HELP)? /
3984 034172  040  040  115  MASS48: .ASCIZ /  M5012-YA TYPE 3  16 BIT TTL INPUT/<15><12>
3985 034240  040  040  115  MASS49: .ASCIZ /  M6010-YA TYPE 4  32 BIT TTL OUTPUT/<15><12>
3986 034307  040  040  116  MASS50: .ASCIZ %  NO I/O MODULE PRESENT%<15><12>
3987 034341  040  040  115  MASS56: .ASCIZ /  M5014 TYPE 10 INPUT COUNTER/<15><12>
3988 034401  040  040  115  MASS57: .ASCIZ /  M5031 TYPE 2  16 BIT ISOLATED CHANGE OF STATE/<15><12>
3989 034463  040  040  115  MASS52: .ASCIZ /  M5016 TYPE 9  QUAD 8 BIT COUNTER/<15><12>
3990 034530  040  040  115  MASS53: .ASCIZ /  M6014 TYPE 11 DUAL 16 BIT COUNTER/<15><12>
3991 034576  040  040  115  MASS58: .ASCIZ /  M6015 TYPE 4  16-BIT ISOLATED DC OUTPUT/<15><12>
3992 034652  040  040  101  MASS59: .ASCIZ /  A631  TYPE 6  12-BIT DAC/<15><12>
3993 034707  040  040  101  MASS60: .ASCIZ %  A020  TYPE 8  HCM A/D 2 WIRE MODE%<15><12>
3994 034755  040  040  101  MASS61: .ASCIZ %  A020  TYPE 8  HCM A/D 3 WIRE MODE%<15><12>
3995 035023  102  101  123  MASS62: .ASCIZ /BASE ADDRESS NOW ==> /
3996 035051  015  012  105  MASS63: .ASCIZ  <15><12>/ENTER NEW BASE ADDRESS : /
3997 035105  015  012  126  MASS64: .ASCIZ  <15><12>/VECTOR ADDRESS NOW ==> /
3998 035137  015  012  105  MASS65: .ASCIZ  <15><12>/ENTER NEW VECTOR ADDRESS : /
3999 035175  015  012  120  MASS66: .ASCIZ <15><12> /PLEASE REENTER THE CORRECT BASE ADDRESS./
4000 035250  015  012  104  MASS67: .ASCIZ <15><12>&DIGITAL I/O MODULE ONLY !&<15><12>
4001 035306  015  012  015  MASS70: .ASCII <15><12><15><12>\*****> WARNING:\
4002 035331  015  012               .ASCII <15><12>
4003 035333  015  012  124          .ASCII <15><12>\THIS TFST ALTERS I/O MODULE OUTPUTS DIRECTLY. BE\
```

```
4004 035415  015  012  101          .ASCII <15><12>\ABSOLUTELY SURE THE CUSTOMER'S EQUIPMENT IS\
4005 035472  015  012  116          .ASCII <15><12>\NOT CONNECTED. THIS COULD BE EXTREMELY DANGEROUS.\
4006 035555  015  012  000          .ASCIZ <15><12>
4007 035560  104  101  124  MASS71: .ASCIZ /DATA PATTERN FOR NEXT BYTE: /
4008 035615  103  101  116  LOCAL:  .ASCIZ /CAN BE RUN ONLY IN LOCAL MODE./<CR><LF>
4009 035656  124  123  124  DH110:  .ASCIZ  /TST#   PC      ADDRESS COUNTER MODULE  ITER/
4010 035722  124  123  124  DH102:  .ASCIZ  /TST#   PC      ADDRESS COUNTER GDDAT   BDDAT   ITER/
4011 035773  124  123  124  DH114:  .ASCIZ  /TST#   PC      ADDRESS COUNTER GDOVF   BDOVF   ITER/
4012 036044  015  012  101  DH33:   .ASCIZ  <15><12>/ADDRESS        CHNUM   STBITS  CH0     CH1     CH2     CH3/
4013 036113  015  012  105  DH34:   .ASCIZ  <15><12>/ERRPC  MODULE  ADDRESS STBITS  CH0     CH1     CH2     CH3/
4014 036171  015  012  105  DH40:   .ASCIZ  <15><12>/ERRPC  TST#    GENERIC /
4015 036217  015  012  105  DH30:   .ASCIZ  <15><12>/ERRPC  MODULE  GDDATA  BDDATA  REGADR /
4016 036264  015  012  104  M80:    .ASCIZ  <15><12>/DBIT DOES NOT DISABLE ALL OUTPUTS /
4017 036331  015  012  101  M81:    .ASCIZ  <15><12>/ADJUSTMENT OF VOLTAGE OUTPUT/
4018 036370  015  012  127  M82:    .ASCIZ  <15><12>/WANT TO ADJUST CURRENT OUTPUT (Y-N)?  /
4019 036441  015  012  105  M83:    .ASCIZ  <15><12>/END OF CALIBRATION/
4020 036466  012  015  125  M85:    .ASCIZ  <12><15>/UNEXPECTED TIME OUT/<12><15>
4021 036516  015  012  104  DANGC:  .ASCIZ <15><12> /DAC DID NOT RESPOND WITH GENERIC CODE 261/
4022 036572  015  012  107  GAINMG: .ASCII <15><12> /GAIN X INPUT VOLTAGE=VOLTAGE READING/
4023 036640  015  012  107          .ASCIZ <15><12> /GAIN = /
4024 036652  015  012  123  DANSEL: .ASCIZ <15><12> /SELECT TEST (A,T,D OR H=HELP): /
4025 036714  015  012  103  DACHN:  .ASCIZ <15><12> /CHANNEL # DESIRED (0,1,2,3): /
4026 036754  015  012  101  TEXT:   .ASCII <15><12> /A - CALIBRATION TEST/
4027 037002  015  012  124          .ASCII <15><12> /T - INTERNAL COMPARATOR TEST/
4028 037040  015  012  104          .ASCIZ <15><12> /D - D-BIT TEST /
4029
4030 037062  015  012  101  CTXTV:  .ASCIZ  <15><12>/ADJUST REFERENCE VOLTAGE (Y-N)?  /
4031
4032
4033 037126  015  012  122          .ASCIZ <15><12> /R66 - CALIB POT FOR CH1 CURRENT ZERO/<15><12>
4034 037177  015  012  122          .ASCII <15><12> /R80 - CALIB POT FOR CH2 CURRENT GAIN/
4035
4036 037245  015  012  123  CM1:    .ASCIZ <15><12> \STEP 1: +10.240V  +/- 2MV (CR)\
4037 037306  015  012  123  CM2:    .ASCIZ <15><12> \STEP 2:   5.120V  +/- 2MV (CR)\
4038 037347  015  012  123  CM3:    .ASCIZ <15><12> \STEP 1:   0.000V  +/- 5MV (CR)\
4039 037410  015  012  123  CM4:    .ASCIZ <15><12> \STEP 2: +10.230V  +/- 40MV (CR)\
4040
4041
4042 037452  015  012  123  CM7:    .ASCII <15><12> /STEP 2: CURRENT OFFSET ADJ.  9.8MV  TOLER. 5MV/
4043 037532  015  012  040          .ASCIZ <15><12> /        OR 19.5 MICRO-AMPS. TOLER. 10 MICRO-AMPS.  (CR)/
4044
4045 037624  015  012  123  CM9:    .ASCII <15><12> /STEP 1: CURRENT GAIN ADJ. TO +10.000V TOLER. 5MV/
4046 037706  015  012  040          .ASCIZ <15><12> /        OR 20.000MA TOLER. 10 MICRO-AMPS (CR)/
4047 037766  015  012  123  CM10:   .ASCII <15><12> /STEP 2: CURRENT OFFSET ADJ. TO +2.000V TOLER. 5MV/
4048 040051  015  012  040          .ASCIZ <15><12> /        OR 4.000 MA.TOLER.  10 MICRO AMPS (CR)/
4049 040132  015  012  110  CM11:   .ASCIZ  <15><12>/HAS CALIB BEEN RE-VERIFIED (Y-N)? /
4050 040177  015  012  127  CM12:   .ASCIZ  <15><12>/WANT TO CALIB THE 4-20 MA RANGE (Y-N)? /
4051 040251  015  012  120  DH31:   .ASCIZ  <15><12>/PC#    ADDRESS EXPECT  WAS/
4052                        DH31:   .EVEN
4053 040302  001212  001132  003462  DT110:  .WORD   $TESTN, $ERRPC, TADDR,  CNUM,   $MUT,   $PASS,
4054 040320  001212  001132  003462  DT102:  .WORD   $TESTN, $ERRPC, TADDR,  CNUM,   $GDDAT, $BDDAT, $PASS,
4055 040340  001212  001132  003456  DT101:  .WORD   $TESTN, $ERRPC, $MUT,   TADDR,  $PASS,
4056 040354  001212  001132  003456  DT100:  .WORD   $TESTN, $ERRPC, $MUT,   TADDR,  $GDDAT, $BDDAT, $PASS,
4057 040374  001132  003456  003510  DT17:   .WORD   $ERRPC, $MUT,   TEMP1,  $TESTN, $GDDAT, $BDDAT, $PASS,
4058 040414  001132  003456  003470  DT20:   .WORD   $ERRPC, $MUT,   COSADR, $TESTN, $GDDAT, $BDDAT, $PASS,
4059 040434  001132  001212  003600  DT23:   .WORD   $ERRPC, $TESTN, TEMP3,
4060 040444  001132  003456  003466  DT7:    .WORD   $ERRPC, $MUT,   TBADDR, $TESTN, $GDDAT, $BDDAT, $PASS,
```

```
4061 040464  001 32  001212  001140  DT1:    .WORD   $ERRPC, $TESTN, $GDDAT, $BDDAT, $PASS,
4062 040500  001132  001212  003456  DT5:    .WORD   $ERRPC, $TESTN, $MUT,   TADDR,  $PASS,
4063 040514  001132  001140  001142  DT21:   .WORD   $ERRPC, $GDDAT, $BDDAT, $PASS,
4064 040526  001132  003456  001140  DT30:   .WORD   $ERRPC, $MUT,   $GDDAT, $BDDAT, TEMP,
4065 040542  001132  003462  003514  DT33:   .WORD   $ERRPC, TADDR,  CHNUM,  TEMP3,  ACOUNT, BCOUNT, ACOUNT, BCOUNT,
4066 040564  001132  003456  003462  DT34:   .WORD   $ERRPC, $MUT,   TADDR,  TEMP3,  ACOUNT, BCOUNT, ACOUNT, BCOUNT,
4067 040606  001132  003462  001140  DT51:   .WORD   $ERRPC, TADDR,  $GDDAT, $BDDAT, LASTCN,
4068 040622  001132  003462  003556  DT52:   .WORD   $ERRPC, TADDR,  BLAST,  BCOUNT, CCOUNT, $BDDAT,
4069 040640  001132  003600  003576  DT57:   .WORD   $ERRPC, TEMP3,  TEMP2,
4070 040650  001132  003456  003466  DT61:   .WORD   $ERRPC, $MUT,   TBADDR, MXNUM,  $TESTN, $GDDAT, $BDDAT,
4071 040670  001132  003456  003466  DT62:   .WORD   $ERRPC, $MUT,   TBADDR, MXNUM,  $TESTN,
4072 040704  001132  003462  003600  DT31:   .WORD   $ERRPC, TADDR,  TEMP3,  TEMP,
4073 040716     000     000     000  DF102:  .BYTE   0,0,0,0,0,0,1
4074 040725     000     000     000  DF100:  .BYTE   0,0,0,0,0,1
4075 040733     000     000     000  DF101:  .BYTE   0,0,0,0,1
4076 040740     000     000     000  DF51:   .BYTE   0,0,0,0
4077 040744     000     000     001  DF52:   .BYTE   0,0,1,1,1
4078                                 .EVEN
```

```
                        .SBTTL  DIGITAL MODULE TEST, M5010

                                ;THIS TEST WILL CHECK M5010 MODULE
                                ;32 BIT NONISOLATED DC SENSE


 040752                 M5010:
                        ;;*****************************************************************
 040752 000004          TST12:  SCOPE
 040754 012737 000012 001212    MOV     #12,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
 040762 112737 000000 001140    MOVB    #0,$GDDAT       ;SET WHAT DATA SHOULD BE
 040770 012737 000004 003460    MOV     #4,BYTNUM       ;SET NUMBER OF BYTES
 040776 004737 015022           JSR     PC,TSTBYT       ;CHECK IF DATA IS CORRECT

 041002 152777 000010 141634    BISB    #TBIT,@CSR
 041010 112737 000377 001140    MOVB    #377,$GDDAT     ;SET WHAT DATA SHOULD BE
 041016 012737 000004 003460    MOV     #4,BYTNUM       ;SET NUMBER OF BYTES
 041024 004737 015022           JSR     PC,TSTBYT       ;CHECK IF DATA IS CORRECT
 041030 142777 000010 141606    BICB    #TBIT,@CSR
 041036 004737 015410           JSR     PC,CLRINT
 041042 000207                  RTS     PC
```

```
                        .SBTTL  DIGITAL MODULE TEST ,M5013


                                ;THIS TEST CHECKS M5013 MODULE
                                ;8 BIT AC SENSE


041044                  M5013:
                        ;;*************************************************************
041044  000004          TST13:  SCOPE
041046  012737  000013  001212          MOV     #13,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
041054  152777  000020  141562          BISB    #DBIT,@CSR      ;SET DISABLE BIT
041062  112737  000000  001140          MOVB    #0,$GDDAT       ;SET WHAT DATA SHOULD BE
041070  012737  000001  003460          MOV     #1,BYTNUM       ;SET NUMBER OF BYTES
041076  004737  015022                  JSR     PC,TSTBYT       ;CHECK IF DATA IS CORRECT

041102  152777  000010  141534          BISB    #TBIT,@CSR      ;SET COMPLIM BIT
041110  004737  015410                  JSR     PC,CLRINT
041114  112737  000377  001140          MOVB    #377,$GDDAT     ;SET WHAT DATA SHOULD BE
041122  012737  000001  003460          MOV     #1,BYTNUM       ;SET NUMBER OF BYTES
041130  004737  015022                  JSR     PC,TSTBYT       ;CHECK IF DATA IS CORRECT
041134  142777  000010  141502          BICB    #TBIT,@CSR
041142  004737  015410                  JSR     PC,CLRINT


041146  012746  000000                  MOV     #PR0,-(SP)      ;SET PSW TO PRIORITY 0
041152  012746  041160                  MOV     #64$,-(SP)
041156  000002                          RTI
041160  000240          64$:    NOP
041162  012777  041316  141460          MOV     #5$,@VECTO      ;SET INTERRUPT VECTOR ADDRESS
041170  012777  000340  141454          MOV     #PR7,@VECTOA
041176  152777  000010  141440          BISB    #TBIT,@CSR      ;START INTERRUPT
041204  152777  000100  141432          BISB    #EBIT,@CSR      ;ENABLE INTERRUPT
041212  013737  002636  003446          MOV     K,ZLOOP
041220  005037  003444          66$:    CLR     YLOOP           ;WAIT
041224  005237  003444          65$:    INC     YLOOP
041230  023727  003444  177777          CMP     YLOOP,#-1
041236  001372                          BNE     65$
041240  005337  003446                  DEC     ZLOOP
041244  001365                          BNE     66$
041246  132737  000001  001226          BITB    #BIT0,$ENV      ;UNDER APT ?
041254  001403                          BEQ     50$             ;NO
041256  005737  001214                  TST     $PASS           ;FIRST PASS ($PASS = 0) ?
041262  001001                          BNE     7$              ;NO
041264                  50$:
041264  104005                          EMT     5
041266  142777  000100  141350  7$:     BICB    #EBIT,@CSR      ;CLEAR CSR
041274  142777  000010  141342          BICB    #TBIT,@CSR
041302  004737  015410                  JSR     PC,CLRINT       ;CLEAR ALL INTERRUPT
041306  012737  007606  000004          MOV     #TMOVEC,ERRVEC
041314  000207                          RTS     PC              ;RETURN TO CALLING PROGRAM D

041316  022626          5$:     CMP     (SP)+,(SP)+     ;RESET STACK
041320  005037  003510                  CLR     TEMP1
041324  113737  002661  003503          MOVB    BASE+1,TEMP+1   ;INITIALIZE TEMP
041332  012737  041640  000004          MOV     #4$,ERRVEC
```

```
     041340                          6$:
     041340  013737  002636  003446          MOV     K,ZLOOP
     041346  005037  003444          68$:    CLR     YLOOP           ;WAIT
     041352  005237  003444          67$:    INC     YLOOP
     041356  023727  003444  000001          CMP     YLOOP,#1
     041364  001372                          BNE     67$
     041366  005337  003446                  DEC     ZLOOP
     041372  001365                          BNE     68$
60   041374  117737  141246  003502          MOVB    @IAR,TEMP       ;CHECK WHICH I/O INTERRUPTED
61   041402  152777  000001  141234          BISB    #RBIT,@CSR      ;HERE IF INTERRUPT
62   041410  123737  003502  003462          CMPB    TEMP,TADDR      ;IS IT MUT
63   041416  001414                          BEQ     2$
64   041420  105777  142056                  TSTB    @TEMP
65   041424  005237  003510                  INC     TEMP1           ;LOOP NO MORE THAN 400 TIMES
66   041430  022737  000400  003510          CMP     #400,TEMP1
67   041436  001340                          BNE     6$
68   041440  005737  001214                  TST     $PASS           ;FIRST PASS ($PASS = 0) ?
69   041444  001001                          BNE     2$              ;NO
70   041446  104005                          EMT     5
71   041450  105777  142006          2$:     TSTB    @TADDR          ;CLEAR INTERRUPT
72   041454  013737  002636  003446          MOV     K,ZLOOP
     041462  005037  003444          70$:    CLR     YLOOP           ;WAIT
     041466  005237  003444          69$:    INC     YLOOP
     041472  023727  003444  000001          CMP     YLOOP,#1
     041500  001372                          BNE     69$
     041502  005337  003446                  DEC     ZLOOP
     041506  001365                          BNE     70$
73   041510  132777  000200  141126          BITB    #FBIT,@CSR      ;CHECK IF INTERRUPT CLEAR
74   041516  001413                          BEQ     8$
75   041520  117737  141122  003502          MOVB    @IAR,TEMP
76   041526  023737  003502  003462          CMP     TEMP,TADDR
77   041534  001002                          BNE     9$
78   041536  104015                          EMT     15
79   041540  000652                          BR      7$
80   041542  004737  015410          9$:     JSR     PC,CLRINT
81   041546  132777  000010  141070  8$:     BITB    #TBIT,@CSR
82   041554  001644                          BEQ     7$
83   041556  012746  000000                  MOV     #PR0,-(SP)      ;SET PSW TO PRIORITY 0
     041562  012746  041570                  MOV     #71$,-(SP)
     041566  000002                          RTI
     041570  000240          71$:    NOP
84   041572  142777  000010  141044          BICB    #TBIT,@CSR      ;CHECK IF CLEARING TBIT CAUSE INTERRUPT
85   041600  013737  002636  003446          MOV     K,ZLOOP
     041606  005037  003444          73$:    CLR     YLOOP           ;WAIT
     041612  005237  003444          72$:    INC     YLOOP
     041616  023727  003444  177777          CMP     YLOOP,#-1
     041624  001372                          BNE     72$
     041626  005337  003446                  DEC     ZLOOP
     041632  001365                          BNE     73$
86   041634  104005                          EMT     5
87   041636  000613                          BR      7$
88   041640  022626                  4$:     CMP     (SP)+,(SP)+
89   041642  104005                          EMT     5
90   041644  000137  041450                  JMP     2$
```

```
                                        .SBTTL  DIGITAL MODULE TEST, M6010, M6010YA


                                                ;THIS TEST CHECKS M6010 MODULE
                                                ;32 BIT NONISOLATED DC OUT


041650                                  M6010:
                                        ;;*****************************************************************
041650  000004                          ;ST14:  SCOPE
041652  012737  000014  001212                  MOV     #14,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
041660  152777  000020  140756                  BISB    #DBIT,@CSR      ;SET D BIT
041666  013700  003462                          MOV     TADDR,R0
041672  105020                                  CLRB    (R0)+           ;CLEAR ALL FOUR BYTES OF I/O REGISTER
041674  105020                                  CLRB    (R0)+
041676  105020                                  CLRB    (R0)+
041700  105020                                  CLRB    (R0)+
041702  005000                                  CLR     R0
041704  005001                          4$:     CLR     R1
041706  005037  003452                          CLR     DBUFF           ;CLEAR IMAGE OF I/O REGISRERS
041712  005037  003454                          CLR     DBUFF+2
041716  116160  002630  003452        3$:     MOVB    PATT(R1),DBUFF(R0);SET DATA PATTERN IN IMIGE
041724  013737  003462  003502                  MOV     TADDR,TEMP
041732  060037  003502                          ADD     R0,TEMP
041736  116177  002630  141536                  MOVB    PATT(R1),@TEMP  ;SET DATA IN I/O REG
041744  005002                                  CLR     R2
041746  013737  003462  003510        2$:     MOV     TADDR,TEMP1
041754  060237  003510                          ADD     R2,TEMP1
041760  117737  141524  001142                  MOVB    @TEMP1,$BDDAT   ;READ I/O REGISTER
041766  123762  001142  003452                  CMPB    $BDDAT,DBUFF(R2);IS DATA OK?
041774  001404                                  BEQ     1$
041776  116237  003452  001140                  MOVB    DBUFF(R2),$GDDAT
042004  104017                                  EMT     17
042006  005202                          1$:     INC     R2
042010  022702  000004                          CMP     #4,R2
042014  001354                                  BNE     2$              ;TEST IF OTHER BYTES ARE OK
042016  005201                                  INC     R1
042020  022701  000004                          CMP     #4,R1           ;LAST PATTERN?
042024  001334                                  BNE     3$              ;TEST NEXT DATA PATTERN
042026  005200                                  INC     R0
042030  022700  000004                          CMP     #4,R0           ;LAST BYTE?
042034  001323                                  BNE     4$              ;NO,TEST NEXT BYTE
042036  000207                                  RTS     PC
```

```
                                .SBTTL  DIGITAL MODULE TEST, M6012, M6013


                                        ;THIS TEST CHECKS MODULES M6012 AND M6013
                                        ;8 BIT AC AND DC OUT



042040                          M6012:
042040                          M6013:
                                ;;*****************************************************************
042040  000004                  TST15:  SCOPE
042042  012737  000015  001212          MOV     #15,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
042050  152777  000020  140566          BISB    #DBIT,@CSR
042056  012737  000125  001140          MOV     #125,$GDDAT     ;SET DATA PATTERN
042064  012737  000001  003460          MOV     #1,BYTNUM       ;SET NUMBER OF BYTES
042072  004737  015166                  JSR     PC,SETBYT       ;SET DATA IN I/O MODULE
042076  012737  000001  003460          MOV     #1,BYTNUM
042104  004737  015022                  JSR     PC,TSTBYT       ;TEST IF DATA IS SET CORRECTLY

042110  012737  000252  001140          MOV     #252,$GDDAT     ;SET DATA PATTERN
042116  012737  000001  003460          MOV     #1,BYTNUM       ;SET NUMBER OF BYTES
042124  004737  015166                  JSR     PC,SETBYT       ;SET DATA IN I/O MODULE
042130  012737  000001  003460          MOV     #1,BYTNUM
042136  004737  015022                  JSR     PC,TSTBYT       ;TEST IF DATA IS SET CORRECTLY

042142  012737  000377  001140          MOV     #377,$GDDAT     ;SET DATA PATTERN
042150  012737  000001  003460          MOV     #1,BYTNUM       ;SET NUMBER OF BYTES
042156  004737  015166                  JSR     PC,SETBYT       ;SET DATA IN I/O MODULE
042162  012737  000001  003460          MOV     #1,BYTNUM
042170  004737  015022                  JSR     PC,TSTBYT       ;TEST IF DATA IS SET CORRECTLY

042174  012737  000000  001140          MOV     #0,$GDDAT       ;SET DATA PATTERN
042202  012737  000001  003460          MOV     #1,BYTNUM       ;SET NUMBER OF BYTES
042210  004737  015166                  JSR     PC,SETBYT       ;SET DATA IN I/O MODULE
042214  012737  000001  003460          MOV     #1,BYTNUM
042222  004737  015022                  JSR     PC,TSTBYT       ;TEST IF DATA IS SET CORRECTLY
042226  000207                          RTS     PC
```

```
                        .SBTTL  DIGITAL MODULE TEST, M6011

                                ;THIS TEST CHECKS MODULE M6011
                                ;ONE SHOT DC OUT


042230                  M6011:
                        ;;*****************************************************************
042230  000004          TST16:  SCOPE
042232  012737  000016  001212          MOV     #16,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
042240  013737  003462  003464          MOV     TADDR,TADDR1
042246  005237  003464                  INC     TADDR1
042252  152777  000020  140364          BISB    #DBIT,@CSR
042260  012777  015402  140400          MOV     #COUNT,@CLKVC   ;SET LOC 100 - CLOCK VECTOR
042266  012777  000340  140374          MOV     #PR7,@CLKVCA    ;SET LOC 102
042274  012746  000000                  MOV     #PR0,-(SP)      ;SET PSW TO PRIORITY 0
042300  012746  042306                  MOV     #64$,-(SP)
042304  000002                          RTI
042306  00024C          64$:    NOP
042310  012737  000252  042550          MOV     #252,TSTDAT     ;FIRST DATA PATTERN

042316  005037  003476          4$:     CLR     CLK
042322  012737  000377  001140          MOV     #377,$GDDAT     ;SET DATA PATTERN
042330  012737  000002  003460          MOV     #2,BYTNUM       ;SET NUMBER OF BYTES
042336  004737  015166                  JSR     PC,SETBYT       ;SET DATA IN I/O MODULE
042342  004737  015022                  JSR     PC,TSTBYT       ;TEST IF DATA IS SET CORRECTLY
042346  005001                          CLR     R1              ;TEST IF LINE CLOCK IS INTERRUPTING
042350  005737  003476          1$:     TST     CLK
042354  001004                          BNE     2$
042356  005201                          INC     R1
042360  001373                          BNE     1$
042362  104025                          EMT     25
042364  000465                          BR      M6011X          ;EXIT TEST

042366  105777  141070          2$      TSTB    @TADDR          ;TEST OUTPUT TO LOCATE ASYNCHRONOUS CLOCK
042372  001003                          BNE     3$
042374  105777  141064                  TSTB    @TADDR1
042400  001411                          BEQ     5$              ;WHEN OUTPUT CLEARS START TEST
042402  023727  003476  001224  3$:     CMP     CLK,#660.       ;WAIT 11 SEC MAX
042410  002766                          BLT     2$
042412  005037  001140                  CLR     $GDDAT          ;PATTERN SHOULD BE CLEAR
042416  004737  015104                  JSR     PC,TSTONE
042422  000446                          BR      M6011X          ;EXIT TEST

042424  005037  003476          5$:     CLR     CLK             ;CLEAR CLOCK COUNTER
042430  013737  042550  001140          MOV     TSTDAT,$GDDAT   ;SET DATA PATTERN
042436  004737  015166                  JSR     PC,SETBYT       ;SET DATA IN I/O MODULE
042442  004737  015022                  JSR     PC,TSTBYT       ;TEST IF DATA IS SET CORRECTLY
042446  105777  141010          6$:     TSTB    @TADDR          ;TEST OUTPUT
042452  001003                          BNE     7$
042454  105777  141004                  TSTB    @TADDR1
042460  001404                          BEQ     10$
042462  022737  001224  003476  7$:     CMP     #660.,CLK       ;WAIT 11 SEC MAX
042470  002766                          BLT     6$
042472  013737  003476  042552  10$:    MOV     CLK,CLKSTP      ;GET TIME
042500  023727  042552  001142          CMP     CLKSTP,#610.    ;LESS THAN 10 SEC?
042506  003404                          BLE     11$             ;YES, GOOD
```

```
    184 042510  005037  001140                CLR     $GDDAT          ;NO, SHOULD BE CLEAR
    185 042514  004737  015104          12$:  JSR     PC,TSTONE
    186 042520  022737  000125  042550  11$:  CMP     #125,TSTDAT     ;LAST DATA?
    187 042526  001404                        BEQ     M6011X          ;YES
    188
    189 042530  012737  000125  042550        MOV     #125,TSTDAT     ;SECOND DATA PATTERN
    190 042536  000667                        BR      4$              ;REPEAT
    191 042540  012777  015400  140120  M6011X: MOV     #NOCLK,@CLKVC   ;RESTORE CLOCK VECTOR
    192 042546  000207                        RTS     PC
    193
    194 042550  000000                TSTDAT: .WORD   0
    195 042552  000000                CLKSTP: .WORD   0
    196
```

```
                                .SBTTL  DIGITAL MODULE TEST, M5011, M5031


                                        ;THIS TEST CHECKS M5011 MODULE
                                        ;16 BIT CHANGE OF STATE SENSE
                                        ;THIS TEST CHECKS M5031 MODULE
                                        ;16 BIT ISOLATED CHANGE OF STATE


208 042554                      M5011:
                                ;;*************************************************************
    042554  000004              TST17:  SCOPE
    042556  012737  000017  001212      MOV     #17,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
209 042564  013737  003462  003470      MOV     TADDR,COSADR
210 042572  062737  000002  003470      ADD     #2,COSADR
211 042600  152777  000030  140036      BISB    #DBIT!TBIT,@CSR;SET DISABLE BIT AND TBIT
212 042606  112737  000377  001140      MOVB    #377,$GDDAT     ;SET WHAT DATA SHOULD BE
    042614  012737  000002  003460      MOV     #2,BYTNUM       ;SET NUMBER OF BYTES
    042622  004737  015022              JSR     PC,TSTBYT       ;CHECK IF DATA IS CORRECT
213
214 042626  142777  000010  140010      BICB    #TBIT,@CSR      ;CLEAR COMPLETION BIT
215 042634  112737  000000  001140      MOVB    #0,$GDDAT       ;SET WHAT DATA SHOULD BE
    042642  012737  000002  003460      MOV     #2,BYTNUM       ;SET NUMBER OF BYTES
    042650  004737  015022              JSR     PC,TSTBYT       ;CHECK IF DATA IS CORRECT
216 042654  004737  015410              JSR     PC,CLRINT
217 042660  012746  000000              MOV     #PR0,-(SP)      ;SET PSW TO PRIORITY 0
    042664  012746  04..                MOV     #64$,-(SP)
    042670  000002                      RTI
    042672  000240              64$:    NOP
218 042674  012777  043030  137746      MOV     #1$,@VECTO      ;SET INTERRUPT VECTOR
219 042702  012777  000340  137742      MOV     #PR7,@VECTOA
220 042710  152777  000010  137726      BISB    #TBIT,@CSR      ;START INTERRUPT
221 042716  152777  000100  137720      BISB    #EBIT,@CSR      ;ENABLE INTERRUPT
222 042724  013737  002636  003446      MOV     K,ZLOOP
    042732  005037  003444      66$:    CLR     YLOOP           ;WAIT
    042736  005237  003444      65$:    INC     YLOOP
    042742  023727  003444  177777      CMP     YLOOP,#-1
    042750  001372                      BNE     65$
    042752  005337  003446              DEC     ZLOOP
    042756  001365                      BNE     66$
223 042760  132737  000001  001226      BITB    #BIT0,$ENV      ;UNDER APT ?
224 042766  001403                      BEQ     50$             ;NO
225 042770  005737  001214              TST     $PASS           ;FIRST PASS ($PASS = 0) ?
226 042774  001001                      BNE     2$              ;NO
227 042776                      50$:
    042776  104005                      EMT     5
228 043000  142777  000100  137636  2$: BICB    #EBIT,@CSR      ;CLEAR CSR
229 043006  142777  000010  137630      BICB    #TBIT,@CSR
230 043014  012737  007606  000004      MOV     #TMOVEC,ERRVEC
231 043022  004737  015410              JSR     PC,CLRINT       ;CLEAR ALL INTERRUPTS
232 043026  000207                      RTS     PC              ;RETURN TO MONITOR
233
234 043030  022626              1$:     CMP     (SP)+,(SP)+     ;ADJUST STACK POINTER
235 043032  005037  003510              CLR     TEMP1
236 043036  113737  002661  003503      MOVB    BASE+1,TEMP+1   ;INITIALIZE TEMP
237 043044  012737  043530  000004      MOV     #10$,ERRVEC
```

```
    238 043052                                  7$:
        043052  013737  002636  003446                  MOV     K,ZLOOP
        043060  005037  003444                  68$:    CLR     YLOOP           ;WAIT
        043064  005237  003444                  67$:    INC     YLOOP
        043070  023727  003444  000001                  CMP     YLOOP,#1
        043076  001372                                  BNE     67$
        043100  005337  003446                          DEC     ZLOOP
        043104  001365                                  BNE     68$
    239 043106  117737  137534  003502                  MOVB    @IAR,TEMP       ;FIND WHICH I/O INTERRUPTED
    240 043114  123737  003502  003462                  CMPB    TEMP,TADDR      ;IS IT MUT
    241 043122  001416                                  BEQ     8$              ;YES
    242 043124  152777  000001  137512                  BISB    #RBIT,@CSR      ;CLEAR INTERRUPT
    243 043132  105777  140344                          TSTB    @TEMP
    244 043136  005237  003510                          INC     TEMP1
    245 043142  022737  000400  003510                  CMP     #400,TEMP1
    246 043150  001340                                  BNE     7$
    247 043152  104005                                  EMT     5
    248 043154  000137  043000                          JMP     2$
    249 043160  142777  000100  137456  8$:     BICB    #EBIT,@CSR
    250 043166  112737  000377  001140                  MOVB    #377,$GDDAT
    251 043174  117737  140270  001142                  MOVB    @COSADR,$BDDAT  ;TEST COS REGISTER
    252 043202  123737  001140  001142                  CMPB    $GDDAT,$BDDAT   ;IS IT ALL ONES
    253 043210  001401                                  BEQ     3$
    254 043212  104020                                  EMT     20
    255 043214  152777  000001  137422  3$:     BISB    #RBIT,@CSR
    256 043222  105777  140242                          TSTB    @COSADR         ;CLEAR INTERRUPT
    257 043226  005037  001140                          CLR     $GDDAT
    258 043232  117737  140232  001142                  MOVB    @COSADR,$BDDAT
    259 043240  001401                                  BEQ     4$
    260 043242  104013                                  EMT     13
    261 043244  005237  003470                  4$:     INC     COSADR
    262 043250  005237  003462                          INC     TADDR           ;TEST NEXT BYTE
    263 043254  013737  002636  003446                  MOV     K,ZLOOP
        043262  005037  003444                  70$:    CLR     YLOOP           ;WAIT
        043266  005237  003444                  69$:    INC     YLOOP
        043272  023727  003444  000001                  CMP     YLOOP,#1
        043300  001372                                  BNE     69$
        043302  005337  003446                          DEC     ZLOOP
        043306  001365                                  BNE     70$
    264 043310  117737  137332  003502                  MOVB    @IAR,TEMP
    265 043316  123737  003502  003462                  CMPB    TEMP,TADDR      ;CHECK IF IT INTERRUPTED
    266 043324  001403                                  BEQ     11$
    267 043326  104005                                  EMT     5
    268 043330  000137  043000                          JMP     2$
    269 043334  112737  000377  001140  11$:    MOVB    #377,$GDDAT
    270 043342  117737  140122  001142                  MOVB    @COSADR,$BDDAT  ;CHECK IF COS REGISTER IS ALL ONES
    271 043350  123737  001140  001142                  CMPB    $GDDAT,$BDDAT
    272 043356  001401                                  BEQ     5$
    273 043360  104020                                  EMT     20
    274 043362  152777  000001  137254  5$:     BISB    #RBIT,@CSR
    275 043370  105777  140074                          TSTB    @COSADR         ;CLEAR INTERRUPT
    276 043374  005037  001140                          CLR     $GDDAT
    277 043400  117737  140064  001142                  MOVB    @COSADR,$BDDAT
    278 043406  001401                                  BEQ     6$
    279 043410  104013                                  EMT     13
    280 043412  005337  003470                  6$:     DEC     COSADR
    281 043416  005337  003462                          DEC     TADDR           ;RESET ADDRESS
```

```
    282 043422  004737 015410                      JSR     PC,CLRINT       ;CLEAR REMAINING INTERRUPTS
    283 043426  132777 000010 137210               BITB    #TBIT,@CSR      ;CHECK IF CLEARING T BIT CAUSE INTERRUPT
    284 043434  001433                             BEQ     9$
    285 043436  012746 000000                      MOV     #PR0,-(SP)      ;SET PSW TO PRIORITY 0
        043442  012746 043450                      MOV     #71$,-(SP)
        043446  000002                             RTI
        043450  000240                  71$:       NOP
    286 043452  142777 000010 137164               BICB    #TBIT,@CSR      ;CLEAR INPUTS
    287 043460  152777 000100 137156               BISB    #EBIT,@CSR
    288 043466  013737 002636 003446               MOV     K,ZLOOP
        043474  005037 003444              73$:    CLR     YLOOP           ;WAIT
        043500  005237 003444              72$:    INC     YLOOP
        043504  023727 003444 177777               CMP     YLOOP,#-1
        043512  001372                             BNE     72$
        043514  005337 003446                      DEC     ZLOOP
        043520  001365                             BNE     73$
    289 043522  104005                             EMT     5
    290 043524  000137 043000              9$:     JMP     2$
    291
    292 043530  022626                    10$:    CMP     (SP)+,(SP)+     ;RESET STACK POINTER
    293 043532  104005                             EMT     5
    294 043534  000137 043000                      JMP     2$
```

```
                                        .SBTTL  DIGITAL MODULE TEST M5012,M5012YA


                                                ;THIS TEST CHECKS M5012 MODULE
                                                ;ISOLATED 16 BIT DC INPUT


                                        M5012:
                                        ;;**********************************************************************
                                        TST20:  SCOPE
043540  000004
043542  012737  000020  001212                  MOV     #20,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
043550  152777  000020  137066                  BISB    #DBIT,@CSR
043556  112737  000000  001140                  MOVB    #0,$GDDAT       ;SET WHAT DATA SHOULD BE
043564  012737  000002  003460                  MOV     #2,BYTNUM       ;SET NUMBER OF BYTES
043572  004737  015022                          JSR     PC,TSTBYT       ;CHECK IF DATA IS CORRECT

043576  152777  000010  137040                  BISB    #TBIT,@CSR
043604  112737  000377  001140                  MOVB    #377,$GDDAT     ;SET WHAT DATA SHOULD BE
043612  012737  000002  003460                  MOV     #2,BYTNUM       ;SET NUMBER OF BYTES
043620  004737  015022                          JSR     PC,TSTBYT       ;CHECK IF DATA IS CORRECT
043624  142777  000010  137012                  BICB    #TBIT,@CSR
043632  004737  015410                          JSR     PC,CLRINT
043636  012746  000000                          MOV     #PR0,-(SP)      ;SET PSW TO PRIORITY 0
043642  012746  043650                          MOV     #64$,-(SP)
043646  000002                                  RTI
043650  000240                          64$:    NOP
043652  012777  044006  136770                  MOV     #6$,@VECTO      ;SET INTERRUPT VECTOR
043660  012777  000340  136764                  MOV     #PR7,@VECTOA
043666  152777  000100  136750                  BISB    #EBIT,@CSR      ;ENABLE INTERRUPT
043674  152777  000010  136742                  BISB    #TBIT,@CSR      ;START INTERRUPT
043702  013737  002636  003446                  MOV     K,ZLOOP
043710  005037  003444                  66$:    CLR     YLOOP           ;WAIT
043714  005237  003444                  65$:    INC     YLOOP
043720  023727  003444  177777                  CMP     YLOOP,#-1
043726  001372                                  BNE     65$
043730  005337  003446                          DEC     ZLOOP
043734  001365                                  BNE     66$
043736  132737  000001  001226                  BITB    #BIT0,$ENV      ;UNDER APT ?
043744  001403                                  BEQ     50$             ;NO
043746  005737  001214                          TST     $PASS           ;FIRST PASS ($PASS = 0) ?
043752  001001                                  BNE     7$              ;NO
043754                                  50$:
043754  104005                                  EMT     5
043756  142777  000100  136660          7$:     BICB    #EBIT,@CSR      ;CLEAR CSR
043764  142777  000010  136652                  BICB    #TBIT,@CSR
043772  012737  007606  000004                  MOV     #TMOVEC,ERRVEC
044000  004737  015410                          JSR     PC,CLRINT       ;CLEAR ALL INTERRUPT
044004  000207                                  RTS     PC              ;RETURN TO MONITOR

044006                                  6$:     ; INTERRUPT ROUTINE

044006  022626                                  CMP     (SP)+,(SP)+     ;RESET STACK POINTER
044010  005037  003510                          CLR     TEMP1
044014  113737  002661  003503                  MOVB    BASE+1,TEMP+1   ;INITIALIZE TEMP
044022  012737  044342  000004                  MOV     #4$,ERRVEC
044030                                  3$:
044030  013737  002636  003446                  MOV     K,ZLOOP
```

```
                044036  005037  003444          68$:    CLR     YLOOP           ;WAIT
                044042  005237  003444          67$:    INC     YLOOP
                044046  023727  003444  000001          CMP     YLOOP,#1
                044054  001372                          BNE     67$
                044056  005337  003446                  DEC     ZLOOP
                044062  001365                          BNE     68$
            336 044064  117737  136556  003502          MOVB    @IAR,TEMP       ;FIND WHICH I/O INTERRUPTED
            337 044072  152777  000001  136544          BISB    #RBIT,@CSR
            338 044100  105777  137376                  TSTB    @TEMP           ;CLEAR INTERRUPT
            339 044104  123737  003502  003462          CMPB    TEMP,TADDR      ;IS IT MUT
            340 044112  001410                          BEQ     2$              ;YES
            341
            342 044114  005237  003510          10$:    INC     TEMP1
            343 044120  022737  000400  003510          CMP     #400,TEMP1
            344 044126  001340                          BNE     3$
            345 044130  104005                          EMT     5
            346 044132  000711                          BR      7$
            347
            348 044134  005237  003462          2$:     INC     TADDR
            349 044140  152777  000001  136476          BISB    #RBIT,@CSR
            350 044146  105777  137310                  TSTB    @TADDR          ;CLEAR INTERRUPT IN NEXT BYTE
            351 044152  005337  003462                  DEC     TADDR           ;RESTORE TADDR
            352 044156  013737  002636  003446          MOV     K,ZLOOP
                044164  005037  003444          70$:    CLR     YLOOP           ;WAIT
                044170  005237  003444          69$:    INC     YLOOP
                044174  023727  003444  000001          CMP     YLOOP,#1
                044202  001372                          BNE     69$
                044204  005337  003446                  DEC     ZLOOP
                044210  001365                          BNE     70$
            353 044212  132777  000200  136424          BITB    #FBIT,@CSR      ;CHECK IF INTERRUPT CLEAR
            354 044220  001413                          BEQ     8$
            355 044222  117737  136420  003502          MOVB    @IAR,TEMP
            356 044230  123737  003502  003462          CMPB    TEMP,TADDR
            357 044236  001002                          BNE     9$
            358 044240  104015                          EMT     15
            359 044242  000645                          BR      7$
            360 044244  004737  015410          9$.     JSR     PC,CLRINT       ;CLEAR ALL REMINDING INTERRUPTS
            361 044250  132777  000010  136366  8$:     BITB    #TBIT,@CSR
            362 044256  001637                          BEQ     7$
            363 044260  012746  000000                  MOV     #PR0,-(SP)      ;SET PSW TO PRIORITY 0
                044264  012746  044272                  MOV     #71$,-(SP)
                044270  000002                          RTI
                044272  000240          71$:    NOP
            364 044274  142777  000010  136342          BICB    #TBIT,@CSR      ;CHECK IF CLEARING T BIT CAUSE INTERRUPT
            365 044302  013737  002636  003446          MOV     K,ZLOOP
                044310  005037  003444          73$:    CLR     YLOOP           ;WAIT
                044314  005237  003444          72$:    INC     YLOOP
                044320  023727  003444  177777          CMP     YLOOP,#-1
                044326  001372                          BNE     72$
                044330  005337  003446                  DEC     ZLOOP
                044334  001365                          BNE     73$
            366 044336  104005                          EMT     5
            367 044340  000606                          BR      7$
            368
            369 044342  022626                  4$:     CMP     (SP)+,(SP)+     ;RESET STACK
            370 044344  000663                          BR      10$
            371
```

```
                                        .SBTTL  TESTING DIGITAL MODULE M5014
                                                ;M5014 DUAL 16-BIT INPUT COUNTER

044346                                  M5014:
                                        ;;*****************************************************************
044346  000004                          TST21:  SCOPE
044350  012737  000021  001212                  MOV     #21,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX

044356  104411                                  SAVREG
044360  012746  000340                          MOV     #PR7,-(SP)      ;SET PSW TO PRIORITY 7
044364  012746  044372                          MOV     #64$,-(SP)
044370  000002                                  RTI
044372  000240                          64$:    NOP
                                                ;ADDRESS INITIALIZATION
044374  013737  003462  003626                  MOV     TADDR,ACNTL     ;INITIALIZE COUNTER A LOW-BYTE
044402  013737  003462  003624                  MOV     TADDR,ACNTH     ;INITIALIZE COUNTER A HIGH-BYTE
044410  062737  000001  003624                  ADD     #1,ACNTH        ;MAKE COUNTER A ADDRESS
044416  013737  003462  003632                  MOV     TADDR,BCNTL     ;INITIALIZE COUNTER B LOW-BYTE
044424  062737  000002  003632                  ADD     #2,BCNTL        ;MAKE COUNTER B ADDRESS
044432  013737  003462  003630                  MOV     TADDR,BCNTH     ;INITIALIZE COUNTER B HIGH-BYTE
044440  062737  000003  003630                  ADD     #3,BCNTH        ;MAKE COUNTER B ADDRESS

                                                ;START WITH A KNOWN CONDITION -- CLEAR THE MODULE

044446  004737  014310                          JSR     PC,KLEER        ;CLEAR MODULE.
044452  152777  000020  136164                  BISB    #DBIT,@CSR      ;SHUT OFF WORLD

                                                ;STATUS  OF  M5014  AT  THIS  TIME  SHOULD  BE  --

                                                ;       COUNTER A = 0           COUNTER B - 0

                                                ;CHECK THAT COUNTERS ARE CLEAR

044460  005037  001140                          CLR     $GDDAT          ;INITIALIZE EXPECTED DATA.
044464  013737  003626  003462                  MOV     ACNTL,TADDR     ;TEST COUNTER A
044472  013737  003626  003636                  MOV     ACNTL,CTRSL
044500  013737  003624  003634                  MOV     ACNTH,CTRSH

044506  117737  137122  001143  1$:     MOVB    @CTRSH,$BDDAT+1 ;READ HIGH BYTE OF COUNTER.
044514  117737  137116  001142          MOVB    @CTRSL,$BDDAT   ;READ LOW BYTE OF COUNTER.
044522  023737  001140  001142          CMP     $GDDAT,$BDDAT   ;IF DATA IS AS EXPECTED
044530  001401                          BEQ     2$              ;-THEN SKIP ERROR

044532  104075                          EMT     75

044534  023737  003632  003462  2$:     CMP     BCNTL,TADDR     ;IF DONE CHECKING CLEAR
044542  001412                          BEQ     TSTPT2          ;-THEN DO PATTERN TEST
044544  013737  003632  003462          MOV     BCNTL,TADDR     ;-ELSE TEST COUNTER B
044552  013737  003632  003636          MOV     BCNTL,CTRSL
044560  013737  003630  003634          MOV     BCNTH,CTRSH
044566  000747                          BR      1$
```

```
                                                ;CHECK THE READ/WRITE CAPABILITIES OF THE COUNTERS

044570  005000                        TSTPT2: CLR     R0              ;INITIALIZE PATTERN POINTER.
044572  142777  000010  136044                BICB    #TBIT,@CSR      ;ASSURE THAT TBIT IS OFF

044600  013737  003626  003462        1$:     MOV     ACNTL,TADDR     ;TEST COUNTER A
044606  013737  003626  003636                MOV     ACNTL,CTRSL
044614  013737  003624  003634                MOV     ACNTH,CTRSH
044622  116037  002630  001140                MOVB    PATT(R0),$GDDAT ;GET EXPECTED PATTERN
044630  116037  002630  001141                MOVB    PATT(R0),$GDDAT+1

044636  152777  000010  136000        2$:     BISB    #TBIT,@CSR      ;INITIALIZE BOARD
044644  113777  001141  136762                MOVB    $GDDAT+1,@CTRSH ;WRITE INTO HIGH BYTE OF COUNTER
044652  113777  001140  136756                MOVB    $GDDAT,@CTRSL   ;WRITE PATTERN INTO LOW BYTE
044660  117737  136750  001143                MOVB    @CTRSH,$BDDAT+1 ;READ COUNTERS HIGH BYTE
044666  117737  136744  001142                MOVB    @CTRSL,$BDDAT   ;READ LOW BYTE OF COUNTER
044674  023737  001140  001142                CMP     $GDDAT,$BDDAT   ;IF PATTERN IS AS EXPECTED
044702  001401                                BEQ     3$              ;-THEN SKIP ERROR

044704  104076                                EMT     76

044706  142777  000010  135730        3$:     BICB    #TBIT,@CSR      ;STOP COUNTING
044714  023737  003632  003462                CMP     BCNTL,TADDR     ;IF COUNTER B JUST TESTED
044722  001414                                BEQ     4$              ;-THEN DO NEXT PATTERN

044724  005137  001140                        COM     $GDDAT          ;-ELSE PRODUCE EXPECTED PATTERN
044730  013737  003632  003462                MOV     BCNTL,TADDR     ; TEST COUNTER B
044736  013737  003632  003636                MOV     BCNTL,CTRSL
044744  013737  003630  003634                MOV     BCNTH,CTRSH
044752  000731                                BR      2$

044754  005200                        4$:     INC     R0              ;MOVE PATTERN POINTER
044756 .122760  000123  002630                CMPB    #123,PATT(R0)   ;IF NOT AT END OF PATTERN TABLE
044764  001305                                BNE     1$              ;-THEN DO NEW PATTERN
```

```
                                                ;CHECK THAT TBIT INITIALIZES MODULE

045766  005037  001140          TINIMD: CLR     $GDDAT          ;INITIALIZE EXPECTED DATA.
045772  112777  177777  136624          MOVB    #-1,@ACNTH      ;LOAD THE HIGH BYTE BUFFER.
045000  112777  177777  136620          MOVB    #-1,@ACNTL      ;LOAD COUNTER A WITH ALL 1'S.
045006  112777  177777  136616          MOVB    #-1,@BCNTL      ;LOAD COUNTER B WITH ALL 1'S.
045014  112777  177777  136602          MOVB    #-1,@ACNTH      ;LOAD THE HIGH BYTE
045022  112777  177777  136600          MOVB    #-1,@BCNTH      ;LOAD COUNTER B HIGH BYTE
045030  112777  000020  135606          MOVB    #DBIT,@CSR      ;ASSURE CSR IS CLEAR.

045036  142777  000010  135600          BICB    #TBIT,@CSR      ;INITIALIZE BOARD.
045044  152777  000010  135572          BISB    #TBIT,@CSR
045052  013737  003626  003462          MOV     ACNTL,TADDR     ;INITIALIZE FIRST TEST ADDRESS
045060  013737  003626  003636          MOV     ACNTL,CTRSL
045066  013737  003624  003634          MOV     ACNTH,CTRSH

045074  117737  136534  001143  1$:     MOVB    @CTRSH,$BDDAT+1 ;READ HIGH BYTE OF TEST COUNTER
045102  117737  136530  001142          MOVB    @CTRSL,$BDDAT   ;READ LOW BYTE OF TEST COUNTER
045110  023737  001140  001142          CMP     $GDDAT,$BDDAT   ;IF DATA IS AS EXPECTED
045116  001401                          BEQ     2$              ;-THEN SKIP ERROR

045120  104077                          EMT     77

045122  023737  003632  003462  2$:     CMP     BCNTL,TADDR     ;IF DONE CHECKING INIT
045130  001412                          BEQ     INTHL2          ;-THEN CHECK INERRUPTS
045132  013737  003632  003462          MOV     BCNTL,TADDR     ;-ELSE TEST COUNTER B.
045140  013737  003632  003636          MOV     BCNTL,CTRSL
045146  013737  003630  003634          MOV     BCNTH,CTRSH
045154  000747                          BR      1$

045156  112777  000020  135460  INTHL2: MOVB    #DBIT,@CSR      ;START INTERNAL FREQ SOURCE
045164  152777  000010  135452          BISB    #TBIT,@CSR
045172  013737  002636  003446          MOV     K,ZLOOP
045200  005037  003444          65$:    CLR     YLOOP           ;WAIT
045204  005237  003444          64$:    INC     YLOOP
045210  023727  003444  000077          CMP     YLOOP,#77
045216  001372                          BNE     64$
045220  005337  003446                  DEC     ZLOOP
045224  001365                          BNE     65$
045226  004737  015410                  JSR     PC,CLRINT       ;CLEAR ANY INTERRUPTS CAUSED BY TBIT.

045232  013737  003626  003462  1$:     MOV     ACNTL,TADDR     ;INITIALIZE FIRST COUNTER TO TEST
045240  013737  003626  003636          MOV     ACNTL,CTRSL
045246  013737  003624  003634          MOV     ACNTH,CTRSH
045254  012777  045456  135366          MOV     #2$,@VECTO      ;INITIALIZE INTERRUPT VECTOR
045262  012777  000340  135362          MOV     #PR7,@VECTOA    ;INITIALIZE INTERRUPT PRIORITY
045270  012737  177776  003642          MOV     #-2,NUM         ;INITIALIZE A COUNT
```

```
                                                ;ALLOW AN INTERRUPT AND WAIT

045276                                  8$:
045276  012746  000000                          MOV     #PR0,-(SP)      ;SET PSW TO PRIORITY 0
045302  012746  045310                          MOV     #66$,-(SP)
045306  000002                                  RTI
045310  000240                          66$:    NOP
045312  152777  000100  135324                  BISB    #EBIT,@CSR      ;ALLOW THE INTERRUPT
045320  113777  003643  136306                  MOVB    NUM+1,@CTRSH    ;CLEAR HIGH BYTE BUFFER
045326  113777  003642  136302                  MOVB    NUM,@CTRSL      ;LOAD COUNTER
045334  013737  002636  003446                  MOV     K,ZLOOP
045342  005037  003444                  68$:    CLR     YLOOP           ;WAIT
045346  005237  003444                  67$:    INC     YLOOP
045352  023727  003444  177777                  CMP     YLOOP,#-1
045360  001372                                  BNE     67$
045362  005337  003446                          DEC     ZLOOP
045366  001365                                  BNE     68$
045370  013737  002636  003446                  MOV     K,ZLOOP
045376  005037  003444                  70$:    CLR     YLOOP           ;WAIT
045402  005237  003444                  69$:    INC     YLOOP
045406  023727  003444  177777                  CMP     YLOOP,#-1
045414  001372                                  BNE     69$
045416  005337  003446                          DEC     ZLOOP
045422  001365                                  BNE     70$

045424  142777  000100  135212                  BICB    #EBIT,@CSR      ;SHUT OFF INTERRUPTS
045432  012746  000340                          MOV     #PR7,-(SP)      ;SET PSW TO PRIORITY 7
045436  012746  045444                          MOV     #71$,-(SP)
045442  000002                                  RTI
045444  000240                          71$:    NOP

045446  104103                                  EMT     103

045450  000532                                  BR      7$              ;SKIP INTERRUPT ROUTINE
```

```
                                        ;TIMEOUT HANDLER FOR INTERRUPT CHECK

045452  022626                  33$:   CMP     (SP)+,(SP)+     ;RESET STACK POINTER
045454  000452                         BR      38$             ;CONTINUE INTERRUPT CHECKS

                                ;ON INTERRUPT DO 2$

045456  022626                  2$:    CMP     (SP)+,(SP)+     ;RESET STACK POINTER
045460  013746  000004                 MOV     ERRVEC,-(SP)    ;SAVE TIMEOUT VECTOR
045464  013746  000006                 MOV     ERRVEC+2,-(SP)  ;SAVE TIMEOUT PRIORITY
045470  013737  045452  000004         MOV     33$,ERRVEC      ;ON TIME OUT GO TO 33$
045476  01273   000340  000006         MOV     #PR7,ERRVEC+2   ;ASSURE PSW REMAINS PRIORITY 7
045504  005037  003510                 CLR     TEMP1           ;PREPARE TO CONSTRUCT INTERRUPT ADDRESS
045510  013737  002660  003502         MOV     BASE,TEMP       ;INITIALIZE TEMP
045516  152777  000001  135120  3$:    BISB    #RBIT,@CSR      ;PREPARE TO CLEAR INTERRUPT

                                        ;CHECK IF COUNTER INTERRUPTS

045524  117737  135116  003502         MOVB    @IAR,TEMP       ;GET LOW BYTE OF INTERRUPT ADDRESS

045532  023737  003626  003462         CMP     ACNTL,TADDR     ;IF COUNTER A IS BEING TESTED
045540  001406                         BEQ     35$             ;-THEN SEE IF COUNTER B INTERRUPTS

045542  023737  003626  003502         CMP     ACNTL,TEMP      ;-ELSE IF A DIDN'T INTERRUPT
045550  001006                         BNE     36$             ;--THEN SKIP ERROR

045552                          34$:
045552  104110                         EMT     110
045554  000410                         BR      37$             ;---CLEAR UNWANTED INTERRUPT

045556  023737  003632  003502  35$:   CMP     BCNTL,TEMP      ;IF COUNTER B INTERRUPTED
045564  001772                         BEQ     34$             ;-THEN IT WAS UNEXPECTED

045566  023737  003502  003462  36$:   CMP     TEMP,TADDR      ;IF COUNTER INTERRUPTED
045574  001411                         BEQ     4$              ;-THEN CLEAN UP.

                                        ;CLEAR UNWANTED INTERRUPT

045576  105777  135700          37$:   TSTB    @TEMP           ;CLEAR UNWANTED INTERRUPT
045602  005237  003510          38$:   INC     TEMP1           ;INCREMENT INTERRUPT+TIMEOUT COUNT
045606  022737  000400  003510         CMP     #400,TEMP1      ;IF NOT EXCESSIVE COUNT NUMBER
045614  001340                         BNE     3$              ;-THEN CHECK AGAIN

045616  104103                         EMT     103

045620  012637  000004          4$:    MOV     (SP)+,ERRVEC    ;RESTORE TIMEOUT VECTOR
045624  012637  000006                 MOV     (SP)+,ERRVEC+2  ;RESTORE TIMEOUT PRIORITY
045630  004737  015410                 JSR     PC,CLRINT       ;CLEAR ALL INTERRUPTS
045634  132777  000200  135002         BITB    #FBIT,@CSR      ;IF INTERRUPT CLEARS
045642  001401                         BEQ     5$              ;THEN SKIP ERROR.

