

IP112, IP113  
IPV12

PROCESS CTRL SS  
CVPCAG1

AH-A961G-M1  
FICHE 1 OF 2

SEP 1982  
COPYRIGHT 77-82  
MADE IN USA

Table with multiple columns and rows of data, including headers like 'IP112', 'IP113', 'IPV12', 'PROCESS CTRL SS', 'CVPCAG1', and various numerical values.

IP112, IP113  
IPV12

PROCESS CTRL SS  
CVPCAG1

AH-A961G-M1  
FICHE 2 OF 2

SEP 1982  
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

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

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

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

COPYRIGHT(C)1977,1982 BY DIGITAL EQUIPMENT CORPORATION.

### 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-COUPPLERS 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 POTENTIOMETER. 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 INTERACTIVE 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 ERRORS 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.

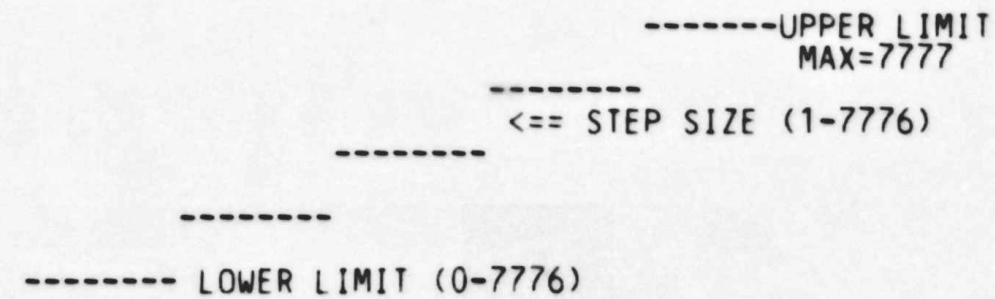
EXAMPLE:

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.



EXAMPLE:

DAC0  
PARAMETERS[Y/N]:Y

DAC 0 (VOLTAGE MODE)  
(Y) WILL REQUEST PARAMETERS  
(N) WILL LEAVE PARAMETERS  
UNCHANGED & SKIP TO NEXT  
DAC

LOWER LIMIT(OCTAL):3000  
UPPER LIMIT(OCTAL):4000  
STEP SIZE(OCTAL):200

SET TO 3000 OR 3.84V  
SET TO 4000 OR 5.12V  
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 SUFFICIENTLY 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 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 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

```
1160 .SBTTL DOCUMENT
1161 123400 $SWR=123400
1162 000001 $TN=1
1163 .SBTTL TRAP CATCHER
      .=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
      .=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
      .=100
1164 000100 000100 NOCLK
1165 000100 015400
1166 000102 000102
1167 000102 000340
1168 000042 000042
1169 000042 000000
1170 000046 000046
1171 000046 011722
1172 000204 000204
1173 000204 000137 007002
1174 .SBTTL BASIC DEFINITIONS
      .SBTTL BASIC DEFINITIONS
      ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
      STACK= 1100
      ERROR =EMT
      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 PIPQ= 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  
1176  
1177  
1178  
1179  
1180  
1181  
1182  
1183  
1184  
1191  
1199  
1206  
1214  
1220  
1248  
1253  
1257  
1267  
1281  
1294  
1304  
1305  
1306

```
000200 .SBTTL BIT DEFINITIONS
000100 FBIT=BIT7 ; FLAG BIT
000040 EBIT=BIT6 ; INTERRUPT ENABLE
000020 MBIT=BIT5 ; MAINTENANCE INTERRUPT
000010 DBIT=BIT4 ; I/O DISABLE BIT
000004 TBIT=BIT3 ; TEST BIT, INVERTS I/O BITS
000002 GBIT=BIT2 ; GENERIC CODE ENABLE
000001 CBIT=BIT1 ; CLEAR I/O
000001 RBIT=BIT0 ; RIF BIT, CLEARS I/O INTERRUPT
```

```
001100 .=1100
.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
000200 200 ;; FOR APT START UP
000044 . =44 ;; POINT TO APT INDIRECT ADDRESS PNTR.
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.
$APTHD:
001100 $HIBTS: .WORD 0 ;; TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
001102 $MBADR: .WORD $MAIL ;; ADDRESS OF APT MAILBOX (BITS 0-15)
001104 $STMT: .WORD 120 ;; RUN TIM OF LONGEST TEST
001106 $PASTM: .WORD 600 ;; RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
001110 $UNITM: .WORD 600 ;; ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
001112 .WORD $ETEND-$MAIL/2 ;; LENGTH MAILBOX-ETABLE (WORDS)
```

1648

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