045644  104074                         EMT     74

045646  012737  177777  001140  5$:    MOV     #-1,$GDDAT      ;INITIALIZE EXPECTED DATA
045654  013737  002636  003446         MOV     K,ZLOOP
045662  005037  003444          73$:   CLR     YLOOP           ;WAIT
045666  005237  003444          72$:   INC     YLOOP
```

```
        045672  023727  003444  000500          CMP     YLOOP,#500
        045700  001372                          BNE     72$
        045702  005337  003446                  DEC     ZLOOP
        045706  001365                          BNE     73$
    567 045710  117737  135720  001143          MOVB    @CTRSH,$BDDAT+1 ;READ HIGH BYTE
    568 045716  117737  135714  001142          MOVB    @CTRSL,$BDDAT   ;READ LOW BYTE BUFFER
    569 045724  023737  001140  001142          CMP     $GDDAT,$BDDAT   ;IF COUNTER HALTED AT -1
    570 045732  001401                          BEQ     7$              ;-THEN SKIP ERROR
    571
    572 045734  104100                          EMT     100
    573
    574                             ;-END 2$
    575
    576 045736  023737  003632  003462  7$:     CMP     BCNTL,TADDR     ;IF FINISHED CHECKING INTERRUPTS
    577 045744  001413                          BEQ     TRANP2          ;-THEN TEST TRANSITION POINTS
    578 045746  013737  003632  003462          MOV     BCNTL,TADDR     ;-ELSE CHECK COUNTER B
    579 045754  013737  003632  003636          MOV     BCNTL,CTRSL
    580 045762  013737  003630  003634          MOV     BCNTH,CTRSH
    581 045770  000137  045276                  JMP     8$
```

```
                                                ;TEST THE TRANSITION POINTS
                                                ;
045774  112777  000020  134642  TRANP2: MOVB    #DBIT,@CSR      ;CLEAR EXTRANEOUS BITS AT CSR.
046002  152777  000010  134634          BISB    #TBIT,@CSR
046010  013737  003626  003462          MOV     ACNTL,TADDR     ;INITIALIZE COUNTER TEST ADDRESS
046016  013701  003624                  MOV     ACNTH,R1        ;INITIALIZE HIGH BYTE ADDRESS
046022  013700  003626                  MOV     ACNTL,R0        ;INITIALIZE LOW BYTE ADDRESS
                                                ;
046026  012737  100000  003642  1$:     MOV     #BIT15,NUM      ;INITIALIZE TRANSITION NUMBER
                                                ;
046034  000257                  5$:     CCC                     ;CLEAR ALL CONDITION CODES
046036  006037  003642                  ROR     NUM             ;DETERMINE NEXT TRANSITION VALUE
046042  103014                  BCC     2$                      ;IF NOT FINISHED THIS COUNTER DO 2$.
                                                ;
046044  023737  003462  003632  6$:     CMP     TADDR,BCNTL     ;IF COUNTER B WAS CHECKED
046052  001453                  BEQ     STST12                  ;-THEN CHECK FREQUENCY
046054  013737  003632  003462          MOV     BCNTL,TADDR     ;-ELSE PREPARE TO TEST COUNTER B
046062  013701  003630                  MOV     BCNTH,R1        ;--LOAD HIGH BYTE ADDRESS
046066  013700  003632                  MOV     BCNTL,R0        ;--LOAD LOW BYTE ADDRESS
046072  000755                  BR      1$                      ;
                                                ;
046074  005005                  2$:     CLR     R5              ;INITIALIZE LOOP COUNT
046076  013737  003642  001140          MOV     NUM,$GDDAT      ;COMPUTE EXPECTED DATA
046104  013737  003642  003502          MOV     NUM,TEMP        ;COMPUTE DATA TO LOAD
046112  005337  003502                  DEC     TEMP
                                                ;
046116  113711  003503                  MOVB    TEMP+1,(R1)     ;LOAD  HIGH BYTE OF COUNTER
046122  113710  003502                  MOVB    TEMP,(R0)       ;LOAD COUNTER
046126  000240                          NOP
046130  000240                          NOP
046132  000240                          NOP
                                                ;
046134  111137  001143          3$:     MOVB    (R1),$BDDAT+1   ;READ HIGH BYTE
046140  111037  001142                  MOVB    (R0),$BDDAT     ;READ LOW BYTE
046144  000240                          NOP
046146  023737  003502  001142          CMP     TEMP,$BDDAT     ;IF COUNT CHANGED
046154  001004                          BNE     4$              ;-THEN CHECK DATA
046156  005305                          DEC     R5              ;-ELSE IF DELAY NOT UP
046160  001365                          BNE     3$              ;--THEN LOOK FOR A COUNT
                                                ;
046162  104101                          EMT     101
                                                ;
046164  000723                          BR      5$
                                                ;
046166  023737  001140  001142  4$:     CMP     $GDDAT,$BDDAT   ;IF COUNT WAS GOOD
046174  001717                          BEQ     5$              ;-THEN SKIP ERROR
                                                ;
046176  104102                          EMT     102
                                                ;
046200  000715                          BR      5$              ;-THEN DO ANOTHER TRANSITION CHECK
```

```
                                        ;TEST THAT COUNTERS COUNT TO WITHIN 20% OF 5KHZ

046202  004737  014310          STST12: JSR     PC,KLEER        ;CLEAR EVERYTHING
046206  112777  000020  134430          MOVB    #DBIT,@CSR      ;SHUT OFF WORLD
046214  152777  000010  134422          BISB    #TBIT,@CSR      ;ENABLE 5KHZ CLOCKING
046222  013737  002636  003446          MOV     K,ZLOOP
046230  005037  003444          65$:    CLR     YLOOP           ;WAIT
046234  005237  003444          64$:    INC     YLOOP
046240  023727  003444  000077          CMP     YLOOP,#77
046246  001372                          BNE     64$
046250  005337  003446                  DEC     ZLOOP
046254  001365                          BNE     65$
046256  004737  015410                  JSR     PC,CLRINT       ;CLEAR ALL INTERRUPTS
046262  012777  015402  134376          MOV     #COUNT,@CLKVC   ;INITIALIZE CLOCK VECTOR
046270  012777  000340  134372          MOV     #PR7,@CLKVCA    ;SET UP PRIORITY FOR CLK INT
046276  012777  046712  134344          MOV     #4$,@VECTO      ;INITIALIZE INTERRUPT VECTOR
046304  012777  000340  134340          MOV     #PR7,@VECTOA    ;SET UP PRIORITY FOR INT

046312  012737  001750  003644          MOV     #1000.,VARI     ;INITIALIZE ALLOWED ERROR
046320  012737  165204  001140          MOV     #-5500.,$GDDAT  ;INITIALIZE EXPECTED DATA
046326  163737  003644  001140          SUB     VARI,$GDDAT     ;SUBTRACT FUDGE FACTOR
046334  005037  003642                  CLR     NUM             ;INITIALIZE PROGRAM POSITION POINTER

                                        ;ALLOW THE INTERRUPTS

046340  152777  000100  134276          BISB    #EBIT,@CSR      ;ENABLE INTERRUPT BIT AT CSR
046346  012746  000000                  MOV     #PR0,-(SP)      ;SET PSW TO PRIORITY 0
046352  012746  046360                  MOV     #66$,-(SP)
046356  000002                          RTI
046360  000240                  66$:    NOP

046362  113777  001141  135234          MOVB    $GDDAT+1,@ACNTH ;LOAD HIGH BYTE BUFFER
046370  113777  001140  135230          MOVB    $GDDAT,@ACNTL   ;LOAD COUNTER A
046376  113777  001140  135226          MOVB    $GDDAT,@BCNTL   ;LOAD COUNTER B

                                        ;TEST THAT THE CLOCK IS ON

046404  005037  003476                  CLR     CLK             ;INITIALIZE CLOCK
046410  005037  003444                  CLR     YLOOP           ;INITIALIZE LOOP COUNT
046414  005737  003476          11$:    TST     CLK             ;IF CLOCK IS COUNTING
046420  001005                          BNE     15$             ;-THEN CHECK THAT COUNTERS COUNT
046422  005237  003444                  INC     YLOOP           ;-ELSE IF CLOCK MAY COUNT
046426  001372                          BNE     11$             ;--THEN CHECK FOR COUNT

046430  104025                          EMT     25
046432  000514                          BR      14$             ;--SKIP CLOCK DEPENDENT TESTS

046434  012737  000001  003642  15$:    MOV     #1,NUM          ;SET A CLOCK FLAG

                                        ;WAIT UP TO ONE SECOND FOR A COUNT

046442  105777  135156          12$:    TSTB    @ACNTH          ;LOAD LOW BYTE BUFFER
046446  127737  135154  001140          CMPB    @ACNTL,$GDDAT   ;IF COUNTER A IS COUNTING
046454  001015                          BNE     13$             ;-THEN CHECK FREQUENCY
046456  105777  135146                  TSTB    @BCNTH          ;-ELSE CHECK COUNTER B
046462  127737  135144  001140          CMPB    @BCNTL,$GDDAT   ;--IF COUNTER B IS COUNTING
046470  001007                          BNE     13$             ;---THEN CHECK FREQUENCY
```

```
                                                ;---ELSE IF 1 SECOND NOT UP
                                                ;----THEN WAIT FOR COUNT

                                                ;WAIT ONE FULL SECOND
                                                ;INDICATE PROGRAM MADE IT HERE

                                                ;INITIALIZE THE CLOCK

                                                ;IF 1 SECOND NOT UP
                                                ;-THEN WAIT

                                                ;DON'T LEAVE THESE BITS HANGING
                                                ;SET PSW TO PRIORITY 7

                                                ;GET CONTENTS OF COUNTER A

                                                ;GET CONTENTS OF COUNTER B

                                                ;GET EXPECTED COUNT
                                                ;DOUBLE THE VARIANCE
                                                ;GET TWO'S COMPLIMENT

                                                ;IF COUNT IS WITHIN 20% OF 5KHZ
                                                ;-THEN SKIP ERROR

                                                ;-ELSE PREPARE FOR ERROR

                                                ;IF COUNT IS WITHIN 20% OF 5KHZ
                                                ;-THEN SKIP ERROR

                                                ;-ELSE PREPARE FOR ERROR
                                                ;--DITTO

                                                ;CHECK TIME BASE

                                                ;DON'T LEAVE BITS HANGING
                                                ;SET PSW TO PRIORITY 7

                                                ;SKIP CLOCK DEPENDENT TESTS
```

```
                                                ;ON INTERRUPT DO 4$

046712  022626                          4$:     CMP     (SP)+,(SP)+     ;RESTORE STACK POINTER
046714  013737  003476  003510                  MOV     CLK,TEMP1       ;GET TIME AT INTERRUPT
046722  013737  002660  003502                  MOV     BASE,TEMP       ;INITIALIZE TEMP
046730  117737  133712  003502                  MOVB    @IAR,TEMP       ;GET INTERRUPTING ADDRESS

046736  023737  003626  003502                  CMP     ACNTL,TEMP      ;IF NOT COUNTER A INTERRUPTING
046744  001013                                  BNE     5$              ;-THEN SKIP ERROR

046746  117737  134652  001143                  MOVB    @ACNTH,$BDDAT+1
046754  117737  134646  001142                  MOVB    @ACNTL,$BDDAT
046762  013737  003626  003462                  MOV     ACNTL,TADDR     ;-ELSE PREPARE FOR ERROR TYPEOUT
046770  104106                                  EMT     106
046772  000445                                  BR      7$              ;CLEAN UP

046774  023737  003632  003502          5$:     CMP     BCNTL,TEMP      ;IF NOT COUNTER B INTERRUPTING
047002  001013                                  BNE     6$              ;-THEN SKIP ERROR

047004  117737  134620  001143                  MOVB    @BCNTH,$BDDAT+1
047012  117737  134614  001142                  MOVB    @BCNTL,$BDDAT
047020  013737  003632  003462                  MOV     BCNTL,TADDR     ;-ELSE PREPARE FOR ERROR TYPEOUT
047026  104106                                  EMT     106

047030  000426                                  BR      7$              ;CLEAN UP

047032  152777  000001  133604          6$:     BISB    #RBIT,@CSR      ;CLEAR UNWANTED INTERRUPT
047040  105777  134436                          TSTB    @TEMP
047044  012746  000000                          MOV     #PR0,-(SP)      ;SET PSW TO PRIORITY 0
047050  012746  047056                          MOV     #69$,-(SP)
047054  000002                                  RTI
047056  000240                          69$:    NOP

047060  005737  003642                          TST     NUM             ;IF NOT AT 11$ MARK
047064  001002                                  BNE     16$             ;-THEN GO TO CORRECT MARK
047066  000137  046414                          JMP     11$             ;-ELSE CONTINUE AT 11$ MARK
047072  022737  000001  003642  16$:     CMP     #1,NUM          ;IF AT THE  1$ MARK
047100  001205                                  BNE     1$              ;-THEN GO TO IT
047102  000137  046442                          JMP     12$             ;-ELSE GO TO THE 12$ MARK

047106  004737  015410          7$:     JSR     PC,CLRINT       ;CLEAR ALL INTERRUPTS
047112  132777  000200  133524                  BITB    #FBIT,@CSR      ;IF INTERRUPT CLEARS
047120  001401                                  BEQ     TSECTB          ;-THEN SKIP ERROR
047122  104104                                  EMT     104
```

```
                                                ;CHECK THAT A 10 SECOND TIME BASE IS CHOSEN

047124                                  TSECTB:
047124  012746  000340                          MOV     #PR7,-(SP)      ;SET PSW TO PRIORITY 7
047130  012746  047136                          MOV     #64$,-(SP)
047134  000002                                  RTI
047136  000240                          64$:    NOP
047140  012737  141520  001140                  MOV     #50000.,$GDDAT  ;INITIALIZE EXPECTED DATA
047146  142777  000110  133470                  BICB    #TBIT!EBIT,@CSR ;ASSURE TBIT AND EBIT ARE CLEAR
047154  012777  047470  133466                  MOV     #4$,@VECTO      ;INITIALIZE INTERRUPT VECTOR
047162  012777  000340  133462                  MOV     #PR7,@VECTOA    ;INITIALIZE INTERRUPT PSW STATUS
047170  152777  000010  133446                  BISB    #TBIT,@CSR      ;INITIALIZE MODULE
047176  004737  015410                          JSR     PC,CLRINT       ;CLEAR ALL INTERRUPTS
047202  105077  134416                          CLRB    @ACNTH          ;CLEAR HIGH BYTE BUFFER
047206  105077  134414                          CLRB    @ACNTL          ;CLEAR COUNTER A
047212  105077  134414                          CLRB    @BCNTL          ;CLEAR COUNTER B
047216  005037  003476                          CLR     CLK             ;INITIALIZE CLOCK COUNT

047222  152777  000100  133414                  BISB    #EBIT,@CSR      ;ALLOW INTERRUPTS
047230  012746  000000                          MOV     #PR0,-(SP)      ;SET PSW TO PRIORITY 0
047234  012746  047242                          MOV     #65$,-(SP)
047240  000002                                  RTI
047242  000240                          65$:    NOP
047244  013737  002636  003446                  MOV     K,ZLOOP
047252  005037  003444                  67$:    CLR     YLOOP           ;WAIT
047256  005237  003444                  66$:    INC     YLOOP
047262  023727  003444  177777                  CMP     YLOOP,#-1
047270  001372                                  BNE     66$
047272  005337  003446                          DEC     ZLOOP
047276  001365                                  BNE     67$
047300  005737  003476                          TST     CLK             ;IF CLOCK IS COUNTING
047304  001014                                  BNE     1$              ;-THEN WAIT 11 SECS
047306  112777  000020  133330                  MOVB    #DBIT,@CSR      ;-ELSE CLEAN UP THE CSR
047314  012746  000340                          MOV     #PR7,-(SP)      ;SET PSW TO PRIORITY 7
047320  012746  047326                          MOV     #68$,-(SP)
047324  000002                                  RTI
047326  000240                          68$:    NOP

047330  104025                                  EMT     25
047332  000137  047546                          JMP     CBITCM          ;  AND CHECK THAT CBIT CLEARS MUT

047336  023727  003476  001332          1$:     CMP     CLK,#730.       ;IF A LITTLE MORE THAN 12 SECONDS NOT UP
047344  002774                                  BLT     1$              ;-THEN WAIT

047346  012746  000340                          MOV     #PR7,-(SP)      ;SET PSW TO PRIORITY 7
047352  012746  047360                          MOV     #69$,-(SP)
047356  000002                                  RTI
047360  000240                          69$:    NOP
047362  112777  000020  133254                  MOVB    #DBIT,@CSR      ;CLEAR EXTRANEOUS BITS
047370  104103                                  EMT     103

047372  013737  003624  003634          5$:     MOV     ACNTH,CTRSH     ;INITIALIZE FIRST TEST ADDRESS
047400  013737  003626  003636                  MOV     ACNTL,CTRSL

047406  013737  003636  003462          2$:     MOV     CTRSL,TADDR     ;LOAD TEST ADDRESS
047414  117737  134214  001143                  MOVB    @CTRSH,$BDDAT+1 ;READ HIGH BYTE OF TEST COUNTER
047422  117737  134210  001142                  MOVB    @CTRSL,$BDDAT   ;READ LOW BYTE OF TEST COUNTER
```

```
    803 047430  023737  001140  001142          CMP     $GDDAT,$BDDAT   ;IF DATA IS AS EXPECTED
    804 047436  001401                           BEQ     3$              ;-THEN SKIP ERROR
    805
    806 047440  104107                           EMT     107
    807
    808 047442  023737  003632  003462  3$:     CMP     BCNTL,TADDR     ;IF DONE CHECKING TIME BASE
    809 047450  001436                           BEQ     CBITCM          ;-THEN CHECK CBIT INIT
    810 047452  013737  003632  003636          MOV     BCNTL,CTRSL     ;-ELSE TEST COUNTER B.
    811 047460  013737  003630  003634          MOV     BCNTH,CTRSH
    812 047466  000747                           BR      2$
    813
    814 047470  022626                  4$:     CMP     (SP)+,(SP)+     ;RESTORE STACK POINTER
    815 047472  013737  002660  003502          MOV     BASE,TEMP       ;INITIALIZE TEMP
    816 047500  117737  133142  003502          MOVB    @IAR,TEMP       ;GET LOW BYTE INTERRUPTING ADDRESS
    817
    818 047506  023737  003632  003502          CMP     BCNTL,TEMP      ;IF B INTERRUPTED
    819 047514  001726                           BEQ     5$              ;-THEN SEE IF TIME BASE IS CORRECT
    820
    821 047516  152777  000001  133120          BISB    #RBIT,@CSR      ;-ELSE CLEAR UNWANTED INTERRUPT
    822 047524  105777  133752                   TSTB    @TEMP
    823 047530  012746  000000                   MOV     #PR0,-(SP)      ;SET PSW TO PRIORITY 0
        047534  012746  047542                   MOV     #70$,-(SP)
        047540  000002                           RTI
        047542  000240                  70$:     NOP
    824 047544  000674                           BR      1$              ;-CONTINUE 12 SECOND WAIT
```

```
826                                             ;CHECK THAT CBIT INITIALIZES MODULE
827
828 047546  005037  001140          CBITCM: CLR     $GDDAT          ;INITIALIZE EXPECTED DATA.
829 047552  112777  177777  134044          MOVB    #-1,@ACNTH      ;LOAD THE HIGH BYTE BUFFER.
830 047560  112777  177777  134040          MOVB    #-1,@ACNTL      ;LOAD COUNTER A WITH ALL 1'S.
831 047566  112777  177777  134036          MOVB    #-1,@BCNTL      ;LOAD COUNTER B WITH ALL 1'S.
832 047574  112777  000020  133042          MOVB    #DBIT,@CSR      ;ASSURE CSR IS CLEAR.
833
834 047602  004737  014310                  JSR     PC,KLEER        ;INITIALIZE SYSTEM
835 047606  152777  000020  133030          BISB    #DBIT,@CSR      ;SHUT OFF WORLD
836
837 047614  013737  003626  003636          MOV     ACNTL,CTRSL     ;INITIALIZE FIRST TEST ADDRESS
838 047622  013737  003624  003634          MOV     ACNTH,CTRSH
839
840 047630  013737  003636  003462  1$:     MOV     CTRSL,TADDR     ;INITIALIZE ADDRESS UNDER TEST
841 047636  117737  133772  001143          MOVB    @CTRSH,$BDDAT+1 ;READ HIGH BYTE OF TEST COUNTER
842 047644  117737  133766  001142          MOVB    @CTRSL,$BDDAT   ;READ LOW BYTE OF TEST COUNTER
843 047652  023737  001140  001142          CMP     $GDDAT,$BDDAT   ;IF DATA IS AS EXPECTED
844 047660  001401                          BEQ     2$              ;-THEN SKIP ERROR
845
846 047662  104075                          EMT     75
847
848 047664  023737  003632  003462  2$:     CMP     BCNTL,TADDR     ;IF DONE CHECKING INIT
849 047672  001407                          BEQ     M5014R          ;-THEN RETURN FROM SUBROUTINE
850 047674  013737  003632  003636          MOV     BCNTL,CTRSL     ;-ELSE TEST COUNTER B.
851 047702  013737  003630  003634          MOV     BCNTH,CTRSH
852 047710  000747                          BR      1$
853 047712  104412                  M5014R: RESREG                  ;RESTORE REGISTERS
854 047714  013737  003626  003462          MOV     ACNTL,TADDR     ;RESTORE TEST ADDRESS FOR LOOPING
855 047722  000207                          RTS     PC              ;RETURN TO MAIN PROGRAM.
```

```
                                        .SBTTL  TESTING DIGITAL MODULE M6014
                                                ;M6014 DUAL 16-BIT OUTPUT COUNTER

                                        M6014:
                                        ;;*****************************************************************
047724  000004                          TST22:  SCOPE
047726  012737  000022  001212                  MOV     #22,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX

047734  012746  000340                          MOV     #PR7,-(SP)      ;SET PSW TO PRIORITY 7
047740  012746  047746                          MOV     #64$,-(SP)
047744  000002                                  RTI
047746  000240                          64$:    NOP
                                                ;ADDRESS INITIALIZATION
047750  013737  003462  003626                  MOV     TADDR,ACNTL     ;INITIALIZE COUNTER A LOW-BYTE
047756  013737  003462  003624                  MOV     TADDR,ACNTH     ;INITIALIZE COUNTER A HIGH-BYTE
047764  062737  000001  003624                  ADD     #1,ACNTH        ;MAKE COUNTER A ADDRESS
047772  013737  003462  003632                  MOV     TADDR,BCNTL     ;INITIALIZE COUNTER B LOW-BYTE
050000  062737  000002  003632                  ADD     #2,BCNTL        ;MAKE COUNTER B ADDRESS
050006  013737  003462  003630                  MOV     TADDR,BCNTH     ;INITIALIZE COUNTER B HIGH-BYTE
050014  062737  000003  003630                  ADD     #3,BCNTH        ;MAKE COUNTER B ADDRESS

                                                ;START WITH A KNOWN CONDITION -- CLEAR THE MODULE

050022  004737  014310                          JSR     PC,KLEER        ;CLEAR MODULE.
050026  152777  000020  132610                  BISB    #DBIT,@CSR      ;SHUT OFF WORLD

                                                ;STATUS  OF  M6014  AT  THIS  TIME  SHOULD  BE  --

                                                ;       COUNTER A = 0           COUNTER B   0

                                                ;CHECK THAT COUNTERS ARE CLEAR

050034  005037  001140                          CLR     $GDDAT          ;INITIALIZE EXPECTED DATA.
050040  013737  003626  003636                  MOV     ACNTL,CTRSL     ;TEST COUNTER A
050046  013737  003624  003634                  MOV     ACNTH,CTRSH

050054  013737  003636  003462  1$.     MOV     CTRSL,TADDR     ;INITIALIZE TEST ADDRESS
050062  117737  133546  001143          MOVB    @CTRSH,$BDDAT+1 ;READ HIGH BYTE OF COUNTER.
050070  117737  133542  001142          MOVB    @CTRSL,$BDDAT   ;READ LOW BYTE OF COUNTER.
050076  023737  001140  001142          CMP     $GDDAT,$BDDAT   ;IF DATA IS AS EXPECTED
050104  001401                          BEQ     2$              ;-THEN SKIP ERROR

050106  104075                          EMT     75

050110  023737  003632  003462  2$:     CMP     BCNTL,TADDR     ;IF DONE CHECKING CLEAR
050116  001407                          BEQ     TSTPT1          ;-THEN DO PATTERN TEST
050120  013737  003632  003636          MOV     BCNTL,CTRSL     ;-ELSE TEST COUNTER B
050126  013737  003630  003634          MOV     BCNTH,CTRSH
050134  000747                          BR      1$
```

```
                                                ;CHECK THE READ/WRITE CAPABILITIES OF THE COUNTERS

050136  005000                        TSTPT1: CLR     R0              ;INITIALIZE PATTERN POINTER.
050140  142777  000010  132476                BICB    #TBIT,@CSR      ;ASSURE THAT TBIT IS OFF

050146  013737  003626  003636        1$:     MOV     ACNTL,CTPSL     ;TEST COUNTER A
050154  013737  003624  003634                MOV     ACNTH,CTRSH
050162  116037  002630  001140                MOVB    PATT(R0),$GDDAT ;GET EXPECTED PATTERN
050170  116037  002630  001141                MOVB    PATT(R0),$GDDAT+1

050176  152777  000010  132440        2$:     BISB    #TBIT,@CSR      ;INITIALIZE BOARD
050204  013737  003636  003462                MOV     CTRSL,TADDR     ;INITIALIZE TEST ADDRESS
050212  113777  001141  133414                MOVB    $GDDAT+1,@CTRSH ;WRITE INTO HIGH BYTE OF COUNTER
050220  113777  001140  133410                MOVB    $GDDAT,@CTRSL   ;WRITE PATTERN INTO LOW BYTE
050226  117737  133402  001143                MOVB    @CTRSH,$BDDAT+1 ;READ COUNTERS HIGH BYTE
050234  117737  133376  001142                MOVB    @CTRSL,$BDDAT   ;READ LOW BYTE OF COUNTER
050242  023737  001140  001142                CMP     $GDDAT,$BDDAT   ;IF PATTERN IS AS EXPECTED
050250  001401                                BEQ     3$              ;-THEN SKIP ERROR

050252  104076                                EMT     76

050254  142777  000010  132362        3$:     BICB    #TBIT,@CSR      ;STOP COUNTING
050262  023737  003632  003462                CMP     BCNTL,TADDR     ;IF COUNTER B JUST TESTED
050270  001411                                BEQ     4$              ;-THEN DO NEXT PATTERN

050272  005137  001140                        COM     $GDDAT          ;-ELSE PRODUCE EXPECTED PATTERN
050276  013737  003632  003636                MOV     BCNTL,CTRSL     ; TEST COUNTER B
050304  013737  003630  003634                MOV     BCNTH,CTRSH
050312  000731                                BR      2$

050314  005200                        4$:     INC     R0              ;MOVE PATTERN POINTER
050316  122760  000123  002630                CMPB    #123,PATT(R0)   ;IF NOT AT END OF PATTERN TABLE
050324  001310                                BNE     1$              ;-THEN DO NEW PATTERN
```

```
                                                ;CHECK THAT TBIT INITIALIZES MODULE

050326  005037  001140          TINIMO: CLR     $GDDAT          ;INITIALIZE EXPECTED DATA.
050332  112777  177777  133264          MOVB    #-1,@ACNTH      ;LOAD THE HIGH BYTE BUFFER.
050340  112777  177777  133260          MOVB    #-1,@ACNTL      ;LOAD COUNTER A WITH ALL 1'S.
050346  112777  177777  133256          MOVB    #-1,@BCNTL      ;LOAD COUNTER B WITH ALL 1'S.
050354  112777  000020  132262          MOVB    #DBIT,@CSR      ;ASSURE CSR IS CLEAR.

050362  152777  000010  132254          BISB    #TBIT,@CSR      ;INITIALIZE BOARD.
050370  013737  003626  003636          MOV     ACNTL,CTRSL     ;INITIALIZE FIRST TEST ADDRESS
050376  013737  003624  003634          MOV     ACNTH,CTRSH

050404  013737  003636  003462  1$:     MOV     CTRSL,TADDR     ;INITIALIZE TEST ADDRESS
050412  117737  133216  001143          MOVB    @CTRSH,$BDDAT+1 ;READ HIGH BYTE OF TEST COUNTER
050420  117737  133212  001142          MOVB    @CTRSL,$BDDAT   ;READ LOW BYTE OF TEST COUNTER
050426  023737  001140  001142          CMP     $GDDAT,$BDDAT   ;IF DATA IS AS EXPECTED
050434  001401                          BEQ     2$              ;-THEN SKIP ERROR

050436  104077                          EMT     77

050440  023737  003632  003462  2$:     CMP     BCNTL,TADDR     ;IF DONE CHECKING INIT
050446  001407                          BEQ     INTHL1          ;-THEN CHECK MODULE INTERRUPTS
050450  013737  003632  003636          MOV     BCNTL,CTRSL     ;-ELSE TEST COUNTER B.
050456  013737  003630  003634          MOV     BCNTH,CTRSH
050464  000747                          BR      1$

                                                ;CHECK THAT MODULE INTERRUPTS

050466  112777  000030  132150  INTHL1: MOVB    #DBIT.TBIT,@CSR ;START INTERNAL FREQ SOURCE
050474  013737  002636  003446          MOV     K,ZLOOP
050502  005037  003444          65$:    CLR     YLOOP           ;WAIT
050506  005237  003444          64$:    INC     YLOOP
050512  023727  003444  177777          CMP     YLOOP,#-1
050520  001372                          BNE     64$
050522  005337  003446                  DEC     ZLOOP
050526  001365                          BNE     65$
050530  004737  015410                  JSR     PC,CLRINT       ;CLEAR ANY INTERRUPTS CAUSED BY TBIT.

050534  013737  003624  003462  1$:     MOV     ACNTH,TADDR     ;INITIALIZE FIRST COUNTER TO TEST
050542  012777  050710  132100          MOV     #2$,@VECTO      ;INITIALIZE INTERRUPT VECTOR
050550  012777  000340  132074          MOV     #PR7,@VECTOA    ;INITIALIZE INTERRUPT PRIORITY
050556  012737  000002  003642          MOV     #2,NUM          ;INITIALIZE A COUNT
```

```
                                                ;ALLOW AN INTERRUPT AND WAIT

050564                                  8$:
050564  012746  000000                        MOV     #PR0,-(SP)      ;SET PSW TO PRIORITY 0
050570  012746  050576                        MOV     #66$,-(SP)
050574  000002                              RTI
050576  000240                          66$:  NOP
050600  152777  000100  132036              BISB    #EBIT,@CSR      ;ALLOW THE INTERRUPT
050606  113777  003643  132646              MOVB    NUM+1,@TADDR    ;CLEAR HIGH BYTE BUFFER
050614  142737  000001  003462              BICB    #BIT0,TADDR     ;PREPARE TO LOAD COUNTER
050622  113777  003642  132632              MOVB    NUM,@TADDR      ;LOAD COUNTER
050630  013737  002636  003446              MOV     K,ZLOOP
050636  005037  003444                  68$:  CLR     YLOOP           ;WAIT
050642  005237  003444                  67$:  INC     YLOOP
050646  023727  003444  177777              CMP     YLOOP,#-1
050654  001372                              BNE     67$
050656  005337  003446                      DEC     ZLOOP
050662  001365                              BNE     68$

050664  104'03                              EMT     103

050666  012746  000340                      MOV     #PR7,-(SP)      ;SET PSW TO PRIORITY 7
050672  012746  050700                      MOV     #69$,-(SP)
050676  000002                              RTI
050700  000240                          69$:  NOP
050702  000537                              BR      7$              ;SKIP INTERRUPT ROUTINE
```

```
                                        ;TIMEOUT HANDLER FOR INTERRUPT CHECK

050704  022626              33$:      CMP     (SP)+,(SP)+     ;RESET STACK POINTER
050706  000452                        BR      38$             ;CONTINUE INTERRUPT CHECKS

                                    ;ON INTERRUPT DO 2$

050710  022626              2$:       CMP     (SP)+,(SP)+     ;RESET STACK POINTER
050712  013746  000004              MOV     ERRVEC,-(SP)    ;SAVE TIMEOUT VECTOR
050716  013746  000006              MOV     ERRVEC+2,-(SP)  ;SAVE TIMEOUT PRIORITY
050722  013737  050704  000004      MOV     33$,ERRVEC      ;ON TIME OUT GO TO 33$
050730  012737  000340  000006      MOV     #PR7,ERRVEC+2   ;ASSURE PSW REMAINS PRIORITY 7
050736  005037  003510              CLR     TEMP1           ;PREPARE TO CONSTRUCT INTERRUPT ADDRESS
050742  013737  002660  003502      MOV     BASE,TEMP       ;INITIALIZE TEMP
050750  152777  000001  131666  3$: BISB    #RBIT,@CSR      ;PREPARE TO CLEAR INTERRUPT

                                    ;CHECK IF COUNTER INTERRUPTS

050756  117737  131664  003502      MOVB    @IAR,TEMP       ;GET LOW BYTE OF INTERRUPT ADDRESS

050764  023737  003626  003462      CMP     ACNTL,TADDR     ;IF COUNTER A I  BEING TESTED
050772  001406                      BEQ     35$             ;-THEN SEE IF COUNTER B INTERRUPTS

050774  023737  003626  003502      CMP     ACNTL,TEMP      ;-ELSE IF A DIDN'T INTERRUPT
051002  001006                      BNE     36$             ;--THEN SKIP ERROR

051004                        34$:
051004  104110                      EMT     110
051006  000410                      BR      37$             ;---CLEAR UNWANTED INTERRUPT

051010  023737  003632  003502  35$:  CMP     BCNTL,TEMP      ;IF COUNTER B INTERRUPTED
051016  001772                      BEQ     34$             ;-THEN IT WAS UNEXPECTED

051020  023737  003502  003462  36$:  CMP     TEMP,TADDR      ;IF COUNTER INTERRUPTED
051026  001411                      BEQ     4$              ;-THEN CLEAN UP.

                                    ;CLEAR UNWANTED INTERRUPT

051030  105777  132446        37$:  TSTB    @TEMP           ;CLEAR UNWANTED INTERRUPT
051034  005237  003510        38$:  INC     TEMP1           ;INCREMENT INTERRUPT+TIMEOUT COUNT
051040  022737  000400  003510      CMP     #400,TEMP1      ;IF NOT EXCESSIVE COUNT NUMBER
051046  001340                      BNE     3$              ;-THEN CHECK AGAIN

051050  104103                      EMT     103

051052  012637  000004        4$:   MOV     (SP)+,ERRVEC    ;RESTORE TIMEOUT VECTOR
051056  012637  000006              MOV     (SP)+,ERRVEC+2  ;RESTORE TIMEOUT PRIORITY
051062  004737  015410              JSR     PC,CLRINT       ;CLEAR ALL INTERRUPTS
051066  132777  000200  131550      BITB    #FBIT,@CSR      ;IF INTERRUPT CLEARS
051074  001401                      BEQ     5$              ;THEN SKIP ERROR.

051076  104074                      EMT     74

051100  005037  001140        5$:   CLR     $GDDAT          ;INITIALIZE EXPE TED DATA
051104  013737  002636  003446      MOV     K,ZLOOP
051112  005037  003444        71$:  CLR     YLOOP           ;WAIT
051116  005237  003444        70$:  INC     YLOOP
```

```
        051122  023727  003444  177777          CMP     YLOOP,#-1
        051130  001372                          BNE     70$
        051132  005337  003446                  DEC     ZLOOP
        051136  001365                          BNE     71$
 1043   051140  052737  000001  003462          BIS     #BIT0,TADDR     ;PREPARE TO READ HIGH BYTE
 1044   051146  117737  132310  001143          MOVB    @TADDR,$BDDAT+1 ;READ HIGH BYTE
 1045   051154  042737  000001  003462          BIC     #BIT0,TADDR     ;PREPARE TO READ LOW BYTE BUFFER
 1046   051162  117737  132274  001142          MOVB    @TADDR,$BDDAT   ;READ LOW BYTE BUFFER
 1047   051170  023737  001140  001142          CMP     $GDDAT,$BDDAT   ;IF COUNTER HALTED AT 0
 1048   051176  001401                          BEQ     7$              ;-THEN SKIP ERROR
 1049
 1050   051200  104100                          EMT     100
 1051
 1052                                  ;-END 2$
 1053
 1054   051202  023737  003632  003462  7$:     CMP     BCNTL,TADDR     ;IF FINISHED CHECKING INTERRUPTS
 1055   051210  001405                          BEQ     TRANP1          ;-THEN TEST TRANSITION POINTS
 1056   051212  013737  003630  003462          MOV     BCNTH,TADDR     ;-ELSE CHECK COUNTER B
 1057   051220  000137  050564                  JMP     8$
```

```
1059 051224  112 77  000030  131412  "RANP1: MOVB    #DBIT!TBIT,@CSR ;CLEAR EXTRANEOUS BITS AT CSR.
1060 051232  01?737  C03624  003462          MOV     ACNTH,TADDR     ;INITIALIZE COUNTER TO TEST
1061
1062 051240  C 2737  100000  003642  1$:     MOV     #BIT15,NUM      ;INITIALIZE TRANSITION NUMBER
1063
1064 051246  000257                  5$:     CCC                     ;CLEAR ALL CONDITION CODES
1065 051250  006037  003642                  ROR     NUM             ;DETERMINE NEXT TRANSITION VALUE
1066 051254  103010                  BCC     2$                      ;IF NOT FINISHED THIS COUNTER DO 2$.
1067
1068 051256  023737  003462  003632  6$:     CMP     TADDR,BCNTL     ;IF COUNTER B WAS CHECKED
1069 051264  001462                  BEQ     STSTIH                  ;-THEN CHECK BOTH TOGETHER
1070 051266  013737  003630  003462          MOV     BCNTH,TADDR     ;-ELSE PREPARE TO TEST COUNTER B
1071 051274  000761                  BR      "$                      ;
1072
1073 051276  005037  003646          2$:     CLR     XLOOP           ;INITIALIZE LOOP COUNT
1074 051302  013737  003642  001140          MOV     NUM,$GDDAT      ;COMPUTE EXPECTED DATA
1075 051310  005337  001140                  DEC     $GDDAT
1076
1077 051314  152737  000001  003462          BISB    #BIT0,TADDR     ;PREPARE TO READ HIGH BYTE
1078 051322  113777  003643  132132          MOVB    NUM+1,@TADDR    ;LOAD  HIGH BYTE OF COUNTER
1079 051330  142737  000001  003462          BICB    #BIT0,TADDR     ;PREPARE TO LOAD COUNTER
1080 051336  113777  003642  132116          MOVB    NUM,@TADDR      ;LOAD COUNTER
1081
1082 051344  152737  000001  003462  3$:     BISB    #BIT0,TADDR     ;PREPARE TO READ HIGH BYTE
1083 051352  117737  132104  001143          MOVB    @TADDR,$BDDAT+1 ;READ HIGH BYTE
1084 051360  142737  000001  003462          BICB    #BIT0,TADDR     ;PREPARE TO READ LOW BYTE
1085 051366  117737  132070  001142          MOVB    @TADDR,$BDDAT   ;READ LOW BYTE
1086 051374  023737  003642  001142          CMP     NUM,$BDDAT      ;IF COUNT CHANGED
1087 051402  001005                  BNE     4$                      ;-THEN CHECK DATA
1088 051404  005337  003646                  DEC     XLOOP           ;-ELSE IF DELAY NOT UP
1089 051410  001355                  BNE     3$                      ;--THEN LOOK FOR A COUNT
1090
1091 051412  104101                  EMT     101
1092
1093 051414  000714                  BR      5$
1094
1095 051416  023737  001140  001142  4$      CMP     $GDDAT,$BDDAT   ;IF COUNT WAS GOOD
1096 051424  001710                  BEQ     5$                      ;-THEN SKIP ERROR
1097
1098 051426  104102                  EMT     102
1099
1100 051430  000706                  BR      5$                      ;-THEN DO ANOTHER TRANSITION CHECK
1101
```

```
1103                                            ;TEST THAT COUNTERS COUNT TO WITHIN 20% OF 2KHZ
1104
1105 051432  105077  132166          STSTIM:   CLRB    @ACNTH          ;LOAD THE HIGH BYTE BUFFER
1106 051436  105077  132164                    CLRB    @ACNTL          ;CLEAR COUNTER A
1107 051442  105077  132164                    CLRB    @BCNTL          ;CLEAR COUNTER B
1108 051446  004737  015410                    JSR     PC,CLRINT       ;ASSURE PENDING INTERRUPTS ARE SERVICED
1109
1110 051452  142777  000010  131164            BICB    #TBIT,@CSR      ;PREPARE TO INITIALIZE BOARD
1111 051460  112777  000030  131156            MOVB    #DBIT.TBIT,@CSR ;INITIALIZE BOARD
1112 051466  004737  015410                    JSR     PC,CLRINT       ;CLEAR ALL INTERRUPTS
1113 051472  012777  015402  131166            MOV     #COUNT,@CLKVC   ;INITIALIZE CLOCK VECTOR
1114 051500  012777  000340  131162            MOV     #PR7,@CLKVCA    ;SET UP PRIORITY FOR CLK INT
1115 051506  012777  052000  131134            MOV     #4$,@VECTO      ;INITIALIZE INTERRUPT VECTOR
1116 051514  012777  000340  131130            MOV     #PR7,@VECTOA    ;SET UP PRIORITY FOR INT
1117
1118 051522  012737  000620  003644            MOV     #400.,VARI      ;INITIALIZE ALLOWED ERROR
1119 051530  012737  004230  001140            MOV     #2200.,$GDDAT   ;INITIALIZE EXPECTED DATA
1120 051536  063737  003644  001140            ADD     VARI,$GDDAT     ;ADD FUDGE FACTOR
1121
1122                                            ;ALLOW THE INTERRUPTS
1123
1124 051544  152777  000100  131072            BISB    #EBIT,@CSR      ;ENABLE INTERRUPT BIT AT CSR
1125 051552  012746  000000                    MOV     #PR0,-(SP)      ;SET PSW TO PRIORITY 0
      051556  012746  051564                    MOV     #64$,-(SP)
      051562  000002                            RTI
      051564  000240                  64$:      NOP
1126
1127 051566  113777  001141  132030            MOVB    $GDDAT+1,@ACNTH ;LOAD HIGH BYTE BUFFER
1128 051574  005037  003476                    CLR     CLK
1129 051600  005037  003646                    CLR     XLOOP           ;INITIALIZE CLOCK CHECK VARIANT
1130 051604  113777  001140  132014            MOVB    $GDDAT,@ACNTL   ;LOAD COUNTER A
1131 051612  113777  001140  132012            MOVB    $GDDAT,@BCNTL   ;LOAD COUNTER B
1132
1133 051620  005237  003646          8$:       INC     XLOOP           ;IF CLOCK SHOULD NOT HAVE INTERRUPTED
1134 051624  001003                            BNE     9$              ;-THEN SEE IF CLOCK COUNTED
1135
1136 051626  104025                            EMT     25
1137 051630  000137  052136                    JMP     CBITCL          ;--SKIP THIS TEST
1138
1139 051634  005737  003476          9$:       TST     CLK             ;IF CLOCK HASN'T COUNTED
1140 051640  001767                            BEQ     8$              ;-THEN SEE IF IT SHOULD HAVE
1141
1142 051642  023727  003476  000074  1$:       CMP     CLK,#60.        ;IF 1 SECOND NOT UP
1143 051650  001374                            BNE     1$              ;-THEN WAIT
1144
1145 051652  012746  000340                    MOV     #PR7,-(SP)      ;SET PSW TO PRIORITY 7
      051656  012746  051664                    MOV     #65$,-(SP)
      051662  000002                            RTI
      051664  000240                  65$:      NOP
1146 051666  142777  000110  130750            BICB    #EBIT.TBIT,@CSR ;DON'T LEAVE THESE BITS HANGING
1147
1148 051674  117737  131724  001143            MOVB    @ACNTH,$BDDAT+1 ;GET CONTENTS OF COUNTER A
1149 051702  117737  131720  001142            MOVB    @ACNTL,$BDDAT
1150 051710  117737  131714  003651            MOVB    @BCNTH,TEMPB+1  ;GET CONTENTS OF COUNTER B
1151 051716  117737  131710  003650            MOVB    @BCNTL,TEMPB
1152
1153 051724  006337  003644                    ASL     VARI            ;DOUBLE THE VARIANCE
```

```
 1154
 1155 051730  023737  003644  001142          CMP     VARI,$BDDAT     ;IF COUNT IS WITHIN 20% OF 2KHZ
 1156 051736  100004                          BPL     2$              ;-THEN SKIP ERROR
 1157
 1158 051740  013737  003626  003462          MOV     ACNTL,TADDR
 1159 051746  104106                          EMT     106
 1160
 1161 051750  023737  003644  003650  2$:     CMP     VARI,TEMPB      ;IF COUNT IS WITHIN 20% OF 2KHZ
 1162 051756  100007                          BPL     3$              ;-THEN SKIP ERROR
 1163
 1164 051760  013737  003632  003462          MOV     BCNTL,TADDR
 1165 051766  013737  003650  001142          MOV     TEMPB,$BDDAT
 1166 051774  104106                          EMT     106
 1167
 1168 051776  000457                  3$:     BR      CBITCL          ;CHECK THAT CBIT CLEARS MODULE.
```

```
1170                                            ;ON INTERRUPT DO 4$
1171
1172 052000  022626                      4$:    CMP     (SP)+,(SP)+     ;RESTORE STACK POINTER
1173 052002  013737  003476  003510             MOV     CLK,TEMP1       ;GET TIME AT INTERRUPT
1174 052010  013737  002660  003502             MOV     BASE,TEMP       ;INITIALIZE TEMP
1175 052016  117737  130624  003502             MOVB    @IAR,TEMP       ;GET INTERRUPTING ADDRESS
1176
1177 052024  023737  003626  003502             CMP     ACNTL,TEMP      ;IF NOT COUNTER A INTERRUPTING
1178 052032  001005                             BNE     5$              ;-THEN SKIP ERROR
1179
1180 052034  013737  003626  003462             MOV     ACNTL,TADDR     ;-ELSE PREPARE FOR ERROR TYPEOUT
1181 052042  104106                             EMT     106
1182 052044  000425                             BR      7$              ;CLEAN UP
1183
1184 052046  023737  003632  003502  5$:        CMP     BCNTL,TEMP      ;IF NOT COUNTER B INTERRUPTING
1185 052054  001005                             BNE     6$              ;-THEN SKIP ERROR
1186
1187 052056  013737  003632  003462             MOV     BCNTL,TADDR     ;-ELSE PREPARE FOR ERROR TYPEOUT
1188 052064  104106                             EMT     106
1189
1190 052066  000414                             BR      7$              ;CLEAN UP
1191
1192 052070  152777  000001  130546  6$:        BISB    #RBIT,@CSR      ;PREPARE TO CLEAR UNEXPECTED INTERRUPT
1193 052076  105777  131400                     TSTB    @TEMP           ;CLEAR THE INTERRUPT
1194 052102  012746  000000                     MOV     #PR0,-(SP)      ;SET PSW TO PRIORITY 0
     052106  012746  052114                     MOV     #66$,-(SP)
     052112  000002                             RTI
     052114  000240                     66$:    NOP
1195 052116  000651                             BR      1$              ;SEE IF TIME IS UP
1196
1197 052120  004737  015410             7$:     JSR     PC,CLRINT       ;CLEAR ALL INTERRUPTS
1198 052124  132777  000200  130512             BITB    #FBIT,@CSR      ;IF INTERRUPT CLEARS
1199 052132  001401                             BEQ     CBITCL          ;-THEN SKIP ERROR
*200 052134  104104                             EMT     104
```

```
1202                                                    ;CHECK THAT CBIT INITIALIZES MODULE
1203
1204 052136  005037  001140          CBITCL: CLR     $GDDAT          ;INITIALIZE EXPECTED DATA.
1205 052142  112777  177777  131454          MOVB    #-1,@ACNTH      ;LOAD THE HIGH BYTE BUFFER.
1206 052150  112777  177777  131450          MOVB    #-1,@ACNTL      ;LOAD COUNTER A WITH ALL 1'S.
1207 052156  112777  177777  131446          MOVB    #-1,@BCNTL      ;LOAD COUNTER B WITH ALL 1'S.
1208 052164  112777  000020  130452          MOVB    #DBIT,@CSR      ;ASSURE CSR IS CLEAR.
1209
1210 052172  004737  014310                  JSR     PC,KLEER        ;INITIALIZE SYSTEM
1211 052176  152777  000020  130440          BISB    #DBIT,@CSR      ;SHUT OFF WORLD
1212
1213 052204  013737  003624  003462          MOV     ACNTH,TADDR     ;INITIALIZE FIRST TEST ADDRESS
1214
1215 052212  117737  131244  001143  1$:     MOVB    @TADDR,$BDDAT+1 ;READ HIGH BYTE OF TEST COUNTER
1216 052220  142737  000001  003462          BICB    #BIT0,TADDR     ;PREPARE TO READ LOW BYTE.
1217 052226  117737  131230  001142          MOVB    @TADDR,$BDDAT   ;READ LOW BYTE OF TEST COUNTER
1218 052234  023737  001140  001142          CMP     $GDDAT,$BDDAT   ;IF DATA IS AS EXPECTED
1219 052242  001401                          BEQ     2$              ;-THEN SKIP ERROR
1220
1221 052244  104075                          EMT     75
1222
1223 052246  023737  003632  003462  2$:     CMP     BCNTL,TADDR     ;IF DONE CHECKING INIT
1224 052254  001404                          BEQ     3$              ;-THEN RETURN FROM SUBROUTINE
1225 052256  013737  003630  003462          MOV     BCNTH,TADDR     ;-ELSE TEST COUNTER B.
1226 052264  000752                          BR      1$
1227 052266  013737  003626  003462  3$:     MOV     ACNTL,TADDR     ;RESTORE ORIGINAL TEST ADDRESS
1228 052274  000207                          RTS     PC              ;RETURN TO MAIN PROGRAM.
```

```
                        .SBTTL  FIELD TESTING DIGITAL MODULE M5016
                        ;M5016 QUAD 8-BIT COUNTER/PRESCALER MODULE TEST
                        ;
                        ;               TEST FLOW
                        ;
                        ;        1.  CLEAR MODULE WITH CBIT
                        ;        2.  CHECK THAT STATUS REGISTER IS CLEAR
                        ;        3.  CHECK THAT ALL COUNTERS ARE CLEAR
                        ;        4.  TEST STATUS REGISTER LOAD AND READ
                        ;        5.  TEST THAT CLEAR ENABLE BIT SELF CLEARS
                        ;        6.  CHECK THAT STATUS REGISTER CLEAR BIT CLEARS
                        ;              EACH COUNTER INDIVIDUALLY
                        ;
                        ;       STORE DIP SWITCH SETTINGS FOR THE FOLLOWING
                        ;       AND PRINT THE SETTINGS IF NOT IN APT MODE
                        ;
                        ;        7.  CHECK THAT COUNTERS COUNT BINARILY UPWARD
                        ;        8.  CHECK THAT COUNTER OVERFLOWS
                        ;        9.  DETERMINE IF LEGAL RADIX
                        ;       10.  DETERMINE IF COUNTERS RESET ON OVERFLOW
                        ;       11.  CHECK THAT OVERFLOW CLEARS
                        ;       12.  CHECK THAT CBIT CLEARS MODULE
                        ;
                        ;LOCAL VARIABLES
                        ;
052276  171000          SR5016: .WORD   171000          ;ADDRESS OF STATUS REGISTER
052300  171001          DR5016: .WORD   171001          ;ADDRESS OF DATA REGISTER
                        ;
052302  000020          OVFS:   .WORD   BIT4            ;OVERFLOW BIT FOR COUNTER 0
052304  000040                  .WORD   BIT5            ;OVERFLOW BIT FOR COUNTER 1
052306  000100                  .WORD   BIT6            ;OVERFLOW BIT FOR COUNTER 2
052310  000200                  .WORD   BIT7            ;OVERFLOW BIT FOR COUNTER 3
                        ;
052312  000000          DSWCH:  .WORD   0               ;COUNTER 0 DIP SWITCH SETTINGS
052314  000000                  .WORD   0               ;COUNTER 1 DIP SWITCH SETTINGS
052316  000000                  .WORD   0               ;COUNTER 2 DIP SWITCH SETTINGS
052320  000000                  .WORD   0               ;COUNTER 3 DIP SWITCH SETTINGS
                        ;
052322  012  015  115  CTASCI: .ASCII <12><15>/M5016/
052331  012  015  040          .ASCIZ  <12><15>/ CTN   1  2  3  4  5  6  7  8  9 10  ADDRESS   /
052413  040  040  000  BLANK2: .ASCIZ  /  /
                        .EVEN
```

```
1273 052416                            M5016:
                                       ;;*****************************************************************
     052416  000004                    TST23:  SCOPE
     052420  012737  000023  001212            MOV     #23,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
1274
1275                                           ;START WITH A KNOWN CONDITION -- CLEAR THE MODULE
1276
1277 052426  004737  014310                    JSR     PC,KLEER        ;CLEAR MODULE.
1278 052432  152777  000020  130204            BISB    #DBIT,@CSR      ;SHUT OFF WORLD
1279 052440  013737  003462  052276            MOV     TADDR,SR5016    ;INITIALIZE STATUS REG ADDRESS
1280 052446  013737  003462  052300            MOV     TADDR,DR5016    ;INITIALIZE DATA REG ADDRESS
1281 052454  052737  000001  052300            BIS     #BIT0,DR5016
1282 052462  104411                            SAVREG                  ;SAVE ALL REGISTERS
1283
1284
1285                                           ;STATUS OF M5016/PRESCALER AT THIS TIME SHOULD BE --
1286
1287                                           ;       STATUS REGISTER = 0     DATA REGISTER = 0
1288
1289                                           ;       -CONTENTS OF COUNTERS ARE-
1290
1291                                           ;       COUNTER 0       CONTAINS 0
1292                                           ;       COUNTER 1       CONTAINS 0
1293                                           ;       COUNTER 2       CONTAINS 0
1294                                           ;       COUNTER 3       CONTAINS 0
1295
1296 052464  005037  001140                    CLR     $GDDAT          ;INITIALIZE EXPECTED DATA
1297
1298                                           ;CHECK THAT STATUS REGISTER IS CLEAR.
1299
1300 052470  117737  177602  001142            MOVB    @SR5016,$BDDAT  ;READ STATUS REGISTER
1301 052476  123737  001140  001142            CMPB    $GDDAT,$BDDAT   ;IF STATUS REGISTER IS CLEAR
1302 052504  001401                            BEQ     CK4CLX          ;-THEN CHECK COUNTERS ARE CLEAR
1303
1304 052506  104075                            EMT     75
```

```
1306
1307                                            ;CHECK THAT ALL COUNTERS ARE CLEAR
1308
1309 052510  005037  003640          CK4CLX: CLR     CNUM            ;INITIALIZE COUNTER NUMBER
1310 052514  004737  015526                  JSR     PC,CNTRC        ;CONTROL C ??????
1311
1312                                    ;DO 1$ WHILE (COUNTER # "CNUM" IS GREATER THAN OR EQUAL TO 3)
1313
1314 052520  022737  000003  003640  1$:     CMP     #3,CNUM         ;IF COUNTER # IS NOT LESS THAN 3
1315 052526  002416                          BLT     CKRWSX          ;-THEN CHECK R/W OF STAT REG
1316 052530  113777  003640  177540          MOVB    CNUM,@SR5016    ;-ELSE LOAD STATUS REG WITH COUNTER #
1317 052536  117737  177536  001142          MOVB    @DR5016,$BDDAT  ;READ DATA REG
1318 052544  123737  001140  001142          CMPB    $GDDAT,$BDDAT   ;IF COUNTER 'CNUM' IS CLEAR
1319 052552  001401                          BEQ     2$              ;-THEN SKIP ERROR
1320
1321 052554  104116                          EMT     116
1322
1323 052556  005237  003640          2$:     INC     CNUM            ;INCREMENT COUNTER NUMBER
1324 052562  000756                          BR      1$              ;SEE IF DONE
1325
1326                                    ;-END 1$
1327
1328                                            ;TEST PRESCALER STATUS REGISTER LOAD & READ
1329
1330 052564  112737  177777  001140  CKRWSX: MOVB    #-1,$GDDAT      ;INIITIALIZE LOAD PATTERN.
1331 052572  005037  001142                  CLR     $BDDAT          ;INITIALIZE BAD DATA STORAGE LOCATION
1332 052576  004737  015526                  JSR     PC,CNTRC        ;CONTROL C ??????
1333
1334                                            ;CHECK R/W BITS BY LOADING COUNTS FROM 1 TO 4
1335                                    ;DO 1$ WHILE (PATTERN "$GDDAT" IS LESS THAN OR EQUAL TO 4)
1336
1337 052602  105237  001140          1$:     INCB    $GDDAT          ;CHANGE LOAD PATTERN.
1338 052606  122737  000004  001140          CMPB    #4,$GDDAT       ;IF DONE CHECKING R/W
1339 052614  100414                          BMI     CKSRCX          ;-THEN CHECK SELF CLEAR ENABLE.
1340
1341                                            ;TEST LOAD & READ
1342
1343 052616  113777  001140  177452          MOVB    $GDDAT,@SR5016  ;LOAD PRESCALER STATUS REGISTER.
1344 052624  117737  177446  001142          MOVB    @SR5016,$BDDAT  ;GET CONTENTS OF STATUS REGISTER
1345 052632  123737  001140  001142          CMPB    $GDDAT,$BDDAT   ;IF STAT REG LOADED CORRECTLY
1346 052640  001760                          BEQ     1$              ;-THEN CHANGE LOAD PATTERN
1347
1348 052642  104111                          EMT     ^11
1349
1350 052644  000756                          BR      1$              ;  CONTINUE READ/WRITE TESTING.
1351
1352                                    ;-END 1$
1353
```

```
                                        ;STATUS OF M5016/PRESCALER AT THIS TIME SHOULD BE --
                                        ;       STATUS REGISTER = 4     DATA REGISTER = 0
                                        ;       -CONTENTS OF COUNTERS ARE-
                                        ;       COUNTER 0       CONTAINS 0
                                        ;       COUNTER 1       CONTAINS 0
                                        ;       COUNTER 2       CONTAINS 0
                                        ;       COUNTER 3       CONTAINS 0
                                        ;TEST THAT CLEAR ENABLE BIT SELF CLEARS
052646  005037  001140          CKSRCX: CLR     $GDDAT          ;PLACE EXPECTED DATA IN $GDDAT
052652  004737  015526                  JSR     PC,CNTRC        ;CONTROL C ???????
052656  105777  177416                  TSTB    @DR5016         ;CLEAR STATUS REG CLEAR BIT
052662  117737  177410  001142          MOVB    @SR5016,$BDDAT  ;READ STATUS REGISTER
052670  023737  001140  001142          CMP     $GDDAT,$BDDAT   ;IF STAT REG CLEARS SELF
052676  001401                          BEQ     CCNTCX          ;-THEN CHECK THAT COUNTERS CLEAR
052700  104112                          EMT     112
                                        ;CHECK THAT STATUS REG CLEARS EACH COUNTER INDIVIDUALLY
052702                          CCNTCX:
052702  012746  000340                  MOV     #PR7,-(SP)      ;SET PSW TO PRIORITY 7
052706  012746  052714                  MOV     #64$,-(SP)
052712  000002                          RTI
052714  000240                  64$:    NOP
052716  005037  003640                  CLR     CNUM            ;INITIALIZE COUNTER NUMBER
052722  142777  000010  127714          BICB    #TBIT,@CSR      ;ASSURE TBIT IS CLEAR
052730  152777  000010  127706          BISB    #TBIT,@CSR      ;TOGGLE THE COUNTERS
052736  142777  000010  127700          BICB    #TBIT,@CSR
```