```
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 $APTPT: .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 $DEV: .WORD ADEV ;;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	034576	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	000011	.WORD	9.	:M5013
002364	000013	.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	

002636 000000  
002640 000042  
002642 177546  
002644 171377  
002646 171376  
002650 000234  
002652 000236  
002654 000000  
002656 000000  
002660 171000  
002662 000001  
002664 000000  
002666 000100  
002670 000102  
002672 015006  
002674 014714  
002676 014714  
002700 014726  
002702 014740  
002704 014752  
002706 015006  
002710 014772  
002712 014772  
002714 014726  
002716 014752  
002720 014714  
002722 000000  
002724 000060  
  
002726 171000  
002730 171001  
002732 171002  
002734 171003  
002736 171004  
002740 171005  
002742 171006  
002744 171007  
002746  
003066  
003206  
003326 177777  
  
003330 000001  
003332 000002  
003334 000010  
003336 000020  
003340 000050  
003342 000100  
003344 000200  
003346 001000  
003350 000000  
003352 000000  
003354 000020  
003356 000100

.EVEN  
K: .WORD 0  
XXDP: .WORD 42  
CLKADR: .WORD 177546  
CSR: .WORD 171377  
IAR: .WORD 171376  
VECTO: .WORD 234  
VECTOA: .WORD 236  
VECT1: .WORD 0  
VECT1A: .WORD 0  
BASE: .WORD 171000  
PASCNT: .WORD 1  
AFLAG: .WORD 0  
CLKVC: .WORD 100  
CLKVCA: .WORD 102  
MOD: .WORD F6015  
          .WORD F5010  
          .WORD F5011  
          .WORD F5012  
          .WORD F5013  
          .WORD F6010  
          .WORD F6011  
          .WORD F6012  
          .WORD F6013  
          .WORD F5012  
          .WORD F6010  
          .WORD F5011  
          .WORD 0  
KBVEC: .WORD 60  
  
LDATA0: .WORD 171000  
HDATA0: .WORD 171001  
LDATA1: .WORD 171002  
HDATA1: .WORD 171003  
LDATA2: .WORD 171004  
HDATA2: .WORD 171005  
LDATA3: .WORD 171006  
HDATA3: .WORD 171007  
RAMP1: .BLKW 40.  
RAMP2: .BLKW 40.  
RAMP3: .BLKW 40.  
ROTFLG: .WORD -1

;TIMING CONSTANT FOR CPU

;CONTROL STATUS REGISTER ADDRESS  
;INTERRUPT ADDRESS REGISTER ADDRESS  
;INTERRUPT VECTOR OF THE IOCM  
;INTERRUPT VECTOR+2 OF THE IOCM

;FIRST I/O

;M6015

;M5012-YA  
;M6010-YA  
;M5031

;TABLE OF A157 GAINS

;TABLE OF A157 GAINS

GAIN: .WORD 1  
          .WORD 2  
          .WORD 10  
          .WORD 20  
          .WORD 50  
          .WORD 100  
          .WORD 200  
          .WORD 1000  
          .WORD 0  
          .WORD 0  
          .WORD 20  
          .WORD 100

;CORRESPONDING OCTAL VALUES TO BE  
;LOADED INTO THE GAIN REGISTER OF A157

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
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
003622 000000 ST2SAV: .WORD 0
003624 000000 ACNTH: 0
003626 000000 ACNTL: 0
003630 000000 BCNTH: 0
003632 000000 BCNTL: 0
003634 000000 CTRSH: 0
003636 000000 CTPSL: 0
003640 000000 CNUM: 0
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 000000 CTABLE: .BLKW 100.
004166 000000 VAREND: .WORD 0
```

;THESE WORDS A-F MUST BE CONTIGUOUS IN CORE

```
;STAT1 SAVE
;STAT2 SAVE
;HIGH BYTE ADDRESS OF COUNTER A
;LOW BYTE ADDRESS OF COUNTER A
;HIGH BYTE ADDRESS OF COUNTER B
;LOW BYTE ADDRESS OF COUNTER B
;COUNTER UNDER TEST HIGH BYTE ADDRESS
;COUNTER UNDER TEST LOW BYTE ADDRESS
;COUNTER NUMBER
```