```
1387                                    ;DO 1$ WHILE (THE NUMBER OF COUNTERS "CNUM" <= 3 )
1388
1389 052744  023727  003640  000003  1$:     CMP     CNUM,#3         ;IF DONE CLEARING ALL COUNTERS
1390 052752  003036                          BGT     BINCNX          ;-THEN CHECK FOR BINARY COUNT
1391
1392 052754  113777  003640  177314          MOVB    CNUM,@SR5016    ;LOAD COUNTER NUMBER TO TEST
1393 052762  152777  000004  177306          BISB    #4,@SR5016      ;CLEAR COUNTER AFTER READ
1394 052770  012737  000001  001140          MOV     #1,$GDDAT       ;INITIALIZE EXPECTED DATA
1395 052776  117737  177276  001142          MOVB    @DR5016,$BDDAT  ;READ COUNTER
1396 053004  023737  001140  001142          CMP     $GDDAT,$BDDAT   ;IF DATA IS AS EXPECTED
1397 053012  001401                          BEQ     2$              ;-THEN SKIP ERROR
1398
1399 053014  104113                          EMT     113
1400
1401 053016  005037  001140          2$:     CLR     $GDDAT          ;INITIALIZE EXPECTED DATA
1402 053022  117737  177252  001142          MOVB    @DR5016,$BDDAT  ;READ COUNTER CNUM
1403 053030  023737  001140  001142          CMP     $GDDAT,$BDDAT   ;IF DATA IS AS EXPECTED
1404 053036  001401                          BEQ     3$              ;-THEN SKIP ERROR
1405
1406 053040  104112                          EMT     112
1407
1408 053042  005237  003640          3$:     INC     CNUM            ;INCREMENT VARIANT
1409 053046  000736                          BR      1$              ;SEE IF DONE
1410
1411                                    ;-END 1$
```

```
1413 053050                            BINCNX:
     053050  012746  C20340                    MOV     #PR7,-(SP)      ;SET PSW TO PRIORITY 7
     053054  012746  053062                    MOV     #64$,-(SP)
     053060  000002                            RTI
     053062  000240                 64$:       NOP
1414 053064  004737  015526                    JSR     PC,CNTRC        ;CONTROL C ??????
1415
1416 053070  012737  000010  003642            MOV     #10,NUM         ;INITIALIZE OVERFLOW POINTER
1417 053076  012737  177777  003640            MOV     #-1,CNUM        ;INITIALIZE COUNTER NUMBER
1418
1419                                           ;INITIALIZE COUNTER DIP SWITCH SETTINGS
1420
1421 053104  005005                            CLR     R5              ;INITIALIZE POINTER REGISTER
1422 053106  005065  052312  .         10$:     CLR     DSWCH(R5)       ;CLEAR CONTENTS OF DIP TABLE
1423 053112  005725                            TST     (R5)+           ;ADVANCE POINTER
1424 053114  022705  000010                    CMP     #10,R5          ;IF NOT FINISHED
1425 053120  001372                            BNE     10$             ;-THEN CONTINUE
1426
1427                                           ;DO 1$ WHILE ( CNUM<=3 )
1428
1429 053122  005237  003640            1$:      INC     CNUM            ;DETERMINE COUNTER NUMBER
1430 053126  023727  003640  000003            CMP     CNUM,#3         ;IF FINISHED BINARY COUNTS
1431 053134  003141                            BGT     CK4INX          ;-THEN CHECK FOR INTERRUPTS
1432
1433 053136  113777  003640  177132            MOVB    CNUM,@SR5016    ;LOAD COUNTER NUMBER TO TEST
1434 053144  152777  000004  177124            BISB    #BIT2,@SR5016   ;PREPARE TO CLEAR COUNTER CNUM
1435 053152  105777  177122                    TSTB    @DR5016         ;CLEAR COUNTER CNUM
1436 053156  152777  000001  127460            BISB    #RBIT,@CSR      ;PREPARE TO CLEAR OVERFLOW
1437 053164  105777  177106                    TSTB    @SR5016         ;CLEAR POSSIBLE OVERFLOW
1438
1439 053170  005037  001140                    CLR     $GDDAT          ;INITIALIZE BINARY COUNT
1440 053174  006337  003642                    ASL     NUM             ;INITIALIZE OVERFLOW POINTER
1441
1442                                           ;-DO 2$ WHILE ( NO OVERFLOW AND CORRECT COUNT )
1443 053200  142777  000010  127436  2$:      BICB    #TBIT,@CSR      ;ASSURE TBIT IS CLEAR
1444 053206  152777  000010  127430            BISB    #TBIT,@CSR      ;CLOCK THE COUNTER
1445 053214  142777  000010  127422            BICB    #TBIT,@CSR      ;DON'T LEAVE BITS HANGING
1446
1447 053222  005237  001140                    INC     $GDDAT          ;TALLY THE COUNTS
1448 053226  133777  003642  177042            BITB    NUM,@SR5016     ;IF COUNTER CNUM OVERFLOWS
1449 053234  001021                            BNE     4$              ;-DETERMINE IF LEGAL RADIX
1450
1451 053236  117737  177036  001142            MOVB    @DR5016,$BDDAT  ;READ COUNTER CNUM
1452 053244  105037  001143                    CLRB    $BDDAT+1        ;CLEAR EXTRANEOUS DATA
1453 053250  023737  001140  001142            CMP     $GDDAT,$BDDAT   ;IF DATA IS AS EXPECTED
1454 053256  001750                            BEQ     2$              ;-THEN DO NEXT COUNT
1455                                   ;--END 2$
1456
1457 053260  023727  001140  000400            CMP     $GDDAT,#400     ;-ELSE IF COUNT REACHES 400
1458 053266  001402                            BEQ     3$              ;--THEN NOTE OVERFLOW ERROR
1459
1460 053270  104113                            EMT     113
1461 053272  000713                            BR      1$              ;---DO NEXT COUNTER
1462
1463 053274                            3$:
     053274  104114                            EMT     114
1464 053276  000711                            BR      1$              ;--DO NEXT COUNTER
```

```
                                        ;DETERMINE IF LEGAL RADIX
                                
053300  012737  000001  003502  4$:   MOV     #BIT0,TEMP      ;INITIALIZE RADIX LOOP VARIANT
                                
                                    ;-DO 40$ WHILE ( NOT ALL LEGAL RADII TESTED )
                                
053306  022737  000400  003502  40$:  CMP     #BIT8,TEMP      ;IF ALL LEGAL RADII NOT TESTED
053314  001002                      BNE     5$              ;-THEN SKIP ERROR
                                
053316  104117                      EMT     117
053320  000700                      BR      1$              ;--DO NEXT COUNTER
                                
053322  006337  003502          5$:  ASL     TEMP            ;COMPUTE LEGAL RADIX TO TEST
053326  023737  003502  001140      CMP     TEMP,$GDDAT     ;IF NOT CORRECT RADIX
053334  001364                      BNE     40$             ;-THEN SEE IF ALL RADII TESTED
                                
                                    ;--END 40$
                                
053336  006237  003502              ASR     TEMP            ;-ELSE COMPUTE CORRECT SWITCH
053342  013705  003640              MOV     CNUM,R5         ;--INITIALIZE OFFSET
053346  006305                      ASL     R5
053350  053765  003502  052312      BIS     TEMP,DSWCH(R5)  ;--SET CORRESPONDING DIP SWITCH
                                
                                    ;CHECK THAT DATA IS RESET ON OVERFLOW
                                
053356  117737  176716  001142      MOVB    @DR5016,$BDDAT  ;--IF COUNTER CNUM NOT ZERO
053364  001003                      BNE     6$              ;---THEN SKIP DIP SWITCH SET
                                
053366  052765  001000  052312      BIS     #BIT9,DSWCH(R5) ;---ELSE COUNTER RESETS ON OVF
                                
                                    ;CHECK THAT OVERFLOW CLEARS
                                
053374  152777  000001  127242  6$:  BISB    #RBIT,@CSR
053402  105777  176670              TSTB    @SR5016         ;CLEAR THE OVERFLOW
053406  005037  001142              CLR     $BDDAT          ;INITIALIZE DATA WORD
053412  117737  176660  001142      MOVB    @SR5016,$BDDAT  ;READ STATUS REGISTER
053420  005037  001140              CLR     $GDDAT          ;INITIALIZE EXPECTED DATA
053424  033737  003642  001142      BIT     NUM,$BDDAT      ;IF DATA IS AS EXPECTED
053432  001633                      BEQ     1$              ;-THEN TEST NEXT COUNTER
                                
053434  104115                      EMT     115
053436  000631                      BR      1$              ;--TEST NEXT COUNTER
                                
                                ;-END 1$
```

```
1512                                            ;CHECK COUNTER FOR INTERRUPTS
1513
1514 053440  012777  054064  127202  CK4INX: MOV     #6$,@VECTO      ;LOAD INTERRUPT VECTOR
1515 053446  012777  000340  127176          MOV     #PR7,@VECTOA    ;LOCK OUT OTHER INTERRUPTS
1516 053454  004737  015526                  JSR     PC,CNTRC        ;CONTROL C ??????
1517 053460  012737  177777  003640          MOV     #-1,CNUM        ;INITIALIZE COUNTER NUMBER
1518 053466  005037  003650                  CLR     TEMPB           ;INITIALIZE INTERRUPT FLAG
1519
1520                                    ;DO 1$ WHILE (CNUM <= 3)
1521
1522 053472  005237  003640          1$:     INC     CNUM            ;ADVANCE COUNTER NUMBER
1523 053476  022737  000003  003640          CMP     #3,CNUM         ;IF NOT FINISHED TESTING INTERRUPTS
1524 053504  002002                          BGE     11$             ;-THEN CONTINUE
1525 053506  000137  054256                  JMP     9$              ;-ELSE CHECK INTERRUPT ERROR
1526
1527 053512  013705  003640          11$:    MOV     CNUM,R5         ;INITIALIZE OFFSET
1528 053516  006305                          ASL     R5
1529 053520  016537  052312  003642          MOV     DSWCH(R5),NUM   ;COMPUTE FULL COUNT
1530 053526  042737  177400  003642          BIC     #177400,NUM     ;-CLEAR EXTRANEOUS BITS
1531 053534  006337  003642                  ASL     NUM             ;-GET RADIX
1532 053540  005337  003642                  DEC     NUM             ;-GET FULL COUNT
1533
1534                                            ;START COUNTER AT 0
1535
1536 053544  113777  003640  176524          MOVB    CNUM,@SR5016    ;LOAD COUNTER NUMBER TO CLEAR
1537 053552  152777  000004  176516          BISB    #4,@SR5016      ;PREPARE TO CLEAR COUNTER
1538 053560  105777  176514                  TSTB    @DR5016         ;CLEAR THE COUNTER
1539 053564  005037  001140                  CLR     $GDDAT          ;INITIALIZE EXPECTED DATA
1540 053570  117737  176504  001142          MOVB    @DR5016,$BDDAT  ;IF THE COUNTER CNUM CLEARS
1541 053576  001401                          BEQ     20$             ;-THEN SKIP THE ERROR
1542
1543 053600  104112                          EMT     112
1544
1545 053602  005037  003502          20$:    CLR     TEMP            ;INITIALIZE LOOP VARIANT
1546                                    ;-DO 2$ WHILE ( TEMP <- NUM )
1547
1548 053606  142777  000010  127030  2$      BICB    #TBIT,@CSR      ;ASSURE TBIT IS CLEAR
1549 053614  152777  000010  127022          BISB    #TBIT,@CSR      ;CLOCK COUNTER
1550 053622  142777  000010  127014          BICB    #TBIT,@CSR      ;DON'T LEAVE BITS HANGING
1551 053630  005237  003502                  INC     TEMP            ;TALLY THE COUNTS
1552 053634  023737  003642  003502          CMP     NUM,TEMP        ;IF NOT FULL COUNT
1553 053642  001361                          BNE     2$              ;-THEN CONTINUE COUNTING
1554                                    ;--END 2$
1555
1556 053644  005037  003502                  CLR     TEMP            ;INITIALIZE LOOP VARIANT
1557
1558                                            ;CLEAR ALL OTHER COUNTERS
1559                                    ;-DO 3$ WHILE (TEMP < 4)
1560
1561 053650  023737  003640  003502  3$:     CMP     CNUM,TEMP       ;IF CNUM = TEMP
1562 053656  001421                          BEQ     4$              ;-THEN ADVANCE VARIANT
1563
1564 053660  022737  000004  003502          CMP     #4,TEMP         ;-ELSE IF 4 <= TEMP
1565 053666  003420                          BLE     5$              ;--THEN FORCE THE INTERRUPT
1566
1567 053670  112777  000004  176400          MOVB    #4,@SR5016      ;--ELSE PREPARE TO CLEAR COUNTER
1568 053676  153777  003502  176372          BISB    TEMP,@SR5016    ;--LOAD COUNTER NUMBER "TEMP"
```

```
1569 053704  105777  176370                    TSTB    @DR5016         ;--CLEAR COUNTER
1570 053710  152777  000001  126726            BISB    #RBIT,@CSR      ;PREPARE TO CLEAR ALL OVERFLOWS
1571 053716  105777  176354                    TSTB    @SR5016         ;CLEAR THE OVERFLOW
1572
1573 053722  005237  003502          4$:       INC     TEMP            ;ADVANCE COUNTER NUMBER
1574 053726  000750                            BR      3$              ;CHECK IF DONE
1575
1576                                 ;--END 3$
1577
1578 053730  004737  015410          5$:       JSR     PC,CLRINT       ;CLEAR ALL POSSIBLE INTERRUPTS
1579
1580                                 ;FORCE AN INTERRUPT
1581
1582 053734  012746  000000                    MOV     #PR0,-(SP)      ;SET PSW TO PRIORITY 0
     053740  012746  053746                    MOV     #64$,-(SP)
     053744  000002                            RTI
     053746  000240          64$:      NOP
1583 053750  142777  000010  126666            BICB    #TBIT,@CSR      ;ASSURE TBIT IS CLEAR
1584 053756  152777  000010  126660            BISB    #TBIT,@CSR      ;CLOCK THE COUNTERS
1585 053764  142777  000010  126652            BICB    #TBIT,@CSR      ;DON'T LEAVE BITS HANGING
1586 053772  152777  000100  126644            BISB    #EBIT,@CSR      ;ENABLE INTERRUPTS
1587
1588 054000  013737  002636  003446            MOV     K,ZLOOP
     054006  005037  003444          66$:      CLR     YLOOP           ;WAIT
     054012  005237  003444          65$:      INC     YLOOP
     054016  023727  003444  000077            CMP     YLOOP,#77
     054024  001372                            BNE     65$
     054026  005337  003446                    DEC     ZLOOP
     054032  001365                            BNE     66$
1589
1590 054034  142777  000100  126602            BICB    #EBIT,@CSR      ;DON'T ALLOW INTERRUPTS
1591 054042  012746  000340                    MOV     #PR7,-(SP)      ;SET PSW TO PRIORITY 7
     054046  012746  054054                    MOV     #67$,-(SP)
     054052  000002                            RTI
     054054  000240          67$:      NOP
1592 054056  000605                            BR      1$              ;TRY NEXT COUNTER
1593
1594                                 ;ON TIMEOUT RESTORE STACK
1595
1596 054060  022626          17$:      CMP     (SP)+,(SP)+     ;RESTORE STACK.
1597 054062  000432                            BR      16$             ;ADD TO INTERRUPT COUNT
1598
1599                                 ;ON INTERRUPT DO 6$
1600
1601 054064  022626          6$:       CMP     (SP)+,(SP)+     ;RESTORE STACK.
1602 054066  013704  000004                    MOV     ERRVEC,R4       ;SAVE ERROR VECTOR.
1603 054072  012737  054060  000004            MOV     #17$,ERRVEC     ;ON TIMEOUT GO TO 17$.
1604 054100  142777  000010  126536            BICB    #TBIT,@CSR      ;ASSURE THAT TBIT IS CLEARED.
1605 054106  005037  003510                    CLR     TEMP1           ;INITIALIZE INTERRUPT TALLY.
1606 054 ''  013737  002660  003502            MOV     BASE,TEMP       ;INITIALIZE TEMP
1607 054120  '52777  000001  126516  7$:       BISB    #RBIT,@CSR      ;PREPARE TO RESET INTERRUPT.
1608
1609                                 ;CHECK IF PRESCALER INTERRUPTED
1610
1611 054126  117737  126514  003502            MOVB    @IAR,TEMP       ;ASSEMBLE INTERRUPTING ADDRESS.
1612 054134  123737  003502  052276            CMPB    TEMP,SR5016     ;IF PRESCALER INTERRUPTS
1613 054142  001417                            BEQ     8$              ;-THEN SET APPROPRIATE DSWTCH
```

```
                                        ;CLEAR THE UNWANTED INTERRUPT
                                        TSTB    @TEMP           ;-ELSE CLEAR INTERRUPT.
                                16$:    INC     TEMP1           ;COUNT NUMBER OF INTERRUPTS.
                                        CMP     #400,TEMP1      ;IF 400 INTERRUPTS NOT COUNTED
                                        BNE     7$              ;-THEN TRY AGAIN.

                                        BICB    #EBIT,@CSR      ;DISABLE INTERRUPTS
                                        MOV     R4,ERRVEC       ;RESTORE ERROR VECTOR
                                        JMP     1$              ;TRY THE NEXT COUNTER

                                8$:     BIS     #BIT8,DSWCH(R5) ;SET APPROPRIATE DSWTCH
                                        MOV     #1,TEMPB        ;SET INTERRUPT OCCURED FLAG

                                        ;CHECK FOR ILLEGAL OVERFLOWS

                                        MOV     OVFS(R5),$GDDAT ;INITIALIZE EXPECTED OVF DATA
                                        MOVB    @SR5016,$BDDAT  ;GET STATUS REGISTER DATA
                                        BIC     #177417,$BDDAT  ;CLEAR EXTRANEOUS DATA
                                        CMP     $GDDAT,$BDDAT   ;IF DATA IS AS EXPECTED
                                        BEQ     18$             ;-THEN SKIP O.ERFLOW ERROR

                                        EMT     114

                                18$:    JMP     7$              ;TRY NEXT COUNTER

                                ;--END 4$

                                9$:     TST     TEMPB           ;INTERRUPTS ?
                                        BNE     10$             ;YES
                                        BITB    #BIT0,$ENV      ;UNDER APT ?
                                        BEQ     50$             ;NO
                                        TST     $PASS           ;FIRST PASS ($PASS = 0) ?
                                        BNE     FHOUSX          ;NO
                                50$:
                                        EMT     103
```

```
1651 054304  005737  001214          10$:    TST     $PASS           ;IF NOT FIRST PASS
1652 054310  001071                          BNE     FHOUSX          ;-THEN SKIP TYPEOUT
1653
1654                                        ;TYPE THE DIP SWITCH SETTINGS IF IN STAND ALONE MODE
1655
1656 054312  005037  003646                 CLR     XLOOP           ;INITIALIZE TYPE LOOP VARIANT
1657
1658
1659 054316  104401  052322          TYPDSW: TYPE    ,CTASCI         ;PRINT A HEADING
1660 054322  013746  003462                  MOV     TADDR,-(SP)     ;;PUSH TADDR ON STACK
1661 054326  104402                          TYPOC                   ;TYPE THE ADDRESS
1662 054330  104401  001203                 TYPE    ,$CRLF          ;TYPE  CARRIAGE RETURN LINE FEED
1663
1664                                        ;DO 5$ WHILE ( DIP SWITCH SETTINGS NOT ALL PRINTED )
1665
1666 054334  013737  003646  003502  5$:     MOV     XLOOP,TEMP      ;SETUP FOR TYPING COUNTER NUM
1667 054342  062737  000060  003502          ADD     #60,TEMP        ;CONVERT TO ASCII NUMBER
1668 054350  104401  052413                 TYPE    ,BLANK2         ;TYPE TWO BLANKS
1669 054354  104401  003502                 TYPE    ,TEMP           ;TYPE COUNTER NUMBER
1670 054360  104401  052413                 TYPE    ,BLANK2         ;TYPE ANOTHER TWO BLANKS
1671 054364  013705  003646                 MOV     XLOOP,R5        ;COMPUTE DSWCH OFFSET
1672 054370  006305                         ASL     R5              ;MULTIPLY COUNTER NUMBER BY 2
1673
1674 054372  012737  000001  003444         MOV     #BIT0,YLOOP     ;INITIALIZE SECOND LOOP VARIANT
1675
1676                                        ;-DO 1$ WHILE (YLOOP < #BIT10 )
1677
1678 054400  104401  052413          1$:     TYPE    ,BLANK2         ;TYPE 2 BLANKS
1679 054404  004737  015526                  JSR     PC,CNTRC        ;CONTROL C ???????
1680 054410  033765  003444  052312         BIT     YLOOP,DSWCH(R5) ;IF THIS SWITCH IS ON
1681 054416  001005                          BNE     2$              ;-THEN TYPE AN ASCII 1
1682
1683 054420  104401  054426                 TYPE    ,10$            ;-ELSE TYPE AN ASCII 0
1684 054424  000404                         BR      3$              ;-SKIP TYPING THE ASCII 1
1685
1686 054426  000060                  10$:    .WORD   '0              ;AN ASCII ZERO
1687 054430  000061                  11$:    .WORD   '1              ;AN ASCII ONE
1688
1689 054432  104401  054430          2$:     TYPE    ,11$            ;TYPE AN ASCII 1
1690
1691 054436  006337  003444          3$:     ASL     YLOOP           ;DETERMINE NEXT SWITCH
1692 054442  032737  002000  003444         BIT     #BIT10,YLOOP    ;IF FINISHED THIS SWITCH PAK
1693 054450  001001                          BNE     4$              ;-THEN TEST THE NEXT
1694 054452  000752                         BR      1$              ;-ELSE DO NEXT SWITCH
1695
1696                                        ;-END 1$
1697
1698 054454  104401  001203          4$:     TYPE    ,$CRLF          ;TYPE CARRIAGE RETURN LINE FEED
1699 054460  005237  003646                 INC     XLOOP           ;INCREMENT LOOP VARIANT
1700 054464  022737  000004  003646         CMP     #4,XLOOP        ;IF NOT FINISHED ALL COUNTERS
1701 054472  001320                          BNE     5$              ;-THEN DO NEXT SWITCH PAK
1702
1703                                        ;-END 5$
```

```
1705
1706 054474  004737  C14310          FHOUSX: JSR     PC,KLEER        ;CLEAR MODULE.
1707 054500  152777  000020  126136          BISB    #DBIT,@CSR      ;SHUT OFF WORLD
1708
1709                                          ;STATUS OF M5016/PRESCALER AT THIS TIME SHOULD BE --
1710
1711                                          ;       STATUS REGISTER = 0     DATA REGISTER = 0
1712
1713                                          ;       -CONTENTS OF COUNTERS ARE-
1714
1715                                          ;       COUNTER 0       CONTAINS 0
1716                                          ;       COUNTER 1       CONTAINS 0
1717                                          ;       COUNTER 2       CONTAINS 0
1718                                          ;       COUNTER 3       CONTAINS 0
1719
1720 054506  005037  001140                  CLR     $GDDAT          ;INITIALIZE EXPECTED DATA
1721
1722                                          ;CHECK THAT STATUS REGISTER IS CLEAR.
1723
1724 054512  117737  175560  001142          MOVB    @SR5016,$BDDAT  ;READ STATUS REGISTER
1725 054520  123737  001140  001142          CMPB    $GDDAT,$BDDAT   ;IF STATUS REGISTER IS CLEAR
1726 054526  001401                          BEQ     CK4CXX          ;-THEN CHECK COUNTERS ARE CLEAR
1727
1728 054530  104075                          EMT     75
```

```
1730
1731                                            ;CHECK THAT ALL COUNTERS ARE CLEAR
1732
1733 054532  005037  003640          CK4CXX: CLR     CNUM            ;INITIALIZE COUNTER NUMBER
1734
1735                                    ;DO 1$ WHILE (COUNTER # ''CNUM'' IS GREATER THAN OR EQUAL TO 3)
1736
1737 054536  022737  000003  003640  1$:     CMP     #3,CNUM         ;IF COUNTER # IS NOT LESS THAN 3
1738 054544  002416                          BLT     MAXRDX          ;-THEN GET MAXIMUM RADIX
1739 054546  113777  003640  175522          MOVB    CNUM,@SR5016    ;-ELSE LOAD STATUS REC WITH COUNTER #
1740 054554  117737  175520  001142          MOVB    @DR5016,$BDDAT  ;READ DATA REG
1741 054562  123737  001140  001142          CMPB    $GDDAT,$BDDAT   ;IF COUNTER 'CNUM' IS CLEAR
1742 054570  001401                          BEQ     2$              ;-THEN SKIP ERROR
1743
1744 054572  104116                          EMT     116
1745
1746 054574  005237  003640          2$:     INC     CNUM            ;INCREMENT COUNTER NUMBER
1747 054600  000756                          BR      1$              ;SEE IF DONE
1748
1749                                    ;-END 1$
1750
```

```
                                        ;GET MAXIMUM RADIX
                                        ;
054602  012737  000400  003502  MAXRDX: MOV     #BIT8,TEMP      ;INITIALIZE LOOP VARIANT
                                        ;
                                        ;TRY EACH SWITCH SETTING UNTIL ONE AGREES STARTING AT 8
                                        ;
                                ;DO 3$ WHILE (TEMP IS NOT MAXIMUM OR NO SWITCHES SET )
054610  006237  003502          3$:     ASR     TEMP            ;GET BIT TO TEST
054614  001463                          BEQ     5$              ;ESCAPE IF NO SWITCHES SET
                                        ;
054616  012737  177777  003640          MOV     #-1,CNUM        ;INITIALIZE COUNTER VARIANT
                                        ;
                                        ;SEE IF ANY COUNTERS ARE SET TO SWITCH SETTING TEMP
                                        ;
                                ;-DO 4$ WHILE   ( COUNTERS NOT ALL DONE )
054624  005237  003640          4$:     INC     CNUM            ;ADVANCE COUNTER NUMBER
054630  022737  000004  003640          CMP     #4,CNUM         ;IF DONE ALL COUNTERS
054636  001764                          BEQ     3$              ;-THEN SEE IF DONE SWITCHES
                                        ;
054640  013705  003640                  MOV     CNUM,R5         ;CONSTRUCT THE OFFSET
054644  006305                          ASL     R5
054646  033765  003502  052312          BIT     TEMP,DSWCH(R5)  ;IF A SWITCH IS DETECTED NOT SET
054654  001763                          BEQ     4$              ;-THEN TRY THE NEXT COUNTER
                                        ;
                                ;--END 4$
                                ;-END 3$
                                        ;
054656  006337  003502                  ASL     TEMP            ;GET MAXIMUM RADIX
                                        ;
054662  006337  003502                  ASL     TEMP            ;GET TWICE MAXIMUM
054666  005337  003502                  DEC     TEMP            ;COMPUTE NUMBER TO MAX BIT SETS
                                        ;
                                        ;CLEAR ALL THE COUNTERS BEFORE MAXING THE COUNTS
                                        ;
054672  005037  003642                  CLR     NUM             ;INITIALIZE LOOP COUNT
                                        ;
                                ;DO 50$ WHILE ( NOT ALL COUNTERS HAVE BEEN CLEARED )
                                        ;
054676  022737  000004  003640  50$:     CMP     #4,CNUM         ;IF ALL COUNTERS CLEARED
054704  001425                          BEQ     52$             ;-THEN FILL COUNTS TO MAXIMUM
                                        ;
054706  113777  003640  175362          MOVB    CNUM,@SR5016    ;LOAD COUNTER NUMBER TO CLEAR
054714  152777  000004  175354          BISB    #4,@SR5016      ;PREPARE TO CLEAR COUNTER
054722  105777  175352                  TSTB    @DR5016         ;CLEAR THE COUNTER
054726  005037  001140                  CLR     $GDDAT          ;INITIALIZE EXPECTED DATA
054732  117737  175342  001142          MOVB    @DR5016,$BDDAT  ;READ COUNTER
054740  023737  001140  001142          CMP     $GDDAT,$BDDAT   ;IF DATA IS AS EXPECTED
054746  001401                          BEQ     51$             ;-THEN SKIP ERROR
                                        ;
054750  104112                          EMT     112
                                        ;
054752  005237  003640          51$:    INC     CNUM            ;PREPARE TO DO NEXT COUNTER
054756  000747                          BR      50$             ;DO NEXT COUNTER
                                        ;
                                ;-END 50$
                                        ;
                                        ;TOGGLE THE COUNTERS TO FULL COUNT TWICE TO GET OVF AND FULL COUNT
```

```
1809
1810 054760  005037  003642          52$:    CLR     NUM             ;INITIALIZE LOOP COUNT
1811
1812                                  ;DO 5$ WHILE (NUM IS LESS THAN TEMP )
1813 054764  023737  003642  003502  5$:     CMP     NUM,TEMP        ;IF DONE
1814 054772  001414                          BEQ     6$              ;-THEN LOAD SR5016
1815
1816 054774  142777  000010  125642          BICB    #TBIT,@CSR      ;ASSURE TBIT IS CLEAR
1817 055002  152777  000010  125634          BISB    #TBIT,@CSR      ;CLOCK THE COUNTERS
1818 055010  142777  000010  125626          BICB    #TBIT,@CSR      ;DON'T LEAVE BITS HANGING
1819
1820 055016  005237  003642                  INC     NUM             ;TALLY THE TOGGLES
1821 055022  000760                          BR      5$              ;CHECK IF DONE
1822
1823                                  ;-END 5$
1824
1825 055024  112777  000007  175244  6$:     MOVB    #7,@SR5016      ;COMPLETE THE FULL HOUSE
1826
1827 055032  004737  014310                  JSR     PC,KLEER        ;CLEAR THE WORLD
1828 055036  152777  000020  125600          BISB    #DBIT,@CSR      ;SHUT OUT WORLD
1829
1830 055044  005037  001140                  CLR     $GDDAT          ;INITIALIZE EXPECTED DATA
1831
1832                                  ;CHECK THAT STATUS REGISTER IS CLEAR
1833
1834 055050  117737  175222  001142          MOVB    @SR5016,$BDDAT  ;READ STATUS REGISTER
1835 055056  123737  001140  001142          CMPB    $GDDAT,$BDDAT   ;IF STATUS REGISTER IS CLEAR
1836 055064  001401                          BEQ     ALLCLX          ;-THEN CHECK COUNTERS ARE CLEAR
1837
1838 055066  104075                          EMT     75
```

```
1840                                        ;CHECK THAT ALL COUNTERS ARE CLEAR
1841
1842 055070  005037  003640          ALLCLX: CLR     CNUM            ;INITIALIZE COUNTER NUMBER
1843
1844                                        ;DO 2$ WHILE (3 IS GREATER THAN OR EQUAL TO COUNTER# ''CNUM'')
1845
1846 055074  022737  000003  003640  2$:     CMP     #3,CNUM         ;IF COUNTER # IS NOT LESS THAN 3
1847 055102  002416                          BLT     4$              ;-THEN CLEAN UP AND EXIT
1848 055104  113777  003640  175164          MOVB    CNUM,@SR5016    ;-ELSE LOAD STATUS REG WITH COUNTER #
1849 055112  117737  175162  001142          MOVB    @DR5016,$BDDAT  ;READ DATA REG
1850 055120  123737  001140  001142          CMPB    $GDDAT,$BDDAT   ;IF COUNTER 'CNUM' IS CLEAR
1851 055126  001401                          BEQ     3$              ;-THEN SKIP ERROR
1852
1853 055130  104116                          EMT     116
1854
1855 055132  005237  003640          3$:     INC     CNUM            ;INCREMENT COUNTER NUMBER
1856 055136  000756                          BR      2$              ;SEE IF DONE
1857
1858                                        ;-END 2$
1859
1860 055140  104412                  4$:     RESREG                  ;RESTORE THE REGISTERS
1861 055142  000207                          RTS     PC
```

```
                                        .SBTTL  MONITOR TO SELECT DAC TESTS
055144  005737  002664          DCAMON: TST     AFLAG                   ;UNDER AUTO MODE ?
055150  001416                          BEQ     DACSTR                  ;NO,BRANCH TO MANUAL
055152  012777  015612  125544          MOV     #KBINT,@KBVEC
055160  152777  000100  123772          BISB    #BIT6,@$TKS     ;ENABLE KB INTERRUPT
055166  004737  055450                  JSR     PC,GTADRS               ;SET UP ADDRESSES FOR DAC
055172  004737  055544                  JSR     PC,TSTADR               ;CHECK PRESENCE OF ADDRESSES
055176  000207                          RTS     PC
055200  004737  055512                  JSR     PC,DDBIT                ;GO DO D-BIT TEST
055204  000207                          RTS     PC                      ;RETURN TO AUTO MONITOR

055206  012737  055206  003652  DACSTR: MOV     #DACSTR,RETURN          ;SET RETURN FOR CONTROL A
055214  004737  055450          DACMON: JSR     PC,GTADRS               ;SET UP ADDRESSES
055220  004737  055544                  JSR     PC,TSTADR               ;CHECK RESPONSE OF ADDRESSES
055224  000207                  DCOUT:  RTS     PC                      ;EXIT TO MONITOR ON ERROR
055226  004737  060112                  JSR     PC,DAGTST               ;CHECK THAT GENERIC CODE OF
055232  104401  036516                  TYPE    ,DANGC                  ;THE DAC IS 261 IF IT IS GO TEST
055236  000207                          RTS     PC                      ;ELSE RETURN TO MAIN MON$PASS
055240  104401  036652          DCMN:   TYPE    ,DANSEL                 ;SELECT TEST A, T,  D,
055244  004737  014310                  JSR     PC,KLEER
055250  104406                          RDCHR                           ;-?-FOR MEANING OF SELECTION
055252  012637  003566                  MOV     (SP)+,ANSW      ;;POP STACK INTO ANSW
055256  104401  003566                  TYPE    ,ANSW
055262  104401  001203                  TYPE    ,$CRLF
055266  122737  000110  003566          CMPB    #'H,ANSW                ;IF QUESTION THE ELIGHTEN
055274  001003                          BNE     1$                      ;USER BY TYPING TEXT  OF TEST
055276  104401  036754                  TYPE    ,TEXT                   ;SELECTION DESCRIPTORS
055302  000756                          BR      DCMN                    ;NOW SELEC TEST
055304  122737  000101  003566  1$:     CMPB    #'A,ANSW                ;SELECT DAC CALIBRATION TEST
055312  001015                          BNE     2$                      ;IF NOT THIS TEST THEN CHECK NEXT
055314  104401  030660                  TYPE    ,M30                    ;DISCONNECT CUSTOMER WIRES
055320  104406                  5$:     RDCHR
055322  012637  003566                  MOV     (SP)+,ANSW      ;;POP STACK INTO ANSW
055326  122737  000015  003566          CMPB    #15,ANSW
055334  001371                          BNE     5$
055336  004737  056132                  JSR     PC,DACTST               ;ELSE DO THIS TEST
055342  000137  055240                  JMP     DCMN                    ;
055346  122737  000124  003566  2$:     CMPB    #'T,ANSW                ;SELECT TBIT (STATUS BITS) TEST
055354  001016                          BNE     3$                      ;IF NOT THIS TEST THEN CHECK NEXT
055356  104401  030660                  TYPE    ,M30
055362  104406                  6$:     RDCHR
055364  012637  003566                  MOV     (SP)+,ANSW      ;;POP STACK INTO ANSW
055370  122737  000015  003566          CMPB    #15,ANSW
055376  001371                          BNE     6$
055400  004737  017306          14$:    JSR     PC,SWLOOP
055404  056672                          INTCOM                          ;DO INTCOM TEST
055406  000137  055240                  JMP     DCMN
055412  122737  000104  003566  3$:     CMPB    #'D,ANSW                ;SELECT DBIT TEST
055420  001005                          BNE     4$                      ;IF NOT THIS TEST THEN CHECK NEXT
055422  004737  017306          9$:     JSR     PC,SWLOOP               ;LOOP HERE
055426  055512                          DDBIT                           ;DO DDBIT TEST
055430  000137  055240                  JMP     DCMN
055434  004737  015526          4$:     JSR     PC,CNTRC                ;CHECK FOR EXIT BY CONTROL ''C''
055440  104401  030374                  TYPE    ,M20                    ;TYPE ? IF NO TST SELECTION FOUND
055444  000137  055240                  JMP     DCMN                    ;GO TO START OF DAC MONITOR
```

```
1919
1920 055450  005000                    GTADRS: CLR     R0                      ;CLEAR ADDRESS POINTER
1921 055452  005001                            CLR     R1                      ;CLEAR ADDRESS COUNTER
1922 055454  013746  003462                    MOV     TADDR,-(SP)     ;;PUSH TADDR ON STACK
1923 055460  013760  003462  002726    1$:     MOV     TADDR,LDATA0(R0)        ;WRITE ADDRESSES
1924 055466  105237  003462                    INCB    TADDR                   ;UPDATE TO NEXT REGISTER
1925 055472  005720                            TST     (R0)+                   ;UPDATE TO NEXT LOCATION
1926 055474  005201                            INC     R1                      ;UPDATE REGISTER COUNTER
1927 055476  022701  000010                    CMP     #10,R1                  ;CHECK FOR EIGHT REGISTERS
1928 055502  001366                            BNE     1$                      ;IF NOT LAST GET NEXT
1929 055504  012637  003462                    MOV     (SP)+,TADDR     ;;POP STACK INTO TADDR
1930 055510  000207                            RTS     PC                      ;AND RETURN
1931
1932                                    .SBTTL  D-BIT   TEST
1933
1934 055512  004737  014310            DDBIT:  JSR     PC,KLEER
1935 055516  152777  000020  125120            BISB    #DBIT,@CSR              ;SET THE D-BIT
1936 055524  004737  055636                    JSR     PC,TSTRGS               ;GO CHECK REGISTERS
1937 055530  142777  000020  125106            BICB    #DBIT,@CSR              ;CLEAR D-BIT WHEN DONE
1938 055536  004737  014310                    JSR     PC,KLEER                ;AND CLEAR THE WORLD
1939 055542  000207                            RTS     PC                      ;AND RETURN
1940
```

```
1942                                            .SBTTL  REGISTER VERIFICATION
1943 055544  013703  000004          TSTADR: MOV      ERRVEC,R3              ;SAVE PRESENT LOC 4
1944 055550  012737  055602  000004          MOV     #2$,ERRVEC              ;SET UP TIME OUT VECTOR
1945 055556  004737  014310                  JSR     PC,KLEER                ;INIT THE SYSTEM
1946 055562  005000                          CLR     R0                      ;CLR ADDRESS LOC
1947 055564  005001                          CLR     R1                      ;CLR ADDRESS COUNTER
1948 055566  013700  003462                  MOV     TADDR, R0               ;GET FIRST REG ADDRESS
1949 055572  105720                  1$:     TSTB    (R0)+                   ;IS IT THERE AND CLEAR
1950 055574  001407                          BEQ     3$                      ;IF IT IS GET NEXT ADDRESS
1951 055576  104027                          EMT     27
1952 055600  000405                          BR      3$                      ;AND THEN GET NEXT ADDRESS
1953 055602                          2$:
     055602  104026                          EMT     26
1954 055604  004737  015526                  JSR     PC,CNTRC                ;CHECK FOR CONTROL "C" EXIT
1955 055610  022626                          CMP     (SP)+,(SP)+             ;ADJUST THE STACK AFTER TIME OUT
1956 055612  000406                          BR      4$                      ;RETURN MAIN MONITOR
1957 055614  005201                  3$:     INC     R1                      ;UPDATE ADDRESS COUNTER
1958 055616  022701  000010                  CMP     #10,R1                  ;CHECK FOR LAST REG
1959 055622  001363                          BNE     1$                      ;IF IT IS THEN
1960 055624  062716  000002                  ADD     #2,(SP)                 ;CONTINUE WITH TEST
1961 055630  010337  000004          4$:     MOV     R3,ERRVEC               ;RESTORE LOC4
1962 055634  000207                          RTS     PC                      ;RETURN
1963
1964 055636                          TSTRGS:
                                     ;;*****************************************************************
     055636  000004                  TST24:  SCOPE
     055640  012737  000024  001212          MOV     #24,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
1965 055646  005037  003564                  CLR      ROTPAT                 ;CLEAR THE PATTRN SPOT
1966 055652  005000                          CLR     R0                      ;CLEAR THE INDEX REGISTER
1967 055654  012737  000007  003540          MOV     #7,KOUNT                ;DO SEVEN FOR NOW
1968 055662  004737  060060          1$:     JSR     PC,ROTDAT               ;NOW GO ROTATE A BIT
1969 055666  113770  003564  002726  2$:     MOVB    ROTPAT,@LDATA0(R0)      ;NOW GO WRITE IT
1970 055674  004737  056112                  JSR     PC,UPREG                ;NOW GET THE NEXT REGISTER
1971 055700  000772                          BR      2$                      ;AND WRITE IN (4 IN ALL)
1972 055702  013737  003564  001140          MOV     ROTPAT,$GDDAT           ;SAVE PATTERN FOR CHECKING
1973 055710  117037  002726  001142  3$:     MOVB    @LDATA0(R0),$BDDAT      ;READ THE RSGISTER
1974 055716  016037  002726  003502          MOV     LDATA0(R0),TEMP         ;AND STORE ITS ADDRESS
1975 055724  123737  001140  001142          CMPB    $GDDAT,$BDDAT           ;SEE IF IT IS GOOD
1976 055732  001401                          BEQ     4$                      ;IF GOOD CONTINUE
1977 055734  104030                          EMT     30
1978 055736  004737  015526          4$:     JSR     PC,CNTRC                ;CHECK FOR CONTROL "C" EXIT
1979 055742  004737  056112                  JSR     PC,UPREG                ;READ NEXT REGISTER
1980 055746  000760                          BR      3$                      ;UNTIL 4 DONE
1981 055750  005337  003540                  DEC     KOUNT                   ;AND UNTIL7 ROTATES
1982 055754  100342                          BPL     1$                      ;NOW GO BACK AND DO IT
1983 055756  005037  003564                  CLR     ROTPAT                  ;CLEAR PATTERN SPOT
1984 055762  012737  000001  003540          MOV     #1,KOUNT                ;DO ONLY 2 THIS TIME
1985 055770  004737  060060          5$:     JSR     PC,ROTDAT               ;GO ROTATE A BIT NOW
1986 055774  113770  003564  002730  6$:     MOVB    ROTPAT,@HDATA0(R0)      ;GO WRITE IT NOW
1987 056002  004737  056112                  JSR     PC,UPREG                ;GET THE NEXT REGISTER
1988 056006  000772                          BR      6$                      ;AND WRITE IN IT (4 IN ALL)
1989 056010  013737  003564  001140          MOV     ROTPAT,$GDDAT           ;SAVE PATTERN FOR CHECKING
1990 056016  112770  000000  002726  7$:     MOVB    #0,@LDATA0(R0)          ;NOW  LET IT TRANSFER
1991 056024  117037  002730  001142          MOVB    @HDATA0(R0),$BDDAT      ;READ THE REGISTER
1992 056032  016037  002730  003502          MOV     HDATA0(R0),TEMP         ;AND STORE ITS ADDRESS
1993 056040  123737  001140  001142          CMPB    $GDDAT,$BDDAT           ;SEE IF IT IS GOOD
1994 056046  001401                          BEQ     8$                      ;IF GOOD CONTINUE
```

REGISTER VERIFICATION

```
1995 056050  104030                        EMT     30
1996 056052  004737  015526          8$:   JSR     PC,CNTRC                ;CHECK FOR CONTROL"C" EXIT
1997 056056  004737  056112                JSR     PC,UPREG                ;READ NEXT REGISTER
1998 056062  000755                        BR      7$                      ;UNTIL 4 ARE DONE
1999 056064  005337  003540                DEC     KOUNT                   ;AND FOR 2 ROTATES
2000 056070  100337                        BPL     5$                      ;GO BACK AND DO IT
2001 056072  005737  002664                TST     AFLAG
2002 056076  001004                        BNE     9$
2003 056100  032737  060000  000176        BIT     #BIT13!BIT14,SWREG
2004 056106  001000                        BNE     9$
2005 056110  000207                  9$:   RTS     PC
2006
2007
2008
2009
2010 056112  022020                  UPREG:CMP     (R0)+,(R0)+                     ;UPDATE INDEX
2011 056114  022700  000020                CMP     #20,R0                  ;UNTIL 4 REGISTERS
2012 056120  001003                        BNE     1$                      ;ARE DONE
2013 056122  005000                        CLR     R0                      ;CLR THE INDEX REG
2014 056124  062716  000002                ADD     #2,(SP)                 ;UPDATE THE STACK
2015 056130  000207                  1$:   RTS     PC                      ;CONTINUE TEST
2016
2017
2018
```

```
                                .SBTTL  DAC CALIBRATION TEST
                        DACTST: JSR     PC,KLEER
                        DATST:  JSR     PC,DECCHN               ;GET CHANEL # AND DECODE IT
                                TYPE    ,CTXTV                  ;DO REFER VOLTAGE CAL Y-N
                                RDCHR                           ;GET Y-N ANSW
                                MOV     (SP)+,ANSW      ;;POP STACK INTO ANSW
                                TYPE    ,ANSW                   ;ECHO IT
                                CMPB    #'N,ANSW                ;IF IT IS A NO
                                BEQ     5$                      ;GO TO CHAN AND
                                CMPB    #'Y,ANSW                ;IS IT A YES?
                                BEQ     20$
                                TYPE    ,M20                    ;?
                                BR      DATST
                        20$:    JSR     PC,CNTRC
                                JSR     PC,DCTST                ;PASS REFER VOLT CAL TST
                        5$:     JSR     PC,CNTRC
                                TYPE    ,M81
                                BR      DTST
                        DCTST:  TYPE    ,CM1                    ;ADJ R135 @ PIN 6 OF E82 FOR 10.24
                                JSR     PC,CRTST                ;USING SWITCHES AT E79 IF NEEDED
                                .                               ;WHEN DONE HIT <CR>
                                TYPE    ,CM2                    ;ADJ R134 @ PIN 60 OF E81 FOR 5.12
                                JSR     PC,CRTST                ;WHEN DONE HIT <CR>
                                RTS     PC
                        DTST:   MOV     XCHAN,R0                ;GET REG INDEX TO CHAN
                                TYPE    ,CM3                    ;ADJ VOLTAGE OFFSET ON CHAN 1 TO 0.000
                        21$:    JSR     PC,CRTST                ;WAIT FOR <CR>
                        1$:     MOVB    #3,@HDATA0 (R0)         ;LOAD ALL ONES INTO
                                MOVB    #377,@LDATA0 (R0)       ;CHAN SELECTED
                                TYPE    ,CM4                    ;CHECK VOLTAGE LEVEL ON CHANNEL SELECTED
                        22$:    JSR     PC,CRTST                ;FOR 10.23V + OR - 10 MIN
                                TYPE    ,M82
                        41$:    RDCHR
                                MOV     (SP)+,ANSW      ;;POP STACK INTO ANSW
                                TYPE    ,ANSW
                                JSR     PC,CNTRC
                                CMPB    #'N,ANSW
                                BNE     40$
                                JMP     7$
                        40$:    CMPB    #'Y,ANSW
                                BNE     41$
                                TYPE    ,CM12                   ;ASK IF 4 TO 20 MA RANGE WANTED
                        23$:    RDCHR                           ;READ ANSWER
                                MOV     (SP)+,ANSW      ;;POP STACK INTO ANSW
                                TYPE    ,ANSW                   ;ECHO ANSWER
                                CMPB    #'Y,ANSW                ;IF YES GO TO 4 TO 20 MA TEST
                                BEQ     5$                      ;ELSE DO 0 TO 20 MA TEST
                                CMPB    #'N,ANSW
                                BEQ     4$
                                JSR     PC,CNTRC
                                BR      23$
                        4$:     MOVB    #3,@HDATA0(R0)          ;RELOAD UPPER BYTE OF CHANEL
                                MOVB    #377,@LDATA0(R0)        ;SELECTED AND LET IT GO (CONV)
```

```
2077 056440  104401  037624                TYPE    ,CM9                    ;ADJ CURRENT GAIN TO 10.000V
2078 056444  004737  015716          25$:  JSR     PC,CRTST                ;OR 20MA
2079 056450  112770  000000  002730        MOVB    #0,@HDATA0(R0)          ;CLEAR UPPER BYTE OF CHANSELECTED
2080 056456  112770  000001  002726        MOVB    #1,@LDATA0(R0)          ;SET 1 TO LOWER BYTE OF
2081                                                                        ;CHANEL SELECTED
2082 056464  104401  037452                TYPE    ,CM7                    ;ADJ CURRENT OFFSET TO 9.8 MV
2083 056470  004737  015716          26$:  JSR     PC,CRTST                ;OR 19.5 MICRO-AMPS
2084
2085 056474  104401  040132                TYPE    ,CM11                   ;ASK TO REVERIFY
2086 056500  104406                27$:  RDCHR
2087 056502  012637  003566                MOV     (SP)+,ANSW      ;;POP STACK INTO ANSW
2088 056506  104401  003566                TYPE    ,ANSW                   ;ECHO ANSWER
2089 056512  004737  015526                JSR     PC,CNTRC
2090 056516  122737  000116  003566        CMPB    #'N,ANSW
2091 056524  001737                      BEQ     4$
2092 056526  122737  000131  003566        CMPB    #'Y,ANSW
2093 056534  001451                      BEQ     7$
2094 056536  000760                      BR      27$                     ;EXIT AFTER ALINEMENT
2095 056540  112770  000003  002730  5$:   MOVB    #3,@HDATA0(R0)          ;LOAD ALL ONES
2096 056546  112770  000377  002726        MOVB    #377,@LDATA0(R0)        ;INTO SELECTED CHANEL
2097 056554  104401  037624                TYPE    ,CM9                    ;ADJ CURRENT GAIN TO
2098 056560  104406                31$:  RDCHR                           ;10.000 V OR 20 MA
2099 056562  012637  003566                MOV     (SP)+,ANSW      ;;POP STACK INTO ANSW
2100 056566  004737  015526                JSR     PC,CNTRC
2101 056572  122737  000015  003566        CMPB    #15,ANSW
2102 056600  001367                      BNE     31$
2103 056602  004737  014310                JSR     PC,KLEER                ;CLEAR ALL REGS TO ZERO
2104 056606  104401  037766                TYPE    ,CM10                   ;ADJ CURRENT OFFSET
2105 056612  004737  015716          32$:  JSR     PC,CRTST                ;FOR 2.00V OR 4.00MA
2106 056616  104401  040132                TYPE    ,CM11
2107 056622  104406                33$:  RDCHR
2108 056624  012637  003566                MOV     (SP)+,ANSW      ;;POP STACK INTO ANSW
2109 056630  104401  003566                TYPE    ,ANSW                   ;ECHO ANSWER
2110 056634  004737  015526                JSR     PC,CNTRC
2111 056640  122737  000116  003566        CMPB    #'N,ANSW
2112 056646  001734                      BEQ     5$
2113 056650  122737  000131  003566        CMPB    #'Y,ANSW
2114 056656  001361                      BNE     33$
2115 056660  104401  036441          7$:   TYPE    ,M83                    ;END OF CALIBRATION
2116 056664  004737  014310                JSR     PC,KLEER
2117 056670  000207                      RTS     PC
```

```
                                        .SBTTL  INTERNAL COMPARATOR CHECK

                                        INTCOM:
                                        ;;*****************************************************************
                                        ;;TST25:  SCOPE
056672  000004                          
056674  012737  000025  001212          MOV     #25,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
056702  152777  000010  123734          BISB    #TBIT,@CSR              ;SET THE TBIT INTO THE CSR
056710  112777  000000  124010          MOVB    #0,@LDATA0              ;START CH 0 CONVERSIONS
056716  112777  000100  124006          MOVB    #100,@LDATA1            ;START CH 1 CONVERSIONS
056724  112777  000200  124004          MOVB    #200,@LDATA2            ;START CH 2 CONVERSIONS
056732  112777  000300  124002          MOVB    #300,@LDATA3            ;START CH 3 CONVERSION
056740  004737  014416                  JSR     PC,DALLY                ;WAIT FOR ALL CHANELS TO FINISH

056744  013700  002730                  MOV     HDATA0,R0               ;GET FIRST REG TO CHECK
056750  111001                  4$:     MOVB    (R0),R1                 ;GET HIGH DATA BYTE
056752  042701  177703                  BIC     #177703,R1              ;CLEAR UNWANTED BITS
056756  022701  000074                  CMP     #74,R1                  ;IF ALL BIT ARE SET
056762  001417                          BEQ       5$                    ;CONTINUE WITH TEST

056764  010137  003502                  MOV     R1,TEMP
056770  006237  003502                  ASR     TEMP
056774  006237  003502                  ASR     TEMP
057000  012737  000017  003600          MOV     #17,TEMP3
057006  104031                          EMT     31
057010  004737  014310                  JSR     PC,KLEER
057014  000207                          RTS     PC
057016  004737  015526                  JSR     PC,CNTRC                ;CHECK FOR CONTROL "C" EXIT
057022  062700  000002          5$:     ADD     #2,R0                   ;UPDATE TO NEXT HIGH DATA REG
057026  023700  002744                  CMP     HDATA3,R0               ;AND CHECK TO SEE IF ALL STATUS
057032  003746                          BLE     4$                      ;BITS ARE SET WHEN DONE
057034  005000                          CLR     R0                      ;CLEAR R0 AND SYSTEM
057036  004737  014310                  JSR     PC,KLEER                ;RIGHT HERE
057042  152777  000010  123574          BISB    #TBIT,@CSR              ;RELOAD THE TBIT
057050  112777  000000  123650          MOVB    #0,@LDATA0              ;CHECK ALL STATUS
057056  112777  000004  123646          MOVB    #4,@LDATA1              ;BITS AT THE 3-LSB
057064  112777  000010  123644          MOVB    #10,@LDATA2             ;DIFFERANCE MARGIN
057072  112777  000014  123642          MOVB    #14,@LDATA3
057100  004737  014416                  JSR     PC,DALLY                ;WAIT FOR IT TO HAPPEN
057104  117702  123620                  MOVB    @HDATA0,R2              ;GET THE STATUS INFO
057110  042702  177703                  BIC     #177703,R2              ;CLEAR UNWANTED BITS
057114  022702  000074                  CMP     #74,    R2              ;ALL FOUR STATUS BITS
057120  001415                          BEQ     6$                      ;SHOULD BE SET ELSE ERROR
057122  010237  003502                  MOV     R2,TEMP
057126  006237  003502                  ASR     TEMP
057132  006237  003502                  ASR     TEMP
057136  012737  000017  003600          MOV     #17,TEMP3
057144  104031                          EMT     31
057146  004737  014310                  JSR     PC,KLEER
057152  000207                          RTS     PC
057154  004737  015526          6$:     JSR     PC,CNTRC                ;CHECK FOR CONTROL "C" EXIT
057160  004737  014310                  JSR     PC,KLEER                ;CLEAR THE SYSTEM
057164  152777  000010  123452          BISB    #TBIT,  @CSR            ;RELOAD THE TBIT
057172  112777  000014  123526          MOVB    #14,@LDATA0             ;CHECK ALL STATUS BIT AT
057200  112777  000010  123524          MOVB    #10,@LDATA1             ;THE DESCENDING 3-LSB
057206  112777  000004  123522          MOVB    #4,@LDATA2              ;DIFFERANCE MARGIN
057214  112777  000000  123520          MOVB    #0,@LDATA3
057222  004737  014416                  JSR     PC,DALLY                ;WAIT FOR IT TO HAPPEN
```

```
2173 057226  117702  123476              MOVB    @HDATA0,R2              ;GET THE STATUS INFO
2174 057232  042702  177703              BIC     #177703,R2              ;CLEAR UNWANTED BITS
2175 057236  005702                      TST     R2                      ;ALL FOUR STATUS BITS
2176 057240  001414                      BEQ     7$                      ;SHOULD BE ZERO ELSE ERROR
2177 057242  010237  003502              MOV     R2,TEMP
2178 057246  006237  003502              ASR     TEMP
2179 057252  006237  003502              ASR     TEMP
2180 057256  005037  003600              CLR     TEMP3
2181 057262  104032                      EMT     32
2182 057264  004737  014310              JSR     PC,KLEER
2183 057270  000207                      RTS     PC
2184 057272  004737  015526        7$:   JSR     PC,CNTRC                ;CHECK FOR CONTROL "C" EXIT
2185
2186 057276  012737  000000  003542      MOV     #0,ACOUNT       ;SET UP 0&10 INPUT TO
2187 057304  012737  000020  003544      MOV     #20,BCOUNT      ;CHANELS 0 TO 3 I.E. 0,20,0,20
2188 057312  012737  000064  003516      MOV     #64,CH0         ;GET GOOD STATUS OUTPUT
2189 057320  012737  000040  003520      MOV     #40,CH1         ;GET ERROR
2190 057326  012737  000070  003522      MOV     #70,CH2         ;STATUS OUTPUTS
2191 057334  012737  000054  003524      MOV     #54,CH3         ;TOTAL OF FOUR
2192 057342  012737  000004  003526      MOV     #4,CH4          ;TO BE LOOKED AT
2193 057350  152777  000010  123266      BISB    #TBIT,@CSR      ;SET THE TBIT IN THE CSR
2194 057356  004737  057622        8$:   JSR     PC,INTLD        ;LOAD INPUT REGS
2195 057362  117737  123342  003502      MOVB    @HDATA0,TEMP    ;GET STATUS BITS
2196 057370  042737  177703  003502      BIC     #177703,TEMP    ;CLEAR UNWANTED BITS
2197 057376  013737  003502  003600      MOV     TEMP,TEMP3
2198 057404  006237  003600              ASR     TEMP3
2199 057410  006237  003600              ASR     TEMP3
2200 057414  004737  057710              JSR     PC,CKICT        ;GO CHECK STATUS BITS
2201 057420  104033                      EMT     33
2202 057422  004737  015526              JSR     PC,CNTRC        ;CHECK OFR CONTROL "C" EXIT
2203 057426  004737  060024              JSR     PC,GOUP         ;GET NEXT INPUT VALUES
2204 057432  022737  001750  003542      CMP     #1750,ACOUNT    ;THIS IS THE LAST INPUT
2205 057440  003746                      BLE     8$              ;IF NOT GET NEXT INPUT
2206
2207 057442  012737  001770  003542      MOV     #1770,ACOUNT    ;SET UP UPPER LIMIT TO DO
2208 057450  012737  001760  003544      MOV     #1760,BCOUNT    ;STATUS BIT COMPLIMENT DOWN
2209 057456  012737  000010  003516      MOV     #10,CH0         ;GET GOOD STATUS OUTPUT
2210 057464  012737  000034  003520      MOV     #34,CH1         ;GET STATUS
2211 057472  012737  000004  003522      MOV     #4,CH2          ;ERROR OUTPUTS
2212 057500  012737  000020  003524      MOV     #20,CH3         ;TOTAL OF FOUR
2213 057506  012737  000070  003526      MOV     #70,CH4         ;TO BE LOOKED AT
2214 057514  004737  057622        9$:   JSR     PC,INTLD        ;LOAD INPUT REGS
2215 057520  117737  123204  003502      MOVB    @HDATA0,TEMP    ;GET STATUS BITS
2216 057526  042737  177703  003502      BIC     #177703,TEMP    ;CLEAR UNWANTED BITS
2217 057534  013737  003502  003600      MOV     TEMP,TEMP3
2218 057542  006237  003600              ASR     TEMP3
2219 057546  006237  003600              ASR     TEMP3
2220 057552  004737  057710              JSR     PC,CKICT        ;GO VERIFY DATA
2221 057556  104033                      EMT     33
2222 057560  004737  015526              JSR     PC,CNTRC        ;CHECK CONTROL "C" EXIT
2223 057564  004737  060042              JSR     PC,GODN         ;GO COUNT DOWN
2224 057570  005737  003544              TST     BCOUNT          ;CHECK FOR LAST INPUT
2225 057574  001347                      BNE     9$              ;IF NOT LAST GET NEXT INPUT
2226 057576  142777  000010  123040      BICB    #TBIT,@CSR      ;CLEAR THE TBIT AND
2227 057604  004737  014310              JSR     PC,KLEER                ;CLEAR THE EMPIRE
2228 057610  032777  020000  121336      BIT     #BIT13,@SWR             ;CHECK INHIBIT PRINTOUT
2229 057616  001000                      BNE     10$
```

INTERNAL COMPARATOR CHECK

```
2230 057620  000207                    10$:    RTS     PC
2231
2232
2233 057622  113777  003543  123100    INTLD:  MOVB    ACOUNT+1,@HDATA0        ;LOAD UPPER CH0 REG
2234 057630  113777  003542  123070            MOVB    ACOUNT,@LDATA0  ;LOAD LOWER CH0 REG
2235 057636  113777  003545  123070            MOVB    BCOUNT+1,@HDATA1        ;LOAD UPPER CH1 REG
2236 057644  113777  003544  123060            MOVB    BCOUNT,@LDATA1  ;LOAD LOWER CH1 REG
2237 057652  113777  003543  123060            MOVB    ACOUNT+1,@HDATA2        ;LOAD UPPER CH2 REG
2238 057660  113777  003542  123050            MOVB    ACOUNT,@LDATA2  ;LOAD LOWER CH2 REG
2239 057666  113777  003545  123050            MOVB    BCOUNT+1,@HDATA3        ;LOAD UPPER CH3 REG
2240 057674  113777  003544  123040            MOVB    BCOUNT,@LDATA3  ;LOAD LOWER CH3 REG
2241 057702  004737  014416                    JSR     PC,DALLY        ;WAIT FOR CONVERSION
2242 057706  000207                            RTS     PC                      ;RETURN
2243
2244 057710  023737  003516  003502    CKICT:  CMP     CH0,TEMP                ;CHECK GOOD DATA
2245 057716  001003                            BNE     1$              ;IF BAD CHECK WHIC CHANEL
2246 057720  062716  000002                    ADD     #2,(SP)         ;ELSE BY PASS ERROR
2247 057724  000207                            RTS     PC                      ;RETURN
2248 057726  005037  003514            1$:     CLR     CHNUM           ;GET CH0
2249 057732  023737  003520  003502            CMP     CH1,TEMP                ;SEE IF THIS CHANEL IS BAD
2250 057740  001430                            BEQ     2$              ;IF YES GO REPORT IT ELSE
2251 057742  012737  000001  003514            MOV     #1,CHNUM        ;GET CH1
2252 057750  023737  003522  003502            CMP     CH2,TEMP                ;SEE IF THIS CHANEL IS BAD
2253 057756  001421                            BEQ     2$              ;IF YES GO REPORT IT ELSE
2254 057760  012737  000002  003514            MOV     #2,CHNUM        ;GET CH2
2255 057766  023737  003524  003502            CMP     CH3,TEMP                ;SEE IF THIS CHANEL IS BAD
2256 057774  001412                            BEQ     2$              ;IF YES GO REPORT IT ELSE
2257 057776  012737  000003  003514            MOV     #3,CHNUM        ;GET CH3
2258 060004  023737  003526  003502            CMP     CH4,TEMP                ;SEE IF THIS CHANEL IS BAD
2259 060012  001403                            BEQ     2$              ;IF YES GO REPORT IT ELSE
2260 060014  104034                            EMT     34
2261 060016  062716  000002                    ADD     #2,(SP)         ;UPDATE STACK AND RETURN
2262 060022  000207                    2$:     RTS     PC              ;TO CONTINUE TEST
2263
2264 060024  062737  000010  003542    GOUP:   ADD     #10,ACOUNT      ;GO UP TO 1760 BY
2265 060032  062737  000010  003544            ADD     #10,BCOUNT      ;INCREMENT OF TEN
2266 060040  000207                            RTS     PC
2267
2268 060042  162737  000010  003542    GODN:   SUB     #10,ACOUNT      ;GO DOWN TO ZERO BY
2269 060050  162737  000010  003544            SUB     #10,BCOUNT      ;INVERSE INCREMENT OF TEN
2270 060056  000207                            RTS     PC              ;RETURN
2271
2272 060060  005237  003326            ROTDAT: INC     ROTFLG                  ;CLEAR THIS FLAG ON FIRST ENTRY
2273 060064  001001                            BNE     1$                      ;CHECK IF ZERO FIRST TIME
2274 060066  000261                            SEC                             ;FIRST TIME IN SET CARRY
2275 060070  106137  003564            1$:     ROLB    ROTPAT                  ;ROTATE PATTERN ONCE
2276 060074  005737  003540                    TST     KOUNT                   ;DID IT GET DONE 8 TIMES
2277 060100  001003                            BNE     2$                      ;IF WE HAVE SET ROTFLG
2278 060102  012737  177777  003326            MOV     #-1,ROTFLG              ;BACK TO MINUS ONE
2279 060110  000207                    2$:     RTS     PC                      ;NOW USE DATA
2280
2281 060112  004737  014310    DAGTST: JSR     PC,KLEER                ;CLEAR IT ALL
2282 060116  152777  000004  122520            BISB    #GBIT,  @CSR            ;SET THE GENERIC BIT IN THE CSR
2283 060124  117700  122576                    MOVB    @LDATA0,R0              ;READ THE GENERIC CODE IN DAC
2284 060130  042700  177400                    BIC     #177400,R0              ;CLEAR THE UPPER BYTE
2285 060134  022700  000261                    CMP     #261,R0                 ;CHECK THAT GENERIC CODE IS 261
2286 060140  001002                            BNE       1$                    ;IF NOT RETURN ERR MESS AND EXIT
```

```
2287 060142  012716  055240                MOV     #DCMN,  (SP)            ;ELSE CONTINUE TO TEST
2288 060146  142777  000004  122470  1$:   BICB    #GBIT,  @CSR            ;BUT FIRST CLEAR GBIT
2289 060154  000207                        RTS     PC
2290
2291
2292 060156  104401  036714          DECCHN: TYPE    ,DACHN                  ;ASK FOR CHANEL NUMBER (0,1,2,3)
2293 060162  104410                        RDOCT                           ;GET CHANEL NUMBER
2294 060164  012637  003566                MOV     (SP)+,ANSW      ;;POP STACK INTO ANSW
2295 060170  022737  000004  003566        CMP     #4,ANSW         ;CHECK FOR RIGHT CHAN
2296 060176  003003                        BGT     1$
2297 060200  104401  030374                TYPE    ,M20                    ;?
2298 060204  000764                        BR      DECCHN
2299 060206  013737  003566  003530  1$:   MOV     ANSW,DCHAN              ;GET DAC CHANEL SELECTED
2300 060214  000241                        CLC                             ;NO END AROUND CARRY WANTED
2301 060216  006337  003566                ASL     ANSW                    ;CONVERT TO VCHAN SELECTION
2302 060222  013737  003566  003532        MOV     ANSW,XCHAN              ;GET ANSW TO INDEX
2303 060230  006337  003532                ASL     XCHAN                   ;MULTIPLY BY 2 AGAIN
2304 060234  013737  003566  003534        MOV     ANSW,VCHAN              ;STORE SELECTED VCHAN
2305 060242  062737  000001  003566        ADD     #1,ANSW                 ;CONVERT TO ICHAN SELECTED
2306 060250  013737  003566  003536        MOV     ANSW,ICHAN              ;STORE SELECTE ICHAN
2307 060256  000207                        RTS     PC                              ;RETURN
2308
```

```
                                        .SBTTL A/D MONITOR
                                        ;THIS IS MONITOR FOR FOR AUTO MODE FOR A/D
060260  005737  002664          ADTST:  TST     AFLAG                   ;UNDER AUTO MODE ?
060264  001431                  ADTST:  BEQ     ATOD                    ;NO,BRANCH TO MANUAL TEST
060266  012777  015612  122430          MOV     #KBINT,@KBVEC
060274  152777  000100  120656          BISB    #BIT6,@$TKS     ;ENABLE KB INTERRUPT
060302  005005                          CLR     R5
060304  004737  017100                  JSR     PC,ADADDR       ;SET ADDRESSES AND CHECK REG.
060310  005705                          TST     R5              ;ADDRESS ERROR?
060312  001015                          BNE     1$
060314  152777  000001  122322          BISB    #RBIT,@CSR
060322  105077  123262                  CLRB    @STAT1
060326  105077  123260                  CLRB    @STAT2
060332  004737  017254                  JSR     PC,GCODE        ;FIND IF SE/DIFF MODE
060336  004737  061620                  JSR     PC,ADLOG        ;TEST LOGIC
060342  004737  063244                  JSR     PC,RUMP         ;TEST MONOTINICITY
060346  000207          1$:     RTS     PC
                                        ;THIS IS MONITOR FOR OPERATOR INTERVENTION TESTS
060350                          ATOD:
060350  012777  015612  122346          MOV     #KBINT,@KBVEC
060356  152777  000100  120574          BISB    #BIT6,@$TKS     ;ENABLE KB INTERRUPT
060364  004737  014310                  JSR     PC,KLEER
060370  012737  060376  003652          MOV     #ADRET,RETURN   ;SET RETURN ADDRESS FOR CONTROL A
060376  005005          ADRET:  CLR     R5
060400  004737  017100                  JSR     PC,ADADDR       ;SET ADDRESSES
060404  005705                          TST     R5              ;ADDRESS ERROR?
060406  001360                          BNE     ATOD
060410  152777  000001  122226          BISB    #RBIT,@CSR
060416  105077  123166                  CLRB    @STAT1          ;SET CHANNEL ZERO-NO MAX
060422  004737  017254                  JSR     PC,GCODE                ;FIND GENERIC CODE

060426  104401  034132          ADMON:  TYPE    ,MASS46         ;WHICH TEST TO RUN?
060432  012737  000014  003456          MOV     #14,$MUT        ;SET MODULE TYPE FOR ERRORS
060440  105077  123144                  CLRB    @STAT1
060444  105077  123142                  CLRB    @STAT2
060450  104406                          RDCHR
060452  012637  003566                  MOV     (SP)+,ANSW      ;;POP STACK INTO ANSW
060456  104401  003566                  TYPE    ,ANSW           ;ECHO
060462  104401  031400                  TYPE    ,MASS0          ;CR,LF
060466  004737  015526                  JSR     PC,CNTRC                ;CHECK FOR CONTROL C
060472  122737  000103  003566          CMPB    #'C,ANSW        ;IS IT CALIBRATION-C

060500  001003                          BNE     1$              ;NO
060502  004737  060712                  JSR     PC,ADCALB       ;EXECUTE CALIBRATION
060506  000747                          BR      ADMON

060510  122737  000104  003566  1$:     CMPB    #'D,ANSW        ;IS IT LOGIC TEST
060516  001005                          BNE     2$              ;NO
```

```
2364 060520  004737  017306          5$:     JSR     PC,SWLOOP
2365 060524  061620                          ADLOG                   ;DO ADLOG TEST
2366 060526  000137  060426                  JMP     ADMON
2367
2368 060532  122737  000115  003566  2$:     CMPB    #'M,ANSW        ;IS IT MONOTONICITY TEST-M
2369 060540  001005                          BNE     10$
2370 060542  004737  017306          8$:     JSR     PC,SWLOOP
2371 060546  063244                          RUMP                    ;DO RUMP TEST
2372 060550  000137  060426                  JMP     ADMON
2373
2374
2375 060554  022737  000130  003566  10$:    CMP     #'X,ANSW
2376 060562  001037                          BNE     15$
2377 060564  104401  033417          17$:    TYPE    ,MASS37         ;WHAT MUX
2378 060570  104410                          RDOCT
2379 060572  012637  003502                  MOV     (SP)+,TEMP      ;;POP STACK INTO TEMP
2380 060576  001404                          BEQ     20$             ;NO MUX 0
2381 060600  122737  000007  003502          CMPB    #7,TEMP         ;MUX TO HIGH?
2382 060606  002003                          BGE     16$
2383 060610  104401  030374          20$:    TYPE    ,M20            ;?
2384 060614  000763                          BR      17$
2385 060616  006337  003502          16$:    ASL     TEMP
2386 060622  006337  003502                  ASL     TEMP
2387 060626  006337  003502                  ASL     TEMP
2388 060632  006337  003502                  ASL     TEMP
2389 060636  006337  003502                  ASL     TEMP
2390 060642  013737  003502  003450          MOV     TEMP,MXNUM
2391 060650  004737  017306                  JSR     PC,SWLOOP
2392 060654  064312                          MUX1            ;DO THIS TEST
2393 060656  000137  060426                  JMP     ADMON
2394 060662  122737  000110  003566  15$:    CMPB    #'H,ANSW
2395 060670  001004                          BNE     14$
2396 060672  104401  033306                  TYPE    ,MASS36         ;TYPE ALL OPTIONS
2397 060676  000137  060426                  JMP     ADMON
2398
2399 060702  104401  030374          14$:    TYPE    ,M20            ;UNKNOWN CHARACTER.
2400 060706  000137  060426                  JMP     ADMON
2401
2402
2403
2404
2405                                .SBTTL  CALIBRATION OF A14 AND MUX
```

```
2407
2408 060712  104401  033417          ADCALB: TYPE    ,MASS37         ;WHICH MUX NUMBER YOU WANT TO
2409 060716  004737  017254                  JSR     PC,GCODE
2410 060722  004737  015526                  JSR     PC,CNTRC
2411                                                                  ;TEST-ZERO FOR A/D
2412 060726  013746  000004                  MOV     ERRVEC,-(SP)    ;;PUSH ERRVEC ON STACK
2413 060732  012737  061570  000004          MOV     #20$,ERRVEC
2414 060740  104410                          RDOCT
2415 060742  012637  003502                  MOV     (SP)+,TEMP      ;;POP STACK INTO TEMP
2416 060746  122737  000007  003502          CMPB    #7,TEMP
2417 060754  002003                          BGE     1$
2418 060756  104401  030374                  TYPE    ,M20            ;?
2419 060762  000753                          BR      ADCALB
2420
2421 060764  006337  003502          1$:     ASL     TEMP            ;SHIFT IT 5 TIMES TO SET MUX #
2422 060770  006337  003502                  ASL     TEMP
2423 060774  006337  003502                  ASL     TEMP
2424 061000  006337  003502                  ASL     TEMP
2425 061004  006337  003502                  ASL     TEMP
2426 061010  104401  033463          5$:     TYPE    ,MASS38         ;WHICH CHANNEL?-ONLY FOR A014
2427 061014  104410                          RDOCT
2428 061016  012637  003600                  MOV     (SP)+,TEMP3     ;;POP STACK INTO TEMP3
2429 061022  105737  003502                  TSTB    TEMP
2430 061026  001022                          BNE     2$
2431 061030  022737  000301  003606          CMP     #301,GBITE      ;IS IT DIFF MODE
2432 061036  001007                          BNE     3$
2433
2434 061040  122737  000007  003600          CMPB    #7,TEMP3        ;CHECK IF CHANNEL NOT TOO HIGH
2435 061046  002130                          BGE     4$              ;OK
2436 061050  104401  033537                  TYPE    ,MASS39         ;TOO HIGH CHANNEL FOR DIFF MODE
2437 061054  000755                          BR      5$
2438
2439 061056  122737  000017  003600  3$:     CMPB    #17,TEMP3       ;IS CHANNEL TOO HIGH?
2440 061064  002121                          BGE     4$              ;OK
2441 061066  104401  033602                  TYPE    ,MASS40         ;TOO HIGH CHANNEL FOR SINGLE END
2442 061072  000746                          BR      5$
2443
2444 061074  113777  003502  122506  2$:     MOVB    TEMP,@STAT1     ;HERE ONLY IF MUX-SET MUX #
2445 061102  152777  000004  121534          BISB    #GBIT,@CSR      ;SET GBIT
2446 061110  117737  122474  003576          MOVB    @STAT1,TEMP2
2447 061116  042737  177400  003576          BIC     #177400,TEMP2   ;FIND FROM GENERIC CODE WHAT KIND
2448 061124  142777  000004  121512          BICB    #GBIT,@CSR      ;OF MUX AND IN WHAT MODE
2449 061132  122737  000342  003576          CMPB    #342,TEMP2      ;IS IT A156 SE?
2450 061140  001007                          BNE     6$              ;NO
2451 061142  122737  000037  003600          CMPB    #37,TEMP3       ;IS CHANNEL TOO HIGH
2452 061150  002067                          BGE     4$              ;NO
2453 061152  104401  033602                  TYPE    ,MASS40         ;TOO HIGH
2454 061156  000714                          BR      5$
2455
2456 061160  005000                  6$:     CLR     R0              ;SET GAIN =1
2457 061162  122737  000322  003576          CMPB    #322,TEMP2      ;IS IT A156 DIFF
2458 061170  001097                          BNE     7$
2459 061172  122737  000017  003600          CMPB    #17,TEMP3       ;IS CHANNEL TOO HIGH?
2460 061200  002053                          BGE     4$              ;NO
2461 061202  104401  033537                  TYPE    ,MASS39         ;YES
2462 061206  000700                          BR      5$
2463
```

```
2464 061210  122737  000323  003576  7$:     CMPB    #323,TEMP2      ;IS IT A157
2465 061216  001031                          BNE     8$              ;NO, UNKNOWN GENERIC CODE.
2466 061220  122737  000017  003600          CMPB    #17,TEMP3       ;IS CHANNEL TOO HIGH?
2467 061226  002003                          BGE     9$              ;NO
2468 061230  104401  033537                  TYPE    ,MASS39         ;YES
2469 061234  000665                          BR      5$
2470                                                                ;HERE ONLY IF A157
2471 061236  104401  033655          9$:     TYPE    ,MASS41         ;WHICH GAIN?
2472 061242  104410                          RDOCT
2473 061244  012637  003602                  MOV     (SP)+,TEMP4     ;;POP STACK INTO TEMP4
2474 061250  005000                          CLR     R0
2475 061252  023760  003602  003330  11$:    CMP     TEMP4,GAINTB(R0)        ;FIND IF GAIN IS CORRECT
2476 061260  001413                          BEQ     21$             ;DO THE CALIBRATION FOR A157
2477 061262  062700  000002                  ADD     #2,R0
2478 061266  005760  003330                  TST     GAINTB(R0)
2479
2480 061272  001367                          BNE     11$
2481 061274  104401  033726                  TYPE    ,MASS42         ;WRONG GAIN SELECTED
2482 061300  000756                          BR      9$
2483
2484
2485 061302  104401  032135          8$:     TYPE    ,MASS11         ;UNKNOWN GENERIC CODE
2486 061306  000601                          BR      ADCALB
2487
2488
2489 061310  116077  003352  122274  21$:    MOVB    GAIN(R0),@STAT2 ;SET GAIN
2490 061316  104401  036572                  TYPE    ,GAINMG
2491 061322  016046  003330                  MOV     GAINTB(R0),-(SP)        ;;SAVE GAINTB(R0) FOR TYPEOUT
     061326  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2492 061330  153777  003600  122252  4$:     BISB    TEMP3,@STAT1
2493 061336  104401  033754                  TYPE    ,MASS43         ;CONNECT VOLTAGE SOURCE, TYPE CR
2494 061342  004737  015716                  JSR     PC,CRTST        ;WAIT FOR CR
2495 061346  104401  034011                  TYPE    ,MASS44         ;TYPE HEADER    -- OCTAL  VOLTAGE--
2496 061352  004737  065502          14$:    JSR     PC,CONV7        ;DO 7 CONV & AVERAGE THEM
2497 061356  012701  000002                  MOV     #2,R1
2498 061362                          13$:
     061362  013737  002636  003446          MOV     K,ZLOOP
     061370  005037  003444          65$:    CLR     YLOOP           ;WAIT
     061374  005237  003444          64$:    INC     YLOOP
     061400  023727  003444  177777          CMP     YLOOP,#-1
     061406  001372                          BNE     64$
     061410  005337  003446                  DEC     ZLOOP
     061414  001365                          BNE     65$
2499 061416  005301                          DEC     R1
2500 061420  001360                          BNE     13$
2501 061422  032737  000004  003654          BIT     #BIT2,AVRSAV
2502 061430  001410                          BEQ     40$
2503 061432  005237  003512                  INC     INTDAT
2504 061436  013746  003512                  MOV     INTDAT,-(SP)    ;;SAVE INTDAT FOR TYPEOUT
     061442  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2505 061444  005337  003512                  DEC     INTDAT
2506 061450  000403                          BR      41$
2507
2508 061452                          40$:
     061452  013746  003512                  MOV     INTDAT,-(SP)    ;;SAVE INTDAT FOR TYPEOUT
     061456  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2509 061460  162737  004000  003512  41$:    SUB     #4000,INTDAT    ;CALCULATE VOLTAGE IN MILLIVOLTS
```

```
2510 061466  013737  003512  001140          MOV     INTDAT,$GDDAT
2511 061474  063737  001140  001140          ADD     $GDDAT,$GDDAT   ;MULTIPLY BY 5
2512 061502  063737  001140  001140          ADD     $GDDAT,$GDDAT
2513 061510  063737  003512  001140          ADD     INTDAT,$GDDAT
2514 061516  042737  177770  003654          BIC     #177770,AVRSAV
2515 061524  013704  003654                  MOV     AVRSAV,R4
2516 061530  116404  003374                  MOVB    AVRTBL(R4),R4
2517 061534  060437  001140                  ADD     R4,$GDDAT
2518
2519 061540  104401  030036                  TYPE    ,M10            ;TAB
2520 061544  104401  030036                  TYPE    ,M10
2521 061550  013746  001140                  MOV     $GDDAT,-(SP)    ;;SAVE $GDDAT FOR TYPEOUT
     061554  104405                          TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
2522 061556  104401  031400                  TYPE    ,MASSO          ;CR,LF
2523 061562  004737  015526                  JSR     PC,CNTRC        ;CONTROL C?
2524 061566  000671                          BR      14$             ;DO IT AGAIN.
2525 061570  022626                  20$:    CMP     (SP)+,(SP)+
2526 061572  104401  031011                  TYPE    ,M84            ;SELECTED MUX DID NOT RESPOND
2527 061576  012637  000004                  MOV     (SP)+,ERRVEC    ;;POP STACK INTO ERRVEC
2528 061602  152777  000001  121034          BISB    #RBIT,@CSR      ;RESET TIME-OUT
2529 061610  105777  122000                  TSTB    @LBYTE
2530 061614  000137  060712                  JMP     ADCALB
```

```
2532                                    .SBTTL  LOGIC TEST OF A014
2533
2534 061620                             ADLOG:
                                        ;;*****************************************************************
     061620  000004                     TST26:  SCOPE
     061622  012737  000026  001212             MOV     #26,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
2535
2536 061630  004737  014310                     JSR       PC    ,KLEER          ;CLEAR THE IOCM WORLD
2537 061634  012746  000000                     MOV     #PR0,-(SP)      ;SET PSW TO PRIORITY 0
   . 061640  012746  061646                     MOV     #64$,-(SP)
     061644  000002                             RTI
     061646  000240                     64$:    NOP
2538 061650  152777  000001  120766             BISB    #RBIT   , @CSR  ;CLEAR RANDOM INTERUPT
2539 061656  005037  001142                     CLR     $BDDAT
2540 061662  117737  121724  001142             MOVB    @STAT2,$BDDAT           ;STORE DATA IN $BDAT
2541 061670  132777  000001  121714             BITB    #BIT0   , @STAT2        ;CHECK BUSY BIT CLEAR
2542 061676  001401                             BEQ       1$            ;IF NO ERROR CONTINUE
2543 061700  104035                             EMT     35
2544
2545 061702  132777  000002  121702  1$:       BITB    #BIT1,@STAT2    ;CHECK CONV DONE IS CLEAR
2546 061710  001401                             BEQ       2$            ;IF NO ERROR CONTINUE
2547 061712  104035                             EMT     35
2548
2549 061714  132777  000004  121670  2$:       BITB    #BIT2,@STAT2    ;CHECK ERROR BIT IS CLEAR
2550 061722  001401                             BEQ       3$            ;IF NO ERROR CONTINUE
2551 061724  104035                             EMT     35
2552
2553 061726  132777  000010  121656  3$:       BITB    #BIT3,@STAT2    ;CHECK MUX TIME OUT IS CLEAR
2554 061734  001401                             BEQ     4$              ;IF NO ERROR CONTINUE
2555 061736  104035                             EMT     35
2556
2557 061740  132777  000360  121644  4$:       BITB    #360,@STAT2     ;CHECK GAIN BITS
2558 061746  001401                             BEQ     BSYCON          ;IF NO ERROR CONTINUE
2559 061750  104035                             EMT     35
2560 061752  012777  062116  120670  BSYCON:   MOV     #10$,@VECT0
2561 061760  012777  000340  120664             MOV     #PR7,@VECT0A
2562 061766  152777  000001  120650             BISB    #RBIT,@CSR      ;CLEAR INTERRUPTS
2563 061774  105777  121610                     TSTB    @STAT1
2564 062000  012700  177777                     MOV     #-1     , R0    ;INITIALIZE COUNTER
2565 062004  152777  000001  121600             BISB    #BIT0   , @STAT2        ;START CONVERSION
2566 062012  005200                     1$:     INC     R0              ;SRATR COUNTER
2567 062014  022700  000300                     CMP     #300    , R0    ;CHECK FOR COUNT OF 100
2568 062020  001002                             BNE       8$            ;STILL WAIT FOR CONVERSION
2569 062022  104040                             EMT     40
2570 062024  000474                             BR      7$
2571 062026  132777  000002  121556  8$:       BITB    #BIT1   , @STAT2        ;CHECK FOR CONVERSION DONE
2572 062034  001766                             BEQ       1$            ;IF NOT DONE THEN LOOP
2573 062036  132777  000001  121546  2$:       BITB    #BIT0   , @STAT2        ;VERIFY BUSY CLEAR
2574 062044  001402                             BEQ       3$            ;IF CLEAR CONTINUE
2575 062046  104036                             EMT     36
2576 062050  000462                             BR      7$
2577 062052  132777  000004  121532  3$:       BITB    #BIT2,@STAT2    ;CHECK ERROR CLEAR
2578 062060  001402                             BEQ     4$              ;CONTINUE IF CLEAR
2579 062062  104037                             EMT     37
2580 062064  000454                             BR      7$
2581 062066  105777  120552             4$:     TSTB    @CSR            ;TST INTERRUPT FLAG
2582 062072  100402                             BMI     5$              ;IT IS SET
```

```
2583 062074  104041                          EMT     41
2584 062076  000447                          BR      7$
2585 062100  152777  000100  120536  5$:     BISB    #EBIT,@CSR      ;ENABLE INTERRUPT
2586 062106  000240                          NOP
2587 062110  000240                          NOP
2588 062112  104041                          EMT     41
2589 062114  000440                          BR      7$
2590
2591
2592 062116  022626                  10$:    CMP     (SP)+,(SP)+     ;ADJUST STACK POINTER
2593 062120  142777  000100  120516          BICB    #EBIT,@CSR
2594 062126  013737  002660  003502          MOV     BASE,TEMP       ;INITIALIZE TEMP
2595 062134  132777  000200  120502  13$:    BITB    #FBIT,@CSR      ;INTERRUPT SET?
2596 062142  001424                          BEQ     12$             ;NO
2597
2598 062144  005037  003510                  CLR     TEMP1
2599 062150  117737  120472  003502          MOVB    @IAR,TEMP       ;FIND WHICH I/O INTERRUPTED
2600 062156  123737  003502  003462          CMPB    TEMP,TADDR      ;IS IT MUT
2601 062164  001414                          BEQ     7$              ;YES
2602 062166  152777  000001  120450          BISB    #RBIT,@CSR      ;CLEAR INTERRUPT
2603 062174  105777  121302                  TSTB    @TEMP
2604 062200  005237  003510                  INC     TEMP1
2605 062204  022737  000400  003510          CMP     #400,TEMP1
2606 062212  001350                          BNE     13$
2607 062214                          12$:
     062214  104041                          EMT     41
2608 062216  152777  000001  120420  7$:     BISB    #RBIT,@CSR      ;CLEAR INTERRUPTS
2609 062224  105777  121364                  TSTB    @LBYTE
2610 062230  004737  015410                  JSR     PC,CLRINT
2611 062234  012746  000000                  MOV     #PR0,-(SP)      ;SET PSW TO PRIORITY 0
     062240  012746  062246                  MOV     #64$,-(SP)
     062244  000002                          RTI
     062246  000240                  64$:    NOP
2612
2613
2614 062250                          CHSEL:
2615 062250  005000                          CLR       R0            ;SET TO START WITH CHANEL ZERO
2616 062252  005001                          CLR       R1            ;CLEAR REFERENCE REGISTER
2617 062254  105077  121330                  CLRB    @STAT1          ;CLEAR CHANEL TO ZERO
2618 062260  105777  121324                  TSTB    @STAT1          ;VERIFY CHANEL IS ZERO
2619 062264  001406                          BEQ       1$            ;IF ZERO CONTINUE
2620 062266  005037  001140                  CLR     $GDDAT          ;CHECK FOR CHANEL 0
2621 062272  117737  121312  001142          MOVB    @STAT1,$BDDAT   ;GET ACTUAL CHANEL
2622 062300  104042                          EMT     42
2623 062302  005200                  1$:     INC       R0            ;GET NEXT CHANEL TO CHECK
2624 062304  110077  121300                  MOVB      R0    , @STAT1        ;WRITE CHANEL SELECTED
2625 062310  117701  121274                  MOVB    @STAT1  , R1    ;READ CHANEL SELECTED
2626 062314  020001                          CMP       R0    , R1    ;VERIFY CORRECT CHANEL
2627 062316  001406                          BEQ       2$            ;CONTINUE IF MATCH
2628 062320  010037  001140                  MOV     R0,$GDDAT       ;STORE CHANEL WRITTEN
2629 062324  010137  001142                  MOV     R1,$BDDAT       ;STORE CHANEL READ
2630 062330  104042                          EMT     42
2631 062332  000431                          BR      CHNSEL
2632 062334  022737  000301  003606  2$:     CMP     #301    ,GBITE  ;CHECK FOR DIFF MODE
2633 062342  001404                          BEQ       3$            ;IF NOT DIFF MODE
2634 062344  022700  000017                  CMP     #17     , R0    ;CHECK FOR 17 CHANELS
2635 062350  001354                          BNE       1$            ;GET NEXT CHANEL UNTIL 17 DONE
```

```
2636 062352  000403                        BR      4$              ;EXIT WHEN DONE
2637 062354  022700  000007          3$:    CMP     #7      , R0    ;ELSE CHECK 7 DIFF CHANELS
2638 062360  001350                        BNE     1$              ;GET NEXT CHANEL UNTIL 7 DONE
2639 062362  132777  000004  121222  4$:    BITB    #BIT2,@STAT2    ;CHECK IF ERROR BIT CLEAR
2640 062370  001401                        BEQ     5$
2641 062372  104055                        EMT     55
2642 062374  105077  121210          5$:    CLRB    @STAT1          ;SET CHANELS TO ZERO
2643 062400  001406                        BEQ     CHNSEL          ;IF ZERO - DONE
2644 062402  005037  001140                CLR     $GDDAT          ;STORE CHANEL WRITTEN
2645 062406  117737  121176  001142        MOVB    @STAT1,$BDDAT   ;STORE CHANEL READ
2646 062414  104042                        EMT     42
2647
2648 062416                        CHNSEL:
2649 062416  022737  000301  003606        CMP     #301    ,GBITE  ;CHECK IF IN DIFF MODE
2650 062424  001003                        BNE     1$              ;IN NOT DO SING END MODE
2651 062426  012700  000010                MOV     #10     , R0    ;START WITH CH10 IN DIFF MODE
2652 062432  000402                        BR      2$              ;AND START CHECK
2653 062434  012700  000020          1$:    MOV     #20     , R0    ;START WITH CH20 IN SING END MODE
2654 062440  152777  000001  120176  2$:    BISB    #RBIT   , @CSR  ;CLEAR ANY INTERRUPTS
2655 062446  105777  121142                TSTB    [illegible]     ;CLEAR RIF BIT
2656 062452  110077  121132          4$:    [illegible]  R0 , @STAT1        ;LOAD NON-EXISTANT CHANEL
2657 062456  132777  000004  121126        BITB    #BIT2   , @STAT2        ;CHECK ERROR BIT SET
2658 062464  001002                        BNE     5$              ;CONTINUE IF SET
2659 062466  104043                        EMT     43
2660 062470  000412                        BR      ADGAN
2661
2662 062472  132777  000200  120144  5$:    BITB    #BIT7,@CSR      ;TEST IF ERROR INTERRUPTED
2663 062500  001002                        BNE     6$
2664 062502  104044                        EMT     44
2665 062504  000404                        BR      ADGAN
2666 062506  005200                  6$:    INC     R0              ;UPDATE TO NEXT CHANEL
2667 062510  022700  000040                CMP     #40     , R0    ;CHECK FOR LAST CHANEL
2668 062514  001351                        BNE     2$              ;CONTINUE IF NOT DONE
2669
2670 062516  012700  000020          ADGAN: MOV     #20,R0          ;GET FIRST GAIN VALUE
2671 062522  152777  000001  120114  1$:    BISB    #RBIT,@CSR      ;CLEAR INTERRUPTS
2672 062530  105077  121054                CLRB    @STAT1
2673 062534  105777  121054                TSTB    @LBYTE
2674 062540  110077  121046                MOVB    R0,@STAT2               ;WRITE GAIN
2675 062544  132777  000004  121040        BITB    #BIT2,@STAT2    ;CHECK FOR ERROR
2676 062552  001002                        BNE     2$              ;BRANCH IF ERROR
2677 062554  104045                        EMT     45
2678 062556  000405                        BR      3$
2679 062560  062700  000020          2$:    ADD     #20,R0          ;GET NEXT GAIN
2680 062564  022700  000400                CMP     #400,R0         ;CHECK FOR LAST
2681 062570  001354                        BNE     1$              ;IF NOT CONTINUE
2682 062572  152777  000001  120044  3$:    BISB    #RBIT,@CSR      ;CLEAR INTERRUPTS
2683 062600  105777  121010                TSTB    @LBYTE
2684
2685 062604  005001                ADTBIT: CLR     R1              ;SET DIFF CHANEL NUMBER
2686 062606  005003                        CLR     R3              ;SET SEND CHANEL NUMB
2687 062610  005037  003500                CLR     NORAMP          ;CLEAR NO RAMP FLAG
2688 062614  152777  000010  120022        BISB    #TBIT , @CSR    ;SET T-BIT
2689 062622  122737  000321  003606  2$:    CMPB    #321,GBITE      ;CHECK  GEN CODE
2690 062630  001403                        BEQ     3$
2691 062632  110177  120752                MOVB    R1,@STAT1               ;WRITE DIFF CHANEL NUMBER
2692 062636  000402                        BR      12$
```

```
2693 062640 110377 120744          3$:   MOVB   R3,@STAT1              ;SELECT SEND CHANEL NUMBER
2694 062644 004737 065502          12$:  JSR    PC,CONV7          ;GO READ A/D DATA
2695 062650 005701                       TST    R1
2696 062652 001011                       BNE    5$                ;IF NOT CHECK NEXT CHANEL
2697 062654 023727 003512 007400         CMP    INTDAT,#7400              ;ELSE CHECK +V OUTPUT ALL ONES
2698 062662 002042                       BGE    9$                ;IF OK CHECK NEXT CHANEL
2699 062664 012737 007400 001140         MOV    #7400,$GDDAT      ;STORE LOWER LIMIT
2700 062672 104046                       EMT    46
2701 062674 000435                 4$:   BR     9$
2702 062676 022701 000001          5$:   CMP    #1     , R1       ;CHECK FOR CHANEL #1
2703 062702 001011                       BNE    7$                ;ELSE GO TO NEXT CHANEL
2704 062704 023727 003512 000400         CMP    INTDAT,#400               ;CHECK IF ABS ZERO IS IN LIMIT
2705 062712 003426                       BLE    9$                ;BRANCH IF OK
2706 062714 012737 000400 001140         MOV    #400,$GDDAT       ;STORE UPPER LIMIT
2707 062722 104046                       EMT    46
2708 062724 000421                 6$:   BR     9$                ;GET NEXT CHANEL
2709 062726 023727 003512 004005   7$:   CMP    INTDAT,#4005      ;CHECK HIGH LIMIT OF ZERO VALUE
2710 062734 003404                       BLE    8$                ;CONTINUE IF OK
2711 062736 012737 004005 001140         MOV    #4005,$GDDAT
2712 062744 104046                       EMT    46
2713 062746 023727 003512 003773   8$:   CMP    INTDAT,#3773      ;CHECK LOW LIMIT OF ZERO VALUE
2714 062754 002023                       BGE    11$               ;CONTINUE IF OK
2715 062756 012737 003773 001140         MOV    #3773,$GDDAT
2716 062764 104046                       EMT    46
2717 062766 000416                       BR     11$
2718 062770 005701                 9$:   TST     R1                ;IS CHANEL #0 SELECTED
2719 062772 001004                       BNE    10$               ;IF NOT CHECK NEXT CHANEL
2720 062774 005201                       INC    R1                ;ELSE UPDATE TO NEXT CHANEL
2721 062776 062703 000002                ADD    #2,R3             ;UPDATE SEND CHANEL
2722 063002 000707                       BR     2$                ;AND GO CHECK IT
2723 063004 022701 000001          10$:  CMP    #1     , R1       ;IS CHANEL #1 SELECTED
2724 063010 001005                       BNE    11$               ;IF NOT CONTINUE
2725 063012 062701 000002                ADD    #2     , R1       ;ELSE UPDATE TO CHANEL #3
2726 063016 062703 000004                ADD    #4,R3             ;UPDATE SEND CHANEL
2727 063022 000677                       BR     2$                ;AND GO CHECK IT
2728 063024 122737 000321 003606   11$:  CMPB   #321,GBITE                ;TEST IF RAMP IS WORKING
2729 063032 001404                       BEQ    14$
2730 063034 112777 000002 120546         MOVB   #2,@STAT1         ;SET RAMP FOR SE
2731 063042 000403                       BR     15$
2732 063044 112777 000004 120536   14$:  MOVB   #4,@STAT1         ;SET RAMP FOR DIFF
2733 063052 005000                 15$:  CLR    R0
2734 063054 005001                       CLR    R1
2735 063056 004737 065502          17$:  JSR    PC,CONV7          ;DO CONVERSIONS
2736 063062 022737 007776 003512         CMP    #7776,INTDAT      ;WAIT FOR TOP OF RAMP
2737 063070 003412                       BLE    16$
2738 063072 105200                       INCB   R0
2739 063074 001370                       BNE    17$
2740 063076 005201                       INC    R1
2741 063100 022701 000050                CMP    #50,R1
2742 063104 002364                       BGE    17$
2743 063106 104056                       EMT    56
2744 063110 005237 003500                INC    NORAMP
2745 063114 000421                       BR     18$
2746
2747 063116 005001                 16$:  CLR    R1
2748 063120 005000                       CLR    R0
2749 063122 004737 065502          20$:  JSR    PC,CONV7          ;WAIT FOR BOTTOM OF RUMP
```

```
2750 063126  022737  000002  003512          CMP     #2,INTDAT
2751 063134  002011                          BGE     18$
2752 063136  105200                          INCB    R0              ;WAIT UP TO 6SEC
2753 063140  001370                          BNE     20$
2754 063142  005201                          INC     R1
2755 063144  022701  000050                  CMP     #50,R1
2756 063150  002364                          BGE     20$
2757 063152  104056                          EMT     56
2758 063154  005237  003500                  INC     NORAMP
2759 063160  105077  120424          18$:    CLRB    @STAT1          ;SELECT CHANNEL 0 ALL ONES
2760 063164  152777  000020  117452          BISB    #DBIT,@CSR      ;SET THE D-BIT
2761 063172  004737  065502                  JSR     PC,CONV7        ;GO DO CONVERSION
2762 063176  012737  003774  001140          MOV     #3774,$GDDAT    ;GET LOW END TOLER
2763 063204  023737  001140  001142          CMP     $GDDAT,$BDDAT
2764 063212  003401                          BLE     43$
2765 063214  104050                          EMT     50
2766 063216  012737  004004  001140  43$:    MOV     #4004,$GDDAT    ;GET HIGH END TOLER
2767 063224  023737  001140  001142          CMP     $GDDAT,$BDDAT
2768 063232  002001                          BGE     44$
2769 063234  104050                          EMT     50
2770 063236  004737  014310          44$:    JSR     PC,KLEER        ;CLEAR THE WORLD
2771 063242  000207                          RTS     PC              ;EXIT
2772
```

```
                                        .SBTTL  A/D MONOTONICITY TEST

                                        ;*************************************************************
                                        ;                       MONTONICITY TEST
                                        ;*************************************************************
2780 063244                             RUMP:
2781                                    ;;************************************************************************
     063244  000004                     TST27:  SCOPE
     063246  012737  000027  001212             MOV     #27,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
2782 063254  012746  000000                     MOV     #PR0,-(SP)      ;SET PSW TO PRIORITY 0
     063260  012746  063266                     MOV     #64$,-(SP)
     063264  000002                             RTI
     063266  000240                     64$:    NOP
2783 063270  005737  003500                     TST     NORAMP
2784 063274  001407                             BEQ     1$
2785 063276  005737  002664                     TST     AFLAG
2786 063302  001003                             BNE     2$
2787 063304  104056                             EMT     56
2788 063306  104401  031200                     TYPE    ,MASS51
2789 063312  000207                     2$:     RTS     PC
2790 063314                             1$:
2791 063314  152777  000001  117322             BISB    #RBIT,@CSR              ;CLEAR DONE FLAG
2792 063322  105777  120264                     TSTB    @STAT2
2793 063326  152777  000010  117310             BISB    #TBIT,@CSR              ;SET T BIT
2794 063334  004737  015410                     JSR     PC,CLRINT
2795 063340  012777  064022  117302 RAMPST:     MOV     #INTR14,@VECTO          ;SET UP INTERRUPT VECTOR
2796 063346  012777  000340  117276             MOV     #340,@VECTOA            ;SET UP PRIORITY
2797 063354  152777  000100  117262             BISB    #EBIT,@CSR              ;ENABLE INTERRUPTS
2798 063362  005004                             CLR     R4                      ;CLEAR ERROR POINTER
2799 063364  004737  065452                     JSR     PC,SETRAM               ;SET RAMP
2800 063370  005037  003442             1$:     CLR     IFLAG                   ;RESET INTERRUPT FLAG
2801 063374  152777  000001  120210             BISB    #BIT0,@STAT2            ;START CONVERSION
2802 063402  132737  000001  003442     2$:     BITB    #BIT0,IFLAG             ;WAIT FOR INTERRUPT
2803 063410  001774                             BEQ     2$
2804 063412  117737  120176  003572             MOVB    @LBYTE,DATALO
2805 063420  117737  120172  003573             MOVB    @HBYTE,DATALO+1
2806 063426  005737  003572                     TST     DATALO                  ;WAIT FOR -10.240 VOLTS
2807 063432  001356                             BNE     1$
2808 063434  005037  003556             BEG:    CLR     BLAST                   ;STORE FIRST CONVERSION
2809 063440  005037  003560                     CLR     LAST
2810 063444  005037  003562                     CLR     LASTCN
2811 063450  005037  003442             ST00:   CLR     IFLAG                   ;RESET INTERRUPT FLAG
2812 063454  152777  000001  120130             BISB    #BIT0,@STAT2            ;START CONVERSION
2813 063462  132737  000001  003442     13$:    BITB    #BIT0,IFLAG             ;WAIT FOR INTERRUPT
2814 063470  001774                             BEQ     13$
2815 063472  117737  120116  003572             MOVB    @LBYTE,DATALO
2816 063500  117737  120112  003573             MOVB    @HBYTE,DATALO+1
2817 063506  013737  003572  003560             MOV     DATALO,LAST
2818 063514  013737  003572  001142             MOV     DATALO,$BDDAT
2819 063522  023737  003556  003560             CMP     BLAST,  LAST
2820 063530  001431                             BEQ     3$
2821 063532  005337  003560                     DEC     LAST
2822 063536  023737  003556  003560             CMP     BLAST,  LAST            ;COMPARE BLAST+1WITH LAST
2823 063544  001416                             BEQ     2$
2824 063546  062737  000002  003560             ADD     #2,     LAST
2825 063554  023737  003556  003560             CMP     BLAST,  LAST
2826 063562  001401                             BEQ     1$
```

```
2827 063564  000431                        BR      4$                      ;ERROR MESSAGE
2828 063566  005737  003562          1$:   TST     LASTCN                  ;IS LASTCN=0 ?
2829 063572  001410                        BEQ     3$                      ;YES
2830 063574  005337  003556                DEC     BLAST                   ;MOVE LAST CONVERSION BEFORE LAST LOCATION
2831 063600  000405                        BR      3$
2832 063602  005737  003562          2$:   TST     LASTCN                  ;IS LASTCN=0 ?
2833 063606  001002                        BNE     3$
2834 063610  005237  003556                INC     BLAST
2835 063614  023727  003560  007777  3$:   CMP     LAST,   #7777           ;IS LAST=10.240 VOLTS ?
2836 063622  001003                        BNE     5$
2837 063624  012737  000001  003562        MOV     #1,     LASTCN          ;SET FLAG AT TOP OF RAMP
2838 063632  005737  003560          5$:   TST     LAST                    ;ARE WE STILL AT MINUS 10 VOLTS ?
2839 063636  001304                        BNE     STCO
2840 063640  005737  003562                TST     LASTCN                  ;IS RAMP GOING UP ?
2841 063644  001701                        BEQ     STCO
2842 063646  000431                        BR      27$                     ;CHECK FOR ERRORS
2843 063650  013737  003556  001140  4$:   MOV     BLAST,  $GDDAT          ;GOOD DATA FOR ERROR MESSAGE
2844 063656  005737  003562                TST     LASTCN                  ;IS RAMP GOING UP ?
2845 063662  001003                        BNE     7$                      ;YES, DECREMENT LAST CONVERSION
2846 063664  062737  000002  003556        ADD     #2,     BLAST           ;RAMP GOING DOWN
2847 063672  005337  003556          7$:   DEC     BLAST                   ;ADJUST BLAST AFTER ERROR
2848 063676  013764  001140  002746        MOV     $GDDAT,RAMP1(R4)
2849 063704  013764  001142  003066        MOV     $BDDAT,RAMP2(R4)
2850 063712  013764  003562  003206        MOV     LASTCN,RAMP3(R4)        ;STORE RAMP SLOPE IN TABLE
2851 063720  062704  000002                ADD     #2,R4                   ;INCREMENT TABLE POINTER
2852 063724  022704  000050                CMP     #40.,R4                 ;IS TABLE FULL
2853 063730  001331                        BNE     3$                      ;NO
2854 063732  005704                27$:    TST     R4                      ;ANY ERRORS ?
2855 063734  001425                        BEQ     OUT                     ;NO ,EXIT
2856 063736  104401  031047                TYPE    ,MASS55                 ;OUT OF RANGE BY TWO BITS
2857 063742  005003                        CLR     R3                      ;POINTER FOR ERROR TABLE
2858 063744  016337  002746  001140  28$:   MOV     RAMP1(R3),$GDDAT
2859 063752  016337  003066  001142        MOV     RAMP2(R3),$BDDAT
2860 063760  016337  003206  003562        MOV     RAMP3(R3),LASTCN
2861 063766  104051                        EMT     51
2862 063770  062703  000002                ADD     #2,R3                   ;INCREMENT DATA POINTER
2863 063774  020403                        CMP     R4,R3                   ;LAST ERROR ?
2864 063776  001362                        BNE     28$                     ;NO
2865 064000  020427  000050                CMP     R4,#40.                 ;TOO MANY ERRORS ?
2866 064004  001001                        BNE     OUT
2867 064006  104054                        EMT     54
2868
2869 064010  004737  015410          OUT:  JSR     PC,CLRINT
2870 064014  004737  014310                JSR     PC,KLEER
2871 064020  000207                        RTS     PC
2872
2873                                        ;*** A014 INTERRRUPT HANDLER ***
2874 064022                          INTR14:
2875 064022  010046                        MOV     R0,-(SP)                ;SAVE R0
2876 064024  013737  002636  003446        MOV     K,ZLOOP
     064032  005037  003444          65$:  CLR     YLOOP                   ;WAIT
     064036  005237  003444          64$:  INC     YLOOP
     064042  023727  003444  000001        CMP     YLOOP,#1
     064050  001372                        BNE     64$
     064052  005337  003446                DEC     ZLOOP
     064056  001365                        BNE     65$
2877 064060  117700  116562                MOVB    @IAR,R0                 ;GET ADDRESS OF MODULE
```

```
2878 064064  042700  177400                    BIC     #177400,R0      ;CLEAR HIGH BYTE
2879 064070  063700  002660                    ADD     BASE,R0         ;MAKE D-BUS ADDRESS
2880 064074  020037  003462                    CMP     R0,TADDR        ;A014 ADDRESS ?
2881 064100  001455                            BEQ     2$              ;YES,BRANCH
2882 064102  104401  064110                    TYPE    ,67$            ;;TYPE ASCIZ STRING
     064106  000411                            BR      66$             ;;GET OVER THE ASCIZ
                                    ;;67$:    .ASCIZ  <CR><LF>/A014 RAMP TEST/
     064132                         66$:
2883 064132  104401  064140                    TYPE    ,69$            ;;TYPE ASCIZ STRING
     064136  000424                            BR      68$             ;;GET OVER THE ASCIZ
                                    ;;69$:    .ASCIZ  <CR><LF>/UNEXPECTED INTERRUPT FROM MODULE AT /
     064210                         68$:
2884 064210  010046                            MOV     R0,-(SP)        ;ADDRESS
2885 064212  104402                            TYPOC                   ;TYPE IT
2886 064214  104401  001203                    TYPE    ,$CRLF
2887 064220  004737  015410                    JSR     PC,CLRINT       ;TRY TO CLEAR IT
2888 064224  012600                            MOV     (SP)+,R0        ;RESTORE R0
2889 064226  022626                            CMP     (SP)+,(SP)+     ;FIXED STACK
2890 064230  000137  063244                    JMP     RUMP            ;TRY AGAIN
2891 064234  152777  000001  116402  2$:       BISB    #RBIT,@CSR      ;CLEAR INTERRUPT
2892 064242  105777  117344                    TSTB    @STAT2          ;THIS MODULE
2893 064246  013737  002636  003446            MOV     K,ZLOOP
     064254  005037  003444          71$:      CLR     YLOOP           ;WAIT
     064260  005237  003444          70$:      INC     YLOOP
     064264  023727  003444  000001            CMP     YLOOP,#1
     064272  001372                            BNE     70$
     064274  005337  003446                    DEC     ZLOOP
     064300  001365                            BNE     71$
2894 064302  005237  003442                    INC     IFLAG           ;SET INTERRUPT FLAG
2895 064306  012600                  1$:       MOV     (SP)+,R0        ;RESTORE R0
2896 064310  000002                            RTI
```

```
                                        .SBTTL  LOGIC TEST OF A156 AND A157
                                        MUX1:
                                        ;;*****************************************************************
064312  000004                  TST30:  SCOPE
064314  012737  000030  001212          MOV     #30,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
064322  012737  000156  003456          MOV     #156,$MUT
064330  013737  003610  003466          MOV     STAT1,TBADDR
064336  105077  117250                  CLRB    @STAT2
064342  005037  001142                  CLR     $BDDAT
064346  013746  000004                  MOV     ERRVEC,-(SP)    ;;PUSH ERRVEC ON STACK
064352  012737  065176  000004          MOV     #30$,ERRVEC
064360  113777  003450  117222          MOVB    MXNUM,@STAT1            ;SET MUX #
064366  013737  003450  001140          MOV     MXNUM,$GDDAT
064374  117737  117210  001142          MOVB    @STAT1,$BDDAT           ;CHECK IF MUX RESPONDS
064402  123737  001140  001142          CMPB    $GDDAT,$BDDAT
064410  001403                          BEQ     1$
064412  104060                          EMT     60
064414  000137  065164                  JMP     13$
064420  113777  003450  117162  1$:     MOVB    MXNUM,@STAT1
064426  004737  017254                  JSR     PC,GCODE
064432  122737  000342  003606          CMPB    #342,GBITE              ;SE GENERIC CODE?
064440  001455                          BEQ     2$
064442  122737  000322  003606          CMPB    #322,GBITE              ;DIF MODE A156
064450  001412                          BEQ     20$
064452  122737  000323  003606          CMPB    #323,GBITE              ;A157?
064460  001406                          BEQ     20$                     ;YES, DIFFERENTIAL MODE
064462  013737  003606  003600          MOV     GBITE,TEMP3
064470  104061                          EMT     61
064472  000137  065164                  JMP     13$
064476  012705  000017          20$:    MOV     #17,R5
064502  005004                          CLR     R4
064504  012737  000040  001140          MOV     #40,$GDDAT
064512  113777  003450  117070          MOVB    MXNUM,@STAT1
064520  113777  001140  117064  43$:    MOVB    $GDDAT,@STAT2           ;SET TOO HIGH CHANNEL
064526  132777  000004  117056          BITB    #BIT2,@STAT2            ;CHECK IF ERROR SET
064534  001001                          BNE     42$
064536  104062                          EMT     62
064540  004737  015526          42$:    JSR     PC,CNTRC
064544  152777  000001  116072          BISB    #RBIT,@CSR              ;CLEAR ERROR BIT
064552  105777  117032                  TSTB    @STAT1
064556  005237  001140                  INC     $GDDAT
064562  005204                          INC     R4
064564  022704  000020                  CMP     #20,R4
064570  001353                          BNE     43$
064572  000402                          BR      21$
064574  012705  000037          2$:     MOV     #37,R5
064600  010537  001140          21$:    MOV     R5,$GDDAT               ;CHECK ALL CHANNELS
064604  053737  003450  001140          BIS     MXNUM,$GDDAT
064612  113777  001140  116770  4$:     MOVB    $GDDAT,@STAT1
064620  117737  116764  001142          MOVB    @STAT1,$BDDAT
064626  123737  001140  001142          CMPB    $GDDAT,$BDDAT
064634  001403                          BEQ     3$
064636  104063                          EMT     63
064640  004737  015526                  JSR     PC,CNTRC
064644  005337  001140          3$:     DEC     $GDDAT
```

```
2953 064650  023737  003450  001140          CMP     MXNUM,$GDDAT
2954 064656  001355                          BNE     4$
2955
2956 064660  005000                          CLR     R0
2957 064662  152777  000001  115754          BISB    #RBIT,@CSR
2958 064670  005037  001140                  CLR     $GDDAT
2959 064674  105777  116712                  TSTB    @STAT2                  ;RIF MODULE
2960 064700  117737  116706  001142          MOVB    @STAT2,$BDDAT           ;CHECK IF ERROR,GAIN & DONE CLEAR
2961 064706  105737  001142                  TSTB    $BDDAT
2962 064712  001401                          BEQ     5$
2963 064714  104064                          EMT     64
2964 064716  152777  000360  116666  5$:     BISB    #360,@STAT2             ;SET WRONG GAIN
2965 064724  132777  000004  116660          BITB    #BIT2,@STAT2
2966 064732  001001                          BNE     6$
2967 064734  104066                          EMT     66
2968 064736  152777  000001  115700  6$:     BISB    #RBIT,@CSR
2969 064744  105777  116642                  TSTB    @STAT2                  ;CLEAR ERROR
2970 064750  001001                          BNE     7$
2971 064752  104067                          EMT     67
2972 064754  005001                  7$:     CLR     R1
2973 064756  156177  003352  116626  22$:    BISB    GAIN(R1),@STAT2
2974 064764  152777  000010  115652          BISB    #TBIT,@CSR
2975 064772  113777  003450  116610          MOVB    MXNUM,@STAT1
2976 065000  152777  000001  116604          BISB    #BIT0,@STAT2            ;START CONVERSION
2977 065006  005200                  9$:     INC     R0
2978 065010  001002                          BNE     8$
2979 065012  104070                          EMT     70
2980 065014  000463                          BR      13$
2981 065016  132777  000002  116566  8$:     BITB    #BIT1,@STAT2            ;WAIT FOR CONV DONE
2982 065024  001770                          BEQ     9$
2983 065026  132777  000004  116556          BITB    #BIT2,@STAT2
2984 065034  001002                          BNE     10$
2985 065036  000452                          BR      13$
2986 065040  104071                          EMT     71
2987 065042  012737  003774  001140  10$:    MOV     #3774,$GDDAT
2988 065050  117737  116540  001142          MOVB    @LBYTE,$BDDAT
2989 065056  117737  116534  001143          MOVB    @HBYTE,$BDDAT+1
2990 065064  042737  170000  001142          BIC     #170000,$BDDAT          ;GET CONVERSION RESULTS
2991 065072  023737  001140  001142          CMP     $GDDAT,$BDDAT
2992 065100  003401                          BLE     11$
2993 065102  104072                          EMT     72
2994 065104  012737  004004  001140  11$:    MOV     #4004,$GDDAT
2995 065112  023737  001140  001142          CMP     $GDDAT,$BDDAT
2996 065120  002001                          BGE     12$
2997 065122  104072                          EMT     72
2998 065124  117737  116460  001142  12$:    MOVB    @STAT1,$BDDAT
2999 065132  001403                          BEQ     23$
3000 065134  005037  001140                  CLR     $GDDAT
3001 065140  104073                          EMT     73
3002 065142  122737  000323  003606  23$:    CMPB    #323,GBITE
3003 065150  001005                          BNE     13$
3004 065152  062701  000002                  ADD     #2,R1
3005 065156  005761  003352                  TST     GAIN(R1)
3006 065162  001275                          BNE     22$
3007 065164  004737  014310          13$:    JSR     PC,KLEER
3008 065170  012637  000004                  MOV     (SP)+,ERRVEC    ;;POP STACK INTO ERRVEC
3009 065174  000207                          RTS     PC
```