```
;TABLE C=CATAGORY NUMBERS OF CURRENT SYSTEM
;END OF VARIABLES
```

004170  
1649  
1650 004170 022626  
1651 004172 027535  
1652 004174 040464  
1653 004176 000000  
1654  
1655  
1656 004200 022655  
1657  
1658  
1659 004202 027535  
1660 004204 040464  
1661 004206 000000  
1662  
1663  
1664 004210 022736  
1665 004212 000000  
1666 004214 000000  
1667 004216 000000  
1668  
1669 004220 022767  
1670 004222 027535  
1671 004224 040464  
1672 004226 000000  
1673  
1674 004230 023015  
1675 004232 027566  
1676 004234 040500  
1677 004236 000000  
1678  
1679 004240 023035  
1680 004242 027535  
1681 004244 040464  
1682 004246 000000  
1683  
1684 004250 023104  
1685 004252 027621  
1686 004254 040444  
1687 004256 000000  
1688  
1689 004260 023121  
1690 004262 000000  
1691 004264 000000  
1692 004266 000000  
1693  
1694 004270 023154

```
.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
$ERRTB:
;;;ERROR 1
EM1      ;;.ASCIZ /CBIT NOT CLEARING IOCM/
DH1      ;;.ASCIZ /PC      TST#      GDDAT      BDDAT      PASS/
DT1      ;;.WORD  $ERRPC, $TESTN, $GDDAT, $BDDAT, $PASS,
0
;;;ERROR 2
EM2      ;;.ASCII /IOCM TST/<15><12>
EM2Y:    ;;.ASCII /CSR DATA ERROR WHEN TESTING /
EM2X:    ;;.ASCIZ / BIT /<15><12>
DH1      ;;.ASCIZ /PC      TST#      GDDAT      BDDAT      PASS/
DT1      ;;.WORD  $ERRPC, $TESTN, $GDDAT, $BDDAT, $PASS,
0
;;;ERROR 3
EM3      ;;.ASCIZ /IOCM IS NOT RESPONDING/<15><12>
0
0
0
EM4      ;;.ASCIZ /DBUS BIT IS FAILING/<15><12>
DH1      ;;.ASCIZ /PC      TST#      GDDAT      BDDAT      PASS/
DT1      ;;.WORD  $ERRPC, $TESTN, $GDDAT, $BDDAT, $PASS,
0
EM5      ;;.ASCIZ /NO INTERRUPT /<15><12>
DH5
DT5
0
EM6
DH1
DT1
0
EM7
DH7
DT7
0
EM10
0
0
0
EM11
```

1695	004272	027021	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

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			

1809	004560	024010	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
1865			

:.ASCII <15><12>/LINEARITY TEST ERROR/<15><12>  
:.ASCIZ /ERRPC ADDRESS TSTPNT MIN MAX WAS (# OF CONVERS)/<15><12>

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

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

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>				



2037	005332	035;22	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				



2151	005620	115145	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

2208	005754	122554	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
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
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
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
2262	006102	121655	DH140	:ERRPC	GDDAT	BDDAT	CHNUM	
2263	006104	122422	DT140	:ERRPC	GDDAT	BDDAT	CHNUM	
2264	006106	122552	DF10					

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 INDEPENDENCE 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
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					

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	OCURED	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						

2370 006360

006360 012706 001114  
 006364 005026  
 006366 022706 001154  
 006372 001374  
 006374 012706 001100  
  
 006400 012737 017410 000020  
 006406 012737 000340 000022  
 006414 012737 020506 000030  
 006422 012737 000340 000032  
 006430 012737 022366 000034  
 006436 012737 000340 000036  
 006444 013737 011752 011744  
 006452 005037 001174  
 006456 112737 000001 001131  
 006464 012737 006464 001124  
  
 006472 013746 000004  
 006476 012737 006532 000004  
 006504 012737 177570 001154  
 006512 012737 177570 001156  
 006520 022777 177777 172426  
 006526 001012  
  
 006530 000403  
 006532 012716 006540  
 006536 000002  
 006540 012737 000176 001154  
 006546 012737 000174 001156  
 006554 012637 000004  
 006560 005037 001214  
 006564 132737 000200 001227  
 006572 001403  
 006574 012737 001230 001154  
 006602

START:  
 .SBTTL INITIALIZE THE COMMON TAGS  
 ;;CLEAR THE COMMON TAGS (%CMTAG) AREA  
 MOV #SCMTAG,R6 ;;FIRST LOCATION TO BE CLEARED  
 CLR (R6)+ ;;CLEAR MEMORY LOCATION  
 CMP #SWR,R6 ;;DONE?  
 BNE -6 ;;LOOP BACK IF NO  
 MOV #STACK,SP ;;SETUP THE STACK POINTER  
 ;;INITIALIZE A FEW VECTORS  
 MOV #SCOPE,@IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE  
 MOV #340,@IOTVEC+2 ;;LEVEL 7  
 MOV #ERROR,@EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE  
 MOV #340,@EMTVEC+2 ;;LEVEL 7  
 MOV #TRAP,@TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS  
 MOV #340,@TRAPVEC+2;LEVEL 7  
 MOV \$ENDCT,\$EOPCT ;;SETUP END-OF-PROGRAM COUNTER  
 CLR \$ESCAPE ;;CLEAR THE ESCAPE ON ERROR ADDRESS  
 MOVB #1,\$ERMAX ;;ALLOW ONE ERROR PER TEST  
 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.  
 MOV @ERRVEC,-(SP) ;;SAVE ERROR VECTOR  
 MOV #64\$,@ERRVEC ;;SET UP ERROR VECTOR  
 MOV #DSWR,SWR ;;SETUP FOR A HARDWARE SWICH REGISTER  
 MOV #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER  
 CMP #-1,@SWR ;;TRY TO REFERENCE HARDWARE SWR  
 BNE 66\$ ;;BRANCH IF NO TIMEOUT TRAP OCCURRED  
 ;;AND THE HARDWARE SWR IS NOT = -1  
 BR 65\$ ;;BRANCH IF NO TIMEOUT  
 64\$: MOV #65\$,(SP) ;;SET UP FOR TRAP RETURN  
 RTI  
 65\$: MOV #SWREG,SWR ;;POINT TO SOFTWARE SWR  
 MOV #DISPREG,DISPLAY  
 66\$: MOV (SP)+,@ERRVEC ;;RESTORE ERROR VECTOR  
 CLR \$PASS ;;CLEAR PASS COUNT  
 BITB #APTSIZE,\$ENVM ;;TEST USER SIZE UNDER APT  
 BEQ 67\$ ;;YES,USE NON-APT SWITCH  
 MOV #SSWREG,SWR ;;NO,USE APT SWITCH REGISTER  
 67\$:

2371  
 2372  
 2373

2374 006602 013746 000004  
 2375 006606 013746 000006  
 2376 006612 012737 006636 000004  
 2377 006620 105777 172334  
 2378 006624 012637 000006  
 2379 006630 012637 000004  
 2380 006634 000415  
 2381  
 2382  
 2383 006636 012737 003404 001160  
 2384 006644 012737 003404 001162  
 2385 006652 012737 003406 001164  
 2386 006660 012737 003406 001166  
 2387 006666 000006  
 2388 006670 004737 007620

;; CHECK FOR CONSOLE TTY MODULE  
 MOV ERRVEC,-(SP) ;SAVE LOCATION 4  
 MOV ERRVEC+2,-(SP) ;SAVE LOCATION 6  
 MOV #1\$,ERRVEC ;NEW TIME-OUT ADDRESS  
 TSTB @\$TKS ;TRAP TO 4 IF NO TTY MODULE  
 MOV (SP)+,ERRVEC+2 ;RESTORE 6  
 MOV (SP)+,ERRVEC ;RESTORE 4  
 BR 2\$ ;BRANCH  
 ;; SET UP DUMMY ADDRESSES IF NO CONSOLE TTY MODULE  
 1\$: MOV #DUMMY,\$TKS  
 MOV #DUMMY,\$TKB  
 MOV #DUMMY1,\$TPS  
 MOV #DUMMY1,\$TPB  
 RTT  
 2\$: JSR PC,CLEARVAR ;INIT VARAIABLES

2389