```
3010 065176  022626          30$:    CMP     (SP)+,(SP)+
3011 065200  104065                  EMT     65
3012 065202  000770                  BR      '3$
```

```
                                .SBTTL          DIGITAL MODULE TEST M6015
                                                ;THIS TEST CHECKS M6015 MODULE
                                                ;16-BIT RETENTIVE ISOLATED DC OUTPUT

065204                          M6015:
                                ;;*****************************************************************
065204  000004                TST31:  SCOPE
065206  012737  000031  001212        MOV     #31,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
065214  152777  000020  115422        BISB    #DBIT,@CSR
065222  013700  003462                MOV     TADDR,R0
065226  105020                        CLRB    (R0)+
065230  105010                        CLRB    (R0)                    ;CLEAR TWO BYTES
065232  005000                        CLR     R0
065234  005001                        CLR     R1
065236  005000                        CLR     R0
065240  005001                4$:     CLR     R1
065242  005037  003452                CLR     DBUFF           ;CLEAR IMAGE OF I/O REGISRERS
065246  005037  003454                CLR     DBUFF+2
065252  116160  002630  003452  3$:   MOVB    PATT(R1),DBUFF(R0);SET DATA PATTERN IN IMIGE
065260  013737  003462  003502        MOV     TADDR,TEMP
065266  060037  003502                ADD     R0,TEMP
065272  116177  002630  116202        MOVB    PATT(R1),@TEMP  ;SET DATA IN I/O REG
065300  005002                        CLR     R2
065302  013737  003462  003510  2$:   MOV     TADDR,TEMP1
065310  060237  003510                ADD     R2,TEMP1
065314  117737  116170  001142        MOVB    @TEMP1,$BDDAT   ;READ I/O REGISTER
065322  123762  001142  003452        CMPB    $BDDAT,DBUFF(R2);IS DATA OK?
065330  001404                        BEQ     1$
065332  116237  003452  001140        MOVB    DBUFF(R2),$GDDAT
065340  104017                        EMT     17
065342  005202                1$:     INC     R2
065344  022702  000002                CMP     #2,R2
065350  001354                        BNE     2$              ;TEST IF OTHER BYTES ARE OK
065352  005201                        INC     R1
065354  022701  000004                CMP     #4,R1           ;LAST PATTERN?
065360  001334                        BNE     3$              ;TEST NEXT DATA PATTERN
065362  005200                        INC     R0
065364  022700  000002                CMP     #2,R0           ;LAST BYTE?
065370  001323                        BNE     4$              ;NO,TEST NEXT BYTE
065372  013700  003462                MOV     TADDR,R0                ;REGISTER ADDRESS
065376  012737  177777  001140        MOV     #-1,$GDDAT              ;ALL 1'S
065404  012737  000002  003460        MOV     #2,BYTNUM               ;WRITE 2 BYTES
065412  004737  015166                JSR     PC,SETBYT               ;GO WRITE IT
065416  005037  001140                CLR     $GDDAT                  ;CLEAR GDDAT
065422  004737  014310                JSR     PC,KLEER                ;SET C BIT
065426  112037  001142                MOVB    (R0)+,$BDDAT            ;LOW BYTE
065432  111037  001142                MOVB    (R0),$BDDAT             ;HI BYTE
065436  023737  001140  001142        CMP     $GDDAT,$BDDAT           ;ALL 0'S READ ?
065444  001401                        BEQ     5$                      ;YES,BRANCH
065446  104075                        EMT     75                      ;C-BIT BROKE
065450                          5$:
065450  000207                        RTS     PC
```

```
                                        .SBTTL  SUBROUTINES
                                
                                
                                
                                
                                
065452  022737  000321  003606  SETRAM: CMP     #321,GBITE
065460  001004                          BNE     1$
065462  112777  000004  116120          MOVB    #4,@STAT1
065470  000403                          BR      2$
065472  112777  000002  116110  1$:     MOVB    #2,@STAT1
065500  000207                  2$:     RTS     PC
                                
                                        ;       THIS SUBROUTINE DOES 8 CONVERSIONS AND STORES AVERAGE IN INTDAT
065502                          CONV7:
065502  117737  116102  003620  3$:     MOVB    @STAT1,ST1SAV           ;SAVE STAT1
065510  117737  116076  003622          MOVB    @STAT2,ST2SAV           ;SAVE STAT2
065516  005037  003512                  CLR     INTDAT
065522  005037  003604                  CLR     TEMP5
065526  152777  000001  116056  2$:     BISB    #BIT0,@STAT2    ;START CONVERSION
065534  132777  000002  116050  1$:     BITB    #BIT1,@STAT2    ;WAIT FOR CONVERSION DONE
065542  001774                          BEQ     1$
065544  152777  000001  115072          BISB    #RBIT,@CSR
065552  117737  116036  003502          MOVB    @LBYTE,TEMP     ;ASSEMBLE DATA
065560  117737  116032  003503          MOVB    @HBYTE,TEMP+1
065566  042737  170000  003502          BIC     #170000,TEMP
065574  063737  003502  003512          ADD     TEMP,INTDAT     ;ADD TO PREVIOUS DATA
065602  005237  003604                  INC     TEMP5
065606  113777  003620  115774          MOVB    ST1SAV,@STAT1
065614  113777  003622  115770          MOVB    ST2SAV,@STAT2
065622  022737  000010  003604          CMP     #10,TEMP5       ;IS IT LAST ONE
065630  001336                          BNE     2$              ;NO
065632  013737  003512  003654          MOV     INTDAT,AVRSAV   ;SAVE TOTAL
065640  006237  003512                  ASR     INTDAT          ;FIND AVERAGE
065644  006237  003512                  ASR     INTDAT
065650  006237  003512                  ASR     INTDAT
065654  042737  170000  003512          BIC     #170000,INTDAT
065662  013737  003512  001142          MOV     INTDAT,$BDDAT
065670  000207                          RTS     PC
```

```
                                        .SBTTL  PARTB MODULE

                                ;*******************************************************************************
                                ;PARTB OF CVPCAG SYSTEM TEST
                                ;CREATED 31 MAY 1979
                                ;BECAUSE OF MAXIMUM MEMORY SIZE LIMITS-16K ON IP300 SYSTEMS, CVPCAG  IS  DIVIDED
                                ;INTO 3 SECTIONS:
                                ;       MONITOR-ERROR HANDLING,MAPPING,COMMANDS
                                ;       PARTA-ALL PRESENT MODULES
                                ;       PARTB-ALL FUTURE MODULES
                                ;*******************************************************************************


                                        .SBTTL A631 MONITOR AND TEST

065672  004737  071460        A631:   JSR     PC,A631AD       ;SET UP ADDRESSES
065676  005737  002664                TST     AFLAG           ;UNDER AUTO MODE?
065702  001405                        BEQ     A631MN          ;NO,BRANCH
065704  004737  014310                JSR     PC,KLEER        ;CLEAR D-BUS
065710  004737  066636                JSR     PC,A631L                ;DO LOGIC TESTS
065714  000207                        RTS     PC              ;RETURN

065716  012706  001100        A631MN: MOV     #1100,SP        ;SET STACK
065722                                TYPTXT <<CR><LF>/A631 12-BIT DAC MONITOR  (H-HELP)/<CR><LF>>
065776  012737  065716  003652        MOV     #A631MN,RETURN  ;ADDRESS FOR CONTROL 'A'

066004                        1$:     TYPTXT </A631:/>
066020  104406                        RDCHR                   ;GET COMMAND
066022  012637  066234                MOV     (SP)+,200$      ;SAVE COMMAND
066026  022737  000003  066234        CMP     #3,200$         ;CONTROL "C" ?
066034  001002                        BNE     3$              ;NO,BRANCH
066036  000137  007240                JMP     MONIT           ;RETURN TO MONITOR
066042  005000                3$:     CLR     R0              ;CLEAR R0  USED AS INDEX
                                ;*******  FIND COMMAND  *******
066044  026037  066236  066234  4$:     CMP     300$(R0),200$   ;A MATCH ?
066052  001423                        BEQ     5$              ;YES, BRANCH
066054  062700  000002                ADD     #2,R0           ;ADVANCE POINTER
066060  005760  066236                TST     300$(R0)        ;END OF TABLE ?
066064  001367                        BNE     4$              ;NO, BRANCH
066066                                TYPTXT  <<15><12> /ILLEGAL COMMAND/<7>>
066120  000731                        BR      1$              ;GO TRY AGAIN

                                ;*******  EXECUTE COMMAND  *******
066122  016037  066264  066132  5$:     MOV     500$(R0),6$     ;GET READY TO TYPE
066130  104401                        TYPE                    ;THE COMMAND
066132  000000                6$:     0                       ;THE MESSAGE TO TYPE
066134  104401  001203                TYPE    ,$CRLF          ;TYPE A CR THEN LF
066140  005037  001214                CLR     $PASS           ;CLEAR PASS COUNT
066144  010037  066132        7$:     MOV     R0,6$           ;SAVE INDEX FOR LOOP ON TEST
066150  004770  066252        9$:     JSR     PC,@400$(R0)    ;GO DO THE TEST
066154  005237  001214                INC     $PASS           ;INCREMENT PASS COUNT
066160  013700  066132                MOV     6$,R0           ;RESTORE INDEX
```

```
     62 066164  023737  001214  002662          CMP     $PASS,PASCNT    ;REQUESTED NUMBER OF PASSES DONE ?
     63 066172  002764                          BLT     7$              ;NO,BRANCH
     64 066174  104401  001203                  TYPE    ,$CRLF
     65 066200  013746  002662                  MOV     PASCNT,-(SP)    ;SAVE IT FOR PRINT
     66 066204  104402                          TYPOC
     67 066206  104401  030040                  TYPE    ,M11            ;END PASS MESSAGE
     68 066212  104401  001203                  TYPE    ,$CRLF
     69 066216  005037  001214                  CLR     $PASS
     70 066222  032777  040000  112724          BIT     #BIT14,@SWR     ;LOOP ON TEST ?
     71 066230  001347                          BNE     9$              ;YES,BRANCH
     72 066232  000664                          BR      1$              ;GET ANOTHER COMMAND
     73
     74
     75                                         ;******* VARIABLE FOR ECHOING ******
     76 066234  000000                  200$:   .WORD   0               ;
```

```
                                                .NLIST BEX
                                                ;******* COMMAND TABLE *******
066236  000110                        300$:     .WORD   'H              ;HELP FILE
066240  000114                                  .WORD   'L              ;DAC LOGIC TEST
066242  000123                                  .WORD   'S              ;DAC-SPOPE ROUTINE
066244  000103                                  .WORD   'C              ;CALIBRATION ROUTINE
066246  000115                                  .WORD   'M              ;MANUAL FIELD OPERATION
066250  000000                                  .WORD   0               ;END OF 300$ TABLE

                                                ;******* TEST POINTERS *******
066252  066446                        400$:     .WORD   A631H           ;HELP FILE
066254  066636                                  .WORD   A631L           ;DAC LOGIC TEST
066256  067766                                  .WORD   A631S           ;DAC-SCOPE
066260  071526                                  .WORD   A631C           ;CALIBRATION ROUTINE
066262  070600                                  .WORD   A631M           ;MANUAL FIELD OPERATION

                                                ;******* HEADER POINTERS *******
066264  066300                        500$:     .WORD   600$            ;HELP FILE
066266  066315                                  .WORD   606$            ;DAC LOGIC TEST
066270  066337                                  .WORD   617$            ;DAC-SCOPE ROUTINE
066272  066364                                  .WORD   609$            ;CALIBRATION ROUTINE
066274  066413                                  .WORD   616$            ;MANUAL FIELD OPERATION
066276  000000                                  .WORD   0               ;END OF TABLE 500$ MARKER

                                                ;******* HEADER MESSAGES *******
066300      040     110     105       600$:     .ASCIZ  / HELP FILE/<15><12>
066315      040     104     101       606$:     .ASCIZ  / DAC LOGIC TEST/<15><12>
066337      040     104     101       617$:     .ASCIZ  / DAC-SCOPE ROUTINE/<15><12>
066364      040     103     101       609$:     .ASCIZ  / CALIBRATION ROUTINE/<15><12>
066413      040     115     101       616$:     .ASCIZ  / MANUAL FIELD OPERATION/<15><12>
                                                .EVEN
                                                ;******* THE HELP PRINTER *******
066446  104401  066460                A631H:    TYPE    ,1$     ;TYPE THE HELP FILE
066452  005726                                  TST     (SP)+
066454  000137  065716                          JMP     A631MN
066460      050     114     051       1$:       .ASCII  /(L)-DAC LOGIC TEST/<CR><LF>
066504      050     115     051                 .ASCII  /(M)-MANUAL OPERATION/<CR><LF>
066532      050     123     051                 .ASCII  /(S)-SCOPE ROUTINE/<CR><LF>
066555      050     103     051                 .ASCIZ  /(C)-CALIBRATION PROCEDURE/<CR><LF>
                                                .EVEN
```

```
                                        .SBTTL DAC LOGIC TEST
                                ;*******************************************************************************
                                ;FUNCTION:
                                ;       DAC12--READ TIMEOUT TEST-- EACH OF THE 8 ADDRESSES ARE READ TO CHECK FOR
                                ;       NO TIMEOUT.
                                ;VARIABLE USAGE:
                                ;       $BDADR-CONTAINS THE AN ADDRESS
                                ;       BTEMP-TEMPORARY
                                ;       CTEMP-TEMPORARY
                                ;       AD0-CONTAINS THE FIRST ADDRESS OF THE MODULE.
                                ;       AD7-CONTAINS THE LAST ADDRESS OF THE MODULE.
                                ;*******************************************************************************

                                        .REM %
                                        DAC12 READ TIMEOUT TEST
                                        :  SET UP TIMEOUT
                                210$:   :  FOR $BDADR=AD0 TO AD7 DO
                                        :  :  READ [$BDADR]
                                        :  END (FOR)
                                        END (DAC12)
                                        DAC12 - TIMOUT HANDLER
                                        %

066612  012637  003412          TMOT12: MOV     (SP)+,BTEMP     ;SAVE PC
066616  012637  003414                  MOV     (SP)+,CTEMP     ;SAVE PSW
066622  104214                          EMT     214
066624  013746  003414                  MOV     CTEMP,-(SP)     ;RESTORE PSW
066630  013746  003412                  MOV     BTEMP,-(SP)     ;RESTORE PC
066634  000002                          RTI

066636                          A631L:
066636                          DAC12:
                                ;;*******************************************************************
066636  000004                  TST32:  SCOPE
066640  012737  000032  001212          MOV     #32,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
066646  152777  000020  113770          BISB    #DBIT,@CSR      ;SET DBIT
066654  004737  015526                  JSR     PC,CNTRC        ;EXIT IF CONTROL C
066660  012737  066612  000004          MOV     #TMOT12,ERRVEC  ;SET UP TIMEOUT VECTOR
066666  012737  000340  000006          MOV     #340,ERRVEC+2   ;
066674  013737  003416  001136          MOV     AD0,$BDADR      ;SET UP FOR LOOP
066702  005337  001136                  DEC     $BDADR          ;
066706  005237  001136          211$:   INC     $BDADR          ;ADVANCE ADDRESS
066712  023737  003434  001136          CMP     AD7,$BDADR      ;DONE LOOP ?
066720  103455                          BLO     DAC13           ;YES, BRANCH
066722  004737  015526                  JSR     PC,CNTRC        ;EXIT IF CONTROL C
066726  105777  112204                  TSTB    @$BDADR         ;READ THE ADDRESS
066732  000765                          BR      211$            ;END (FOR)
```

```
                        ;*****************************************************************************
                        ;FUNCTION:
                        ;       DAC13--WRITE TIMEOUT TEST-- EACH OF THE 8 ADDRESSES ARE WRITTEN TO CHECK
                        ;       FOR NO TIMEOUT. NO DATA CHECKING IS DONE.
                        ;VARIABLE USAGE:
                        ;       $BDADR-CONTAINS THE AN ADDRESS
                        ;       BTEMP-TEMPORARY
                        ;       CTEMP-TEMPORARY
                        ;       AD0-CONTAINS THE FIRST ADDRESS OF THE MODULE.
                        ;       AD7-CONTAINS THE LAST ADDRESS OF THE MODULE.
                        ;*****************************************************************************

                                .REM %
                                DAC13 WRITE TIMEOUT TEST
                                :  SET UP TIMEOUT
                        210$:   ;  FOR $BDADR=AD0 TO AD7 DO
                                :  :  WRITE [$BDADR]
                                :  END (FOR)
                                END (DAC13)
                                DAC13 - TIMOUT HANDLER
                                %
066734  101     066     063     EM215: .ASCIZ  /A631 ERROR -TIMEOUT WHEN WRITING ADDRESS/
                        ;DH67:  .ASCIZ  /ERRPC  TST#    ADDRESS PASCNT/
                        ;DT67:  .WORD $ERRPC, $TESTN, $BDADR, $PASS,
067006  000000  000000  000000  DF1:    .WORD 0,0,0,0,0,0,0,0,

067030  012637  003412          TMOT13: MOV     (SP)+,BTEMP     ;SAVE PC
067034  012637  003414                  MOV     (SP)+,CTEMP     ;SAVE PSW
067040  104215                  EMT     215
067042  013746  003414                  MOV     CTEMP,-(SP)     ;RESTORE PSW
067046  013746  003412                  MOV     BTEMP,-(SP)     ;RESTORE PC
067052  000002                  RTI

067054                          DAC13:
                                ;;*************************************************************
067054  000004                TST33:  SCOPE
067056  012737  000033  001212          MOV     #33,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
067064  152777  000020  113552          BISB    #DBIT,@CSR      ;SET DBIT
067072  004737  015526                  JSR     PC,CNTRC        ;EXIT IF CONTROL C
067076  012737  067030  000004          MOV     #TMOT13,ERRVEC  ;SET UP TIMEOUT VECTOR
067104  012737  000340  000006          MOV     #340,ERRVEC+2   ;
067112  013737  003416  001136          MOV     AD0,$BDADR      ;SET UP FOR LOOP
067120  005337  001136                  DEC     $BDADR          ;
067124  005237  001136        211$:     INC     $BDADR          ;ADVANCE ADDRESS
067130  023737  003434  001136          CMP     AD7,$BDADR      ;DONE LOOP ?
067136  103425                  BLO     DAC14           ;YES, BRANCH
067140  004737  015526                  JSR     PC,CNTRC        ;EXIT IF CONTROL C
067144  105077  111766                  CLRB    @$BDADR         ;WRITE THE ADDRESS
067150  000765                  BR      211$            ;END (FOR)
```

```
                        ;*****************************************************************************
                        ;FUNCTION:
                        ;       DAC14--RAM MEMORY TEST-- EACH OF THE 8 WORDS ARE TREATED AS ONE LONG  64
                        ;       BIT WORD. THE 64 BIT WORD IS SET = 1 THEN A BIT IS ROTATED THRU  THE  64
                        ;       BITS. THE 64 BIT WORD IS NOW SET = -1 THEN A 0 IS ROTATED  THRU  THE  64
                        ;       BITS. AFTER EACH ROTATION THE 64 BIT WORD IS CHECKED.
                        ;VARIABLE USAGE:
                        ;       X--TEMPORARY STORAGE (R0)
                        ;       M[X]--FOUR WORDS USED TO DUPLICATE THE RAM. (10$)
                        ;       COUNT--CONTAINS THE LOOP COUNT. (R1)
                        ;       KEY--DETERMINES WHEN TO COMPLEMENT DATA. (R2)
                        ;       AD0-AD7--CONTAIN THE MODULE ADDRESSES.
                        ;*****************************************************************************


                                .REM %
                                DAC14 - RAM MEMORY TEST
                                :  CLEAR FOUR WORDS OF M[X]
                                :  M[0]=1
                                :  KEY=0
                        210$:   ;  REPEAT
                                :  :  COUNT=0
                        220$:   ;  :  REPEAT
                                :  :  :  [AD0]=M[0]
                                :  :  :  [AD1]=M[1]
                                :  :  :  [AD2]=M[2]
                                :  :  :  [AD3]=M[3]
                                :  :  :  [AD4]=M[4]
                                :  :  :  [AD5]=M[5]
                                :  :  :  [AD6]=M[6]
                                :  :  :  [AD7]=M[7]
                        230$:   ;  :  :  FOR X=0 TO 7 DO
                                :  :  :  :  IF [AD0+X]<>M[X] THEN REPORT ERROR
                                :  :  :  END (FOR)
                        240$:   ;  :  :  ROTATE 64. BIT NUMBER (M) BY ONE
                                :  :  :  COUNT=COUNT+1
                                :  :  UNTIL (COUNT=64.)
                                :  :  KEY=KEY+1
                                :  :  COMPLIMENT 64. BIT WORD
                        250$:   ;  UNTIL (KEY=2)
                                END (DAC14 - RAM MEMORY TEST)
                                %
067152  101     066     063     EM216: .ASCIZ  /A631 ERROR - RAM MEMORY FAILURE/
                        ; DH71: .ASCIZ  /ERRPC  TST#    BDADR   GDDAT   BDDAT   PASS/
                        ; DT71: .WORD $ERRPC, $TESTN, $BDADR, $GDDAT, $BDDAT, $PASS,
                        ;DF1    .WORD 0,0,0,0,0,0,0,0,
```

```
                                                ;DAC14 - RAM MEMORY TEST
067212                                  DAC14:
                                        ;;*****************************************************************
067212  000004                          TST34:  SCOPE
067214  012737  000034  001212                  MOV     #34,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
067222  152777  000020  113414                  BISB    #DBIT,@CSR      ;SET DBIT
067230  012737  076146  000004                  MOV     #TIMOUT,ERRVEC  ;RESET TIMEOUT VECTOR
067236  005037  067542                          CLR     10$             ;-CLEAR 4 WORDS
067242  005037  067544                          CLR     10$+2           ;
067246  005037  067546                          CLR     10$+4           ;
067252  005037  067550                          CLR     10$+6
067256  012737  000001  067542                  MOV     #1,10$          ;-M[0]=1
067264  005002                                  CLR     R2              ;-KEY=0
067266  005001                          210$:   CLR     R1              ;--COUNT=0

067270  004737  015526                  220$:   JSR     PC,CNTRC        ;EXIT IF CONTROL C
067274  113777  067542  114114                  MOVB    10$,@AD0        ;---[AD0]=M[0]
067302  113777  067543  114110                  MOVB    10$+1,@AD1      ;---[AD1]=M[1]
067310  113777  067544  114104                  MOVB    10$+2,@AD2      ;---[AD2]=M[2]
067316  113777  067545  114100                  MOVB    10$+3,@AD3      ;---[AD3]=M[3]
067324  113777  067546  114074                  MOVB    10$+4,@AD4      ;---[AD4]=M[4]
067332  113777  067547  114070                  MOVB    10$+5,@AD5      ;---[AD5]=M[5]
067340  113777  067550  114064                  MOVB    10$+6,@AD6      ;---[AD6]=M[6]
067346  113777  067551  114060                  MOVB    10$+7,@AD7      ;---[AD7]=M[7]

067354  012700  177777                          MOV     #-1,R0          ;---FOR X=0 TO 7 DO
067360  005200                          231$:   INC     R0              ;
067362  022700  000007                          CMP     #7,R0           ;END OF LOOP ?
067366  103426                                  BLO     240$            ;YES, BRANCH
067370  004737  015526                          JSR     PC,CNTRC        ;EXIT IF CONTROL C
067374  013737  003416  003410                  MOV     AD0,ATEMP       ;-----IF [AD0+X]<>M[X] THEN REPORT ERROR
067402  060037  003410                          ADD     R0,ATEMP        ;
067406  117737  113776  001142                  MOVB    @ATEMP,$BDDAT   ;SAVE DATA
067414  123760  001142  067542                  CMPB    $BDDAT,10$(R0)  ;CORRECT DATA ?
067422  001756                                  BEQ     231$            ;YES, BRANCH
067424  113737  003410  001136                  MOVB    ATEMP,$BDADR    ;
067432  116037  067542  001140                  MOVB    10$(R0),$GDDAT  ;
067440  104216                                  EMT     216
067442  000746                                  BR      231$            ;---END (FOR)

067444  000241                          240$:   CLC                     ;---ROTATE NUMBER BY ONE
067446  006137  067542                          ROL     10$             ;
067452  006137  067544                          ROL     10$+2           ;
067456  006137  067546                          ROL     10$+4           ;
067462  006137  067550                          ROL     10$+6           ;
067466  103003                                  BCC     241$            ;WRAP AROUND BIT ?
067470  052737  000001  067542                  BIS     #BIT0,10$       ;YES, DO IT
067476  005201                          241$:   INC     R1              ;---COUNT=COUNT+1
067500  022701  000100                          CMP     #64.,R1         ;COUNT=64. ?
067504  001271                                  BNE     220$            ;NO, BRANCH
067506  005202                                  INC     R2              ;--KEY=KEY+1

067510  005137  067542                          COM     10$             ;--COMPLIMENT 64. BIT WORD
067514  005137  067544                          COM     10$+2           ;
067520  005137  067546                          COM     10$+4           ;
067524  005137  067550                          COM     10$+6           ;
067530  022702  000002                          CMP     #2,R2           ;KEY=2 ?
```

```
    324 067534  001454                        BNE     210$            ;NO, BRANCH
    325 067536  000137  057616                JMP     DAC15           ;DO NEXT TEST
    326
    327 067542                          10$:  .BLKW 4                 ;'M'' THE 64 BIT WORD
```

```
                        ;*****************************************************************************
                        ;FUNCTION:
                        ;       DAC15--GENERIC CODE TEST-- EACH OF THE 8 ADDRESS ARE CHECKED TO  CONTAIN
                        ;       THE CORRECT GENERIC CODE WHICH IS 262.
                        ;VARIABLE USAGE:
                        ;       $BDADR-CONTAINS THE ADDRESS
                        ;       $BDDAT-CONTAINS THE GENERIC CODE READ.
                        ;       AD0-CONTAINS THE FIRST ADDRESS OF THE MODULE.
                        ;       AD7-CONTAINS THE LAST ADDRESS OF THE MODULE.
                        ;*****************************************************************************

                                .REM %
                                DAC15 - GENERIC CODE TEST
                                :  SET GBIT
                        211$:   ;  FOR $BDADR=AD0 TO AD7 DO
                                :  $BDDAT=[$BDADR]
                                :  IF $BDDAT<>#262 THEN REPORT ERROR
                                :  END (FOR)
                                :  CLR GBIT
                                END (DAC15 - GENERIC CODE TEST)
                                %
 067552     101     066     063   EM217: .ASCIZ  /A631 ERROR - GENERIC CODE INCORRECT/
                        ;DH71:  .ASCIZ  /ERRPC  TST#    BDADR   GDDAT   BDDAT   PASS/
                        ;DT71:  .WORD $ERRPC, $TESTN, $BDADR, $GDDAT, $BDDAT, $PASS,
                        ;DF1    .WORD 0,0,0,0,0,0,0,0,

 067616                         DAC15:
                        ;*****************************************************************
 067616  000004                 TST35:  SCOPE
 067620  012737  000035  001212         MOV     #35,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
 067626  152777  000020  113010         BISB    #DBIT,@CSR      ;SET DBIT
 067634  004737  015526               JSR     PC,CNTRC        ;EXIT IF CONTROL C
 067640  112777  000002  112776         MOVB    #CBIT,@CSR      ;CLEAR THE SUBSYSTEM
 067646  012737  001000  003410         MOV     #1000,ATEMP     ;SET UP COUNT
 067654  005337  003410         201$:   DEC     ATEMP           ;
 067660  001375                       BNE     201$            ;
 067662  152777  000004  112754         BISB    #GBIT,@CSR      ;SET THE GBIT
 067670  013737  003416  001136         MOV     AD0,$BDADR      ;SET UP FOR
 067676  005337  001136               DEC     $BDADR          ;
 067702  005037  001142               CLR     $BDDAT
 067706  005237  001136         211$:   INC     $BDADR          ;MAKE NEXT ADDRESS
 067712  023737  003434  001136         CMP     AD7,$BDADR      ;END OF LOOP ?
 067720  103416                       BLO     220$            ;YES, BRANCH
 067722  004737  015526               JSR     PC,CNTRC        ;EXIT IF CONTROL C
 067726  117737  111204  001142         MOVB    @$BDADR,$BDDAT  ;SAVE THE CODE
 067734  122737  000262  001142         CMPB    #262,$BDDAT     ;CORRECT CODE ?
 067742  001761                       BEQ     211$            ;YES, BRANCH
 067744  012737  000262  001140         MOV     #262,$GDDAT     ;
 067752  104217                       EMT     217
 067754  000754                       BR      211$            ;END (FOR)
 067756  142777  000004  112660 220$:   BICB    #GBIT,@CSR      ;CLEAR THE GBIT
 067764  000207               221$:   RTS     PC
```

```
                        .SBTTL  A631 DAC-SCOPE
                ;*******************************************************************************
                ;FUNCTION:
                ;       THE OUTPUTS OF THE DAC'S ARE TO BE OBSERVED USING A SCOPE. EACH DAC  IS
                ;       CONTINUALLY LOADED. THE VALUES THAT ARE LOADED ARE  SET  UP  DURING  THE
                ;       INITIAL DIALOG. THE DAC(S) MAY BE SETUP BY ANSWERING THE DAC,  LOWLIMIT,
                ;       HIGHLIMIT, STEP,  QUESTIONS. IF YOU DESIRE NOT TO  CHANGE  A  DAC'S
                ;       VALUES SIMPLY TYPE CARRAIGE RETURN. AN ANSWER OF "YES" TO  THE  EXECUTE
                ;       QUESTION CAUSES ALL DACS TO BE LOADED  AND  RUN  THRU  THERE  RESPECTIVE
                ;       PARAMETERS CONTINUOUSLY. THE ONLY WAY TO EXIT THIS LOOP IS VIA CONTROL C
                ;       OR CONTROL A.;DAC OUTPUTS:
                ;       DAC0 VOLTAGE + BERG PIN 5,  TERMINAL STRIP 5
                ;                    - BERG PIN 6,  TERMINAL STRIP 6
                ;            CURRENT + BERG PIN X,  TERMINAL STRIP 7
                ;                    - BERG PIN X,  TERMIIAL STRIP 8
                ;
                ;       DAC1 VOLTAGE + BERG PIN 13, TERMINAL STRIP 13
                ;                    - BERG PIN 14, TERMINAL STRIP 14
                ;            CURRENT + BERG PIN X,  TERMINAL STRIP 15
                ;                    - BERG PIN X,  TERMIIAL STRIP 16
                ;
                ;       DAC2 VOLTAGE + BERG PIN 37, TERMINAL STRIP 21
                ;                    - BERG PIN 38, TERMINAL STRIP 22
                ;            CURRENT + BERG PIN X,  TERMINAL STRIP 23
                ;                    - BERG PIN X,  TERMIIAL STRIP 24
                ;
                ;       DAC3 VOLTAGE + BERG PIN 47, TERMINAL STRIP 31
                ;                    - BERG PIN 48, TERMINAL STRIP 32
                ;            CURRENT + BERG PIN X,  TERMINAL STRIP 33
                ;                    - BERG PIN X,  TERMIIAL STRIP 34
                ;
                ;VARIABLE USAGE:
                ;       EACH VARIABLE IS THE START OF A  TABLE.  THE  TABLE
                ;       ENTRIES CORRESPOND  TO  THE  APPROPRIATE  DAC.  THE
                ;       TABLE CONSISTS OF 4 WORDS. WORD 1-DAC 0...ETC.
                ;       SCLL--LOWER LIMIT. [0..3]
                ;       SCHL--HIGH LIMIT. [0..3]
                ;       SCSTP--THE STEP SIZE TO INCREMENT BY. [0..3]
                ;       SCCUR--THE CURRENT VALUE. [0..3]
                ;       R1--DAC#
                ;       R2-TEMPORARY
                ;*******************************************************************************
```

```
                        .REM %
A631S:                  ;DAC-SCOPE ROUTINE
                        :  R1=0
210$:                   ;  REPEAT
                        :  :  REPEAT
                        :  :  :  ASK DAC(R1/2) PARAMETERS[Y/N]?
                        :  :  :  IF (YES)
                        :  :  :  :  THEN (*THIS DAC*)
                        :  :  :  :  :  ASK LOWLIMIT, STORE IN SCLL[R1]
                        :  :  :  :  :  ASK HIGH LIMIT, STORE IN SCHL[R1]
                        :  :  :  :  :  ASK STEP, STORE IN SCSTP[R1]
220$:                   ;  :  :  R1=R1+2
                        :  :  UNTIL (R1>6)
230$:                   ;  :  R1=0
                        :  :  EXECUTE[Y/N] ?
                        ;  UNTIL (YES)
240$:                   ;  INITIAL COUNT (SCCUR=SCLL)
250$:                   ;  REPEAT
260$:                   ;  :  FOR R1=0 TO 6 BY 2 DO
                        :  :  :  SCCUR[R1]=SCCUR[R1]+SCSTP[R1]
                        :  :  :  IF SCCUR[R1]>SCHL[R1]
                        :  :  :  :  THEN (*ADJUST VALUE*)
                        :  :  :  :  :  SCCUR[R1]=SCLL[R1]
                        :  :  :  LOAD DAC[R1]
                        :  :  :  END (*FOR*)
270$:                   ;  UNTIL (^C) OR (^A)
                        END (*DAC-SCOPE*)
                        %
```

```
                                                ;-R1=0
                                                ;-REPEAT
                                                ;--REPEAT
                                                ;---ASK DAC(R1/2) PARAMETERS[Y/N]?
                                                ;---IF (YES)
                                                ;----THEN (*THIS DAC*)
                                                ;-----ASK LOWLIMIT, STORE IN SCLL[R1]
                                                ;-----ASK HIGH LIMIT, STORE IN SCHL[R1]
                                                ;-----ASK STEP, STORE IN SCSTP[R1]
067766  005001                  A631S:  CLR     R1
067770                          210$:   TYPTXT  <<15><12> /DAC/>
070004  010146                         MOV     R1,-(SP)        ;TYPE NUMBER
070006  006216                         ASR     (SP)            ;DIVIDE BY 2
070010  104403                         TYPOS                   ;ONE DIGIT
070012     001                         .BYTE   1               ;
070013     000                         .BYTE   0               ;SUPPRESS ZEROS
070014                                  TYPTXT  <% PARAMETERS[Y/N]:%>
070044  104406                         RDCHR                   ;GET RESPONSE
070046  122726  000131                 CMPB    #'Y,(SP)+       ;IS IT YES ?
070052  001067                         BNE     220$            ;NO, BRANCH
070054                                  TYPTXT </YES/>
070066  104401  001203                 TYPE    ,$CRLF
070072                                  TYPTXT  </LOWER LIMIT(OCTAL):/>
070124  104410                         RDOCT
070126  012661  070540                 MOV     (SP)+,SCLL(R1)  ;STORE VALUE
070132                                  TYPTXT  </UPPER LIMIT(OCTAL):/>
070164  104410                         RDOCT
070166  012661  070550                 MOV     (SP)+,SCHL(R1)  ;STORE VALUE
070172                                  TYPTXT  </STEP SIZE(OCTAL):/>
070222  104410                         RDOCT
070224  012661  070560                 MOV     (SP)+,SCSTP(R1);STORE VALUE
070230  000405                         BR      221$

                                        ;---R1=R1+2
                                        ;--UNTIL (R1>6)
070232                          220$:   TYPTXT </NO/>
070244  004737  015526         22'$:   JSR     PC,CNTRC        ;EXIT IF CONTROL C
070250  062701  000002                 ADD     #2,R1           ;ADVANCE THE DAC #'S
070254  020127  000006                 CMP     R1,#6           ;R1>6 ?
070260  101002                         BHI     230$            ;YES, BRANCH
070262  000137  067770                 JMP     210$            ;NO, JUMP

```

```
                                                ;--R1=0
                                                ;--EXECUTE[Y/N] ?
                                                ;-UNTIL (YES)
070266  005001                        230$:   CLR     R1              ;RESET TO ZERO
070270                                          TYPTXT  <<15><12> %EXECUTE [Y/N]:%>
070320  104406                                RDCHR
070322  122726  000131                        CMPB    #'Y,(SP)+       ;YES ?
070326  001413                                BEQ     240$            ;YES, BRANCH
070330  004737  015526                        JSR     PC,CNTRC        ;EXIT IF CONTROL C
070334                                          TYPTXT </NO/>
070346  104401  001203                        TYPE    ,$CRLF
070352  000137  067770                        JMP     210$            ;
                                                ;-INITAILIZE THE CURRENT COUNT
070356  013737  070540  070570        240$:   MOV     SCLL+0,SCCUR+0  ;DAC0 START
070364  013737  070542  070572                MOV     SCLL+2,SCCUR+2  ;DAC1 START
070372  013737  070544  070574                MOV     SCLL+4,SCCUR+4  ;DAC2 START
070400  013737  070546  070576                MOV     SCLL+6,SCCUR+6  ;DAC3 START
                                                ;-REPEAT
                                                ;--FOR R1=0 TO 6 BY 2 DO
                                                ;---SCCUR[R1]=SCCUR[R1]+SCSTP[R1]
                                                ;---IF SCCUR[R1]>SCHL[R1]
                                                ;----THEN (*ADJUST VALUE*)
                                                ;-----SCCUR[R1]=SCLL[R1]
                                                ;---LOAD DAC[R1]
                                                ;---END (*FOR*)
                                                ;-UNTIL (^C) OR (^A)
070406                                  250$: TYPTXT </YES/>
070420  104401  001203                        TYPE    ,$CRLF          ;
070424  013737  003416  003410        260$:   MOV     AD0,ATEMP       ;SET UP ADDRESS
070432  013737  003420  003412                MOV     AD1,BTEMP       ;SET UP ADDRESS
070440  012701  177776                        MOV     #-2,R1          ;SET UP COUNT
070444  062701  000002                261$:   ADD     #2,R1           ;
070450  022701  000006                        CMP     #6,R1           ;END OF LOOP ?
070454  103763                                BLO     260$            ;YES, BRANCH
070456  116177  070571  112726              MOVB    SCCUR+1(R1),@BTEMP;LOAD DAC(R1)
070464  116177  070570  112716              MOVB    SCCUR(R1),@ATEMP        ;
070472  066161  070560  070570              ADD     SCSTP(R1),SCCUR(R1);SET UP VALUE
070500  026161  070570  070550              CMP     SCCUR(R1),SCHL(R1); ABOVE LIMIT ?
070506  101403                              BLOS    262$            ;NO, BRANCH
070510  016161  070540  070570              MOV     SCLL(R1),SCCUR(R1);RESET VALUE
070516  062737  000002  003410  262$:       ADD     #2,ATEMP        ;MAKE NEXT ADDRESS
070524  062737  000002  003412              ADD     #2,BTEMP        ;MAKE NEXT ADDRESS
070532  004737  015526                      JSR     PC,CNTRC        ;EXIT IF CONTROL C
070536  000742                              BR      261$

                                                ;*** ROUTINE TABLES ***
070540                                  SCLL:   .BLKW 4 ;LOW LIMIT FOR EACH DAC
070550                                  SCHL:   .BLKW 4 ;UPPER LIMIT FOR EACH DAC
070560                                  SCSTP:  .BLKW 4 ;STEP SIZE FOR EACH DAC
070570                                  SCCUR:  .BLKW 4 ;CURRENT VALUE FOR EACH DAC
```

```
                        .SBTTL A631 FIELD MANUAL OPERATION
                ;*******************************************************************************
                ;FUNCTION:
                ;        THIS ROUTINE ALLOWS THE SETTING OF ANY DAC TO  ANY  VALUE  IN  THE
                ;       0-7777 (OCTAL) RANGE. TYPING AN OCTAL NUMBER  IN  REPONSE  TO  THE  DAC#
                ;       QUESTION SETS THE DAC TO THE VALUE ENTERED.  TYPING  JUST  A  <CR>  DOES
                ;       NOTHING BUT GET YOU TO THE NEXT QUESTION. EXITING THIS ROUTINE  IS  DONE
                ;       VIA CONTROL C.
                ;VARIABLE USAGE:
                ;       FLAG--USED TO INDICATE AN OCTAL NUMBER WAS ENTERED.
                ;       (R0)
                ;       DACX--USED TO INDICATE WHICH DAC TO OPERATE ON.
                ;       (R1)
                ;       NUMB--CONTAINS THE OCTAL NUMBER TO BE LOADED. (R2
                ;       CH--HOLDS THE CHARACTER (R3)
                ;*******************************************************************************

                        .REM %
                        :(A631 MANUAL FIELD ROUTINE)
                A631M:  ;  REPEAT
                        ;  :  FOR DACX=0 TO 3 DO
                210$:   ;  :  :  REPEAT
                        ;  :  :  :  CASE DACX OF
                220$:   ;  :  :  :  :  0: TYPE DAC0 MESSAGE
                        ;  :  :  :  :  :  TYPE DAC CONTENTS
                230$:   ;  :  :  :  :  1: TYPE DAC1 MESSAGE
                        ;  :  :  :  :  :  TYPE DAC CONTENTS
                240$:   ;  :  :  :  :  2: TYPE DAC2 MESSAGE
                        ;  :  :  :  :  :  TYPE DAC CONTENTS
                250$:   ;  :  :  :  :  3: TYPE DAC3 MESSAGE
                        ;  :  :  :  :  :  TYPE DAC CONTENTS
                        ;  :  :  :  :  END (CASE)
                260$:   ;  :  :  :  NUMB=0
                        ;  :  :  :  FLAG=0
                        ;  :  :  :  GET(CHARACTER)
                        ;  :  :  :  IF CNTRLC THEN EXIT
                270$:   ;  :  :  :  WHILE (CH=OCTAL_VALUE) DO
                        ;  :  :  :  :  FLAG=1
                        ;  :  :  :  :  NUMB=CH+(NUMB * 8)
                        ;  :  :  :  :  GET(CHARACTER)
                        ;  :  :  :  :  IF CNTRLC THEN EXIT
                        ;  :  :  :  :  END (WHILE)
                280$:   ;  :  :  UNTIL (NUMB <= 7777)
                        ;  :  :  IF FLAG=1
                        ;  :  :  :  THEN /LOAD THE DAC/
                        ;  :  :  :  :  SET UP DAC WORD
                        ;  :  :  :  :  FLAG=0
                        ;  :  :  :  :  CASE DACX OF
                290$:   ;  :  :  :  :  :  0: LOAD DAC0
                300$:   ;  :  :  :  :  :  1: LOAD DAC1
                310$:   ;  :  :  :  :  :  2: LOAD DAC2
                320$:   ;  :  :  :  :  :  3: LOAD DAC3
                        ;  :  :  :  :  :  END (CASE)
                        ;  :  END (FOR)
                        ;  UNTIL (CONTROL C)
                        %
```

```
                                                ;-REPEAT
                                                ;--FOR DACX=0 TO 3 DO
                                                ;---REPEAT
                                                ;----CASE DACX OF
070600  000400                          A631M:  BR      202$            ;DON'T INCREMENT YET
070602                                  200$:
070602  012701  177777                  202$:   MOV     #-1,R1          ;SET UP FOR LOOP
070606  005201                          201$:   INC     R1              ;ADVANCE COUNT
070610  022701  000003                          CMP     #3,R1           ;EXIT LOOP ?
070614  103772                                  BLO     200$            ;YES, BRANCH
070616  022701  000000                  210$:   CMP     #0,R1           ;DACX=0 ?
070622  001413                                  BEQ     220$            ;YES, BRANCH
070624  022701  000001                          CMP     #1,R1           ;DACX=1 ?
070630  001432                                  BEQ     230$            ;YES, BRANCH
070632  022701  000002                          CMP     #2,R1           ;DACX=2 ?
070636  001451                                  BEQ     240$            ;YES, BRANCH
070640  022701  000003                          CMP     #3,R1           ;DACX=3 ?
070644  001470                                  BEQ     250$            ;YES, BRANCH
070646  000000                                  HALT    ;R1 CAN'T BE THAT HIGH
070650  000776                                  BR      .-2             ;FIX IT
070652                                  220$:   TYPTXT  <<15><12> /DAC0 [OCTAL]: />
070702  117746  112510                        MOVB    @AD0,-(SP)      ;
070706  117766  112506  000001                MOVB    @AD1,1(SP)      ;
070714  000465                                BR      251$            ;
070716                                  230$:   TYPTXT  <<15><12> /DAC1 [OCTAL]: />
070746  117746  112450                        MOVB    @AD2,-(SP)      ;
070752  117766  112446  000001                MOVB    @AD3,1(SP)      ;
070760  000443                                BR      251$            ;
070762                                  240$:   TYPTXT  <<15><12> /DAC2 [OCTAL]: />
071012  117746  112410                        MOVB    @AD4,-(SP)      ;
071016  117766  112406  000001                MOVB    @AD5,1(SP)      ;
071024  000421                                BR      251$            ;
071026                                  250$:   TYPTXT  <<15><12> /DAC3 [OCTAL]: />
071056  117746  112350                        MOVB    @AD6,-(SP)      ;
071062  117766  112346  000001                MOVB    @AD7,1(SP)      ;
071070  104403                          251$:   TYPOS
071072     004                                  .BYTE   4               ;FOUR DIGITS
071073     001                                  .BYTE   1               ;DON'T SUPPRESS
071074                                          TYPTXT </ />
```

```
                                                ;---NUMB=0
                                                ;----FLAG=0
                                                ;----GET(CHARACTER)
                                                ;----IF CNTRLC THEN EXIT
071104  005002                        260$:   CLR     R2              ;
071106  005000                                CLR     R0              ;
071110  104406                                RDCHR
071112  012603                                MOV     (SP)+,R3        ;SAVE CHARACTER
071114  042703  000200                        BIC     #BIT7,R3        ;STRIP PARITY BIT
071120  004737  015526                        JSR     PC,CNTRC        ;EXIT IF CONTROL C
071124  110337  071136                        MOVB    R3,262$         ;STORE CHAR
071130  104401  071136                        TYPE    ,262$           ;
071134  000401                                BR      270$            ;
071136  000000                        262$:   .WORD   0               ;CHAR STORAGE

                                                ;----WHILE (CH=OCTAL_VALUE) DO
                                                ;-----FLAG=1
                                                ;-----NUMB=CH+(NUMB * 8)
                                                ;-----GET(CHARACTER)
                                                ;-----IF CNTRLC THEN EXIT
                                                ;-----END (WHILE)
071140  022703  000060                270$:   CMP     #60,R3          ;>59 ?
071144  101030                                BHI     280$            ;NO, BRANCH
071146  022703  000067                        CMP     #67,R3          ;<70 ?
071152  103425                                BLO     280$            ;NO,BRANCH
071154  042703  177770                        BIC     #177770,R3      ;STRIP UNNECESSARY GARBAGE
071160  012700  000001                        MOV     #1,R0           ;SET THE FLAG
071164  006302                                ASL     R2              ;X8
071166  006302                                ASL     R2              ;
071170  006302                                ASL     R2              ;
071172  060302                                ADD     R3,R2           ;CH+
071174  104406                                RDCHR                   ;
071176  012603                                MOV     (SP)+,R3        ;
071200  042703  000200                        BIC     #BIT7,R3        ;STRIP PARITY BIT
071204  004737  015526                        JSR     PC,CNTRC        ;EXIT IS CONTROL C
071210  110337  071222                        MOVB    R3,272$         ;PREPARE TO PRINT
071214  104401  071222                        TYPE    ,272$           ;
071220  000401                                BR      273$            ;
071222  000000                        272$:   .WORD   0               ;USED TO EHCO CHAR
071224  000745                        273$:   BR      270$            ;

                                                ;--UNTIL (NUMB <= 7777)
071226  020227  007777                280$:   CMP     R2,#7777        ;R2<=7777 ?
071232  101423                                BLOS    281$            ;YES, BRANCH
071234                                        TYPTXT  <<15><12> /*** NUMBER TOO LARGE ***/>
071276  000137  070616                        JMP     210$            ;TRY AGAIN
```

```
                                                ;---IF FLAG=1
                                                ;----THEN /LOAD THE DAC/
                                                ;-----SET UP DAC WORD
                                                ;-----CASE DACX OF
                                                ;-UNTIL (CONTROL C)
071302  010237  071456          281$:   MOV     R2,20$          ;FOR BYTE OPERATIONS
071306  005700                          TST     R0              ;FLAG=0 ?
071310  001453                          BEQ     321$            ;YES, BRANCH
071312  005000                          CLR     R0              ;CLEAR IT
071314  022701  000000                  CMP     #0,R1           ;DAC0 ?
071320  001413                          BEQ     290$            ;YES, BRANCH
071322  022701  000001                  CMP     #1,R1           ;DAC1 ?
071326  001420                          BEQ     300$            ;YES, BRANCH
071330  022701  000002                  CMP     #2,R1           ;DAC2 ?
071334  001424                          BEQ     310$            ;YES, BRANCH
071336  022701  000003                  CMP     #3,R1           ;DAC3 ?
071342  001430                          BEQ     320$            ;YES, BRANCH
071344  000000                          HALT    ;R1 CAN'T BE THAT HIGH
071346  000776                          BR      .-2             ;FIX IT
071350  113777  071457  112042  290$:   MOVB    20$+1,@AD1      ;SET UP HIGH BYTE
071356  113777  071456  112032          MOVB    20$,@AD0        ;LOAD THE WORD
071364  000137  071440                  JMP     321$            ;
071370  113777  071457  112026  300$:   MOVB    20$+1,@AD3      ;SET UP HIGH BYTE
071376  113777  071456  112016          MOVB    20$,@AD2        ;LOAD THE WORD
071404  000415                          BR      321$            ;
071406  113777  071457  112014  310$:   MOVB    20$+1,@AD5      ;SET UP HIGH BYTE
071414  113777  071456  112004          MOVB    20$,@AD4        ;LOAD THE WORD
071422  000406                          BR      321$            ;
071424  113777  071457  112002  320$:   MOVB    20$+1,@AD7      ;SET UP HIGH BYTE
071432  113777  071456  111772          MOVB    20$,@AD6        ;LOAD THE WORD
071440  022701  000003          321$:   CMP     #3,R1           ;DAC3 ?
071444  001002                          BNE     322$            ;NO, BRANCH
071446  104401  001203                  TYPE    ,$CRLF          ;
071452  000137  070606          322$:   JMP     201$            ;

                                        ;******* VARIABLE STORAGE *******
071456  000000                  20$:    .WORD   0               ;TEMPORARY STORAGE
```

```
                        ;*****************************************************************************
                        ;THIS ROUTINE SETS UP THE A631 TEST ADDRESSES (AD0-AD7).
                        ;*****************************************************************************


071460  005001          A631AD: CLR     R1
071462  013700  003462          MOV     TADDR,R0        ;A631 ADDRESS
071466  020127  000020  1$:     CMP     R1,#16.         ;LAST ADDRESS ?
071472  001406                  BEQ     2$              ;YES,BRANCH
071474  010061  003416          MOV     R0,ADC(R1)      ;SET ADDRESS
071500  005200                  INC     R0              ;ADD 1 TO ADDRESS
071502  005201                  INC     R1
071504  005201                  INC     R1
071506  000767                  BR      1$              ;DO IT AGAIN
071510  000207          2$:     RTS     PC

                        ;*****************************************************************************
                        ;THIS ROUTINE WAITS FOR A KEYBOARD RESPONSE.
                        ;*****************************************************************************


071512  105777  107442  WAIT:   TSTB    @$TKS           ;KEY STRUCK ?
071516  100375                  BPL     WAIT            ;NO,BRANCH
071520  004737  015526          JSR     PC,CNTRC
071524  000207                  RTS     PC
```

A631 CALIBRATION ROUTINE

```
                        .SBTTL A631 CALIBRATION ROUTINE

                ;*************************************************************
                ;       A631 12 BIT DAC CALIBRATION ROUTINE
                ;*************************************************************


                        .REM *
                        A631 CALIBRATION ROUTINE
                A631C:  :  ASK "VOLTAGE OR CURRENT?"
                        :  IF ANSWER = VOLTAGE
                        :  :  THEN
                        :  :  :  PRINT "SET E66 TO 11010100"
                        :  :  :  FOR R0 = 0 TO 3
                        :  :  :  :  PRINT "CONNECT EDC TO TERMINALS X & Y"
                        :  :  :  :  PRINT "SET EDC TO 0000.0"
                        :  :  :  :  PRINT "HIT A KEY WHEN READY"
                        :  :  :  :  REPEAT UNTIL KEYSTRUCK
                        :  :  :  :  LOAD DAC [R0] = 0000
                        :  :  :  :  REPEAT UNTIL KEYSTRUCK
                        :  :  :  :  PRINT "SET EDC TO 10.2375"
                        :  :  :  :  PRINT "ADJUST POT R XX UNTIL METER NULLS"
                        :  :  :  :  LOAD DAC [R0] WITH 7777
                        :  :  :  :  REPEAT UNTIL KEYSTRUCK
                        :  :  :  END (FOR)
                        :  IF ANSWER = CURRENT
                        :  :  THEN
                        :  :  :  PRINT "SET E66 TO 10101010"
                        :  :  :  PRINT "SET EDC TO NULL MODE"
                        :  :  :  FOR R0 = 0 TO 3
                        :  :  :  :  PRINT "SET EDC TO 2.5 MV"
                        :  :  :  :  PRINT "CONNECT 500 OHM .01% RESISTOR TO TERMINALS X &Y"
                        :  :  :  :  PRINT "CONNECT EDC ACROSS RESISTOR"
                        :  :  :  :  PRINT "HIT A KEY WHEN READY"
                        :  :  :  :  REPEAT UNTIL KEYSTRUCK
                        :  :  :  :  LOAD DAC[R0] WITH 0001
                        :  :  :  :  PRINT "ADJUST R XX UNTIL METER NULLS"
                        :  :  :  :  REPEAT UNTIL KEYSTRUCK
                        :  :  :  :  PRINT "SET EDC TO 10.2375 VOLTS"
                        :  :  :  :  PRINT "ADJUST R XX UNTIL EDC NULLS"
                        :  :  :  :  LOAD DAC[R0] WITH 7777
                        :  :  :  :  REPEAT UNTIL KEYSTRUCK
                        :  :  :  END (FOR)
                        END (A631 CALIBRATION)
                        *
```

```
                        ;*****************************************************************************
                        ;A631 FIELD SERVICE CALIBRATION ROUTINE
                        ;*****************************************************************************


071526                  A631C:  TYPTXT </A631 FIELD SERVICE CALIBRATION PROCEDURE/>
071606  104401  001203          TYPE    ,$CRLF

071612  000240          5$:     NOP
071614                  S5:     TYPTXT <<CR><LF>*VOLTAGE OR CURRENT    'C] ?*>
071660  104406                  RDCHR                       ;GET RESPONSE
071662  012600                  MOV     (SP)+,R0            ;SAVE IT
071664  020027  000126          CMP     R0,#'V              ;VOLTAGE ?
071670  001410                  BEQ     200$                ;YES, BRANCH
071672  020027  000103          CMP     R0,#'C              ;CURRENT ?
071676  001002                  BNE     8$                  ;NO,BRANCH
071700  000137  072626          JMP     300$                ;DO CURRENT CALIBRATION
071704  004737  015526  8$:     JSR     PC,CNTRC            ;CHECK FOR EXIT
071710  000741                  BR      S5                  ;TRY AGAIN

071712                  200$:   TYPTXT </VOLTAGE/<CR><LF>>
071732  104401  073622          TYPE    ,411$               ;''SET E66 TO VOLTAGE MODE
071736  104401  073703          TYPE    ,412$               ;-
071742  104401  073747          TYPE    ,414$               ;''HIT A KEY WHEN READY''
071746  104406                  RDCHR
071750  005726                  TST     (SP)+               ;CLEAR STACK
071752  104401  001203          TYPE    ,$CRLF
071756  104401  073410          TYPE    ,410$               ;SET EDC TO NULL MODE
071762  104401  001203          TYPE    ,$CRLF
071766  012700  177777          MOV     #-1,R0              ;SET UP FOR LOOP
071772  005200          205$:   INC     R0                  ;BUMP FOR LOOP
071774  020027  000003          CMP     R0,#3               ;LAST DAC ?
072000  101305                  BHI     S5                  ;YES,BRANCH
072002                          TYPTXT <<CR><LF><LF>/   VOLTAGE CALIBRATION DAC />
072046  010046                  MOV     R0,-(SP)
072050  104403                  TYPOS
072052     001                  .BYTE 1
072053     000                  .BYTE 0
072054  104401  001203          TYPE    ,$CRLF
072060                          TYPTXT <<CR><LF>/SET EDC TO 0.0000/>
072112  104401  001203          TYPE    ,$CRLF
072116                          TYPTXT  </CONNECT EDC TO SCREW TERMINALS />
072164  010001                  MOV     R0,R1
072166  006301                  ASL     R1                  ;OFFSET TO TERMINAL TABLE
072170  016137  074412  072200  MOV     30$(R1),7$          ;TERMINAL NUMBERS
072176  104401                  TYPE
072200  000000          7$:     .WORD 0
072202  104401  001203          TYPE    ,$CRLF
072206  104401  073747          TYPE    ,414$               ;''HIT A KEY WHEN READY''
072212  104406                  RDCHR                       ;WAIT FOR REPONSE
072214  005726                  TST     (SP)+               ;CLEAR STACK
072216  104401  001203          TYPE    ,$CRLF
072222  104401  001203          TYPE    ,$CRLF
072226                          TYPTXT </VOLTAGE OFFSET SHOULD BE LESS THAN (+ OR -) 2.5 MVOLT./>
072324                          TYPTXT <<CR><LF>/  NOTE: THERE IS NO VOLTAGE OFFSET ADJUSTMENT./>
072414                          TYPTXT <<CR><LF>/         IF NOT WITHIN ABOVE LIMITS - REPLACE MODULE./>
```

```
    871 072512  104401  001203                      TYPE    ,$CRLF
    872 072516  012746  000000                      MOV     #0,-(SP)        ;VALUE TO LOAD
    873 072522  004737  073362                      JSR     PC,400$         ;LOAD IT
    874 072526  104401  073747                      TYPE    ,414$
    875 072532  104406                             RDCHR                   ;WAIT FOR RESPONSE
    876 072534  005726                             TST     (SP)+           ;CLEAR STACK
    877 072536  104401  001203                      TYPE    ,$CRLF
    878 072542  104401  001203                      TYPE    ,$CRLF
    879 072546  104401  074056                      TYPE    ,417$           ;''SET EDC TO 10.2375''
    880 072552  104401  073770                      TYPE    ,415$           ;ADJ R XX
    881 072556  010001                             MOV     R0,R1
    882 072560  006301                             ASL     R1              ;OFFSET TO TABLE
    883 072562  016137  074370  072572              MOV     20$(R1),1$      ;POT #
    884 072570  104401                             TYPE
    885 072572  000000                      1$:    .WORD 0
    886 072574  104401  074007                      TYPE    ,416$
    887 072600  012746  007777                      MOV     #7777,-(SP)     ;VALUE TO LOAD
    888 072604  004737  073362                      JSR     PC,400$         ;LOAD IT
    889 072610  104406                             RDCHR                   ;WAIT FOR RESPONSE
    890 072612  005726                             TST     (SP)+           ;CLEAR STACK
    891 072614  152777  000002  110022              BISB    #CBIT,@CSR      ;CLEAR DAC
    892 072622  000137  071772                      JMP     205$            ;NEXT DAC
    893
```

```
                                300$:   TYPTXT </CURRENT/<CR><LF>>
072646  104401  073622                  TYPE    ,411$           ;SET E66
072652  104401  073725                  TYPE    ,413$           ;-TO CURRENT MODE
072656  104401  073747                  TYPE    ,414$           ;HIT ANY KEY WHEN READY
072662  104406                          RDCHR                   ;WAIT FOR RESPONSE
072664  005726                          TST     (SP)+           ;CLEAR STACK
072666  104401  073410                  TYPE    ,410$           ;''SET EDC TO NULL MODE''

072672  012700  177777                  MOV     #-1,R0          ;SET UP FOR LOOP
072676  005200                  305$:   INC     R0              ;BUMP R0
072700  020027  000003                  CMP     R0,#3           ;LAST DAC ?
072704  101402                          BLOS    9$              ;NO,BRANCH
072706  000137  071614                  JMP     S5              ;JUMP
072712                          9$:     TYPTXT <<CR><LF>/       CURRENT CALIBRATION DAC />
072754  010046                          MOV     R0,-(SP)
072756  104403                          TYPOS
072760     001                          .BYTE 1
072761     000                          .BYTE 0
072762  104401  001203                  TYPE    ,$CRLF
072766  104401  001203                  TYPE    ,$CRLF
072772                                  TYPTXT </CONNECT 500 OHM .01% RESISTOR TO  />
073044                                  TYPTXT <<CR><LF>/   SCREW TERMINALS />
073100  010001                          MOV     R0,R1           ;GET INDEX
073102  006301                          ASL     R1              ;* 2
073104  016137  074112  073114          MOV     10$(R1),2$      ;ADDRESS OF MESSAGE
073112  104401                          TYPE
073114  000000                  2$:     .WORD 0
073116                                  TYPTXT <<CR><LF>/SET EDC TO 2.5 MVOLT./>
073154                                  TYPTXT <<CR><LF>/CONNECT EDC ACROSS RESISTOR./<CR><LF>>
073224  104401  073747                  TYPE    ,414$           ;HIT A KEY WHEN READY
073230  104406                          RDCHR                   ;WAIT FOR RESPONSE
073232  005726                          TST     (SP)+           ;CLEAR STACK
073234  104401  073770                  TYPE    ,415$
073240  010001                          MOV     R0,R1           ;GET DAC #
073242  006301                          ASL     R1              ;* 2
073244  016137  074346  073254          MOV     15$(R1),3$      ;ADDRESS OF MESSAGE
073252  104401                          TYPE
073254  000000                  3$:     .WORD   0               ;POT #
073256  104401  074007                  TYPE    ,416$           ;''UNTIL EDC NULLS''
073262  012746  000001                  MOV     #1,-(SP)        ;VALUE TO LOAD
073266  004737  073362                  JSR     PC,400$         ;LOAD IT
073272  104406                          RDCHR                   ;WAIT FOR RESPONSE
073274  005726                          TST     (SP)+           ;CLEAR STACK
073276  104401  001203                  TYPE    ,$CRLF
073302  104401  074056                  TYPE    ,417$           ;''SET EDC TO 10.2375 V''
073306  104401  073770                  TYPE    ,415$           ;''ADJUST RXX''
073312  010001                          MOV     R0,R1           ;DAC #
073314  006301                          ASL     R1              ;* 2
073316  016137  074370  073326          MOV     20$(R1),4$      ;ADDRESS OF MESSAGE
073324  104401                          TYPE
073326  000000                  4$:     .WORD 0
073330  104401  074007                  TYPE    ,416$           ;''UNTIL METER NULLS''
073334  012746  007777                  MOV     #7777,-(SP)     ;VALUE TO LOAD
073340  004737  073362                  JSR     PC,400$         ;GO LOAD IT
073344  104406                          RDCHR                   ;WAIT FOR REPONSE
073346  005726                          TST     (SP)+           ;CLEAR STACK
```

```
    952 073350  152477  000002  107266          BISB    #CBIT,@CSR      ;CLEAR DACS
    953 073356  000137  072676                  JMP     305$            ;DO IT AGAIN
```

```
                                ;*****************************************************************
                                ;       CALL:   MOV     DAC#,R0
                                ;               MOV     #VALUE,-(SP)
                                ;               JSR     PC,400$
                                ;*****************************************************************

 073362  010001                400$:   MOV     R0,R1           ;DON'T DESTROY R0
 073364  006301                        ASL     R1              ;* 2
 073366  063701  003420                ADD     AD1,R1          ;HIGH BYTE ADDRESS
 073372  116611  000003                MOVB    3(SP),@R1       ;LOAD HIGH BYTE
 073376  005301                        DEC     R1              ;LOW BYTE ADDRESS
 073400  116611  000002                MOVB    2(SP),@R1       ;LOAD LOW BYTE
 073404  012616                        MOV     (SP)+,(SP)      ;ADJUST STACK
 073406  000207                        RTS     PC

 073410     015     012     075  410$:   .ASCII <CR><LF>/==> SET EDC TO NULL MODE <==/<CR><LF>
 073450     075     075     076          .ASCII *==>CALIBRATIONS ARE TO BE WITHIN (+ OR -)1.25MVOLT (1/2LSB)<==*
 073546     015     012     075          .ASCIZ <CR><LF>/==>SET METER SENSITIVITY ACCORDINGLY<==/<CR><LF>
 073622     123     105     124  411$:   .ASCII /SET E66 AS FOLLOWS 1=ON 0=OFF/<CR><LF>
 073661     061     040     062          .ASCIZ /1 2 3 4 5 6 7 8/<CR><LF>
 073703     061     040     061  412$:   .ASCIZ /1 1 0 1 0 1 0 0/<CR><LF>        ;VOLTAGE MODE
 073725     061     040     060  413$:   .ASCIZ /1 0 1 0 1 0 1 0/<CR><LF>        ;CURRENT MODE
 073747     074     103     122  414$:   .ASCIZ /<CR> TO CONTINUE/
 073770     015     012     101  415$:   .ASCIZ <CR><LF>/ADJUST POT R/
 074007     040     125     116  416$:   .ASCIZ / UNTIL EDC NULLS.   <CR> TO CONTINUE/<CR><LF>
 074056     123     105     124  417$:   .ASCIZ /SET EDC TO 10.2375 VOLTS./<CR><LF>
                                .EVEN

                                ;*****************************************************************
                                ;       CURRENT SCREW TERMINALS.
                                ;*****************************************************************

 074112  074122                10$:    11$     ;DAC 0
 074114  074164                        12$     ;DAC 1
 074116  074232                        13$     ;DAC 2
 074120  074300                        14$     ;DAC 3
 074122     040     050     053  11$:    .ASCIZ / (+)7 (-)8 OR BERG PINS (+)7 (-)8/
 074164     040     050     053  12$:    .ASCIZ / (+)15 (-)16 OR BERG PINS (+)15 (-)16/
 074232     040     050     053  13$:    .ASCIZ / (+)23 (-)24 OR BERG PINS (+)49 (-)50/
 074300     040     050     053  14$:    .ASCIZ / (+)33 (-)34 OR BERG PINS (+)39 (-)40/
                                .EVEN


                                ;*****************************************************************
                                ;CURRENT OFFSET TRIMPOT
                                ;*****************************************************************

 074346  074356                15$:    16$     ;DAC 0
 074350  074361                        17$     ;DAC 1
 074352  074363                        18$     ;DAC 2
 074354  074365                        19$     ;DAC 3
 074356     061     061     000  16$:    .ASCIZ /11/             ;R11
 074361     070     000          17$:    .ASCIZ /8/              ;R8
 074363     066     000          18$:    .ASCIZ /6/              ;R6
 074365     063     000          19$:    .ASCIZ /3/              ;R3
```

```
                                .EVEN

                                ;*****************************************************************
                                ;CURRENT AND VOLTAGE  GAIN TRIMPOT
                                ;*****************************************************************

074370  074400                  20$:    21$
074372  074403                          22$
074374  074405                          23$
074376  074407                          24$
074400    061   060   000       21$:    .ASCIZ /10/             ;R10
074403    067   000             22$:    .ASCIZ /7/              ;R7
074405    064   000             23$:    .ASCIZ /4/              ;R4
074407    062   000             24$:    .ASCIZ /2/              ;R2
                                .EVEN

                                ;*****************************************************************
                                ;VOLTAGE DAC OUTPUT SCREW TERMINALS

074412  074422                  30$:    31$     ;DAC 0
074414  074464                          32$     ;DAC 1
074416  074532                          33$     ;DAC 2
074420  074600                          34$     ;DAC 3
074422    040   050   053       31$:    .ASCIZ / (+)5 (-)6 OR BERG PINS (+)5 (-6)/
074464    040   050   053       32$:    .ASCIZ / (+)13 (-)14 OR BERG PINS (+)13 (-)14/
074532    040   050   053       33$:    .ASCIZ / (+)21 (-)22 OR BERG PINS (+)47 (-)48/
074600    040   050   053       34$:    .ASCIZ / (+)31 (-)32 OR BERG PINS (+)37 (-)38/
```

```
1049
1050                                    .SBTTL  A020 A/D CONVERTER MONITOR
1051                                ;*************************************************************
1052                                ;*      A020 MONITOR - THE MONITOR GETS YOU THERE            *
1053                                ;*************************************************************
1054
1055                                    ;*******AUTOMATIC TEST *******
1056 074646  005737  002664        AUT020: TST     AFLAG           ;UNDER AUTO MODE ?
1057 074652  001407                        BEQ     A20SRT          ;NO,BRANCH
1058 074654  004737  014310                JSR     PC,KLEER        ;CLEAR D BUS
1059 074660  004737  075466                JSR     PC,ADSET        ;SET UP AD0-AD3
1060 074664  004737  075534                JSR     PC,A20TE1       ;DO THE TEST
1061 074670  000207                        RTS     PC
1062
1063                                    ;******* THE INTERROGATOR *******
1064 074672                        A20SRT:
1065 074672  012737  076146  000004 A20MON: MOV     #TIMOUT,ERRVEC  ;SET UP TIMEOUT VECTOR
1066 074700  012737  000340  000006        MOV     #340,ERRVEC+2   ;SET UP PRIORITY
1067 074706  012706  001100                MOV     #1100,SP        ;RESET STACK POINTER
1068 074712  012777  000340  105732 1$:    MOV     #340,@VECTOA    ;SET PRIORITY = 7
1069 074720  004737  014310                JSR     PC,KLEER        ;CLEAR THE D BUS
1070 074724  005037  001214                CLR     $PASS           ;CLEAR PASS COUNT
1071 074730  104401  075227                TYPE    ,220$           ;TYPE HEADER
1072 074734  004737  112646                JSR     PC,PRI7         ;SET PRIORITY 7
1073 074740  105777  106452                TSTB    @AD0            ;CHECK FOR TIMEOUT
1074 074744  104406                       RDCHR                   ;READ FIRST CHARACTER
1075 074746  112637  075200                MOVB    (SP)+,200$      ;SAVE FIRST LETTER
1076 074752  004737  015526                JSR     PC,CNTRC        ;CONTROL C ?
1077 074756  113737  075200  075174        MOVB    200$,100$       ;SET UP ECHO
1078 074764  104401  075174                TYPE    ,100$           ;
1079 074770  104401  001203                TYPE    ,$CRLF          ;TYPE CR,LF
1080 074774  042737  100200  075200        BIC     #100200,200$    ;STRIP BIT7 OF CHARACTERS
1081 075002  005000                        CLR     R0              ;RESET COMMAND POINTER
1082
1083                                    ;*******  FIND COMMAND  *******
1084 075004  026037  075326  075200 4$:    CMP     300$(R0),200$   ;A MATCH ?
1085 075012  001410                        BEQ     5$              ;YES, BRANCH
1086 075014  062700  000002                ADD     #2,R0           ;ADVANCE POINTER
1087 075020  005760  075326                TST     300$(R0)        ;END OF TABLE ?
1088 075024  001367                        BNE     4$              ;NO, BRANCH
1089 075026  104401  075202                TYPE    ,210$           ;COMMAND NOT FOUND
1090 075032  000727                        BR      1$              ;GO TRY AGAIN
1091
```

```
1093                                                    ;******* EXECUTE COMMAND *******
1094 075034  004737  075466              5$:      JSR     PC,ADSET        ;SET THE ADDRESS
1095 075040  016037  075344  075050              MOV     500$(R0),6$     ;GET READY TO TYPE
1096 075046  104401                              TYPE                    ;THE COMMAND
1097 075050  000000                      6$:     0                       ;THE MESSAGE TO TYPE
1098 075052  104401  001203                      TYPE    ,$CRLF          ;TYPE A CR THEN LF
1099 075056  016037  075336  003652              MOV     400$(R0),RETURN ;SET UP CNTRLA RETURN
1100 075064  004777  106562              10$:    JSR     PC,@RETURN      ;GO DO THE TEST
1101 075070  022737  075144  003652              CMP     #7$,RETURN
1102 075076  001705                              BEQ     1$
1103 075100  005237  001214                      INC     $PASS           ;PASS COUNTER
1104 075104  023737  002662  001214              CMP     PASCNT,$PASS    ;DONE ?
1105 075112  001364                              BNE     10$             ;NO,BRANCH
1106 075114  104401  075300                      TYPE    ,223$           ;PASS COMPLETE(S)
1107 075120  013746  001214                      MOV     $PASS,-(SP)     ;
1108 075124  104402                              TYPOC                   ;
1109 075126  005037  001214                      CLR     $PASS
1110 075132  032777  040000  104014              BIT     #BIT14,@SWR     ;LOOP ON TEST ?
1111 075140  001351                              BNE     10$             ;YES BRANCH
1112 075142  000663                              BR      1$
1113
1114                                                    ;******* HELP PRINTER *******
1115 075144  005000                      7$:     CLR     R0              ;SET UP R0
1116 075146  016037  075344  075156      8$:     MOV     500$(R0),9$     ;GET READY TO TYPE
1117 075154  104401                              TYPE                    ;THE COMMAND
1118 075156  000000                      9$:     0                       ;THE MESSAGE TO TYPE
1119 075160  062700  000002                      ADD     #2,R0           ;ADVANCE POINTER
1120 075164  005760  075344                      TST     500$(R0)        ;END OF TABLE ?
1121 075170  001366                              BNE     8$              ;NO, KEEP PRINTING
1122 075172  000207                              RTS     PC
1123
1124                                                    ;******* MESSAGES & THINGS *******
1125                                                    .NLIST BEX              ;TYPE NEATLY THE .ASCIZ LINES
1126 075174  000000                      100$:   .WORD   0               ;INITAILLY SET TO 0,OTHERWISE TEMP
1127 075176  000000                      101$:   .WORD   0               ;TEMPORARY
1128 075200  000000                      200$:   .WORD   0               ;COMMAND LINE
1129 075202    040    103    117        210$:   .ASCIZ  / COMMAND NOT FOUND/<15><12>
1130 075227    015    012    101        220$:   .ASCII <15><12> "A020 A/D CONVERTER MONITOR (H=HELP)"
1131 075274    015    012    043                .ASCIZ <15><12> /#/
1132 075300    015    012    120        223$:   .ASCIZ <15><12> /PASS COMPLETE(S)    /
1133                                            .EVEN
1134
1135
```

```
                                        ;******* COMMAND TABLES *******
075326  000114                  300$:   .WORD   'L                  ;LOGIC TEST
075330  000110                          .WORD   'H                  ;HELP FILE
075332  000103                          .WORD   'C                  ;CALIBRATION
075334  000000                          .WORD   0                   ;END OF 300$ TABLE

                                        ;******* TEST POINTERS *******
075336  075534                  400$:   .WORD   A20TE1              ;LOGIC TEST
075340  075144                          .WORD   7$                  ;HELP FILE
075342  105426                          .WORD   A20CAL              ;CALIBRATION

                                        ;******* HEADER POINTERS *******
075344  075354                  500$:   .WORD   600$                ;LOGIC TEST
075346  075402                          .WORD   601$                ;HELP FILE
075350  075427                          .WORD   602$                ;CALIBRATION
075352  000000                          .WORD   0                   ;END OF TABLE 500$ MARKER

                                        ;******* HEADER MESSAGES *******
075354  050     114     051     600$:   .ASCIZ  /(L)-A020 LOGIC TEST/<15><12>
075402  050     110     051     601$:   .ASCIZ  /(H)-A020 HELP FILE/<15><12>
075427  050     103     051     602$:   .ASCIZ  #(C)-A020 CALIBRATION/MANUAL #<15><12>
                                        .EVEN

                                        ;******* SET UP ADDRESSES *******
075466  013737  003462  003416  ADSET:  MOV     TADDR,AD0           ;SET BASE ADDRESS
075474  013737  003416  003420          MOV     AD0,AD1             ;
075502  005237  003420                  INC     AD1                 ;MAKE SECOND ADDRESS
075506  013737  003420  003422          MOV     AD1,AD2             ;
075514  005237  003422                  INC     AD2                 ;MAKE THIRD ADDRESS
075520  013737  003422  003424          MOV     AD2,AD3             ;
075526  005237  003424                  INC     AD3                 ;MAKE FOURTH ADDRESS
075532  000207                          RTS     PC
```

```
                                        .SBTTL  A20 LOGIC TEST
                                ;*******************************************************************************
                                ;THE TEST WILL VERIFY MOST OF THE LOGIC ON THE MOTHER BOARD. THE DAUGHTER  BOARD
                                ;DOES NOT HAVE TO BE PRESENT FOR THIS TEST TO FUNCTION.  MOUNTING  THE  DAUGHTER
                                ;BOARD WILL NOT INTERFERE WITH THE TEST.
                                ;*******************************************************************************

                                ;*******************************************************************************
                                ;TST1-ALL FOUR ADDRESSES ARE READ TO CHECK THAT TIMEOUT DOES NOT OCCUR.
                                ;*******************************************************************************

1182 C75534                     A20TE1:
                                ;;*************************************************************
075534  000004                  TST36:  SCOPE
075536  012737  000036  001212          MOV     #36,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
1183 075544  004737  015526              JSR     PC,CNTRC        ;GOTO MONITOR IF CNTRLC
1184 075550  013737  003416  001136      MOV     @0,$BDADR       ;$BDADR = 171000
1185 075556  012737  075640  000004      MOV     #TMOUT,ERRVEC   ;ESTABLISH TIMEOUT RETURN
1186
1187                                    ;******* CHECK FOR NO TIMEOUT *******
1188 075564  105777  103346              TSTB    @$BDADR         ;171000 EXIST ?
1189 075570  005237  001136              INC     $BDADR          ;MAKE NEXT ADDRESS
1190 075574  004737  015526              JSR     PC,CNTRC        ;GOTO MONITOR IF CNTRC
1191 075600  105777  103332              TSTB    @$BDADR         ;171001 EXIST ?
1192 075604  005237  001136              INC     $BDADR          ;MAKE NEXT ADDRESS
1193 075610  004737  015526              JSR     PC,CNTRC        ;GOTO MONITOR IF CNTRC
1194 075614  105777  103316              TSTB    @$BDADR         ;171002 EXIST ?
1195 075620  005237  001136              INC     $BDADR          ;MAKE NEXT ADDRESS
1196 075624  004737  015526              JSR     PC,CNTRC        ;GOTO MONITOR IF CNTRC
1197 075630  105777  103302              TSTB    @$BDADR         ;171003 EXIST ?
1198 075634  000410                      BR      A20TE2
1199 075636  000736                      BR      A20TE1 ;REPLACE ABOVE BRANCH WITH NOP TO LOOP
1200
1201                                    ;*******TIMEOUT TRAPS HERE *******
1202 075640  012637  001136      TMOUT:  MOV     (SP)+,$BDADR    ;SAVE OLD PC
1203 075644  012746  075652              MOV     #1$,-(SP)       ;TO LOAD PC
1204 075650  000002                      RTI                     ;THIS ONE TOO
1205 075652                      1$:
075652  104120                          EMT     120
1206 075654  000207                      RTS     PC
```

```
1208                                    ;*****************************************************************************
1209                                    ;TST2-THE FIRST, SECOND, AND FOURTH ADDRESS ARE WRITTEN TO 1'S TO CHECK THAT
1210                                    ;TIMEOUT WILL OCCUR. THEN THE THIRD ADDRESS IS WRITTEN AND CHECKED THAT NO
1211                                    ;TIMEOUT OCCURS.
1212                                    ;*****************************************************************************
1213
1214 075656                            A20TE2:
                                        ;;************************************************************
     075656  000004                     TST37:  SCOPE
     075660  012737  000037  001212             MOV     #37,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
1215 075666  004737  015526                     JSR     PC,CNTRC        ;GOTO MONITOR IF CNTRC
1216 075672  012737  076004  000004             MOV     #2$,ERRVEC      ;SET UP TIMEOUT RETURN
1217
1218                                            ;******* CHECK FOR TIMEOUT *******
1219 075700  013737  003416  001136  1$:        MOV     AD0,$BDADR      ;SET UP 171000
1220 075706  105077  103224                     CLRB    @$BDADR         ;TRY WRITING
1221 075712  004737  015526                     JSR     PC,CNTRC        ;GOTO MONITOR IF CNTRC
1222 075716  104121                             EMT     121
1223 075720  013737  003420  001136             MOV     AD1,$BDADR      ;SET UP 171001
1224 075726  105077  103204                     CLRB    @$BDADR         ;TRY WRITING
1225 075732  004737  015526                     JSR     PC,CNTRC        ;GOTO MONITOR IF CNTRC
1226 075736  104121                             EMT     121
1227 075740  013737  003424  001136             MOV     AD3,$BDADR      ;SET UP 171003
1228 075746  105077  103164                     CLRB    @$BDADR         ;TRY WRITING
1229 075752  004737  015526                     JSR     PC,CNTRC        ;GOTO MONITOR IF CNTRC
1230 075756  104121                             EMT     121
1231
1232                                            ;******* CHECK FOR NO TIMEOUT *******
1233 075760  012737  075640  000004             MOV     #TMOUT,ERRVEC   ;SET UP TIMEOUT RETURN
1234 075766  013737  003422  001136             MOV     AD2,$BDADR      ;SET UP 171002
1235 075774  105077  103136                     CLRB    @$BDADR         ;TRY WRITING
1236 076000  000404                             BR      A20TE3
1237 076002  000725                             BR      A20TE2 ;REPLACE ABOVE BRANCH WITH NOP TO LOOP
1238
1239                                            ;******* WRITE TIMEOUT RETURN *******
1240 076004  062716  000014          2$:        ADD     #14,(SP)        ;ADJUST STACK
1241 076010  000002                             RTI                     ;RETURN TO PC+16
```

```
                        ;*****************************************************************************
                        ;TST3-THE FOURTH ADDRESS IS CHECKED TO BE CLEARED BY THE C  BIT.  ALSO  THE  CSR
                        ;(171377) IS CHECKED TO BE CLEARED.
                        ;*****************************************************************************

                        A20TE3:
                        ;;************************************************************
 076012  000004         TST40:  SCOPE
 076014  012737  000040  001212          MOV     #40,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
1249 076022  005037  001142             CLR     $BDDAT          ;RESET THE DATA
1250 076026  004737  015526             JSR     PC,CNTRC        ;GOTO MONITOR IF CNTRC
1251 076032  012737  076146  000004     MOV     #TIMOUT,ERRVEC  ;SET UP TIMEOUT FOR ALL
1252
1253                                    ;******* CHECK THAT AD3 = 0 *******
1254 076040  112777  000002  104576     MOVB    #CBIT,@CSR      ;CLEAR THE THING
1255 076046  012700  001000             MOV     #1000,R0        ;SET UP DELAY
1256 076052  005300                 1$: DEC     R0              ;DECREASE BY ONE
1257 076054  001376                     BNE     1$              ;LOOP TILL = 0
1258 076056  004737  015526             JSR     PC,CNTRC        ;GOTO MONITOR IF CNTRC
1259 076062  105777  105336             TSTB    @AD3            ;171003 = 0 ?
1260 076066  001411                     BEQ     2$              ;YES, BRANCH
1261 076070  005037  001140             CLR     $GDDAT          ;SET UP GOOD DATA
1262 076074  013737  003424  001136     MOV     AD3,$BDADR      ;SET UP 171003
1263 076102  117737  103030  001142     MOVB    @$BDADR,$BDDAT  ;SET UP BAD DATA
1264 076110  104135                     EMT     135
1265
1266                                    ;******* CHECK THAT CSR = 0 *******
1267 076112  105777  104526         2$: TSTB    @CSR            ;CSR = 0 ?
1268 076116  001411                     BEQ     3$              ;YES, BRANCH
1269 076120  005037  001140             CLR     $GDDAT          ;SET UP GOOD DATA
1270 076124  013737  002644  001136     MOV     CSR,$BDADR      ;SET UP TYPEOUT
1271 076132  117737  104506  001142     MOVB    @CSR,$BDDAT     ;SET UP TYPEOUT
1272 076140  104136                     EMT     136
1273
1274 076142  000412                 3$: BR      A20IE4
1275 076144  000722                     BR      A20TE3  ;REPLACE ABOVE BRANCH WITH NOP TO LOOP
1276
1277                                    ;******* GENERAL TIMEOUT ROUTINE *******
1278 076146  012637  001136     TIMOUT: MOV     (SP)+,$BDADR    ;SAVE PC
1279 076152  012746  076160             MOV     #1$,-(SP)       ;TO LOAD PC
1280 076156  000002                     RTI                     ;THIS ONE TOO !
1281 076160  004737  015526         1$: JSR     PC,CNTRC        ;GOTO MONITOR IF CNTRC
1282 076164  104134                     EMT     134
1283 076166  000207                     RTS     PC
```

```
                                ;*****************************************************************************
                                ;TST4-THE G BIT IS SET AND EACH OF THE FOUR ADDRESSES ARE READ. THE GENERIC CODE
                                ;OF 324 OR 304 IS EXPECTED FROM EACH ADDRESS.
                                ;*****************************************************************************

076170                          A20TE4:
                                ;;************************************************************
076170  000004                  TST41:  SCOPE
076172  012737  000041  001212          MOV     #41,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
076200  005037  001142                  CLR     $BDDAT          ;CLEAR THIS WORD
076204  004737  015526                  JSR     PC,CNTRC        ;GOTO MONITOR IF CNTRC

                                        ;******* ESTABLISH CORRECT CODE *******
076210  112777  000002  104426          MOVB    #CBIT,@CSR      ;CLEAR THE THING
076216  012700  001000                  MOV     #1000,R0        ;SET UP DELAY
076222  005300                  1$:     DEC     R0              ;DECREASE BY ONE
076224  001376                          BNE     1$              ;LOOP TILL = 0
076226  112777  000004  104410          MOVB    #GBIT,@CSR      ;SET UP TO READ GENERIC CODE
076234  013737  003416  001136          MOV     @D0,$BDADR      ;SET UP 171000
076242  013737  003416  076456          MOV     @D0,200$        ;SAVE ADDRESS
076250  062737  000004  076456          ADD     #4,200$         ;EXIT ADDRESS
076256  012737  000324  001140  2$:     MOV     #324,$GDDAT     ;PREPARE TYPEOUT
076264  004737  015526                  JSR     PC,CNTRC        ;GOTO MONITOR IF CNTRC
076270  127737  102642  001140          CMPB    @$BDADR,$GDDAT  ;CORRECT GENERIC CODE ?
076276  001414                          BEQ     3$              ;YES, BRANCH
076300  012737  000304  001140          MOV     #304,$GDDAT     ;PREPARE TYPEOUT
076306  127737  102624  001140          CMPB    @$BDADR,$GDDAT  ;OTHER GENERIC CODE CORRECT ?
076314  001405                          BEQ     3$              ;YES, BRANCH
076316  117737  102614  001142          MOVB    @$BDADR,$BDDAT  ;SET UP TYPEOUT
076324  104123                          EMT     123
076326  000207                          RTS     PC

                                        ;******* CHECK LOCATIONS FOR CORRECT CODE *******
076330  013737  001140  003606  3$:     MOV     $GDDAT,GBITE    ;STORE FOR SAFE KEEPING
076336  005237  001136                  INC     $BDADR          ;ADVANCE POINTER
076342  023737  076456  001136          CMP     200$,$BDADR     ;MORE ADDRESSES TO GO ?
076350  001413                          BEQ     4$              ;NO, BRANCH
076352  004737  015526                  JSR     PC,CNTRC        ;GOTO MONITOR IF CNTRC
076356  127737  102554  003606          CMPB    @$BDADR,GBITE   ;CORRECT CODE ?
076364  001761                          BEQ     3$              ;YES, BRANCH
076366  117737  102544  001142          MOVB    @$BDADR,$BDDAT  ;SET UP TYPEOUT
076374  104123                          EMT     123
076376  000207                          RTS     PC

                                        ;******* CHECK C BIT CLEARS AD3 *******
076400  112777  000002  104236  4$:     MOVB    #CBIT,@CSR      ;CLEAR THE THING
076406  012700  001000                  MOV     #1000,R0        ;SET UP DELAY
076412  005300                  5$:     DEC     R0              ;DECREASE BY ONE
076414  001376                          BNE     5$              ;LOOP TILL = 0
076416  004737  015526                  JSR     PC,CNTRC        ;GOTO MONITOR IF CNTRC
076422  105777  104776                  TSTB    @AD3            ;171003 = 0 ?
076426  001411                          BEQ     6$              ;YES, BRANCH
076430  005037  001140                  CLR     $GDDAT          ;SET UP GOOD DATA
076434  013737  003424  001136          MOV     AD3,$BDADR      ;SET UP 171003
076442  117737  102470  001142          MOVB    @$BDADR,$BDDAT  ;SET UP BAD DATA
076450  104135                          EMT     135
076452  000402                  6$:     BR      A20TE5
```

```
1339 076454  000045                        BR      A20TE4 ;RFPLACE ABOVE BRANCH WITH NOP TO LOOP
1340
1341                                       ;******* VARIABLES  *******
1342 076456  000000                 200$:  .WORD   0               ;STORAGE FOR EXIT ADDRESS
```

```
1344                                    ;*****************************************************************************
1345                                    ;TST5-DATA (NOT = 0) IS LOADED INTO THE CONVERTED CHANNEL BITS(4-7) OF THE THIRD
1346                                    ;ADDRESS. THIS SHOULD CAUSE THE ERROR BIT (BIT 2) OF THE FOURTH ADDRESS TO SET.
1347                                    ;*****************************************************************************
1348
1349 076460                            A20TE5:
                                        ;;************************************************************
     076460  000004                    TST42:  SCOPE
     076462  012737  000042  001212            MOV     #42,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
1350 076470  005037  001142                    CLR     $BDDAT          ;CLEAR THIS WORD
1351 076474  112777  000002  104142            MOVB    #CBIT,@CSR      ;CLEAR THE WORLD
1352 076502  012700  001000                    MOV     #1000,R0        ;SET UP DELAY
1353 076506  005300                    6$:     DEC     R0              ;
1354 076510  001376                            BNE     6$              ;LOOP TILL ZERO
1355 076512  004737  015526                    JSR     PC,CNTRC        ;GOTO MONITOR IF CNTRC
1356 076516  012737  000010  076742            MOV     #10,200$        ;SET UP FIRST BIT
1357 076524  022737  000304  003606            CMP     #304,GBITE      ;3 WIRE GENERIC CODE ?
1358 076532  001402                            BEQ     1$              ;YES, BRANCH
1359 076534  106337  076742                    ASLB    200$            ;STEP ONE PATTERN
1360
1361 076540  113777  076742  104654    1$:     MOVB    200$,@AD2       ;SET ERROR CONDITION
1362 076546  117737  104072  001142            MOVB    @CSR,$BDDAT     ;SAVE THE CSR
1363 076554  004737  015526                    JSR     PC,CNTRC        ;GOTO MONITOR IF CNTRC
1364 076560  105737  001142                    TSTB    $BDDAT          ;F BIT SET ?
1365 076564  100407                            BMI     2$              ;YES, BRANCH
1366 076566  013737  002644  001136            MOV     CSR,$BDADR      ;
1367 076574  012737  000200  001140            MOV     #200,$GDDAT     ;
1368 076602  104127                            EMT     127
1369
1370 076604  012737  000004  001140    2$:     MOV     #4,$GDDAT       ;SET UP GOOD DATA
1371 076612  013737  076742  003542            MOV     200$,ACOUNT     ;SET UP DATA
1372 076620  117737  104600  001142            MOVB    @AD3,$BDDAT     ;SAVE DATA
1373 076626  013737  003424  001136            MOV     AD3,$BDADR      ;SET UP FAILING ADDRESS
1374 076634  004737  015526                    JSR     PC,CNTRC        ;GOTO MONITOR IF CNTRC
1375 076640  123737  001140  001142            CMPB    $GDDAT,$BDDAT   ;CORRECT DATA ?
1376 076646  001401                            BEQ     3$              ;YES, BRANCH
1377 076650  104137                            EMT     137
1378
1379 076652  106337  076742            3$:     ASLB    200$            ;MAKE NEXT PATTERN
1380
1381                                    ;******* CHECK C BIT CLEARS AD3 *******
1382 076656  112777  000002  103760            MOVB    #CBIT,@CSR      ;CLEAR THE THING
1383 076664  012700  001000                    MOV     #1000,R0        ;SET UP DELAY
1384 076670  005300                    4$:     DEC     R0              ;DECREASE BY ONE
1385 076672  001376                            BNE     4$              ;LOOP TILL = 0
1386 076674  004737  015526                    JSR     PC,CNTRC        ;GOTO MONITOR IF CNTRC
1387 076700  105777  104520                    TSTB    @AD3            ;171003 = 0 ?
1388 076704  001411                            BEQ     5$              ;YES, BRANCH
1389 076706  005037  001140                    CLR     $GDDAT          ;SET UP GOOD DATA
1390 076712  013737  003424  001136            MOV     AD3,$BDADR      ;SET UP 171003
1391 076720  117737  102212  001142            MOVB    @$BDADR,$BDDAT  ;SET UP BAD DATA
1392 076726  104135                            EMT     135
1393 076730  105737  076742            5$:     TSTB    200$            ;200$ = 0 ?
1394 076734  001301                            BNE     1$              ;NO, BRANCH
1395 076736  000402                            BR      A20TE6          ;ONWARD BRANCH
1396 076740  000647                            BR      A20TE5 ;REPLACE ABOVE BRANCH WITH NOP TO LOOP
1397
```

```
1398                                    ;******* CONSTANTS *******
1399 076742  000000                200$:   .WORD   0       ;DATA PATTERN STROAGE
```

```
1401                            ;*****************************************************************************
1402                            ;TST6-WITH THE ERROR BIT IS SET THE INTERRUPT SYSTEM IS TURNED ON. AN  INTERRUPT
1403                            ;SHOULD OCCUR, CAUSING THE IAR (171376) TO CONTAIN THE DEVICE ADDRESS. THE C BIT
1404                            ;IS THEN SET AND THE ERROR AND THE INTERRUPTS ARE CHECKED TO BE CLEARED.
1405                            ;*****************************************************************************
1406
1407 076744                     A20TE6:
                                ;;************************************************************
      076744  000004            TST43:  SCOPE
      076746  012737  000043  001212    MOV     #43,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
1408  076754  004737  112556            JSR     PC,PRIO         ;SET PRIORITY ZERO
1409  076760  005037  001142            CLR     $BDDAT          ;CLEAR THIS WORD
1410  076764  004737  015526            JSR     PC,CNTRC        ;EXIT IF ^C
1411  076770  012777  077062  103652    MOV     #2$,@VECTO      ;SET UP NEW VECTOR
1412
1413                                    ;****** CHECK INTERRUPT ON ERROR *******
1414  076776  112777  000002  103640    MOVB    #CBIT,@CSR      ;SET CBIT
1415  077004  012700  001000            MOV     #1000,R0        ;SET UP DELAY
1416  077010  005300            1$:     DEC     R0              ;DECREASE BY ONE
1417  077012  001376                    BNE     1$              ;LOOP TILL = 0
1418  077014  013737  003422  001136    MOV     AD2,$BDADR      ;IN CASE OF TIMEOUT
1419  077022  112777  000200  104372    MOVB    #200,@AD2       ;SET 171002 = 200 (ILLEGAL)
1420  077030  112777  000100  103606    MOVB    #EBIT,@CSR      ;ENABLE INTERRUPTS
1421  077036  000240                    NOP                     ;WAIT FOR INTERRUPTS
1422  077040  000240                    NOP                     ;WAIT FOR INTERRUPTS
1423  077042  142777  000100  103574    BICB    #EBIT,@CSR      ;DISABLE INTERRUPTS
1424  077050  004737  015526            JSR     PC,CNTRC        ;GOTO MONITOR IF CNTRC
1425  077054  000240                    NOP                     ;DALLY A BIT
1426  077056  104125                    EMT     125
1427  077060  000465                    BR      4$
1428
1429  077062  005066  000002    2$:     CLR     2(SP)           ;LSI11 EXTRA CODE
1430  077066  012716  077074            MOV     #3$,(SP)        ;TO LOAD PSW
1431  077072  000002                    RTI                     ;THIS ONE TOO !
1432  077074                    3$:
      077074  013737  002636  003446    MOV     K,ZLOOP
      077102  005037  003444    65$:    CLR     YLOOP           ;WAIT
      077106  005237  003444    64$:    INC     YLOOP
      077112  023727  003444  000001    CMP     YLOOP,#1
      077120  001372                    BNE     64$
      077122  005337  003446            DEC     ZLOOP
      077126  001365                    BNE     65$
1433  077130  132777  000200  103506            BITB    #FBIT,@CSR              ;ANY INTERRUPTS ?
1434  077136  001001                            BNE     7$                      ;YES, BRANCH
1435  077140  000422                            BR      8$              ;ERROR CONDITION
1436  077142  005037  003502            7$:     CLR     TEMP                    ;START FROM SRATCH
1437  077146  117737  103474  003502            MOVB    @IAR,TEMP               ;STORE ADDRESS
1438  077154  123737  003502  003416            CMPB    TEMP,AD0                ;CORRECT INTERRUPT ADDRESS ?
1439  077162  001424                            BEQ     4$                      ;YES, BRANCH
1440  077164  152777  000001  103452            BISB    #RBIT,@CSR              ;PREPARE TO CLEAR
1441  077172  062737  002660  003502            ADD     #BASE,TEMP              ;MAKE AN ADDRESS
1442  077200  105777  104276                    TSTB    @TEMP                   ;DO IT
1443  077204  000733                            BR      3$                      ;TRY AGAIN
1444  077206  004737  015526            8$:     JSR     PC,CNTRC        ;GOTO MONITOR IF CNTRC
1445  077212  013737  003502  001136            MOV     TEMP,$BDADR     ;SET UP ADDRESS
1446  077220  005037  001140                    CLR     $GDDAT          ;SET UP GOOD DATA
1447  077224  113737  003502  001142            MOVB    TEMP,$BDDAT     ;SET UP BAD DATA
```

```
1448 077232  104 40                          EMT     140
1449
1450                                         ;******* CHECK CBIT CLEARS INTERRUPT *******
1451 077234  012777  077342  103406  4$:     MOV     #IOTRAP,@VECTO  ;SET UP TRAP VECTOR
1452 077242  112777  000002  103374          MOVB    #CBIT,@CSR      ;SET C BIT
1453 077250  012700  001000                  MOV     #1000,R0        ;SET UP DELAY
1454 077254  005300                  5$:     DEC     R0              ;DECREASE BY ONE
1455 077256  001376                          BNE     5$              ;LOOP TILL = 0
1456 077260  112777  000100  103356          MOVB    #EBIT,@CSR      ;ENABLE INTERRUPTS
1457 077266  000240                          NOP                     ;WAIT FOR INTERRUPT
1458 077270  000240                          NOP                     ;WAIT FOR INTERRUPT
1459 077272  142777  000100  103344          BICB    #EBIT,@CSR      ;CLEAR BIT
1460 077300  000240                          NOP                     ;JUST IN CASE
1461 077302  004737  015526                  JSR     PC,CNTRC        ;GOTO MONITOR IF CNTRC
1462 077306  105777  104112                  TSTB    @AD3            ;171003 = 0 ?
1463 077312  001411                          BEQ     6$              ;YES, BRANCH
1464 077314  005037  001140                  CLR     $GDDAT          ;SET UP GOOD DATA
1465 077320  013737  003424  001136          MOV     AD3,$BDADR      ;SET UP 171003
1466 077326  117737  101604  001142          MOVB    @$BDADR,$BDDAT  ;SET UP BAD DATA
1467 077334  104135                          EMT     135
1468
1469 077336  000412                  6$:     BR      A20TE7
1470 077340  000601                          BR      A20TE6 ;REPLACE ABOVE BRANCH WITH NOP TO LOOP
1471
```

```
1473                                            ;******* UNEXPECTED INTERRUPT HANDLER *******
1474 077342  012637  C01136          IOTRAP: MOV     (SP)+,$BDADR    ;SAVE PC
1475 077346  012746  077354                  MOV     #1$,-(SP)       ;TO LOAD PC
1476 077352  000002                          RTI                     ;THIS ONE TOO !
1477 077354  004737  015526          1$:     JSR     PC,CNTRC        ;GOTO MONITOR IF CNTRC
1478 077360  104126                          EMT     126
1479 077362  000207                          RTS     PC
```

```
                                ;*****************************************************************************
                                ;TST7-THE ERROR BIT IS SET, THEN THE R BIT IS SET TO CLEAR THE  ERROR  BIT.  THE
                                ;INTERRUPT SYSTEM IS THEN SET UP AND CLEARED BY SETTING THE R BIT.
                                ;*****************************************************************************

                                A20TE7:
                                ;;*************************************************************
077364  000004                  TST44:  SCOPE
077366  012737  000044  001212          MOV     #44,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
077374  004737  015526                  JSR     PC,CNTRC        ;GOTO MONITOR IF CNTRC
077400  005037  001142                  CLR     $BDDAT          ;SET UP BAD DATA
077404  005037  001140                  CLR     $GDDAT          ;SET UP GOOD DATA
077410  012777  077342  103232          MOV     #IOTRAP,@VECTO  ;SET UP NEW VECTOR
077416  013737  003424  001136          MOV     AD3,$BDADR      ;SET UP ADDRESS
077424  012777  000340  103220          MOV     #340,@VECTOA    ;SET PRIORITY SEVEN

                                        ;******* CHECK THAT R BIT CLEARS FLAG *******
077432  112777  000002  103204          MOVB    #CBIT,@CSR      ;SET C BIT
077440  012700  001000                  MOV     #1000,R0        ;SET UP DELAY
077444  005300                  1$:     DEC     R0              ;DECREASE BY ONE
077446  001376                          BNE     1$              ;LOOP TILL = 0
077450  112777  000200  103744          MOVB    #200,@AD2       ;SET 171002 = 200 (ILLEGAL)
077456  112777  000001  103160          MOVB    #RBIT,@CSR      ;GET READY TO CLEAR FLAG
077464  013737  003424  001136          MOV     AD3,$BDADR      ;SET UP DATA
077472  105777  101440                  TSTB    @$BDADR         ;CLEAR THIS DEVICE BY READING
077476  004737  015526                  JSR     PC,CNTRC        ;GOTO MONITOR IF CNTRC
077502  105777  101430                  TSTB    @$BDADR         ;ERROR FLAG RESET ?
077506  001401                          BEQ     2$              ;YES, BRANCH
077510  104141                          EMT     141

                                        ;****** CHECK THAT INTERRUPT IS CLEARED *******
077512  005077  103134          2$:     CLR     @VECTOA         ;SET TRAP PRIORITY = 0
077516  012777  077570  103124          MOV     #3$,@VECTO      ;SET UP VECTOR
077524  112777  000200  103670          MOVB    #200,@AD2       ;SET UP ILLEGAL CONDITION
077532  112777  000001  103104          MOVB    #RBIT,@CSR      ;CLEAR THE CONDITION
077540  105777  103652                  TSTB    @AD0            ;THIS DEVICE ONLY
077544  112777  000100  103072          MOVB    #EBIT,@CSR      ;ENABLE INTERRUPTS
077552  000240                          NOP                     ;WAIT FOR INTERRUPT
077554  000240                          NOP                     ;WAIT FOR INTERRUPT
077556  142777  000100  103060          BICB    #EBIT,@CSR      ;DISABLE INTERRUPTS
077564  000240                          NOP                     ;WAIT FOR INTERRUPT
077566  000453                          BR      4$

077570                          3$:
077570  013737  002636  003446          MOV     K,ZLOOP
077576  005037  003444          65$:    CLR     YLOOP           ;WAIT
077602  005237  003444          64$:    INC     YLOOP
077606  023727  003444  000001          CMP     YLOOP,#1
077614  001372                          BNE     64$
077616  005337  003446                  DEC     ZLOOP
077622  001365                          BNE     65$
077624  132777  000200  103012          BITB    #FBIT,@CSR              ;ANY INTERRUPTS ?
077632  001001                          BNE     7$                      ;YES, BRANCH
077634  000002                          RTI
077636  005037  003502          7$:     CLR     TEMP                    ;START FROM SRATCH
077642  117737  103000  003502          MOVB    @IAR,TEMP               ;STORE ADDRESS
077650  123737  003502  003416          CMPB    TEMP,AD0                ;CORRECT INTERRUPT ADDRESS ?
```

```
1528 077656  001411                          BEQ     8$                      ;YES, BRANCH
1529 077660  063737  C02660  003502          ADD     BASE,TEMP               ;MAKE NEXT ADDRESS
1530 077666  152777  000001  102750          BISB    #RBIT,@CSR              ;PREPARE TO CLEAR
1531 077674  105777  103602                  TSTB    @TEMP                   ;DO IT
1532 077700  000733                          BR      3$                      ;TRY AGAIN
1533 077702  004737  015526          8$:     JSR     PC,CNTRC        ;GOTO MONITOR IF CNTRC
1534 077706  104142                          EMT     142
1535 077710  012777  077342  102732          MOV     #IOTRAP,@VECTO  ;RESET TRAP VECTOR
1536
1537                                          ;******* CHECK C BIT CLEARS AD3 *******
1538 077716  004737  112646          4$:     JSR     PC,PRI7                 ;SET CPU PRIORITY TO 7
1539 077722  112777  000002  102714          MOVB    #CBIT,@CSR              ;CLEAR THE THING
1540 077730  012700  001000                  MOV     #1000,R0                ;SET UP DELAY
1541 077734  005300                  5$:     DEC     R0                      ;DECREASE BY ONE
1542 077736  001376                          BNE     5$                      ;LOOP TILL = 0
1543 077740  004737  015526                  JSR     PC,CNTRC                ;GOTO MONITOR IF CNTRC
1544 077744  105777  103454                  TSTB    @AD3                    ;171003 = 0 ?
1545 077750  001411                          BEQ     6$                      ;YES, BRANCH
1546 077752  005037  001140                  CLR     $GDDAT                  ;SET UP GOOD DATA
1547 077756  013737  003424  001136          MOV     AD3,$BDADR              ;SET UP 171003
1548 077764  117737  101146  001142          MOVB    @$BDADR,$BDDAT  ;SET UP BAD DATA
1549 077772  104135                          EMT     135
1550
1551 077774  000402                  6$:     BR      A20T10
1552 077776  000137  077364                  JMP     A20TE7 ;REPLACE ABOVE BRANCH WITH NOP TO LOOP
```

```
                                ;*****************************************************************************
                                ;TST10-THIS TEST CHECKS THAT THE CHANNEL NUMBER JUST WRITTEN CAN BE READ BACK
                                ;IMMEDIATELY. THE CHANNEL IS LOADED INTO BITS 0-3 AND READ BACK FROM BITS 0-3 OF
                                ;ADDRESS TWO.
                                ;       R1=DATA PATTERN
                                ;*****************************************************************************

100C02                          A20T10:
                                ;;************************************************************
100002  000004                  TST45:  SCOPE
100004  012737  000045  001212          MOV     #45,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
100012  012701  177777                  MOV     #-1,R1          ;SET UP PATTERN
100016  005037  001140                  CLR     $GDDAT          ;SET UP GOOD DATA
100022  005037  001142                  CLR     $BDDAT          ;SET UP BAD DATA
100026  112777  000002  102610          MOVB    #CBIT,@CSR      ;SET CBIT
100034  012700  001000                  MOV     #1000,R0        ;SET UP DELAY
100040  005300                  1$:     DEC     R0              ;DECREASE THE COUNT
100042  001376                          BNE     1$              ;LOOP TILL=0

100044  004737  015526          2$:     JSR     PC,CNTRC        ;EXIT IF ^C
100050  005201                          INC     R1              ;MAKE NEXT PATTERN
100052  013737  003422  001136          MOV     AD2,$BDADR      ;SET UP ADDRESS
100060  110177  101052                  MOVB    R1,@$BDADR      ;LOAD THE CHANNEL NUMBER
100064  117737  101046  001142          MOVB    @$BDADR,$BDDAT  ;GET DATA FROM A20
100072  042737  000360  001142          BIC     #360,$BDDAT     ;STRIP BITS TO IGNORE
100100  010137  001140                  MOV     R1,$GDDAT       ;SET UP GOOD DATA
100104  023737  001142  001140          CMP     $BDDAT,$GDDAT   ;DATA OK?
100112  001401                          BEQ     4$              ;YES, BRANCH
100114  104143                          EMT     143

100116  112777  000002  102520  4$:     MOVB    #CBIT,@CSR      ;SET THE CBIT
100124  012700  001000                  MOV     #1000,R0        ;SET UP DELAY
100130  005300                  5$:     DEC     R0              ;
100132  001376                          BNE     5$              ;LOOP TILL = 0
100134  013737  002644  001136          MOV     CSR,$BDADR      ;GET CSR
100142  105777  100770                  TSTB    @$BDADR         ;CSR CLEAR?
100146  001406                          BEQ     6$              ;YES, BRANCH
100150  117737  100762  001142          MOVB    @$BDADR,$BDDAT  ;SET UP BAD DATA
100156  005037  001140                  CLR     $GDDAT          ;SET UP GOOD DATA
100162  104144                          EMT     144

100164  022701  000017          6$:     CMP     #17,R1          ;DO MORE PATTERNS ?
100170  001407                          BEQ     7$              ;NO, BRANCH
100172  022737  000324  003606          CMP     #324,GBITE      ;16 CHANNELS ?
100200  001721                          BEQ     2$              ;YES, BRANCH
100202  022701  000007                  CMP     #7,R1           ;DO MORE PATTERNS ?
100206  001316                          BNE     2$              ;YES, BRANCH

                                ;******* CHECK C BIT CLEARS AD3 *******
100210  112777  000002  102426  7$:     MOVB    #CBIT,@CSR      ;CLEAR THE THING
100216  012700  001000                  MOV     #1000,R0        ;SET UP DELAY
100222  005300                  8$:     DEC     R0              ;DECREASE BY ONE
100224  001376                          BNE     8$              ;LOOP TILL = 0
100226  004737  015526                  JSR     PC,CNTRC        ;GOTO MONITOR IF CNTRC
100232  105777  103166                  TSTB    @AD3            ;171003 = 0 ?
100236  001411                          BEQ     9$              ;YES, BRANCH
100240  005037  001140                  CLR     $GDDAT          ;SET UP GOOD DATA
```

```
1608 100244  013737  003424  001136          MOV     AD3,$BDADR      ;SET UP 171003
1609 100252  117737  100660  001142          MOVB    @$BDADR,$BDDAT  ;SET UP BAD DATA
1610 100260  104135                          EMT     135
1611
1612 100262  000401                  9$:     BR      A20T11
1613 100264  000646                          BR      A20T10  ;REPLACE ABOVE BRANCH WITH A NOP TO LOOP
```

```
                                ;*****************************************************************************
                                ;TST11-THIS TEST OVERLOADS THE NUMBER OF CHANNELS INPUT AND CHECKS FOR THE ERROR
                                ;BIT BEING SET. THE C BIT IS THEN SET, AND THE OVERLOAD CONDITION IS CHECKED  TO
                                ;BE CLEARED.
                                ;*****************************************************************************

100266                          A20T11:
                                ;;************************************************************
100266  000004                  TST46:  SCOPE
100270  012737  000046  001212          MOV     #46,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
100276  005037  001142          CLR     $BDDAT          ;CLEAR THIS WORD
100302  004737  015526          JSR     PC,CNTRC        ;EXIT IF ^C
100306  013737  003424  001136  MOV     AD3,$BDADR      ;SET UP ERROR ADDRESS

                                ;******* ESTABLISH SYNC WITH EOC *******
100314  112777  000002  102322  MOVB    #CBIT,@CSR      ;CLEAR THE WORLD
100322  012700  001000          MOV     #1000,R0        ;SET UP DELAY
100326  005300          11$:    DEC     R0              ;DECREASE THE COUNT
100330  001376                  BNE     11$             ;LOOP TILL = 0
100332  112777  000001  103062  MOVB    #1,@AD2         ;LOAD A CHANNEL
100340  005000                  CLR     R0              ;SET UP DELAY
100342  132777  000002  103054  12$:    BITB    #BIT1,@AD3      ;DONE BIT SET ?
100350  001013                  BNE     14$             ;YES, BRANCH
100352  005300                  DEC     R0              ;DECREASE DELAY
100354  001372                  BNE     12$             ;LOOP TILL ZERO
100356  117737  103042  001142  MOVB    @AD3,$BDDAT     ;
100364  012737  000002  001140  MOV     #2,$GDDAT       ;
100372  104146                  EMT     146
100374  000137  101060          JMP     A20T12          ;

                                ;******* LOAD FIRST CHANNEL *******
100400  112777  000002  102236  14$:    MOVB    #CBIT,@CSR      ;CLEAR THE WORLD
100406  012700  001000          MOV     #1000,R0        ;SET UP DELAY
100412  005300          1$:     DEC     R0              ;DECREASE THE COUNT
100414  001376                  BNE     1$              ;LOOP TILL = 0
100416  012737  000001  001140  MOV     #1,$GDDAT       ;EXPECTED DATA
100424  112777  000007  102770  MOVB    #7,@AD2         ;LOAD CHANNEL
100432  117737  100500  001142  MOVB    @$BDADR,$BDDAT  ;GET STATUS
100440  122737  000001  001142  CMPB    #1,$BDDAT       ;ONLY ACTIVE BIT?
100446  001410                  BEQ     2$              ;YES, BRANCH
100450  005037  001140          CLR     $GDDAT          ;SET UP GOOD DATA
100454  105737  001142          TSTB    $BDDAT          ;IS ALL CLEAR ?
100460  001403                  BEQ     2$              ;YES, BRANCH
100462  104122                  EMT     122
100464  000137  101060          JMP     A20T12          ;GO TO NEXT TEST
```

```
1659                                                    ;******* LOAD SECOND CHANNEL *******
1660 100470  112777  000007  102724  2$:     MOVB    #7,@AD2         ;LOAD CHANNEL
1661 100476  117737  100434  001142          MOVB    @$BDADR,$BDDAT  ;GET STATUS
1662 100504  122737  000201  001142          CMPB    #201,$BDDAT     ;ACTIVE & BUSY
1663 100512  001412                          BEQ     3$              ;YES, BRANCH
1664 100514  012737  000200  001140          MOV     #200,$GDDAT     ;SET UP GOOD DATA
1665 100522  122737  000200  001142          CMPB    #200,$BDDAT     ;ONLY BUSY ?
1666 100530  001403                          BEQ     3$              ;YES, BRANCH
1667 100532  104124                          EMT     124
1668 100534  000137  101060                  JMP     A20T12          ;GO TO NEXT TEST
1669
1670                                                    ;******* LOAD THIRD CHANNEL *******
1671 100540  112777  000007  102654  3$:     MOVB    #7,@AD2         ;LOAD CHANNEL
1672 100546  117737  100364  001142          MOVB    @$BDADR,$BDDAT  ;GET STATUS
1673 100554  122737  000205  001142          CMPB    #205,$BDDAT     ;ACTIVE,BUSY,& ERROR ONLY?
1674 100562  001412                          BEQ     4$              ;YES, BRANCH
1675 100564  012737  000204  001140          MOV     #204,$GDDAT     ;SET UP GOOD DATA
1676 100572  122737  000204  001142          CMPB    #204,$BDDAT     ;BUSY & ERROR ?
1677 100600  001403                          BEQ     4$              ;YES, BRANCH
1678 100602  104130                          EMT     130
1679 100604  000137  101060                  JMP     A20T12          ;GO TO NEXT TEST
1680
1681                                                    ;******* LOAD FOURTH CHANNEL *******
1682 100610  112777  000007  102604  4$:     MOVB    #7,@AD2         ;LOAD CHANNEL
1683 100616  117737  100314  001142          MOVB    @$BDADR,$BDDAT  ;GET STATUS
1684 100624  122737  000205  001142          CMPB    #205,$BDDAT     ;ACTIVE,BUSY & ERROR ONLY?
1685 100632  001412                          BEQ     5$              ;YES,BRANCH
1686 100634  012737  000204  001140          MOV     #204,$GDDAT     ;SET UP GOOD DATA
1687 100642  122737  000204  001142          CMPB    #204,$BDDAT     ;BUSY & ERROR ?
1688 100650  001403                          BEQ     5$              ;YES, BRANCH
1689 100652  104131                          EMT     131
1690 100654  000137  101060                  JMP     A20T12          ;GO TO NEXT TEST
1691
1692                                                    ;******* CHECK THAT C BIT REALLY DOES CLEAR *******
1693 100660  112777  000002  101756  5$:     MOVB    #CBIT,@CSR      ;CLEAR EVERYTHING
1694 100666  012700  001000                  MOV     #1000,R0        ;SET UP DELAY
1695 100672  005300                  6$.     DEC     R0              ;DECREASE COUNT
1696 100674  001376                          BNE     6$              ;LOOP TILL = 0
1697 100676  105777  102522                  TSTB    @AD3            ;STATUS REG CLEAR ?
1698 100702  001413                          BEQ     7$              ;YES, BRANCH
1699 100704  013737  003424  001136          MOV     AD3,$BDADR      ;SET UP ERROR TYPEOUT
1700 100712  117737  102506  001142          MOVB    @AD3,$BDDAT     ;SET UP ERROR TYPEOUT
1701 100720  005037  001140                  CLR     $GDDAT          ;SET UP ERROR TYPEOUT
1702 100724  104132                          EMT     132
1703 100726  000137  101060                  JMP     A20T12          ;GO TO NEXT TEST
```