```
2391      .SBTTL  TYPE PROGRAM NAME
          ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
          INC    #-1      ;;FIRST TIME?
          BNE    68$      ;;BRANCH IF NO
          CMP    #$ENDAD,@#42  ;;ACT-11?
          BEQ    68$      ;;BRANCH IF YES
          TYPE   ,69$      ;;TYPE ASCIZ STRING
          BR     68$      ;;GET OVER THE ASCIZ
          ;;69$: .ASCIZ <CRLF>*CVPCAGO MDC PROCESS CONTROL I/O SUBSYSTEM TEST*<CRLF>
          68$:
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 ;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,$WREG ;SET SWR13=1
2409 007104 005037 001214 3$:    CLR    $PASS ;CLEAR PASS COUNT
2410 007110 000137 012472 JMP    M ;UNDER APT - JUMP TO MAPPER ROUTINE
2411
2412 ;
2413 ; IN STANDALONE MODE
2414
2414 007114 012777 015612 173602 4$:    MOV    #KBINT,@KBVEC
          007114 152777 000100 172030 BISB   #BIT6,@$TKS ;ENABLE KB INTERRUPT
2415 007130 012746 000000 MOV    #PRO,-(SP) ;SET PSW TO PRIORITY 0
          007134 012746 007142 MOV    #64$,-(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
```