```
1705
1706 100732  112777  000007  102462  7$:     MOVB    #7,@AD2          ;LOAD A CHANNEL
1707 100740  132777  000004  102456          BITB    #4,@AD3          ;ANY ERRORS ?
1708 100746  001414                          BEQ     8$               ;NO,BRANCH
1709 100750  012737  000004  001140          MOV     #4,$GDDAT        ;SET UP ERROR TYPEOUT
1710 100756  117737  102442  001142          MOVB    @AD3,$BDDAT      ;SET UP ERROR TYPEOUT
1711 100764  013737  003424  001136          MOV     AD3,$BDADR       ;SET UP ERROR TYPEOUT
1712 100772  104133                          EMT     133
1713 100774  000137  101060                  JMP     A20T12           ;GO TO NEXT TEST
1714
1715                                         ;******* CHECK C BIT CLEARS AD3 *******
1716 101000  112777  000002  101636  8$:     MOVB    #CBIT,@CSR       ;CLEAR THE THING
1717 101006  012700  001000                  MOV     #1000,R0         ;SET UP DELAY
1718 101012  005300                  9$:     DEC     R0               ;DECREASE BY ONE
1719 101014  001376                          BNE     9$               ;LOOP TILL = 0
1720 101016  105777  102402                  TSTB    @AD3             ;171003 = 0 ?
1721 101022  001413                          BEQ     10$              ;YES, BRANCH
1722 101024  005037  001140                  CLR     $GDDAT           ;SET UP GOOD DATA
1723 101030  013737  003424  001136          MOV     AD3,$BDADR       ;SET UP 171003
1724 101036  117737  100074  001142          MOVB    @$BDADR,$BDDAT   ;SET UP BAD DATA
1725 101044  104135                          EMT     135
1726 101046  000137  101060                  JMP     A20T12           ;GO TO NEXT TEST
1727
1728 101052  000402                  10$:    BR      A20T12
1729 101054  000137  100266                  JMP     A20T11 ;REPLACE ABOVE BRANCH WITH NOP TO LOOP
```

```
                                ;*****************************************************************************
                                ;TST12-THIS TEST CHECKS THAT THE CHANNEL NUMBER WRITTEN CAN BE READ IN THE
                                ;CONVERTED DATA CHANNEL BITS (4-7) OF ADDRESS THREE AFTER A CONVERSION USING
                                ;DUMB MODE.
                                ;       R1=DATA PATTERN
                                ;       R2=EXPECTED DATA
                                ;*****************************************************************************

1739 101060                     A20T12:
                                ;;************************************************************************
     101060  000004             TST47:  SCOPE
     101062  012737  000047  001212      MOV     #47,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
1740 101070  005037  001142              CLR     $BDDAT          ;SET THIS TO 0
1741 101074  005001                      CLR     R1              ;SET UP DATA PATTERN
1742 101076  005002                      CLR     R2              ;SET UP EXPECTED DATA PATTERN
1743
1744 101100  004737  015526      1$:     JSR     PC,CNTRC        ;EXIT IF ^C
1745 101104  112777  000002  101532      MOVB    #CBIT,@CSR      ;SET THE C BIT
1746 101112  012700  001000              MOV     #1000,R0        ;SET UP DELAY
1747 101116  005300              2$:     DEC     R0              ;DECREASE COUNT
1748 101120  001376                      BNE     2$              ;LOOP TILL = 0
1749 101122  110177  102274              MOVB    R1,@AD2         ;LOAD CHANNEL
1750 101126  004737  015526              JSR     PC,CNTRC        ;GOTO MONITOR IF CNTRC
1751 101132  127727  102266  000001      CMPB    @AD3,#1         ;STATUS REG OK?
1752 101140  001416                      BEQ     3$              ;YES, BRANCH
1753 101142  127727  102256  000000      CMPB    @AD3,#0         ;BUSY ONLY ?
1754 101150  001412                      BEQ     3$              ;YES, BRANCH
1755 101152  117737  102246  001142      MOVB    @AD3,$BDDAT     ;SETUP BAD DATA
1756 101160  013737  003424  001136      MOV     AD3,$BDADR      ;SETUP BAD ADDRESS
1757 101166  012737  000001  001140      MOV     #1,$GDDAT       ;SETUP GOOD DATA
1758 101174  104145                      EMT     145
1759
1760 101176  005000              3$:     CLR     R0              ;SET UP 74MS DELAY
1761 101200  012703  000001              MOV     #1,R3           ;MORE SET UP
1762 101204  004737  015526              JSR     PC,CNTRC        ;GOTO MONITOR IF CNTRC
1763 101210  137727  102210  000002  4$: BITB    @AD3,#2         ;DONE BIT SET?
1764 101216  001016                      BNE     5$              ;YES, BRANCH
1765 101220  005300                      DEC     R0              ;DECREASE COUNT
1766 101222  001372                      BNE     4$              ;LOOP TILL = 0
1767 101224  005303                      DEC     R3              ;DECREASE COUNT
1768 101226  001370                      BNE     4$              ;LOOP TILL = 0
1769 101230  117737  102170  001142      MOVB    @AD3,$BDDAT     ;SETUP BAD DATA
1770 101236  012737  000002  001140      MOV     #2,$GDDAT       ;SETUP GOOD DATA
1771 101244  013737  003424  001136      MOV     AD3,$BDADR      ;SETUP BAD ADDRESS
1772 101252  104146                      EMT     146
1773
1774 101254  004737  015526      5$:     JSR     PC,CNTRC        ;GOTO MONITOR IF CNTRC
1775 101260  127727  102140  000002      CMPB    @AD3,#2         ;ANY OTHER BITS THAN DONE?
1776 101266  001416                      BEQ     6$              ;NO, BRANCH
1777 101270  127727  102130  000003      CMPB    @AD3,#3         ;DONE AND ACTIVE BITS ?
1778 101276  001412                      BEQ     6$              ;YES, BRANCH
1779 101300  117737  102120  001142      MOVB    @AD3,$BDDAT     ;SETUP BAD DATA
1780 101306  012737  000002  001140      MOV     #2,$GDDAT       ;SETUP GOOD DATA
1781 101314  013737  003424  001136      MOV     AD3,$BDADR      ;SETUP BAD ADDESS
1782 101322  104147                      EMT     147
```

```
1784 101324  127702  102072          6$:   CMPB   @AD2,R2          ;INPUT CHANNEL=OUTPUT CHANNEL?
1785 101330  001414                        BEQ    7$              ;YES, BRANCH
1786 101332  117737  102064  001142        MOVB   @AD2,$BDDAT     ;SETUP
1787 101340  013737  003422  001136        MOV    AD2,$BDADR      ;SETUP
1788 101346  010237  001140                MOV    R2,$GDDAT       ;SETUP
1789 101352  042737  177400  001140        BIC    #177400,$GDDAT  ;CLEAR UPPER BYTE
1790 101360  104150                        EMT    150
1791
1792 101362  005037  001142          7$:   CLR    $BDDAT          ;CLEAR THE WORD
1793 101366  132777  000200  101250        BITB   #FBIT,@CSR              ;ANY INTERRUPTS ?
1794 101374  001420                        BEQ    14$                     ;YES, BRANCH
1795 101376  117737  101244  001142        MOVB   @IAR,$BDDAT             ;STORE ADDRESS
1796 101404  123737  001142  003416        CMPB   $BDDAT,AD0              ;CORRECT INTERRUPT ADDRESS ?
1797 101412  001417                        BEQ    13$                     ;YES, BRANCH
1798 101414  063737  002660  001142        ADD    BASE,$BDDAT             ;MAKE NEXT ADDRESS
1799 101422  152777  000001  101214        BISB   #RBIT,@CSR              ;PREPARE TO CLEAR
1800 101430  105777  077506                TSTB   @$BDDAT                 ;DO IT
1801 101434  000752                        BR     7$                      ;TRY AGAIN
1802 101436  013737  002646  001136  14$:  MOV    IAR,$BDADR      ;SET UP BAD ADDRESS
1803 101444  005037  001140                CLR    $GDDAT          ;SHOULD BE ZERO
1804 101450  104140                        EMT    140
1805
1806 101452  112777  000001  101164  13$:  MOVB   #RBIT,@CSR       ;CLEAR DEVICE
1807 101460  105777  101740                TSTB   @AD3             ;DO IT
1808 101464  004737  015526                JSR    PC,CNTRC         ;GOTO MONITOR IF CNTRC
1809 101470  117737  101730  001142        MOVB   @AD3,$BDDAT
1810 101476  142737  000001  001142        BICB   #1,$BDDAT        ;IGNORE ACTIVE BIT
1811 101504  105737  001142                TSTB   $BDDAT           ;ALL CLEAR ?
1812 101510  001406                        BEQ    8$               ;YES, BRANCH
1813 101512  005037  001140                CLR    $GDDAT
1814 101516  013737  003424  001136        MOV    AD3,$BDADR
1815 101524  104170                        EMT    170
1816
1817 101526  005201                  8$:   INC    R1               ;MAKE NEXT PATTERN
1818 101530  022701  000020                CMP    #20,R1           ;LAST PATTERN ?
1819 101534  001417                        BEQ    10$              ;YES, BRANCH
1820 101536  022737  000324  003606        CMP    #324,GBITE       ;16 CHANNELS ?
1821 101544  001403                        BEQ    9$               ;NO, BRANCH
1822 101546  022701  000007                CMP    #7,R1            ;LAST PATTERN ?
1823 101552  001410                        BEQ    10$              ;YES, BRANCH
1824 101554  010102                  9$:   MOV    R1,R2            ;SAVE PATTERN
1825 101556  006302                        ASL    R2               ;SHIFT
1826 101560  006302                        ASL    R2               ;R2
1827 101562  006302                        ASL    R2               ;4
1828 101564  006302                        ASL    R2               ;TIMES
1829 101566  060102                        ADD    R1,R2            ;MAKE WORD COMPLETE
1830 101570  000137  101100                JMP    1$               ;DO SOME MORE
1831
1832                                       ;******* CHECK C BIT CLEARS AD3 *******
1833 101574  112777  000002  101042  10$:  MOVB   #CBIT,@CSR       ;CLEAR THE THING
1834 101602  012700  001000                MOV    #1000,R0         ;SET UP DELAY
1835 101606  005300                  11$:  DEC    R0               ;DECREASE BY ONE
1836 101610  001376                        BNE    11$              ;LOOP TILL = 0
1837 101612  004737  015526                JSR    PC,CNTRC         ;GOTO MONITOR IF CNTRC
1838 101616  105777  101602                TSTB   @AD3             ;171003 = 0 ?
1839 101622  001411                        BEQ    12$              ;YES, BRANCH
1840 101624  005037  001140                CLR    $GDDAT           ;SET UP GOOD DATA
```

A20 LOGIC TEST

```
1841 101630  013737  003424  001136          MOV     AD3,$BDADR      ;SET UP 171003
1842 101636  117737  007274  001142          MOVB    @$BDADR,$BDDAT  ;SET UP BAD DATA
1843 101644  104135                          EMT     135
1844
1845 101646  000402                  12$:    BR      A20T13
1846 101650  000137  101060                  JMP     A20T12 ;REPLACE ABOVE BRANCH WITH NOP TO LOOP
```

```
                                ;*****************************************************************************
                                ;TST13-THIS TEST CHECKS THAT THE CHANNEL NUMBERS WRITTEN ARE READ BACK IN THE
                                ;CONVERTED DATA CHANNEL BITS (4-7) AFTER A CONVERSION IS DONE USING SMART MODE.
                                ;THIS TEST USES THE INTERRUPT SYSTEM TO GATHER THE DATA.
                                ;       R0=TEMPORARY
                                ;       R3=ACTUAL DATA READ
                                ;       R4=TEMPORARY
                                ;*****************************************************************************

                                A20T13:
                                ;;*****************************************************************
101654  000004                  TST50:  SCOPE
101656  012737  000050  001212          MOV     #50,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
101664  005037  001140          CLR     $GDDAT          ;SET UP GOOD DATA
101670  005037  001142          CLR     $BDDAT          ;SET UP BAD DATA
101674  004737  015526          JSR     PC,CNTRC        ;EXIT IF ^C

                                ;******* CLEAR EVERYTHING, SETUP TRAPS, ENABLE INTERRUPTS *******
101700  112777  000002  100736  1$:     MOVB    #CBIT,@CSR      ;CLEAR THE THING
101706  012700  001000          MOV     #1000,R0        ;SET UP DELAY
101712  005300                  2$:     DEC     R0              ;DECREASE COUNT
101714  001376                  BNE     2$              ;SCOPE TILL = 0
101716  012777  077342  100724  MOV     #IOTRAP,@VECTO  ;SET UP UNEXPECTED INTERRUPT
101724  012777  000340  100720  MOV     #340,@VECTOA    ;SET TRAP PRIORITY TO 7
101732  004737  112556          JSR     PC,PRIO         ;SET CPU PRIORITY TO ZERO
101736  112777  000100  100700  MOVB    #EBIT,@CSR      ;ENABLE INTERRUPTS

                                ;******* LOAD 2 CHANNELS, CHECK STATUS(AD3) *******
101744  005037  003442          CLR     IFLAG           ;SET INTERRUPT FLAG = 0
101750  112777  000001  101444  MOVB    #1,@AD2         ;LOAD 1ST CHANNEL #
101756  112777  000002  101436  MOVB    #2,@AD2         ;LOAD 2CD CHANNEL #
101764  117737  101434  001142  MOVB    @AD3,$BDDAT     ;GET THE BYTE
101772  123727  001142  000201  CMPB    $BDDAT,#201     ;BUSY, ACTIVE ONLY?
102000  001414                  BEQ     3$              ;YES, BRANCH
102002  142777  000100  100634  BICB    #EBIT,@CSR      ;SHUT OFF INTERRUPTS
102010  012737  000201  001140  MOV     #201,$GDDAT     ;SET UP GOOD DATA
102016  013737  003424  001136  MOV     AD3,$BDADR      ;LOAD ADDRESS NUMBER
102024  104151                  EMT     151
102026  000137  103546          JMP     A20T14          ;ABORT TEST

                                ;******* WAIT FOR INTERRUPT #1 *******
102032  012777  103410  100610  3$:     MOV     #400$,@VECTO    ;SET UP TRAP VECTOR
102040  005004                  CLR     R4              ;SET UP DELAY
102042  005737  003442          4$:     TST     IFLAG           ;ANY INTERRUPTS OCCUR ?
102046  001022                  BNE     5$              ;YES, BRANCH
102050  162704  000001          SUB     #1,R4           ;DECREASE THE COUNT
102054  001372                  BNE     4$              ;LOOP TILL = 0
102056  142777  000100  100560  BICB    #EBIT,@CSR      ;DISABLE INTERRUPTS
102064  013737  002644  001136  MOV     CSR,$BDADR      ;SET UP ADDRESS
102072  117737  100546  001142  MOVB    @CSR,$BDDAT     ;SAVE CSR
102100  012737  000001  003542  MOV     #1,ACOUNT       ;INTERRUPT NUMBER
102106  104205                  EMT     205
102110  000137  103546          JMP     A20T14          ;ABORT TEST
```

```
1899                                                    ;******* CHECK RECV'D DATA *******
1900 102114  012777  C77342  100526  5$:      MOV     #IOTRAP,@VECTO  ;SET UP UNEXPECTED INTERRUPT
1901 102122  122703  000022                   CMPB    #22,R3          ;CHANNEL DATA OK ?
1902 102126  001416                           BEQ     10$             ;YES, BRANCH
1903 102130  142777  000100  100506           BICB    #EBIT,@CSR      ;SHUT OFF INTERRUPTS
1904 102136  010337  001142                   MOV     R3,$BDDAT       ;BAD DATA
1905 102142  012737  000022  001140           MOV     #22,$GDDAT      ;SAVE GOOD DATA
1906 102150  013737  003422  001136           MOV     AD2,$BDADR      ;SET UP ADDRESS
1907 102156  104152                           EMT     152
1908 102160  000137  103546                   JMP     A20T14          ;ABORT TEST
1909
1910                                                    ;******* LOAD 2 CHANNELS, CHECK STATUS(AD3) *******
1911 102164  005037  003442           10$:    CLR     IFLAG           ;RESET INTERRUPT FLAG
1912 102170  112777  000003  101224           MOVB    #3,@AD2         ;LOAD 3RD CHANNEL #
1913 102176  112777  000004  101216           MOVB    #4,@AD2         ;LOAD 4TH CHANNEL #
1914 102204  117737  101214  001142           MOVB    @AD3,$BDDAT     ;SAVE THE DATA
1915 102212  123727  001142  000201           CMPB    $BDDAT,#201     ;BUSY, ACTIVE ONLY?
1916 102220  001414                           BEQ     11$             ;YES, BRANCH
1917 102222  142777  000100  100414           BICB    #EBIT,@CSR      ;SHUT OFF INTERRUPTS
1918 102230  012737  000201  001140           MOV     #201,$GDDAT     ;
1919 102236  013737  003424  001136           MOV     AD3,$BDADR      ;SET UP ADDRESS NUMBER
1920 102244  104153                           EMT     153
1921 102246  000137  103546                   JMP     A20T14          ;ABORT TEST
1922
1923                                                    ;******* WAIT FOR INTERRUPT #2 *******
1924 102252  012777  103410  100370  11$:     MOV     #400$,@VECTO    ;SET UP TRAP VECTOR
1925 102260  005004                           CLR     R4              ;SET UP DELAY
1926 102262  005737  003442           12$:    TST     IFLAG           ;ANY INTERRUPTS OCCUR ?
1927 102266  001022                           BNE     13$             ;YES, BRANCH
1928 102270  162704  000001                   SUB     #1,R4           ;DECREASE THE COUNT
1929 102274  001372                           BNE     12$             ;LOOP TILL = 0
1930 102276  142777  000100  100340           BICB    #EBIT,@CSR      ;DISABLE INTERRUPTS
1931 102304  013737  002644  001136           MOV     CSR,$BDADR      ;SET UP ADDRESS
1932 102312  117737  100326  001142           MOVB    @CSR,$BDDAT     ;SAVE CSR
1933 102320  012737  000002  003542           MOV     #2,ACOUNT       ;INTERRUPT NUMBER
1934 102326  104205                           EMT     205
1935 102330  000137  103546                   JMP     A20T14          ;ABORT TEST
1936
1937                                                    ;******* CHECK RECV'D DATA *******
1938 102334  012777  077342  100306  13$:     MOV     #IOTRAP,@VECTO  ;SET UP UNEXPECTED INTERRUPT
1939 102342  122703  000044                   CMPB    #44,R3          ;CHANNEL DATA OK ?
1940 102346  001416                           BEQ     20$             ;YES, BRANCH
1941 102350  142777  000100  100266           BICB    #EBIT,@CSR      ;SHUT OFF INTERRUPTS
1942 102356  010337  001142                   MOV     R3,$BDDAT       ;BAD DATA
1943 102362  012737  000044  001140           MOV     #44,$GDDAT      ;SAVE GOOD DATA
1944 102370  013737  003422  001136           MOV     AD2,$BDADR      ;SET UP ADDRESS
1945 102376  104154                           EMT     154
1946 102400  000137  103546                   JMP     A20T14          ;ABORT TEST
```

```
                                                ;******* LOAD A CHANNEL, CHECK STATUS(AD3) *******
102404  005037  003442          20$:    CLR     IFLAG           ;RESET INTERRUPT FLAG
102410  112777  000005  101004          MOVB    #5,@AD2         ;LOAD 5TH CHANNEL #
102416  117737  101002  001142          MOVB    @AD3,$BDDAT     ;GET THE BYTE
102424  123727  001142  000201          CMPB    $BDDAT,#201     ;BUSY, ACTIVE ONLY?
102432  001414                          BEQ     21$             ;YES, BRANCH
102434  142777  000100  100202          BICB    #EBIT,@CSR      ;SHUT OFF INTERRUPTS
102442  012737  000201  001140          MOV     #201,$GDDAT     ;SET UP GOOD DATA
102450  013737  003424  001136          MOV     AD3,$BDADR      ;LOAD ADDRESS NUMBER
102456  104155                          EMT     155
102460  000137  103546                  JMP     A20T14          ;ABORT TEST

                                                ;******* WAIT FOR INTERRUPT #3 *******
102464  012777  103410  100156  21$:    MOV     #400$,@VECTO    ;SET UP TRAP VECTOR
102472  005004                          CLR     R4              ;SET UP DELAY
102474  005737  003442          22$:    TST     IFLAG           ;ANY INTERRUPTS OCCUR ?
102500  001022                          BNE     23$             ;YES, BRANCH
102502  162704  000001                  SUB     #1,R4           ;DECREASE THE COUNT
102506  001372                          BNE     22$             ;LOOP TILL = 0
102510  142777  000100  100126          BICB    #EBIT,@CSR      ;DISABLE INTERRUPTS
102516  013737  002644  001136          MOV     CSR,$BDADR      ;SET UP ADDRESS
102524  117737  100114  001142          MOVB    @CSR,$BDDAT     ;SAVE CSR
102532  012737  000003  003542          MOV     #3,ACOUNT       ;INTERRUPT NUMBER
102540  104205                          EMT     205
102542  000137  103546                  JMP     A20T14          ;ABORT TEST

                                                ;******* CHECK RECV'D DATA *******
102546  012777  077342  100074  23$:    MOV     #IOTRAP,@VECTO  ;SET UP UNEXPECTED INTERRUPT
102554  122703  000065                  CMPB    #65,R3          ;CHANNEL DATA OK ?
102560  001416                          BEQ     30$             ;YES, BRANCH
102562  142777  000100  100054          BICB    #EBIT,@CSR      ;SHUT OFF INTERRUPTS
102570  010337  001142                  MOV     R3,$BDDAT       ;BAD DATA
102574  012737  000065  001140          MOV     #65,$GDDAT      ;SAVE GOOD DATA
102602  013737  003422  001136          MOV     AD2,$BDADR      ;SET UP ADDRESS
102610  104156                          EMT     156
102612  000137  103546                  JMP     A20T14          ;ABORT TEST

                                                ;******* CHECK STATUS(AD3) *******
102616  005037  003442          30$:    CLR     IFLAG           ;RESET INTERRUPT FLAG
102622  117737  100576  001142          MOVB    @AD3,$BDDAT     ;GET THE BYTE
102630  123727  001142  000001          CMPB    $BDDAT,#1       ;ACTIVE ONLY?
102636  001414                          BEQ     31$             ;YES, BRANCH
102640  142777  000100  077776          BICB    #EBIT,@CSR      ;SHUT OFF INTERRUPTS
102646  012737  000001  001140          MOV     #1,$GDDAT       ;SET UP GOOD DATA
102654  013737  003424  001136          MOV     AD3,$BDADR      ;LOAD ADDRESS NUMBER
102662  104157                          EMT     157
102664  000137  103546                  JMP     A20T14          ;ABORT TEST
```

```
                                        ;******* WAIT FOR INTERRUPT #4 *******
102670  012777  103410  077752  31$:   MOV     #400$,@VECTO    ;SET UP TRAP VECTOR
102676  005004                        CLR     R4              ;SET UP DELAY
102700  005737  003442          32$:   TST     IFLAG           ;ANY INTERRUPTS OCCUR ?
102704  001022                        BNE     33$             ;YES, BRANCH
102706  162704  000001                SUB     #1,R4           ;DECREASE THE COUNT
102712  001372                        BNE     32$             ;LOOP TILL = 0
102714  142777  000100  077722        BICB    #EBIT,@CSR      ;DISABLE INTERRUPTS
102722  013737  002644  001136        MOV     CSR,$BDADR      ;SET UP ADDRESS
102730  117737  077710  001142        MOVB    @CSR,$BDDAT     ;SAVE CSR
102736  012737  000004  003542        MOV     #4,ACOUNT       ;INTERRUPT NUMBER
102744  104205                        EMT     205
102746  000137  103546                JMP     A20T14          ;ABORT TEST

                                        ;******* CHECK RECV'D DATA *******
102752  012777  077342  077670  33$:   MOV     #IOTRAP,@VECTO  ;SET UP UNEXPECTED INTERRUPT
102760  122703  000105                CMPB    #105,R3         ;CHANNEL DATA OK ?
102764  001416                        BEQ     40$             ;YES, BRANCH
102766  142777  000100  077650        BICB    #EBIT,@CSR      ;SHUT OFF INTERRUPTS
102774  010337  001142                MOV     R3,$BDDAT       ;BAD DATA
103000  012737  000105  001140        MOV     #105,$GDDAT     ;SAVE GOOD DATA
103006  013737  003422  001136        MOV     AD2,$BDADR      ;SET UP ADDRESS
103014  104160                        EMT     160
103016  000137  103546                JMP     A20T14          ;ABORT TEST

                                        ;******* CHECK STATUS(AD3) *******
103022  005037  003442          40$:   CLR     IFLAG           ;RESET INTERRUPT FLAG
103026  117737  100372  001142        MOVB    @AD3,$BDDAT     ;SAVE THE DATA
103034  123727  001142  000001        CMPB    $BDDAT,#1       ; ACTIVE ONLY?
103042  001414                        BEQ     41$             ;YES, BRANCH
103044  142777  000100  077572        BICB    #EBIT,@CSR      ;SHUT OFF INTERRUPTS
103052  012737  000001  001140        MOV     #1,$GDDAT       ;
103060  013737  003424  001136        MOV     AD3,$BDADR      ;SET UP ADDRESS NUMBER
103066  104161                        EMT     161
103070  000137  103546                JMP     A20114          ;ABORT TEST

                                        ;******* WAIT FOR INTERRUPT #5 *******
103074  012777  103410  077546  41$:   MOV     #400$,@VECTO    ;SET UP TRAP VECTOR
103102  005004                        CLR     R4              ;SET UP DELAY
103104  005737  003442          42$:   TST     IFLAG           ;ANY INTERRUPTS OCCUR ?
103110  001022                        BNE     43$             ;YES, BRANCH
103112  162704  000001                SUB     #1,R4           ;DECREASE THE COUNT
103116  001372                        BNE     42$             ;LOOP TILL = 0
103120  142777  000100  077516        BICB    #EBIT,@CSR      ;DISABLE INTERRUPTS
103126  013737  002644  001136        MOV     CSR,$BDADR      ;SET UP ADDRESS
103134  117737  077504  001142        MOVB    @CSR,$BDDAT     ;SAVE CSR
103142  012737  000005  003542        MOV     #5,ACOUNT       ;INTERRUPT NUMBER
103150  104203                        EMT     203
103152  000137  103546                JMP     A20T14          ;ABORT TEST
```

```
2048                                            ;******* CHECK RECV'D DATA *******
2049 103156  012777  077342  077464  43$:      MOV     #IOTRAP,@VECTO  ;SET UP UNEXPECTED INTERRUPT
2050 103164  122703  000125                     CMPB    #125,R3         ;CHANNEL DATA OK ?
2051 103170  001413                            BEQ     50$             ;YES, BRANCH
2052 103172  010337  001142                     MOV     R3,$BDDAT       ;BAD DATA
2053 103176  012737  000125  001140            MOV     #125,$GDDAT     ;SAVE GOOD DATA
2054 103204  013737  003422  001136            MOV     AD2,$BDADR      ;SET UP ADDRESS
2055 103212  104162                            EMT     162
2056 103214  000137  103546                    JMP     A20T14          ;ABORT TEST
2057
2058                                            ;******* CHECK STATUS(AD3) *******
2059 103220  005037  003442          50$:      CLR     IFLAG           ;RESET INTERRUPT FLAG
2060 103224  117737  100174  001142            MOVB    @AD3,$BDDAT     ;SAVE THE DATA
2061 103232  005737  001142                     TST     $BDDAT          ;ALL CLEAR ?
2062 103236  001434                            BEQ     60$             ;YES, BRANCH
2063 103240  022737  000001  001142            CMP     #1,$BDDAT       ;ACTIVE SET ?
2064 103246  001410                            BEQ     55$             ;YES, BRANCH
2065 103250  005037  001140                     CLR     $GDDAT          ;
2066 103254  013737  003424  001136            MOV     AD3,$BDADR      ;SET UP ADDRESS NUMBER
2067 103262  104163                            EMT     163
2068 103264  000137  103546                    JMP     A20T14          ;ABORT TEST
2069
2070                                            ;******* CHECK ACTIVE CLEARS AFTER DELAY *******
2071 103270  005004                  55$:      CLR     R4              ;SET UP DELAY
2072 103272  117737  100126  001142  56$:      MOVB    @AD3,$BDDAT     ;SAVE STATUS
2073 103300  005737  001142                     TST     $BDDAT          ;STATUS NOW CLEAR ?
2074 103304  001411                            BEQ     60$             ;YES, BRANCH
2075 103306  162704  000001                     SUB     #1,R4           ;COUNT DOWN
2076 103312  001367                            BNE     56$             ;LOOP TILL ZERO
2077 103314  005037  001140                     CLR     $GDDAT          ;
2078 103320  013737  003424  001136            MOV     AD3,$BDADR      ;SET ADDRESS
2079 103326  104204                            EMT     204
2080
2081                                            ;******* CHECK C BIT CLEARS AD3 *******
2082 103330  112777  000002  077306  60$:      MOVB    #CBIT,@CSR      ;CLEAR THE THING
2083 103336  012700  001000                     MOV     #1000,R0        ;SET UP DELAY
2084 103342  005300                  61$:      DEC     R0              ;DECREASE BY ONE
2085 103344  001376                            BNE     61$             ;LOOP TILL = 0
2086 103346  004737  015526                     JSR     PC,CNTRC        ;GOTO MONITOR IF CNTRC
2087 103352  105777  100046                     TSTB    @AD3            ;171003 = 0 ?
2088 103356  001411                            BEQ     62$             ;YES, BRANCH
2089 103360  005037  001140                     CLR     $GDDAT          ;SET UP GOOD DATA
2090 103364  013737  003424  001136            MOV     AD3,$BDADR      ;SET UP 171003
2091 103372  117737  075540  001142            MOVB    @$BDADR,$BDDAT  ;SET UP BAD DATA
2092 103400  104016                            EMT     16
2093
2094 103402  000461                  62$:      BR      A20T14
2095 103404  000137  101654                    JMP     A20T13 ;REPLACE ABOVE BRANCH WITH NOP TO LOOP
```

```
2097                                                    ;******* INTERRUPT HANDLER *******
2098 103410  005237  003442            400$:   INC     IFLAG           ;SET INTERRUPT OCCURED
2099 103414                            401$:
     103414  013737  002636  003446            MOV     K,ZLOOP
     103422  005037  003444            65$:    CLR     YLOOP           ;WAIT
     103426  005237  003444            64$:    INC     YLOOP
     103432  023727  003444  000001            CMP     YLOOP,#1
     103440  001372                            BNE     64$
     103442  005337  003446                    DEC     ZLOOP
     103446  001365                            BNE     65$
2100 103450  132777  000200  077166            BITB    #FBIT,@CSR              ;ANY INTERRUPTS ?
2101 103456  001001                            BNE     402$                    ;YES, BRANCH
2102 103460  000002                            RTI
2103 103462  005037  003502            402$:   CLR     TEMP                    ;START FROM SRATCH
2104 103466  117737  077154  003502            MOVB    @IAR,TEMP               ;STORE ADDRESS
2105 103474  123737  003502  003416            CMPB    TEMP,AD0                ;CORRECT INTERRUPT ADDRESS ?
2106 103502  001411                            BEQ     403$                    ;YES, BRANCH
2107 103504  063737  002660  003502            ADD     BASE,TEMP               ;MAKE NEXT ADDRESS
2108 103512  152777  000001  077124            BISB    #RBIT,@CSR              ;PREPARE TO CLEAR
2109 103520  105777  077756                    TSTB    @TEMP                   ;DO IT
2110 103524  000733                            BR      401$                    ;TRY AGAIN
2111
2112 103526  117703  077670            403$:   MOVB    @AD2,R3         ;GET CONVERTED CHANNEL
2113 103532  152777  000001  077104            BISB    #RBIT,@CSR      ;SET UP TO CLEAR
2114 103540  105777  077660                    TSTB    @AD3            ;DO IT
2115 103544  000002                            RTI
```

```
2117                                    ;*****************************************************************************
2118                                    ;TST14-THIS TEST CHECKS FOR A DATA OVER RUN CONDITION BY LOADING TWO CHANNELS AND
2119                                    ;NOT READING THE FIRST CHANNEL LOADED UNTIL THE SECOND ONE HAS BEEN CONVERTED.
2120                                    ;*****************************************************************************
2121
2122 103546                            A20T14:
                                        ;;*************************************************************
     103546  000004                     TST51:  SCOPE
     103550  012737  000051  001212             MOV     #51,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
2123 103556  005037  001142                     CLR     $BDDAT          ;CLEAR THIS LOCATION
2124 103562  004737  015526                     JSR     PC,CNTRC        ;EXIT IF ^C
2125
2126 103566  112777  000002  077050             MOVB    #CBIT,@CSR      ;CLEAR THE WORLD
2127 103574  012700  001000                     MOV     #1000,R0        ;SET UP DELAY
2128 103600  005300                     1$:     DEC     R0              ;COUNT DOWN
2129 103602  001376                             BNE     1$              ;LOOP TILL = 0
2130
2131 103604  112777  000001  077610             MOVB    #1,@AD2         ;LOAD 1ST CHANNEL
2132 103612  112777  000002  077602             MOVB    #2,@AD2         ;LOAD 2ND CHANNEL
2133 103620  005000                             CLR     R0              ;SET COUNT = 0
2134 103622  132777  000002  077574     2$:     BITB    #BIT1,@AD3      ;DONE BIT SET ?
2135 103630  001024                             BNE     4$              ;YES, BRANCH
2136 103632  005300                             DEC     R0              ;DECREASE COUNTT
2137 103634  001372                             BNE     2$              ;LOOP TILL = 0
2138 103636  004737  015526                     JSR     PC,CNTRC        ;GOTO MONITOR IF CNTRC
2139 103642  012737  000002  001140             MOV     #2,$GDDAT
2140 103650  117737  077550  001142             MOVB    @AD3,$BDDAT
2141 103656  013737  003424  001136             MOV     AD3,$BDADR
2142 103664  042737  177775  001142             BIC     #177775,$BDDAT  ;IGNORE OTHER BITS
2143 103672  104164                             EMT     164
2144
2145 103674  004737  015526                     JSR     PC,CNTRC        ;GOTO MONITOR IF CNTRC
2146 103700  005000                             CLR     R0              ;RESET THE COUNT
2147 103702  117737  077516  001142     4$:     MOVB    @AD3,$BDDAT     ;GET THE DATA
2148 103710  022737  000012  001142             CMP     #12,$BDDAT      ;DATA OVER AND DONE
2149 103716  001411                             BEQ     5$              ;YES, BRANCH
2150 103720  005300                             DEC     R0              ;DECREASE THE COUNT
2151 103722  001367                             BNE     4$              ;KEEP TRYING
2152 103724  013737  003424  001136             MOV     AD3,$BDADR
2153 103732  012737  000012  001140             MOV     #12,$GDDAT
2154 103740  104165                             EMT     165
2155
2156 103742  117737  077454  001142     5$:     MOVB    @AD2,$BDDAT     ;SAVE THE DATA
2157 103750  122737  000022  001142             CMPB    #22,$BDDAT      ;CHANNEL #'S INTACT?
2158 103756  001407                             BEQ     6$              ;YES, BRANCH
2159 103760  013737  003422  001136             MOV     AD2,$BDADR      ;
2160 103766  012737  000022  001140             MOV     #22,$GDDAT      ;
2161 103774  104166                             EMT     166
```

```
2163 103776  112,77  000001  076640  6$:     MOVB    #RBIT,@CSR      ;CLEAR THE CONDITION
2164 104004  105777  C77414                  TSTB    @AD3            ;DO IT
2165 104010  004737  015526                  JSR     PC,CNTRC        ;GOTO MONITOR IF CNTRC
2166 104014  117737  077404  001142          MOVB    @AD3,$BDDAT     ;GET THE DATA
2167 104022  105737  001142                  TSTB    $BDDAT          ;ALL CLEAR?
2168 104026  001406                          BEQ     7$              ;YES, BRANCH
2169 104030  013737  003424  001136          MOV     AD3,$BDADR      ;
2170 104036  005037  001140                  CLR     $GDDAT          ;
2171 104042  104167                          EMT     167
2172
2173                                         ;******* CHECK C BIT CLEARS AD3 *******
2174 104044  112777  000002  076572  7$:     MOVB    #CBIT,@CSR      ;CLEAR THE THING
2175 104052  012700  001000                  MOV     #1000,R0        ;SET UP DELAY
2176 104056  005300                  8$:     DEC     R0              ;DECREASE BY ONE
2177 104060  001376                          BNE     8$              ;LOOP TILL = 0
2178 104062  004737  015526                  JSR     PC,CNTRC        ;GOTO MONITOR IF CNTRC
2179 104066  105777  077332                  TSTB    @AD3            ;171003 = 0 ?
2180 104072  001411                          BEQ     9$              ;YES, BRANCH
2181 104074  005037  001140                  CLR     $GDDAT          ;SET UP GOOD DATA
2182 104100  013737  003424  001136          MOV     AD3,$BDADR      ;SET UP 171003
2183 104106  117737  075024  001142          MOVB    @$BDADR,$BDDAT  ;SET UP BAD DATA
2184 104114  104135                          EMT     135
2185
2186 104116  000207                  9$:     RTS     PC
2187 104120  000137  103546                  JMP     A20T14 ;REPLACE ABOVE WITH NOP TO LOOP
```

```
                        .SBTTL  A020 CALIBRATION/MANUAL ROUTINE
                ;*******************************************************************************
                ;A020 CALIBRATION ROUTINE-- THIS ROUTINE IS BASICALLY IN TWO PARTS. ONE PART  IS
                ;USED FOR MANUAL OBSERVATION OF DATA; AND THE OTHER FOR CALIBRATION.
                ;
                ;THE CALIBRATION/MANUAL ROUTINE WILL DO GROUPS OF 32 CONVERSIONS ANY CHANNEL AND
                ;PRINT THE RESULTS.
                ;
                ;THE OFFSET MUST BE CALIBRATED BEFORE CALIBRATING THE GAINS. THE VOLTAGE  SOURCE
                ;USED WILL DETERMINE THE ACCURACY OF THE CALIBRATION.
                ;
                ;WHENEVER ERROR MESSAGES ARE PRINTED THE AVERAGE IS MEANINGLESS.
                ;
                ;REFER TO THE MANUAL FOR SWITCH SETTINGS ABOUT THE CHANNELS AND GAINS.
                ;
                ;THE OFFSET MUST BE CALIBRATED AT THE HIGHEST GAIN (LOWEST  FULL  SCALE RANGE).
                ;
                ;THE CALIBRATIONS MAYBE DONE IN ONE OF TWO WAYS: THROUGH THE BERG  CONNECTOR  OR
                ;BY APPLING A VOLTAGE DIRECTLY TO TP1&TP2.  ALWAYS  PEMOVE  THE  CUSTOMERS  BERG
                ;CABLE BEFORE CONNECTING THE VOLTAGE SOURCE.
                ;
                ;TYPING CONTROL C WILL CAUSE A RETURN TO THE MONITOR.
                ;TYPING CONTROL A WILL CAUSE THE CALIBRATION ROUTINE TO RESTART.
                ;CHANNEL#=TEMP1 VALUE=TEMP2
                ;GCODE=TEMP3    ERRORS=TEMP4
                ;AVERAGE=TEMP5  OFFSET=R0
                ;*******************************************************************************

                .REM %
                A20CAL--A020 FIELD CALIBRATION ROUTINE
                        ;  SET CBIT TO CLEAR WORLD
                        :  SET UP TIMEOUT VECTOR
                        :  CALL ADSET (SET UP ADDRESSES)
                20$:    ;  REPEAT
                        :  :  IF (USING TP1 &TP2)
                        :  :  :  THEN SET DBIT
                30$:    :  :  IF ADJUST OFFSET
                        :  :  :  THEN
                        :  :  :  :  CALL 500$ (GET CHANNEL #)
                        :  :  :  :  VALUE=0
                        :  :  :  :  TYPE REMOVE CUSTOMER CABLE AND INPUT 0.0V
                        :  :  :  :  :  PRESS A KEY WHEN READY.
                        :  :  :  :  CALL CALRTN (CONVERSION ROUTINE)
                40$:    ;  :  IF CALIBRATE GAIN
                        :  :  :  THEN
                        :  :  :  :  CALL 500$ (GET CHANNEL #)
                        :  :  :  :  TYPE SELECT VOLTAGE TO CALIBRATE FROM TABLE
                        :  :  :  :  TYPE REMOVE CUSTOMER CABLE AND PRESS KEY WHEN READY
                        :  :  :  :  GET #X
                        :  :  :  :  OFFSET=(6*X)-6
                        :  :  :  :  VALUE=CALTAB(OFFSET)
                        :  :  :  :  TOL=CALTAB+2(OFFSET)
                        :  :  :  :  TYPE MESSAGE CALTAB+4(OFFSET)
                        :  :  :  :  CALL CALRTN (CONVERSION ROUTINE)
                50$:    ;  :  IF MANUAL PART
                        :  :  :  THEN
                        :  :  :  :  CALL 500$
```

```
2246                                    :  :  :  :  VALUE=-1
2247                                    :  :  :  :  CALL CALTRN
2248                                    :  UNTIL (CONTROL C) OR (CONTROL A) OR (CONTROL Z)
2249                                    END A20CAL
2250
2251                                    A020-GETS A CHANNEL #
2252                            500$:   ;  REPEAT
2253                                    :  :  GET CHANNEL#
2254                                    :  :  GET GCODE
2255                                    :  :  IF GCODE=304
2256                                    :  :  :  THEN X=17
2257                                    :  :  :  ELSE X=7
2258                                    :  UNTIL (CHANNEL# <=X)
2259
2260                            CALRTN--CONVERSION_ROUTINE
2261                            300$:   ;  REPEAT
2262                                    :  :  LOAD CHANNEL #
2263                                    :  :  CALL LDTABL (DO 32 CONVERSIONS)
2264                                    :  :  SAVE ERRORS
2265                                    :  :  IF SWR_BIT13=0
2266                                    :  :  :  THEN (PRINT OUT RESULTS)
2267                                    :  :  :  :  CALL AVERG (FIND AVERAGE)
2268                                    :  :  :  :  TYPE AVERAGE
2269                                    :  :  :  :  CALL TYPERR (TYPE ANY ERRORS)
2270                                    :  :  :  :  IF AVERAGE=VALUE
2271                                    :  :  :  :  :  THEN TYPE BELL
2272                            310$:   ;  UNTIL (CONTROL C) OR (CONTROL A)
2273                                    END CONVERSION_ROUTINE
2274
2275                            TMOUT---TIMEOUT TRAPS HERE
2276                            400$:   ;  CLEAN STACK
2277                                    :  TYPE (TIMEOUT OCCURED HERE)
2278                                    :  TYPOC PC
2279                                    :  JMP TO START
2280                                    END TIMEOUT
2281                                    %
```

```
                                                ;******* CALIBRATION VALUES *******
                                                ;VALUE  TOL.    MESSAGE
104124  037200  000001  104440  CALTAB:  37200.  1.      1$      ;51.200V
104132  037200  000001  104455           37200.  1.      2$      ;50.000V
104140  006200  000001  104472           6200.   1.      3$      ;51.200V W/O FS PRECISION SOURCE
104146  006315  000001  104507           6315.   1.      4$      ;50.000V W/O FS PRECISION SOURCE
104154  037200  000001  105245           37200.  1.      33$     ;25.600V
104162  037200  000001  105262           37200.  1.      34$     ;25.000V
104170  014400  000001  105277           14400.  1.      35$     ;25.600V W/O FS PRECISION SOURCE
104176  014632  000001  105314           14632.  1.      36$     ;25.000V W/O FS PRECISION SOURCE
104204  037200  000001  104524           37200.  1.      5$      ;20.480V
104212  037200  000001  104541           37200.  1.      6$      ;20.000V
104220  017500  000001  104556           17500.  1.      7$      ;20.480V W/O FS PRECISION SOURCE
104226  020000  000001  104573           20000.  1.      10$     ;20.000V W/O FS PRECISION SOURCE
104234  037200  000001  104610           37200.  1.      11$     ;10.240V
104242  037200  000001  104625           37200.  1.      12$     ;10.000V
104250  037200  000001  104642           37200.  1.      13$     ;5.1200V
104256  037200  000001  104657           37200.  1.      14$     ;5.0000V
104264  037200  000001  105331           37200.  1.      37$     ;2.5600V
104272  037200  000001  105346           37200.  1.      40$     ;2.5000V
104300  037200  000001  104674           37200.  1.      15$     ;2.0480V
104306  037200  000001  104711           37200.  1.      16$     ;2.0000V
104314  037200  000001  104726           37200.  1.      17$     ;1.0240V
104322  037200  000001  104743           37200.  1.      20$     ;1.0000V
104330  037200  000001  105363           37200.  1.      41$     ;500.0MV
104336  037200  000001  105404           37200.  1.      42$     ;482.8MV
104344  037200  000001  104761           37200.  1.      21$     ;204.80MV
104352  037200  000001  105003           37200.  1.      22$     ;200.00MV
104360  037200  000001  105025           37200.  1.      23$     ;102.40MV
104366  037200  000001  105047           37200.  1.      24$     ;100.00MV
104374  037200  000001  105071           37200.  1.      25$     ;51.200MV
104402  037200  000001  105113           37200.  1.      26$     ;50.000MV
104410  037200  000002  105135           37200.  2.      27$     ;20.480MV
104416  037200  000002  105157           37200.  2.      30$     ;20.000MV
104424  037200  000004  105201           37200.  4.      31$     ;10.240MV
104432  037200  000004  105223           37200.  4.      32$     ;10.000MV

                                                ;******* ASCII MESSAGES *******
104440      065     060     056  1$:      .ASCIZ /50.000 VOLTS/
104455      064     070     056  2$:      .ASCIZ /48.828 VOLTS/
104472      061     060     056  3$:      .ASCIZ /10.000 VOLTS/
104507      061     060     056  4$:      .ASCIZ /10.000 VOLTS/
104524      062     060     056  5$:      .ASCIZ /20.000 VOLTS/
104541      061     071     056  6$:      .ASCIZ /19.531 VOLTS/
104556      061     060     056  7$:      .ASCIZ /10.000 VOLTS/
104573      061     060     056  10$:     .ASCIZ /10.000 VOLTS/
104610      061     060     056  11$:     .ASCIZ /10.000 VOLTS/
104625      071     056     067  12$:     .ASCIZ /9.7656 VOLTS/
104642      065     056     060  13$:     .ASCIZ /5.0000 VOLTS/
104657      064     056     070  14$:     .ASCIZ /4.8828 VOLTS/
104674      062     056     060  15$:     .ASCIZ /2.0000 VOLTS/
104711      061     056     071  16$:     .ASCIZ /1.9531 VOLTS/
104726      061     056     060  17$:     .ASCIZ /1.0000 VOLTS/
104743      060     056     071  20$:     .ASCIZ /0.97656 VOLTS/
104761      062     060     060  21$:     .ASCIZ /200.00 MILLIVOLTS/
105003      061     071     065  22$:     .ASCIZ /195.31 MILLIVOLTS/
105025      061     060     060  23$:     .ASCIZ /100.00 MILLIVOLTS/
```

```
2340 105047     071     067     056  24$:    .ASCIZ /97.656 MILLIVOLTS/
2341 105071     065     060     056  25$:    .ASCIZ /50.000 MILLIVOLTS/
2342 105113     064     070     056  26$:    .ASCIZ /48.828 MILLIVOLTS/
2343 105135     062     060     056  27$:    .ASCIZ /20.000 MILLIVOLTS/
2344 105157     061     071     056  30$:    .ASCIZ /19.531 MILLIVOLTS/
2345 105201     061     060     056  31$:    .ASCIZ /10.000 MILLIVOLTS/
2346 105223     071     056     067  32$:    .ASCIZ /9.7656 MILLIVOLTS/
2347 105245     062     065     056  33$:    .ASCIZ /25.000 VOLTS/
2348 105262     062     064     056  34$:    .ASCIZ /24.414 VOLTS/
2349 105277     061     060     056  35$:    .ASCIZ /10.000 VOLTS/
2350 105314     061     060     056  36$:    .ASCIZ /10.000 VOLTS/
2351 105331     062     056     065  37$:    .ASCIZ /2.5000 VOLTS/
2352 105346     062     056     064  40$:    .ASCIZ /2.4414 VOLTS/
2353 105363     065     060     060  41$:    .ASCIZ /500.0 MILLIVOLTS/
2354 105404     064     070     070  42$:    .ASCIZ /488.28 MILLIVOLTS/
2355                                         .EVEN
```

```
                                                ;******* FIELD CALIBRATION ROUTINE *******
                                                ;-SET CBIT TO CLEAR WORLD
                                                ;-SET UP TIMEOUT VECTOR
                                                ;-CALL ADSET (SET UP ADDRESSES)
2361 105426  112777  000002  075210  A20CAL: MOVB    #CBIT,@CSR      ;SET TO CLEAR THE DBUS
2362 105434  012737  001000  003502         MOV     #1000,TEMP      ;SET UP DELAY
2363 105442  005337  003502          1$:    DEC     TEMP            ;DECREASE THE COUNT
2364 105446  001375                        BNE     1$              ;LOOP TILL ZERO
2365
2366 105450  012737  111204  000004         MOV     #400$,ERRVEC    ;SET UP TIMEOUT TRAP
2367 105456  012737  000340  000006         MOV     #340,ERRVEC+2   ;SET PRIORITY TO 7
2368 105464  000404                        BR      21$             ;
2369
2370                                            ;-REPEAT
2371                                            ;--IF (USING TP1 &TP2)
2372                                            ;---THEN SET DBIT
2373 105466  104401  122576          20$:   TYPE    ,NO$            ;TYPE 'NO'
2374 105472  004737  015526                 JSR     PC,CNTRC        ;EXIT IS CONTROL C
2375 105476  104401  110544          21$:   TYPE    ,290$           ;WILL YOU USE TP1 & TP2 ?
2376 105502  104406                        RDCHR                   ;GET A CHARACTER
2377 105504  022726  000131                 CMP     #'Y,(SP)+       ;A YES RESPONSE ?
2378 105510  001010                        BNE     30$             ;NO, BRANCH
2379 105512  104401  122572                 TYPE    ,YES$           ;TYPE 'YES'
2380 105516  152777  000020  075120         BISB    #DBIT,@CSR      ;SET THE DBIT TO ISOLATE TP1 & TP2
2381 105524  104401  110514                 TYPE    ,289$           ;SETTING D BIT
2382 105530  000407                        BR      31$
2383
2384                                            ;--IF ADJUST OFFSET
2385                                            ;---THEN
2386                                            ;----CALL 500$ (GET CHANNEL #)
2387                                            ;----VALUE=0
2388                                            ;----TYPE REMOVE CUSTOMER CABLE AND INPUT 0.0V
2389                                            ;-----PRESS A KEY WHEN READY.
2390                                            ;----CALL CALRTN (CONVERSION ROUTINE)
2391 105532  142777  000020  075104  30$:    BICB    #DBIT,@CSR      ;CLEAR THE D BIT
2392 105540  004737  015526                 JSR     PC,CNTRC        ;EXIT IS CONTROL C
2393 105544  104401  122576                 TYPE    ,NO$            ;TYPE 'NO'
2394 105550  104401  106260          31$:   TYPE    ,200$           ;DO YOU WISH TO ADJUST OFFSET ?
2395 105554  104406                        RDCHR                   ;GET A CHARACTER
2396 105556  022726  000131                 CMP     #'Y,(SP)+       ;A YES RESPONSE ?
2397 105562  001015                        BNE     40$             ;NO, BRANCH
2398 105564  104401  122572                 TYPE    ,YES$           ;TYPE 'YES'
2399 105570  004737  110702                 JSR     PC,500$         ;GET CHANNEL #
2400 105574  005037  003576                 CLR     TEMP2           ;SET UP A VALUE OF 0.0 VOLTS
2401 105600  104401  106366                 TYPE    ,230$           ;INPUT 0.0V
2402 105604  104406                        RDCHR                   ;WAIT FOR RESPONSE
2403 105606  005726                        TST     (SP)+           ;CLEAN STACK OF CHARACTER
2404 105610  004737  111014                 JSR     PC,300$         ;GO DO CONVERSIONS
2405 105614  000404                        BR      41$
2406
```

```
                                                ;-IF CALIBRATE GAIN
                                                ;--THEN
                                                ;---CALL 500$ (GET CHANNEL #)
                                                ;---TYPE SELECT VOLTAGE TO CALIBRATE FROM TABLE AND ENTER NUMBER
                                                ;---TYPE REMOVE CUSTOMER CABLE AND PRESS A KEY WHEN READY
105616  004737  015526          40$:  JSR     PC,CNTRC          ;IS CONTROL C
105622  104401  122576                TYPE    ,NO$              ;TYPE 'NO'
105626  104401  106335          41$:  TYPE    ,210$             ;DO TOU WISH TO CALIBRATE GAIN?
105632  104406                      RDCHR                       ;GET A CHARACTER
105634  022726  000131              CMP     #'Y,(SP)+         ;A YES RESPONSE
105640  001160                      BNE     50$               ;NO,BRANCH
105642  104401  122572              TYPE    ,YES$             ;TYPE'YES'
105646  004737  110702              JSR     PC,500$           ;GET A CHANNEL NUMBER
105652  104401  106465              TYPE    ,240$             ;DISCONNECT AND PRESS KEY
105656  104406                      RDCHR                     ;
105660  005726                      TST     (SP)+             ;
105662  104401  001203              TYPE    ,$CRLF
                                                ;--- GET #X
                                                ;---OFFSET=(6*X)-6
                                                ;---VALUE=CALTAB(OFFSET)
                                                ;---TOL=CALTAB+2(OFFSET)
                                                ;---TYPE MESSAGE CALTAB+4(OFFSET)
                                                ;---CALL CALRTN (CONVERSION ROUTINE)
105666  104401  106600          46$:  TYPE    ,245$             ;''PRINT CALIBRATION TABLE ?''
105672  104406                      RDCHR                       ;GET A CHARACTER
105674  022726  000131              CMP     #'Y,(SP)+         ;A YES RESPONSE
105700  001405                      BEQ     47$               ;YES, BRANCH
105702  004737  015526              JSR     PC,CNTRC          ;CONTROL C ?
105706  104401  122576              TYPE    ,NO$              ;NO
105712  000404                      BR      42$
105714  104401  122572          47$:  TYPE    ,YES$             ;TYPE'YES'
105720  104401  106652              TYPE    ,241$             ;PRINT TABLE
105724  104401  110310          42$:  TYPE    ,243$             ;''ENTER NUMBER FROM TABLE''
105730  104410                      RDOCT                     ;
105732  021627  000042              CMP     (SP),#42          ;TOO LARGE ?
105736  101404                      BLOS    43$               ;NO,BRANCH
105740  005726                      TST     (SP)+             ;CLEAR CHARACTER
105742  104401  110360              TYPE    ,242$             ;TOO LARGE !
105746  000747                      BR      46$               ;TRY AGAIN
105750  022716  000001          43$:  CMP     #1,(SP)           ;BELOW 1?
105754  101404                      BLOS    45$               ;NO,BRANCH
105756  005726                      TST     (SP)+             ;CLEAR CHARACTER
105760  104401  110360              TYPE    ,242$             ;ERROR TOO LOW
105764  000740                      BR      46$               ;
105766  012746  000006          45$:  MOV     #6.,-(SP)         ;MULTIPLIER
105772  004737  017644              JSR     PC,@#$MULT        ;MULTIPLY
105776  012600                      MOV     (SP)+,R0          ;THE OFFSET
106000  005726                      TST     (SP)+             ;IGNORE MSB
106002  162700  000006              SUB     #6,R0             ;ADJUST FOR START OF 1
106006  016037  104124  003576      MOV     CALTAB(R0),TEMP2;
106014  016037  104126  111220      MOV     CALTAB+2(R0),CALTOL;
106022  016037  104130  106052      MOV     CALTAB+4(R0),44$;PREPARE TO TYPE MESSAGE
106030  104401  106036              TYPE    ,65$              ;;TYPE ASCIZ STRING
106034  000405                      BR      64$               ;;GET OVER THE ASCIZ
                              ;;65$:  .ASCIZ  <CR><LF>/INPUT /
```

```
      106050
 2463 106050  104401
 2464 106052  000000
 2465 106054  104401  106062
      106060  000415

      106114
 2466 106114  013746  003576
      106120  104402
 2467 106122  104401  106130
      106126  000411

      106152
 2468 106152  013746  111220
 2469 106156  104403
 2470 106160     001
 2471 106161     000
 2472 106162  104401  106170
      106166  000403

      106176
 2473 106176  004737  111014
```

```
64$:
        TYPE
44$:    0                       ;MESSAGE ADDRESS
        TYPE    ,67$            ;;TYPE ASCIZ STRING
        BR      66$             ;;GET OVER THE ASCIZ
;;67$:  .ASCIZ  / FOR AN AVERAGE VALUE OF /
66$:
        MOV     TEMP2,-(SP)     ;;SAVE TEMP2 FOR TYPEOUT
        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
        TYPE    ,69$            ;;TYPE ASCIZ STRING
        BR      68$             ;;GET OVER THE ASCIZ
;;69$:  .ASCIZ  <CR><LF>%TOLERANCE +/- %
68$:
        MOV     CALTOL,-(SP)
        TYPOS
        .BYTE   1
        .BYTE   0
        TYPE    ,71$            ;;TYPE ASCIZ STRING
        BR      70$             ;;GET OVER THE ASCIZ
;;71$:  .ASCIZ  / LSB/
70$:
        JSR     PC,300$         ;
```

```
                                                ;--IF MANUAL PART
                                                ;---THEN
                                                ;----CALL 500$
                                                ;----VALUE=-1
                                                ;----CALL CALTRN
106202  004737  015526            50$:  JSR     PC,CNTRC        ;EXIT IS CONTROL C
106206  104401  122576                  TYPE    ,NO$            ;TYPE 'NO'
106212  104401  106314            51$:  TYPE    ,201$           ;DO MANUAL OPERATIONS ?
106216  104406                        RDCHR                   ;GET CHARACTER
106220  022726  000131                CMP     #'Y,(SP)+       ;A YES RESPONSE ?
106224  001402                        BEQ     52$             ;YES, BRANCH
106226  000137  105466                JMP     20$             ;NO, BRANCH
106232  104401  122572            52$:  TYPE    ,YES$           ;TYPE 'YES'
106236  004737  110702                JSR     PC,500$         ;GET CHANNEL NUMBER
106242  012737  177777  003576        MOV     #-1,TEMP2       ;NEVER GET TO THIS NUMBER
106250  004737  111014                JSR     PC,300$         ;GO DO CONVERSIONS
106254  000137  105476                JMP     21$             ;
```

```
2494                                        ;******* MESSAGES *******
2495 106260  015  012  101  200$:  .ASCIZ  <15><12> #ADJUST THE OFFSET [Y/N] ?#
2496 106314  015  012  115  201$:  .ASCIZ  <15><12> #MANUAL [Y/N] ?#
2497 106335  015  012  103  210$:  .ASCIZ  <15><12> #CALIBRATE GAIN [Y/N] ?#
2498 106366  015  012  122  230$:  .ASCIZ  <15><12> /REMOVE CUSTOMER CABLE AND INPUT 0.0V, PRESS A KEY WHEN READY/
2499 106465  015  012  122  240$:  .ASCII  <15><12> /REMOVE CUSTOMER CABLE AND CONNECT VOLTAGE SOURCE/
2500 106547  054  040  120          .ASCIZ  /, PRESS A KEY WHEN READY/
2501 106600  015  012  120  245$:  .ASCIZ  <15><12> #PRINT CALIBRATION VOLTAGE TABLE [Y/N] ?#
2502 106652  015  012  012  241$:  .ASCII  <15><12><12> /VALUE    VOLTAGE/
2503 106672  011  011  011          .ASCII  /                                                VALUE   VOLTAGE/
2504 106715  015  012  055          .ASCII  <15><12>     /---------------/
2505 106736  011  011  011          .ASCII    /                                              ---------------/
2506 106763  015  012  061          .ASCII  <15><12> /1     51.200V/
2507 106776  011  011  011          .ASCII  /                                                23      2.0480V/
2508 107016  015  012  062          .ASCII  <15><12> /2     50.000V/
2509 107031  011  011  011          .ASCII  /                                                24      2.0000V/
2510 107051  015  012  063          .ASCII  <15><12> %3     51.200V W/O FULL SCALE PRECISION SOURCE%
2511 107124  011  011  062          .ASCII  /                25      1.0240V/
2512 107140  015  012  064          .ASCII  <15><12> %4     50.000V W/O FULL SCALE PRECISION SOURCE%
2513 107213  011  011  062          .ASCII  /                26      1.0000V/
2514 107227  015  012  065          .ASCII  <15><12> /5     25.600V/
2515 107242  011  011  011          .ASCII   /                                               27      512.0MV/
2516 107262  015  012  066          .ASCII  <15><12> /6     25.000V/
2517 107275  011  011  011          .ASCII   /                                               30      500.0MV/
2518 107315  015  012  067          .ASCII  <15><12> %7     25.600V W/O FULL SCALE PRECISION SOURCE%
2519 107370  011  011  063          .ASCII  /                31      204.80MV/
2520 107405  015  012  061          .ASCII  <15><12> %10    25.000V W/O FULL SCALE PRECISION SOURCE%
2521 107461  011  011  063          .ASCII  /                32      200.00MV/
2522 107476  015  012  061          .ASCII  <15><12> /11    20.480V/
2523 107512  011  011  011          .ASCII  /                                                33      102.40MV/
2524 107533  015  012  061          .ASCII  <15><12> /12    20.000V/
2525 107547  011  011  011          .ASCII  /                                                34      100.00MV/
2526 107570  015  012  061          .ASCII  <15><12> %13    20.480V W/O FULL SCALE PRECISION SOURCE%
2527 107644  011  011  063          .ASCII  /                35      51.200MV/
2528 107661  015  012  061          .ASCII  <15><12> %14    20.000V W/O FULL SCALE PRECISION SOURCE%
2529 107735  011  011  063          .ASCII  /                36      50.000MV/
2530 107752  015  012  061          .ASCII  <15><12> /15    10.240V/
2531 107766  011  011  011          .ASCII  /                                                37      20.480MV/
2532 110007  015  012  061          .ASCII  <15><12> /16    10.000V/
2533 110023  011  011  011          .ASCII  /                                                40      20.000MV/
2534 110044  015  012  061          .ASCII  <15><12> /17    5.1200V/
2535 110060  011  011  011          .ASCII  /                                                41      10.240MV/
2536 110101  015  012  062          .ASCII  <15><12> /20    5.0000V/
2537 110115  011  011  011          .ASCII  /                                                42      10.000MV/
2538 110136  015  012  062          .ASCII  <15><12> /22    2.5000V/
2539 110152  015  012  062          .ASCII  <15><12> /21    2.5600V/
2540 110166  015  012  012          .ASCII  <15><12><12> /INPUT A VALUE FROM THE TABLE WHICH/
2541 110233  015  012  115          .ASCIZ  <15><12> /MATCHES THE VOLTAGE YOU ARE CALIBRATING TO/
2542 110310  015  012  111  243$:  .ASCIZ  <15><12> /INPUT VALUE FROM TABLE [1-42 OCTAL]: /
2543 110360  015  012  116  242$:  .ASCIZ  <15><12> /NUMBER NOT IN RANGE !!/
2544 110411  015  012  124  260$:  .ASCIZ  <15><12> /TIMEOUT OCCURED PC= /
2545 110440  015  012  117  270$:  .ASCIZ  <15><12> /OCTAL AVERAGE =/
2546 110462  015  012  105  280$:  .ASCIZ  <15><12> /ENTER CHANNEL #(OCTAL) /
2547 110514  015  012  052  289$:  .ASCIZ  <15><12> /*** SETTING D BIT ***/
2548 110544  015  012  127  290$:  .ASCIZ  <15><12> #WILL YOU USE TP1 & TP2 [Y/N] ?#
2549 110605  015  012  103  291$:  .ASCIZ  <15><12> /CHANNEL# TOO LARGE FOR 3-WIRE MODE (GCODE=304, E6-5 IS ON)/
2550                                        .EVEN
```

```
2552                                                ;******* GETS A CHANNEL NUMBER ******
2553 110702  004737  C15526          500$:  JSR     PC,CNTRC         ;EXIT IF CONTROL C
2554 110706  104401  110462                 TYPE    ,280$            ;ENTER CHANNEL #
2555 110712  104410                        RDOCT                    ;
2556 110714  012637  003510                MOV     (SP)+,TEMP1      ;SAVE CHANNEL #
2557 110720  152777  000004  071716        BISB    #GBIT,@CSR       ;ENABLE GCODE READ
2558 110726  005037  003600                CLR     TEMP3            ;INIT THIS LOCATION
2559 110732  117737  072460  003600        MOVB    @ADO,TEMP3       ;SAVE THE CODE
2560 110740  142777  000004  071676        BICB    #GBIT,@CSR       ;DISABLE GCODE READ
2561 110746  022737  000324  003600        CMP     #324,TEMP3       ;IS IT THIS GENERIC CODE ?
2562 110754  001004                        BNE     501$             ;NO, BRANCH
2563 110756  012737  000017  003502        MOV     #17,TEMP         ;SET UPPER LIMIT TO 17
2564 110764  000403                        BR      502$             ;
2565 110766  012737  000007  003502  501$:  MOV     #7,TEMP          ;SET UPPER LIMIT TO 7
2566 110774  023737  003510  003502  502$:  CMP     TEMP1,TEMP       ;CHANNEL# > TEMP ?
2567 111002  101403                        BLOS    503$             ;NO, BRANCH
2568 111004  104401  110605                TYPE    ,291$            ;# TOO LARGE
2569 111010  000734                        BR      500$             ;TRY AGAIN
2570 111012  000207                503$:   RTS     PC
2571
2572                                                ;******* CONVERSION ROUTINE *******
2573 111014  013737  111220  111222  300$:  MOV     CALTOL,CALUP     ;SET UP UPPER LIMIT
2574 111022  063737  003576  111222         ADD     TEMP2,CALUP      ;
2575 111030  013737  003576  111224         MOV     TEMP2,CALLO      ;SET UP LOWER LIMIT
2576 111036  163737  111220  111224         SUB     CALTOL,CALLO     ;
2577 111044  004737  015526                JSR     PC,CNTRC         ;EXIT IF CONTROL C
2578 111050  013746  003510                MOV     TEMP1,-(SP)      ;LOAD CHANNEL NUMBER TO CONVERT
2579 111054  004737  111516                JSR     PC,LDTABL        ;DO 32 CONVERSIONS
2580 111060  012637  003602                MOV     (SP)+,TEMP4      ;SAVE ANY ERRORS
2581
2582 111064  032777  020000  070062        BIT     #BIT13,@SWR      ;INHIBIT PRINTOUT ?
2583 111072  001041                        BNE     301$             ;YES, BRANCH
2584
2585 111074  004737  112274                JSR     PC,AVERG         ;FIND THE TABLE AVERAGE
2586 111100  012637  003604                MOV     (SP)+,TEMP5      ;SAVE THE AVERAGE
2587 111104  104401  110440                TYPE    ,270$            ;OCTAL AVERAGE =
2588 111110  013746  003604                MOV     TEMP5,-(SP)      ;READY TO TYPE
2589 111114  104402                        TYPOC                    ;TYPE VALUE
2590 111116  013746  003602                MOV     TEMP4,-(SP)      ;PREPARE TO TYPE ERRORS
2591 111122  004737  111226                JSR     PC,TYPERR        ;TYPE THE ERRORS
2592 111126  023727  003604  040000        CMP     TEMP5,#40000     ;NEG ZERO ?
2593 111134  001002                        BNE     302$             ;NO, BRANCH
2594 111136  005037  003604                CLR     TEMP5            ;MAKE POSITIVE ZERO
2595 111142  023737  003604  111222  302$:  CMP     TEMP5,CALUP      ;TEMP5>CALUP ?
2596 111150  101012                        BHI     301$             ;YES, BRANCH
2597 111152  023737  003604  111224         CMP     TEMP5,CALLO      ;TEMP5<CALLO ?
2598 111160  103406                        BLO     301$             ;NO, BRANCH
2599 111162  022737  177777  003576         CMP     #-1,TEMP2        ;MANUAL TEST ?
2600 111170  001402                        BEQ     301$             ;YES, BRANCH (NO BELL)
2601 111172  104401  001176                TYPE    ,$BELL           ;RING THE BELL
2602 111176  004737  015526          301$:  JSR     PC,CNTRC         ;
2603 111202  000704                        BR      300$             ;
2604
2605                                                ;******* TIMEOUT HANDLER *******
2606 111204  104401  110411          400$:  TYPE    ,260$            ;TIMEOUT OCCURED PC=
2607 111210  012616                        MOV     (SP)+,(SP)       ;SAVE PC
2608 111212  104402                        TYPOC                    ;
```

```
 2609 111214  000137  006360                    JMP     START
 2610
 2611                                           ;******* VARIABLE *******
 2612 111220  000000                 CALTOL:   0                ;HOLDS TOLERANCE
 2613 111222  000000                 CALUP:    0                ;UPPER LIMIT
 2614 111224  000000                 CALLO:    0                ;LOWER LIMIT
```

```
                                        ;*****************************************************************************
                                        ;THIS ROUTINE WILL ANALYZE BITS 15,14,13,11 OF THE WORD ON THE STACK  AND  PRINT
                                        ;THE APPROPRIATE ERROR MESSAGE
                                        ;*****************************************************************************
                                        ;
111226  032766  100000  000002  TYPERR: BIT     #BIT15,2(SP)     ;ERROR BIT SET ?
111234  001402                          BEQ     1$               ;NO, BRANCH
111236  104401  111312                  TYPE    ,20$             ;
111242  032766  040000  000002  1$:     BIT     #BIT14,2(SP)     ;OVER RANGE BIT SET ?
111250  001402                          BEQ     2$               ;NO, BRANCH
111252  104401  111343                  TYPE    ,21$             ;
111256  032766  020000  000002  2$:     BIT     #BIT13,2(SP)     ;DATA OVERRUN SET ?
111264  001402                          BEQ     3$               ;NO, BRANCH
111266  104401  111402                  TYPE    ,22$             ;
111272  032766  004000  000002  3$:     BIT     #BIT11,2(SP)     ;TOO MANY CONVERSIONS ?
111300  001402                          BEQ     4$               ;NO, BR'NCH
111302  104401  111440                  TYPE    ,23$
111306  012616                  4$:     MOV     (SP)+,(SP)       ;MOVE THE RETURN PC
111310  000207                          RTS     PC
111312     015     012     105  20$:    .ASCIZ <15><12> /ERROR-- ERROR BIT SET /
111343     015     012     105  21$:    .ASCIZ <15><12> /ERROR-- OVER RANGE BIT SET  /
111402     015     012     105  22$:    .ASCIZ <15><12> /ERROR-- DATA OVER RUN SET  /
111440     015     012     105  23$:    .ASCIZ <15><12> /ERROR--RECEIVED INCORRECT # OF CONVERSION /
                                        .EVEN
                                        ;*****************************************************************************
                                        ;THIS ROUTINE WILL FILL A TABLE WITH 32 CONVERSIONS. THE CHANNEL TO BE CONVERTED
                                        ;IS SENT VIA THE STACK. BIT15=1 IF ERROR, BIT13=1 IF  DATA  OVERUN,  BIT14=1  IF
                                        ;OVER RANGE ARE SENT BACK VIA THE STACK THE TABLE IS FILLED IN SMART MODE
                                        ;
                                        ;CALL:
                                        ;       MOV     CHANNEL,-(SP)
                                        ;       JSR     PC, LDTABL
                                        ;       MOV     (SP)+,ANYWHERE
                                        ;*****************************************************************************
                                        ;
111516  017746  071126          LDTABL: MOV     @VECTO,-(SP)             ;SAVE OLD VECTOR
111522  012777  111732  071120          MOV     #19$,@VECTO              ;SET UP NEW VECTOR
111530  004737  112556                  JSR     PC,PRIO
111534  010046                          MOV     R0,-(SP)                 ;SAVE R0
111536  010146                          MOV     R1,-(SP)                 ;SAVE R1
111540  005000                          CLR     R0                       ;SET=0
111542  005001                          CLR     R1                       ;SET=0
                                        ;
                                        ;******* INITIALIZE THE TABLE *******
111544  005060  112170          1$:     CLR     TABLE1(R0)       ;CLEAR A ELEMENT
111550  062700  000002                  ADD     #2,R0            ;INCREASE THE OFFSET
111554  022700  000100                  CMP     #100,R0          ;END OF TABLE ?
111560  001371                          BNE     1$               ;NO, BRANCH
111562  005000                          CLR     R0               ;RESET R0
111564  152777  000001  071052          BISB    #RBIT,@CSR               ;CLEAR THE UNIT
111572  105777  071626                  TSTB    @AD3                     ;DO IT
111576  152777  000100  071040          BISB    #EBIT,@CSR       ;ENABLE INTERRUPTS
111604  116677  000010  071610  2$:     MOVB    10(SP),@AD2              ;LOAD CHANNEL TO BE CONVERTED
111612  132777  000004  071604          BITB    #BIT2,@AD3               ;IS ERROR BIT SET ?
111620  001407                          BEQ     3$                       ;NO,BRANCH
111622  004737  112646                  JSR     PC,PRI7                  ;SET PRIORITY SEVEN
111626  052766  100000  000010          BIS     #BIT15,10(SP)            ;SET COUNT ERROR
```

```
2673 111634  000137  111712                    JMP     5$                      ;AND LEAVE
2674 111640  105777  071560          3$:       TSTB    @AD3                    ;BUSY BIT SET?
2675 111644  100775                            BMI     3$                      ;YES, BRANCH
2676 111646  062700  000002                    ADD     #2,R0                   ;STEP OFFSET
2677 111652  020027  000100                    CMP     R0,#100                 ;ALL DONE?
2678 111656  001352                            BNE     2$                      ;NO, BRANCH
2679 111660  132777  000001  071536  4$:       BITB    #BIT0,@AD3              ;CHECK THE ACTIVE BIT
2680 111666  001374                            BNE     4$
2681 111670  142777  000100  070746            BICB    #EBIT,@CSR
2682 111676  022701  000100                    CMP     #100,R1                 ;ALL CONVERSIONS DONE?
2683 111702  001403                            BEQ     5$                      ;YES, BRANCH
2684 111704  052766  004000  000010            BIS     #BIT11,10(SP)           ;SET COUNT ERROR
2685 111712  142777  000100  070724  5$:       BICB    #EBIT,@CSR              ;DISABLE INTERRUPTS
2686 111720  012601                            MOV     (SP)+,R1                ;RESTORE R1
2687 111722  012600                            MOV     (SP)+,R0                ;RESTORE R0
2688 111724  012677  070720                    MOV     (SP)+,@VECT0            ;RESTORE VECT0
2689 111730  000207                            RTS     PC
2690
2691                                           ;*******INTERRUPT HANDLER *******
2692 111732                          19$:
     111732  013737  002636  003446            MOV     K,ZLOOP
     111740  005037  003444          65$:      CLR     YLOOP           ;WAIT
     111744  005237  003444          64$:      INC     YLOOP
     111750  023727  003444  000001            CMP     YLOOP,#1
     111756  001372                            BNE     64$
     111760  005337  003446                    DEC     ZLOOP
     111764  001365                            BNE     65$
2693 111766  132777  000200  070650            BITB    #FBIT,@CSR              ;ANY INTERRUPTS ?
2694 111774  001001                            BNE     20$                     ;YES, BRANCH
2695 111776  000002                            RTI
2696 112000  005037  003502          20$:      CLR     TEMP                    ;START FROM SRATCH
2697 112004  117737  070636  003502            MOVB    @IAR,TEMP               ;STORE ADDRESS
2698 112012  063737  002660  003502            ADD     BASE,TEMP               ;MAKE NEXT ADDRESS
2699 112020  023737  003502  003416            CMP     TEMP,AD0                ;CORRECT INTERRUPT ADDRESS ?
2700 112026  001406                            BEQ     24$                     ;YES, BRANCH
2701 112030  152777  000001  070606            BISB    #RBIT,@CSR              ;PREPARE TO CLEAR
2702 112036  105777  071440                    TSTB    @TEMP                   ;DO IT
2703 112042  000733                            BR      19$                     ;TRY AGAIN
2704 112044  117761  071346  112170  24$:      MOVB    @AD0,TABLE1(R1)         ;GET DATA
2705 112052  117761  071342  112171            MOVB    @AD1,TABLE1+1(R1)       ;GET DATA
2706 112060  032761  100000  112170            BIT     #BIT15,TABLE1(R1)       ;OVER RANGE?
2707 112066  001403                            BEQ     21$                     ;NO, BRANCH
2708 112070  052766  040000  000014            BIS     #BIT14,14(SP)           ;SET RETURN TO INDICATE OVR
2709 112076  152777  000001  070540  21$:      BISB    #RBIT,@CSR              ;CLEAR THE INTERRUPT
2710 112104  132777  000010  071312            BITB    #BIT3,@AD3              ;OVER RUN SET ?
2711 112112  001405                            BEQ     22$                     ;NO, BRANCH
2712 112114  052766  020000  000014            BIS     #BIT13,14(SP)           ;SET THE BIT
2713 112122  012716  111712                    MOV     #5$,(SP)                ;SET FAST RETURN
2714 112126  062701  000002          22$:      ADD     #2,R1                   ;STEP OFFSET
2715 112132  013737  002636  003446            MOV     K,ZLOOP
     112140  005037  003444          67$:      CLR     YLOOP           ;WAIT
     112144  005237  003444          66$:      INC     YLOOP
     112150  023727  003444  001000            CMP     YLOOP,#1000
     112156  001372                            BNE     66$
     112160  005337  003446                    DEC     ZLOOP
     112164  001365                            BNE     67$
2716 112166  000002                            RTI
```

```
2717 112170                    TABLE1: .BLKW   42      ;SAVE 32(D) LOCATIONS AND MORE
2718
2719
```

```
                                ;*************************************************************************
                                ;THIS ROUTINE WILL FIND THE AVERAGE OF THE DATA IN THE TABLE.  THE  NEW  AVERAGE
                                ;IS RETURNED ON THE STACK.
                                ;CALL:
                                ;       JSR     PC,AVERG
                                ;       MOV     (SP)+,AVERAGE
                                ;*************************************************************************

112274  010046                  AVERG:  MOV     R0,-(SP)        ;SAVE R0
112276  005037  112550                  CLR     5$              ;CLEAR EXTRA BITS
112302  005037  112552                  CLR     6$              ;CLEAR LSB'S
112306  005037  112554                  CLR     7$              ;CLEAR MSB'S
112312  005000                          CLR     R0              ;SET UP OFFSET
112314  042760  100000  112170  1$:     BIC     #BIT15,TABLE1(R0);STRIP OVERRANGE BIT
112322  032760  040000  112170          BIT     #BIT14,TABLE1(R0);NEG VALUE ?
112330  001413                          BEQ     10$             ;NO, BRANCH
112332  042760  040000  112170          BIC     #BIT14,TABLE1(R0);CLEAR SIGN BIT
112340  005760  112170                  TST     TABLE1(R0)      ;NEG ZERO ?
112344  001405                          BEQ     10$             ;YES, BRANCH
112346  005460  112170                  NEG     TABLE1(R0)      ;MAKE NEGATIVE
112352  062737  177777  112554          ADD     #177777,7$      ;ADD IN SIGN BITS
112360  066037  112170  112552  10$:    ADD     TABLE1(R0),6$   ;ANY OVERFLOW ?
112366  005537  112554                  ADC     7$              ;ADD TO MSB'S
112372  062700  000002                  ADD     #2,R0           ;STEP OFFSET
112376  022700  000100                  CMP     #100,R0         ;END OF TABLE ?
112402  001344                          BNE     1$              ;NO, BRANCH

112404  012600                          MOV     (SP)+,R0        ;RESTORE R0
112406  012737  000005  112546          MOV     #5,4$           ;SET UP DIV BY 8
112414  000241                  2$:     CLC
112416  006037  112554                  ROR     7$              ;DIVIDE
112422  006037  112552                  ROR     6$              ;BY
112426  006037  112550                  ROR     5$              ;2
112432  005337  112546                  DEC     4$              ;DECREASE THE COUNT
112436  001366                          BNE     2$              ;NOT DONE YET, BRANCH
112440  005737  112552                  TST     6$              ;NEG VALUE ?
112444  100014                          BPL     11$             ;NO, BRANCH
112446  005137  112552                  COM     6$              ;START 2'S COMPL
112452  005137  112550                  COM     5$              ;MAKE 1'S COM
112456  062737  020000  112550          ADD     #20000,5$       ;INC 5$
112464  005537  112552                  ADC     6$              ;GET CARRY
112470  052737  040000  112552          BIS     #BIT14,6$       ;INDICATE NEGATIVE
112476  005737  112550          11$:    TST     5$              ;MSB OF EXTRA SET ?
112502  100014                          BPL     3$              ;NO, BRANCH
112504  013737  112552  112550          MOV     6$,5$           ;SAVE VALUE
112512  042737  140000  112550          BIC     #140000,5$      ;STRIP SIGN BIT
112520  022737  037777  112550          CMP     #37777,5$       ;MAX COUNT ?
112526  001402                          BEQ     3$              ;NO, BRANCH
112530  005237  112552                  INC     6$              ;ADD HALF A BIT
112534  011646                  3$:     MOV     (SP),-(SP)      ;MOVE RETURN PC
112536  013766  112552  000002          MOV     6$,2(SP)        ;PUT AVERAGE ON STACK
112544  000207                          RTS     PC

112546  000000                  4$:     .WORD   0               ;COUNT STORAGE
112550  000000                  5$:     .WORD   0               ;EXTRA BITS STORAGE
112552  000000                  6$:     .WORD   0               ;LSB'S STORAGE
112554  000000                  7$:     .WORD   0               ;MSB'S STORAGE
```

```
                                                ;******* SET PRIORITY *******
112556  012737  000000  112666  PRIO:   MOV     #0,PRIO         ;SAVE PRIORITY
112564  000433                        BR      PRIOR
112566  012737  000040  112666  PRI1:   MOV     #40,PRIO        ;SAVE PRIORITY
112574  000427                        BR      PRIOR
112576  012737  000100  112666  PRI2:   MOV     #100,PRIO       ;SAVE PRIORITY
112604  000423                        BR      PRIOR
112606  012737  000140  112666  PRI3:   MOV     #140,PRIO       ;SAVE PRIORITY
112614  000417                        BR      PRIOR
112616  012737  000200  112666  PRI4:   MOV     #200,PRIO       ;SAVE PRIORITY
112624  000413                        BR      PRIOR
112626  012737  000240  112666  PRI5:   MOV     #240,PRIO       ;SAVE PRIORITY
112634  000407                        BR      PRIOR
112636  012737  000300  112666  PRI6:   MOV     #300,PRIO       ;SAVE PRIORITY
112644  000403                        BR      PRIOR
112646  012737  000340  112666  PRI7:   MOV     #340,PRIO       ;SAVE PRIORITY
112654  011646                PRIOR:  MOV     (SP),-(SP)      ;SAVE RETURN PC
112656  013766  112666  000002        MOV     PRIO,2(SP)      ;SET RETURN PS
112664  000002                        RTI
112666  000000                PRIO:   0                       ;STORAGE
```

```
2801 112670  015  012  124  EM120:  .ASCIZ  <15><12> /TIMEOUT OCCURED WHEN READING ADDRESS /
2802 112740  015  012  116  EM121:  .ASCIZ  <15><12> /NO TIMEOUT OCCURED WHEN ADDRESS WAS WRITTEN/
2803 113016  015  012  102  EM122:  .ASCIZ  <15><12> /BAD STATUS WHEN FIRST CHANNEL WAS LOADED/
2804 113071  015  012  107  EM123:  .ASCII  <15><12>/GENERIC CODE OF 304 OR 324 NOT FOUND/
2805 113137  015  012  106          .ASCII  <15><12>/FATAL ERROR, RETURNING TO MONITOR UNLESS/
2806 113211  015  012  114          .ASCIZ  <15><12>/LOOP ON ERROR IS SELECTED./
2807 113246  015  012  102  EM124:  .ASCIZ  <15><12> /BAD STATUS WHEN LOADING SECOND CHANNEL/
2808 113317  116  117  040  EM125:  .ASCIZ  /NO INTERRUPT ON ERROR CONDITION/
2809 113357  015  012  125  EM126:  .ASCIZ  <15><12> /UNEXPECTED INTERRUPT/
2810 113406  015  012  127  EM127:  .ASCII  <15><12> /WRITING ILLEGAL CHANNEL NUMBER DID NOT/
2811 113456  015  012  103          .ASCIZ  <15><12> /CAUSE FLAG (BIT 7) TO SET IN CSR/
2812 113521  015  012  127  EM130:  .ASCII  <15><12> /WHEN THIRD CHANNEL LOADED TO OVERLOAD/
2813 113570  015  012  111          .ASCIZ  <15><12>/INPUT, BAD STATUS OCCURED/
2814 113624  015  012  127  EM131:  .ASCII  <15><12>         /WHEN FOURTH CHANNEL LOADED TO OVERLOAD/
2815 113674  015  012  111          .ASCIZ  <15><12>/INPUT, BAD STATUS OCCURED/
2816 113730  015  012  103  EM132:  .ASCIZ  <15><12>/C BIT DIDN'T CLEAR STATUS REG AFTER OVERLOADING CHANNEL INPUTS/
2817 114031  015  012  101  EM133:  .ASCII  <15><12>/AFTER OVERLOADING INPUT,SETTING C BIT DID/
2818 114104  015  012  116          .ASCIZ  <15><12> /NOT ENABLE NORMAL OPERATION./
2819 114143  015  012  125  EM134:  .ASCIZ  <15><12> /UNEXPECTED TIMEOUT, FATAL ERROR/
2820 114205  015  012  123  EM135:  .ASCII  <15><12> /SETTING C BIT IN THE IOCM CSR DID NOT/
2821 114254  015  012  103          .ASCIZ  <15><12> /CLEAR THE STATUS REG ON THE A020/
2822 114317  015  012  124  EM136:  .ASCII  <15><12> /THE IOCM CSR WAS NOT CLEARED BY/
2823 114360  015  012  124          .ASCIZ  <15><12> /THE C BIT BEING SET IN THE CSR/
2824 114421  015  012  101  EM137:  .ASCII  <15><12> /AFTER WRITING ILLEGAL CHANNEL NUMBER/
2825 114467  015  012  124          .ASCIZ  <15><12> /THE ERROR BIT WAS NOT SET IN THE STATUS WORD/
2826 114546  015  012  111  EM140:  .ASCII  <15><12> /IAR HAS WRONG ADDRESS AFTER AN INTERRUPT/
2827 114620  015  012  111          .ASCIZ  <15><12> /INTERRUPT CAUSED BY ERROR BIT BEING SET/
2828 114672  015  012  127  EM141:  .ASCIZ  <15><12>/WHEN ERROR BIT WAS SET,R BIT DIDN'T CLEAR ERROR BIT/
2829 114760  015  012  122  EM142:  .ASCII  <15><12> /R BIT WAS SET CLEARING THE ERROR BIT/
2830 115026  015  012  102          .ASCIZ  <15><12> /BUT INTERRUPT OCCURRED ANYWAY/
2831 115066  015  012  103  EM143:  .ASCIZ  <15><12> /CHANNEL WRITTEN WAS NOT SAME AS CHANNEL READ/
2832 115145  015  012  103  EM144:  .ASCII  <15><12> /C BIT DIDN'T CLEAR CONVERSION IN PROGRESS/
2833 115220  015  012  123          .ASCIZ  <15><12> /STATUS REG IN ERROR/
2834 115246  015  012  111  EM145:  .ASCII  <15><12>/IMMEDIATLY AFTER FIRST CHANNEL WAS LOADED/
2835 115321  015  012  124          .ASCIZ  <15><12> /THE STATUS REG WAS IN ERROR (DUMB MODE)/
2836 115373  015  012  104  EM146:  .ASCIZ  <15><12> /DONE BIT NOT SET IN TIME AFTER CONVERSION STARTED (DUMB MODE)/
2837 115473  015  012  101  EM147:  .ASCIZ  <15><12> /AFTER CONVERSION DONE THE STATUS REG WAS NOT CORRECT (DUMB MODE)/
2838 115576  015  012  103  EM150:  .ASCIZ  <15><12> /CHANNEL DATA LOADED WAS NOT THE DATA READ (DUMB MODE)/
2839 115666  015  012  101  EM151:  .ASCII  <15><12> /AFTER LOADING TWO CHANNEL NUMBERS ON STACK/
2840 115742  015  012  123          .ASCIZ  <15><12> /STATUS REG HAS AN ERROR (SMART MODE)/
2841 116011  015  012  101  EM152:  .ASCII  <15><12> /AFTER FIRST INTERRUPT A CHANNEL WAS READ/
2842 116063  015  012  101          .ASCIZ  <15><12> /AND FOUND TO BE IN ERROR (SMART MODE)/
2843 116133  015  012  101  EM153:  .ASCIZ  <15><12> /AFTER LOADING 4 CHANNELS STATUS REG IS IN ERROR (SMART MODE)/
2844 116232  015  012  101  EM154:  .ASCIZ  <15><12> /AFTER SECOND INTERRUPT CHANNEL READ IS IN ERROR (SMART MODE)/
2845 116331  015  012  101  EM155:  .ASCIZ  <15><12> /AFTER LOADING 5 CHANNELS THE STATUS REGISTOR IS IN ERROR (SMART MO
2846 116441  015  012  101  EM156:  .ASCIZ  <15><12> /AFTER THIRD INTERRUPT CHANNEL READ IS IN ERROR (SMART MODE)/
2847 116537  015  012  101  EM157:  .ASCIZ  <15><12> /AFTER THIRD INTERRUPT STATUS IS IN ERROR (SMART MODE)/
2848 116627  015  012  101  EM160:  .ASCIZ  <15><12> /AFTER FOURTH INTERRUPT CHANNEL READ IS IN ERROR (SMART MODE)/
2849 116726  015  012  101  EM161:  .ASCIZ  <15><12> /AFTER FOURTH INTERRUPT STATUS REG IS IN ERROR (SMART MODE)/
2850 117023  015  012  101  EM162:  .ASCIZ  <15><12> /AFTER FIFTH INTERRUPT CHANNEL READ IS IN ERROR (SMART MODE)/
2851 117121  015  012  101  EM163:  .ASCII  <15><12> /AFTER FIFTH INTERRUPT THE LAST ONE EXPECTED,/
2852 117177  015  012  124          .ASCIZ  <15><12> /THE STATUS HAD OTHER THAN ACTIVE (BIT0) SET/
2853 117255  015  012  101  EM164:  .ASCIZ  <15><12> /AFTER LOADING TWO CHANNELS DONE WAS NOT SET IN TIME (SMART MODE)/
2854 117360  015  012  104  EM165:  .ASCIZ  <15><12> /DONE AND DATA OVERRUN NOT SET IN TIME/
2855 117430  015  012  101  EM166:  .ASCII  <15><12> /AFTER A DATA OVERRUN CONDITION PREVIOUS CHANNEL/
2856 117511  015  012  127          .ASCIZ  <15><12> /WAS NOT INTACT (SMART MODE)/
2857 117547  015  012  101  EM167:  .ASCII  <15><12> /AFTER DATA OVERRUN R BIT DID NOT CLEAR/
```

```
2858 117617      015     012     123             .ASCIZ  <15><12> /STATUS REGISTOR (SMART MODE)/
2859 117656      015     012     122  EM170:     .ASCIZ <15><12> /R BIT DIDN'T CLEAR STATUS REG (DUMB MODE)/
2860 117732      015     012     104  EM171:     .ASCIZ <15><12> /D BIT NOT SET, OVERANGE OCCURED/
2861 117774      015     012     104  EM172:     .ASCIZ <15><12> /D BIT SET, OVERANGE DID NOT OCCUR/
2862 120040      015     012     103  EM173:     .ASCIZ <15><12> /CHANNEL INDEPENDENCE ERROR, OVERANGE BIT NOT SET/
2863 120123      015     012     103  EM174:     .ASCII <15><12> /CHANNEL INDEPENDENCE ERROR, AVERAGE NOT NEAR 0/
2864 120203      015     012     117             .ASCIZ <15><12> /OR OVERANGE SET/
2865 120225      015     012     103  EM175:     .ASCIZ <15><12> /CHANNEL INDEPENDENCE STATUS ERROR/
2866 120271      015     012     101  EM200:     .ASCIZ <15><12> /A020   ACCURACY ERROR/
2867 120317      015     012     107  EM201:     .ASCIZ <15><12> /GAIN TEST ERROR/
2868 120341      015     012     111  EM202:     .ASCIZ <15><12> /IAR HAD WRONG ADDRESS UPON INTERRUPT/
2869 120410      015     012     115  EM203:     .ASCIZ <15><12> /MORE COVERSIONS RETURNED THAN LOADED IN BUFFER/
2870 120471      015     012     110  EM204:     .ASCIZ <15><12> /HIGH COMMON MODE TEST FAILURE/
2871 120531      015     012     116  EM205:     .ASCIZ <15><12> /NO INTERRUPT OCCURED /
2872 120561      015     012     127  EM206:     .ASCIZ <15><12> #WHEN THE A/D WAS ALL DONE AND AFTER A DELAY, ACTIVE REMAINED ON#
2873 120663      110     103     115  EM207:     .ASCIZ   /HCM TEST -- LOW READING BELOW LOW LIMIT/
2874 120733      110     103     115  EM210:     .ASCIZ   /HCM TEST -- HIGH READING ABOVE UPPER LIMIT/
2875 121006      110     103     115  EM211:     .ASCIZ   /HCM TEST -- WINDOW READING TOO LARGE/
2876 121053      110     103     115  EM212:     .ASCIZ   /HCM TEST -- COMMOM MODE READING TOO LARGE/
2877
2878 121125      115     117     104  DH15:      .ASCIZ  /MODULE TSTNUM  PC      ITERCNT/
2879 121156      105     122     122  DH24:      .ASCIZ  /ERRPC  MODULE  TSTNUM  ADDR    GDDAT   BDDAT   ITERCNT DATA/
2880 121240      105     122     122  DH65:      .ASCIZ  /ERRPC  MODULE  LO-LIM  LO-DAT  HI-LIM  HI-DAT  WINDOW  CURRENT WINDOW/
2881 121337      115     117     104  DH66:      .ASCIZ  /MOD    TSTNUM  ERRPC   INTRPT# ADDR    CSR/
2882 121401      105     122     122  DH67:      .ASCIZ  /ERRPC  MODULE  TSTNUM  BDADR   ITERCNT/
2883 121443      105     122     122  DH71:      .ASCIZ  /ERRPC  MODULE  TSTNUM  ADDR    GDDAT   BDDAT   ITERCNT/
2884 121520      105     122     122  DH74:      .ASCIZ  /ERRPC  MODULE  TSTNUM  ITERCNT/
2885 121554      105     122     122  DH75:      .ASCIZ  /ERRPC  MODULE  TSTNUM  ACTUAL PC ITERCNT/
2886 121622      105     122     122  DH103:     .ASCIZ  /ERRPC  MODULE  BDADR   ITERCNT/
2887 121655      105     122     122  DH140:     .ASCIZ /ERRPC   MODULE  CHANNEL /
2888 121703      105     122     122  DH142:     .ASCIZ /ERRPC   MODULE  ERRCH   TSTCH   GENCODE BDDAT/
2889 121752      105     122     122  DH143:     .ASCIZ /ERRPC   MODULE  GDDAT   BDDAT   ERRCH   TSTCH   GENCODE OVR-RANGE/
2890 122041      105     122     122  DH144:     .ASCIZ /ERRPC   MODULE  ADDR    GDDAT   BDDAT   ERRCH   TSTCH   GENCODE/
2891 122123      105     122     122  DH145:     .ASCIZ  /ERRPC  MODULE  GDDAT   BDDAT   GAIN    STATUS/
2892 122170      105     122     122  DH150:     .ASCIZ  /ERRPC  MODULE  GDDAT   BDDAT   STATUS  GAIN    CHNUM/
2893                                             .EVEN
2894 122244  003456  001212  001136  DT15:      .WORD   $MUT,   $TESTN, $BDADR, $PASS,
2895 122256  001132  003456  001212  DT24:      .WORD   $ERRPC,$MUT,  $TESTN, $BDADR, $GDDAT, $BDDAT, $PASS, ACOUNT,
2896 122300  001132  003456  003542  DT65:      .WORD   $ERRPC,$MUT,ACOUNT,BCOUNT,CCOUNT,DCOUNT,ECOUNT,FCOUNT,
2897 122322  003456  001212  001132  DT66:      .WORD   $MUT, $TESTN, $ERRPC, ACOUNT, $BDADR,   $BDDAT,
2898 122340  001132  003456  001212  DT67:      .WORD   $ERRPC,$MUT,  $TESTN, $BDADR, $PASS,
2899 122354  001132  003456  001212  DT71:      .WORD   $ERRPC,$MUT,  $TESTN, $BDADR, $GDDAT, $BDDAT, $PASS,
2900 122374  001132  003456  001212  DT74:      .WORD   $ERRPC,$MUT, $TESTN, $PASS,
2901 122406  001132  003456  001212  DT75:      .WORD   $ERRPC,$MUT, $TESTN,$BDADR, $PASS,
2902 122422  001132  003456  003514  DT140:     .WORD   $ERRPC,$MUT, CHNUM,
2903 122432  001132  003456  003542  DT142:     .WORD   $ERRPC, $MUT, ACOUNT, BCOUNT, GCODE, CCOUNT,
2904 122450  001132  003456  001140  DT143:     .WORD   $ERRPC, $MUT, $GDDAT, $BDDAT,ACOUNT, BCOUNT, GCODE, CCOUNT,
2905 122472  001132  003456  001136  DT144:     .WORD   $ERRPC, $MUT, $BDADR, $GDDAT, $BDDAT, ACOUNT, BCOUNT,  GCODE,
2906 122514  001132  003456  001140  DT145:     .WORD   $ERRPC, $MUT, $GDDAT, $BDDAT, BCOUNT, ACOUNT,
2907 122532  001132  003456  001140  DT150:     .WORD   $ERRPC, $MUT, $GDDAT, $BDDAT, ACOUNT, BCOUNT, CHNUM,
2908
2909 122552      000     000     000  DF10:      .BYTE   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 ;THIS COVERS ANYTHING
2910
2911 122572      131     105     123  YES$:      .ASCIZ          /YES/
2912 122576      116     117     000  NO$:       .ASCIZ          /NO/
```

END OF SOURCE FOR DPM50 ASSEMBLY

```
                                .SBTTL   END OF SOURCE FOR DPM50 ASSEMBLY

                        ;*******************************************************************
                        ;       THIS MODULE MUST BE INCLUDED IN THE ASSEMBLY
                        ;       FOR DPM50, OTHERWISE A .END WILL NOT BE SEEN.
                        ;*******************************************************************

006360                          .END    START
```

SYMBOL TABLE

```
A       010332
ABASE = 000000
ACDW1 = 000000
ACDW2 = 000000
ACNTH   003624
ACNTL   003626
ACOUNT  003542
ACPUOP= 000000
ADADDR  017100
ADCALB  060712
ADDW0 = 000000
ADDW1 = 000000
ADDW10= 000000
ADDW11= 000000
ADDW12= 000000
ADDW13= 000000
ADDW14= 000000
ADDW15= 000000
ADDW2 = 000000
ADDW3 = 000000
ADDW4 = 000000
ADDW5 = 000000
ADDW6 = 000000
ADDW7 = 000000
ADDW8 = 000000
ADDW9 = 000000
ADEVCT= 000000
ADEVM = 000000
ADGAN   062516
ADLOG   061620
ADMON   060426
ADRET   060376
ADSET   075466
ADTBIT  062604
ADTST   060260
AD0     003416
AD1     003420
AD2     003422
AD3     003424
AD4     003426
AD5     003430
AD6     003432
AD7     003434
AENV  = 000000
AENVM = 000000
AFATAL= 000000
AFLAG   002664
ALLCLX  055070
AMADR1= 000000
AMADR2= 000000
AMADR3= 000000
AMADR4= 000000
AMAMS1= 000000
AMAMS2= 000000
AMAMS3= 000000
AMAMS4= 000000
AMSGAD  000000
AMSGLG= 000000
AMSGTY= 000000
AMTYP1= 000000
AMTYP2= 000000
AMTYP3= 000000
AMTYP4= 000000
ANSW    003566
APASS = 000000
APRIOR= 000000
APTCSU= 000040
APTENV= 000001
APTSIZ= 000200
APTSPO= 000100
ASWREG= 000000
ATABL   001456
ATEMP   003410
ATESTN= 000000
ATOD    060350
AUNIT = 000000
AUSWR = 000000
AUTO    010760
AUTO20  074646
AVECT1= 000000
AVECT2= 000000
AVERG   112274
AVRSAV  003654
AVRTBL  003374
A20CAL  105426
A20MON  074672
A20SRT  074672
A20TE1  075534
A20TE2  075656
A20TE3  076012
A20TE4  076170
A20TE5  076460
A20TE6  076744
A20TE7  077364
A20T10  100002
A20T11  100266
A20T12  101060
A20T13  101654
A20T14  103546
A631    065672
A631AD  071460
A631C   071526
A631H   066446
A631L   066636
A631M   070600
A631MN  065716
A631S   067766
B       007404
BASE    002660
BCNTH   003630
BCNTL   003632
BCOUNT  003544
BEG     063434
BINCNX  053050
BITSET  015260
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BLANK2  052413
BLAST   003556
BPTVEC= 000014
BSYCON  061752
BTABL   001716
BTEMP   003412
BYTE    002240
BYTNUM  003460
CALLO   111224
CALTAB  104124
CALTOL  111220
CALUP   111222
CAT     002322
CBIT  = 000002
CBITCL  052136
CBITCM  047546
CCNTCX  052702
CCOUNT  003546
CHKBYT  015620
CHNSEL  062416
CHNUM   003514
CHSEL   062250
CH0     003516
CH1     003520
CH2     003522
CH3     003524
CH4     003526
CKICT   057710
CKRWSX  052564
CKSRCX  052646
CK4CLX  052510
CK4CXX  054532
CK4INX  053440
CLEAR   007374
CLEARV  007620
CLK     003476
CLKADR  002642
CLKSTP  042552
CLKVC   002666
CLKVCA  002670
CLRINT  015410
CMDASC  007210
CMMAND  007160
CM1     037245
CM10    037766
CM11    040132
CM12    040177
CM2     037306
CM3     037347
CM4     037410
CM7     037452
CM9     037624
CNTRC   015526
CNUM    003640
CONV    003570
CONVCT  003574
CONV7   065502
COSADR  003470
COUNT   015402
CR    = 000015
CRLF  = 000200
CRTST   015716
CSR     002644
CSS     003436
CTABLE  003656
CTASCI  052322
CTEMP   003414
CTRSH   003634
CTRSL   003636
CTXTV   037062
DACHN   036714
DACMON  055214
DACSTR  055206
DACTST  056132
DAC12   066636
DAC13   067054
DAC14   067212
DAC15   067616
DAGTST  060112
DALLY   014416
DANGC   036516
DANSEL  036652
DATALO  003572
DATST   056136
DBIT  = 000020
DBUFF   003452
DCAMON  055144
DCHAN   003530
DCMN    055240
DCOUNT  003550
DCOUT   055224
DCTST   056230
DDBIT   055512
DDISP = 177570
DECCHN  060156
DF1     067006
DF10    122552
DF100   040725
DF101   040733
DF102   040716
DF51    040740
DF52    040744
DH1     027535
DH100   027434
DH101   027501
DH102   035722
DH103   121622
DH110   035656
DH114   035773
DH140   121655
DH142   121703
DH143   121752
DH144   122041
DH145   122123
DH15    121125
DH150   122170
DH21    027671
DH23    027715
DH24    121156
DH30    036217
DH31    040251
DH33    036044
DH34    036113
DH40    036171
DH5     027566
DH57    031242
DH61    031266
DH62    031341
DH65    121240
DH66    121337
DH67    121401
DH7     027621
DH71    121443
DH74    121520
DH75    121554
DISPLA  001156
DISPRE  000174
DMUT    003472
DR5016  052300
DSWCH   052312
DSWR  = 177570
DTST    056252
DT1     040464
DT100   040354
```

SYMBOL TABLE

```
DT101   040340      EM120   112670      EM202   120341      ERRVEC= 000004      LDATA2  002736
DT102   040320      EM121   112740      EM203   120410      F       012256      LDATA3  002742
DT110   040302      EM122   113016      EM204   120471      FBIT  = 000200      LDTABL  111516
DT140   122422      EM123   113071      EM205   120531      FCOUNT  003554      LF    = 000012
DT142   122432      EM124   113246      EM206   120561      FHOUSX  054474      LOCAL   035615
DT143   122450      EM125   113317      EM207   120663      F5010   014714      LOGIC   011722
DT144   122472      EM126   113357      EM21    023513      F5011   014714      LONGIN  003474
DT145   122514      EM127   113406      EM210   120733      F5012   014726      LOPTST  016162
DT15    122244      EM13    023242      EM211   121006      F5013   014740      M       012472
DT150   122532      EM130   113521      EM212   121053      F6010   014752      MASS0   031400
DT17    040374      EM131   113624      EM213   027146      F6011   015006      MASS10  032073
DT20    040414      EM132   113730      EM215   066734      F6012   014772      MASS11  032135
DT21    040514      EM133   114031      EM216   067152      F6013   014772      MASS12  032170
DT23    040434      EM134   114143      EM217   067552      F6015   015006      MASS13  032230
DT24    122256      EM135   114205      EM22    023555      GAIN    003352      MASS14  032253
DT30    040526      EM136   114317      EM23    023631      GAINMG  036572      MASS15  032300
DT31    040704      EM137   114421      EM24    023666      GAINTB  003330      MASS17  032327
DT33    040542      EM14    023310      EM25    023734      GBIT  = 000004      MASS18  032355
DT34    040564      EM140   114546      EM26    023775      GBITE   003606      MASS19  032363
DT5     040500      EM141   114672      EM27    024051      GCOD    003440      MASS2   031403
DT51    040606      EM142   114760      EM3     022736      GCODE   017254      MASS20  032423
DT52    040622      EM143   115066      EM30    024115      GENER   002404      MASS21  032464
DT57    040640      EM144   115145      EM31    024157      GODN    060042      MASS22  032514
DT61    040650      EM145   115246      EM32    024232      GOUP    060024      MASS23  032543
DT62    040670      EM146   115373      EM33    024307      GROUP   007642      MASS25  032606
DT65    122300      EM147   115473      EM34    024343      GTADRS  055450      MASS26  032633
DT66    122322      EM15    023353      EM35    024430      H       007150      MASS27  032677
DT67    122340      EM150   115576      EM36    024501      HBYTE   003616      MASS28  032702
DT7     040444      EM151   115666      EM37    024547      HDATA0  002730      MASS29  032765
DT71    122354      EM152   116011      EM4     022767      HDATA1  002734      MASS3   031415
DT74    122374      EM153   116133      EM40    024610      HDATA2  002740      MASS30  033037
DT75    122406      EM154   116232      EM41    024653      HDATA3  002744      MASS31  033115
DUMMY   003404      EM155   116331      EM42    024716      HT    = 000011      MASS32  033201
DUMMY1  003406      EM156   116441      EM43    024757      I       013166      MASS33  033213
EBIT  = 000100      EM157   116537      EM44    025034      IAR     002646      MASS34  033242
ECOUNT  003552      EM16    023416      EM45    025073      ICHAN   003536      MASS35  033276
EMTVEC= 000030      EM160   116627      EM46    025140      IFLAG   003442      MASS36  033306
EM1     022626      EM161   116726      EM47    025212      INTCOM  056672      MASS37  033417
EM10    023121      EM162   117023      EM5     023015      INTDAT  003512      MASS38  033463
EM100   026251      EM163   117121      EM50    025240      INTHL1  050466      MASS39  033537
EM101   026321      EM164   117255      EM53    025302      INTHL2  045156      MASS4   031473
EM102   026352      EM165   117360      EM54    025414      INTLD   057622      MASS40  033602
EM103   026424      EM166   117430      EM55    025456      INTR14  064022      MASS41  033655
EM104   026443      EM167   117547      EM56    025531      IOTRAP  077342      MASS42  033726
EM105   026504      EM17    023104      EM57    025572      IOTVEC= 000020      MASS43  033754
EM106   026531      EM170   117656      EM6     023035      K       002636      MASS44  034011
EM107   026573      EM171   117732      EM60    025621      KBINT   015612      MASS45  034051
EM11    023154      EM172   117774      EM61    025655      KBVEC   002724      MASS46  034132
EM110   026615      EM173   120040      EM62    025724      KCLOCK  014454      MASS48  034172
EM111   026654      EM174   120123      EM63    025765      KLEER   014310      MASS49  034240
EM112   026710      EM175   120225      EM7     023104      KOUNT   003540      MASS5   031555
EM113   026742      EM2     022655      EM73    026022      L       007356      MASS50  034307
EM114   026777      EM2X    022724      EM74    026074      LAST    003560      MASS51  031200
EM115   027030      EM2Y    022670      EM75    026124      LASTCN  003562      MASS52  034463
EM116   027062      EM20    023461      EM76    026161      LBYTE   003614      MASS53  034530
EM117   027124      EM200   120271      EM77    026207      LDATA0  002726      MASS55  031047
EM12    023212      EM201   120317      ERROR = 104000      LDATA1  002732      MASS56  034341
```

SYMBOL TABLE

```
MASS57  034401        M7      030034        SAVREG= 104411        TBIT   = 000010        TST36   075534
MASS58  034576        M30     036264        SCCUR   070570        TBITVE= 000014        TST37   075656
MASS59  034652        M81     036331        SCHL    070550        TEMP    003502        TST4    013426
MASS6   031630        M82     036370        SCLL    070540        TEMPB   003650        TST40   076012
MASS60  034707        M83     036441        SCOPE  = 000004       TEMP1   003510        TST41   076170
MASS61  034755        M84     031011        SCSTP   070560        TEMP2   003576        TST42   076460
MASS62  035023        M85     036466        SETBYT  015166        TEMP3   003600        TST43   076744
MASS63  035051        NOCLK   015400        SETCLK  007002        TEMP4   003602        TST44   077364
MASS64  035105        NORAMP  003500        SETPTN  015742        TEMP5   003604        TST45   100002
MASS65  035137        NO$     122576        SETRAM  065452        TEXT    036754        TST46   100266
MASS66  035175        NUM     003642        SR5016  052276        TIMOUT  076146        TST47   101060
MASS67  035250        OUT     064010        STACK  = 001100       TINIMD  044766        TST5    013572
MASS7   031671        OVFS    052302        START   006360        TINIMO  050326        TST50   101654
MASS70  035306        PASCNT  002662        STAT1   003610        TIOCM   013230        TST51   103546
MASS71  035560        PATT    002630        STAT2   003612        TKVEC  = 000060       TST6    013642
MASS8   031750        PIRQ   = 177772       STCO    063450        TMOT12  066612        TST7    013706
MASS9   032020        PIRQVE= 000240        STKLMT= 177774        TMOT13  067030        TYPDS  = 104405
MAXRDX  054602        PRIO    112666        STSTIH  051432        TMOUT   075640        TYPDSW  054316
MBIT   = 000040       PRIOR   112654        STSTI2  046202        TMOVEC  007606        TYPE   = 104401
MESCSS  027731        PRIO    112556        ST1SAV  003620        TPVEC  = 000064       TYPERR  111226
MESTBL  002156        PRI1    112566        ST2SAV  003622        TRANP1  051224        TYPOC  = 104402
MOD     002672        PRI2    112576        SWLOOP  017306        TRANP2  045774        TYPON  = 104404
MODUL   002546        PRI3    112606        SWR     001154        TRAPVE= 000034        TYPOS  = 104403
MONDAT  016044        PRI4    112616        SWREG   000176        TRTVEC= 000014        UPREG   056112
MONIT   007240        PRI5    112626        SW0    = 000001       TSECTB  047124        VAREND  004166
MUT     002466        PRI6    112636        SW00   = 000001       TSTADR  055544        VARI    003644
MUX1    064312        PRI7    112646        SW01   = 000002       TSTBYT  015022        VARSTA  003404
MXNUM   003450        PR0    = 000000       SW02   = 000004       TSTDAT  042550        VCHAN   003534
M0      030016        PR1    = 000040       SW03   = 000010       TSTONE  015104        VECT0   002650
M1      030012        PR2    = 000100       SW04   = 000020       TSTPT1  050136        VECT0A  002652
M10     030036        PR3    = 000140       SW05   = 000040       TSTPT2  044570        VECT1   002654
M11     030040        PR4    = 000200       SW06   = 000100       TSTRGS  055636        VECT1A  002656
M12     030062        PR5    = 000240       SW07   = 000200       TST1    013230        W       012002
M2      030022        PR6    = 000300       SW08   = 000400       TST10   013752        WAIT    071512
M20     030374        PR7    - 000340       SW09   = 001000       TST11   014046        XCHAN   003532
M22     030403        PS     = 177776       SW1    = 000002       TST12   040752        XLOOP   003646
M23     030564        PSW    = 177776       SW10   = 002000       TST13   041044        XXDP    002640
M27     030623        PWRVEC= 000024        SW11   = 004000       TST14   041650        XXX     003506
M3      030024        RAMPST  063340        SW12   = 010000       TST15   042040        YES$    122572
M30     030660        RAMP1   002746        SW13   = 020000       TST16   042230        YLOOP   003444
M4      030026        RAMP2   003066        SW14   = 040000       TST17   042554        ZLOOP   003446
M5      030030        RAMP3   003206        SW15   = 100000       TST2    013316        $APTBU  001236
M5010   040752        RBIT   = 000001       SW2    = 000004       TST20   043540        $APTHD  001100
M5011   042554        RDCHR  = 104406       SW3    = 000010       TST21   044346        $APTPT  001360
M5012   043540        RDLIN  = 104407       SW4    = 000020       TST22   047724        $ATYC   021604
M5013   041044        RDOCT  = 104410       SW5    = 000040       TST23   052416        $ATY1   021560
M5014   044346        RERROR  003504        SW6    = 000100       TST24   055636        $ATY3   021566
M5014R  047712        RESREG= 104412        SW7    = 000200       TST25   056672        $ATY4   021576
M5016   052416        RESVEC= 000010        SW8    = 000400       TST26   061620        $AUTOB  001150
M6      030032        RETURN  003652        SW9    = 001000       TST27   063244        $BASE   001406
M6010   041650        ROTDAT  060060        S5      071614        TST3    013362        $BDADR  001136
M6011   042230        ROTFLG  003326        T       010720        TST30   064312        $BDDAT  001142
M6011X  042540        ROTPAT  003564        TABLE   015230        TST31   065204        $BELL   001176
M6012   042040        RUMP    063244        TABLE1  112170        TST32   066636        $CDW1   001412
M6013   042040        R6     =%000006       TADDR   003462        TST33   067054        $CDW2   001414
M6014   047724        R7     =%000007       TADDR1  003464        TST34   067212        $CHARC  021330
M6015   065204        S       007340        TBADDR  003466        TST35   067616        $CMTAG  001114
```

SYMBOL TABLE

```
$CM3  = 000000        $ENV    001226        $LFLG   022023        $OMODE  022270        $TN   = 000052
$CNTLG  020317        $ENVM   001227        $LPADR  001122        $OVER   017606        $TPB    001166
$CNTLU  020312        $EOP    011722        $LPERR  001124        $PASS   001214        $TPFLG  001173
$CPUOP  001234        $EOPCT  011744        $MADR1  001364        $PASTM  001106        $TPS    001164
$CRLF   001203        $ERFLG  001117        $MADR2  001370        $POWER  022616        $TRAP   022366
$DBLK   021550        $ERMAX  001131        $MADR3  001374        $PWRDN  022450        $TRAP2  022410
$DDW0   001416        $ERROR  020506        $MADR4  001400        $PWRMG  022604        $TRP  = 000013
$DDW1   001420        $ERRPC  001132        $MAIL   001206        $PWRUP  022522        $TRPAD  022422
$DDW10  001442        $ERRTB  004170        $MAMS1  001362        $QUES   001202        $TSTM   001104
$DDW11  001444        $ERRTY  020716        $MAMS2  001366        $RDCHR  017756        $TSTNM  001116
$DDW12  001446        $ERTTL  001126        $MAMS3  001372        $RDLIN  020152        $TTYIN  020302
$DDW13  001450        $ESCAP  001174        $MAMS4  001376        $RDOCT  020346        $TYPDS  021334
$DDW14  001452        $ETABL  001226        $MBADR  001102        $RDSZ = 000010        $TYPE   021052
$DDW15  001454        $ETEND  001456        $MFLG   022022        $RESRE  022330        $TYPEC  021264
$DDW2   001422        $FATAL  001210        $MNEW   020335        $RTNAD  012000        $TYPEX  021332
$DDW3   001424        $FFLG   022024        $MSGAD  001222        $SAVRE  022272        $TYPOC  022052
$DDW4   001426        $FILLC  001172        $MSGLG  001224        $SAVR6  022614        $TYPON  022066
$DDW5   001430        $FILLS  001171        $MSGTY  001206        $SCOPE  017410        $TYPOS  022026
$DDW6   001432        $GDADR  001134        $MSWR   020324        $SETUP= 000027        $UNIT   001220
$DDW7   001434        $GDDAT  001140        $MTYP1  001363        $STUP = 177777        $UNITM  001110
$DDW8   001436        $GET42  011756        $MTYP2  001367        $SVLAD  017552        $USWR   001232
$DDW9   001440        $HIBTS  001100        $MTYP3  001373        $SWR  = 123400        $VECT1  001402
$DEVCT  001216        $HIOCT  020504        $MTYP4  001377        $SWREG  001230        $VECT2  001404
$DEVM   001410        $ICNT   001120        $MULT   017644        $SWRMK= 000000        $XTSTR  017410
$DOAGN  011776        $ILLUP  022610        $MUT    003456        $SW08T  017622        $$GET4= 000000
$DTBL   021540        $INTAG  001151        $NULL   001170        $TESTN  001212        $$SW08= 000012
$ENDAD  011766        $ITEMB  001130        $NWTST= 000000        $TKB    001162        $OFILL  022267
$ENDCT  011752        $LF     001204        $OCNT   022266        $TKS    001160        .$X   = 001100
. ABS.  122601     000
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  55368 WORDS  ( 217 PAGES)
DYNAMIC MEMORY:  21558 WORDS  ( 82 PAGES)
ELAPSED TIME:  00:21:40
CVPCAG/ENABLE:ABS:AMA,CVPCAG/-SP/CR=SYSMAC/ML,MONIT,PARTA,PARTB,END
```