```

2432                                     ;TABLE OF COMMAND ROUTINES
2433
2434 007160                               CMMAND:
2435 007160 007340                       .WORD S           ;SYSTEM TEST
2436 007162 010332                       .WORD A           ;MODULE TEST
2437 007164 007404                       .WORD B           ;CHANGE BASE ADDRESS
2438 007166 012472                       .WORD M           ;MAP OF DBUS DEVICES
2439 007170 007150                       .WORD H           ;TYPE OPERATOR OPTIONS
2440 007172 013166                       .WORD I           ;IOCM TEST
2441 007174 010720                       .WORD T           ;SET SWREG
2442 007176 007356                       .WORD L           ;LOAD ITERATION COUNT
2443 007200 012256                       .WORD F           ;FIELD TEST
2444 007202 012002                       .WORD W           ;WRAP AROUND TEST
2445 007204 007374                       .WORD CLEAR      ;CLEAR THE SYSTEM
2446 007206 000000                       .WORD 0          ;END OF TABLE FLAG
2447
2448                                     ;TABLE OF ASCII COMMANDS
2449
2450 007210                               CMDASC:
2451 007210 000123                       .WORD 'S         ;SYSTEM TEST
2452 007212 000101                       .WORD 'A         ;MODULE TEST
2453 007214 000102                       .WORD 'B         ;CHANGE BASE ADDRESS
2454 007216 000115                       .WORD 'M         ;MAP OF DBUS DEVICES
2455 007220 000110                       .WORD 'H         ;TYPE OPERATOR OPTIONS
2456 007222 000111                       .WORD 'I         ;IOCM TEST
2457 007224 000124                       .WORD 'T         ;SET SWREG
2458 007226 000114                       .WORD 'L         ;LOAD ITERATION COUNT
2459 007230 000106                       .WORD 'F         ;FIELD TEST
2460 007232 000127                       .WORD 'W         ;WRAP AROUND TEST
2461 007234 000003                       .WORD 3          ;CONTROL C, CLEAR THE SYSTEM
2462 007236 000000                       .WORD 0          ;END OF TABLE FLAG
2463
2464 007240 012706 001100                 MONIT: MOV #1100,R6      ;SET THE STACK
2465 007244 005037 002664                 CLR AFLAG          ;INITIALIZE AUTO RUN FLAG
2466 007250 012777 000340 173374         MOV #340,@VECTOA  ;DISABLE INTERRUPTS
2467
2468 007256                               1$:
2469 007256 012701 177776                 MOV #-2,R1        ;INITIALIZE TABLE POINTER
2470 007262 104401 032170                 TYPE ,MASS12     ;QUERY THE OPERATOR
2471 007266 104406                       RDCHR            ;READ THE KEYBOARD INPUT
2472 007270 012600                       MOV (SP)+,R0     ;POP STACK INTO R0
2473 007272 010037 003566                 MOV R0,ANSW      ;PREPARE TO ECHO CHARACTER
2474 007276 104401 003566                 TYPE ,ANSW       ;ECHO THE CHARACTER
2475 007302                               2$:
2476 007302 062701 000002                 ADD #2,R1        ;ADVANCE TABLE POINTER
2477 007306 020061 007210                 CMP R0,CMDASC(R1);IF TABLE AGREES WITH INPUT
2478 007312 001406                       BEQ 3$          ;--THEN EXECUTE THE COMMAND
2479
2480 007314 005761 007210                 TST CMDASC(R1)  ;--ELSE IF NOT AT END OF TABLE
2481 007320 001370                       BNE 2$          ;--THEN LOOK FOR TABLE MATCH
2482
2483 007322 104401 001202                 TYPE ,SQUES     ;--ELSE UNINTELLIGIBLE ENTRY
2484 007326 000753                       BR 1$           ;---TRY AGAIN
2485 007330                               3$:
2486 007330 104401 001203                 TYPE ,SCRLF     ;EXECUTE THE COMMAND
2487 007334 000171 007160                 JMP @CMMAND(R1)
2488                                     ;SYSTEM TEST
  
```

```

2489 007340
2490 007340 012737 C00001 002664 S:      MOV      #1,AFLAG      ;SET MONITOR LOC FOR AUTO RUN
2491 007346 005037 001214      CLR      $PASS        ;INITIALIZE PASS NUMBER
2492 007352 000137 010760      JMP      AUTO
2493
2494
2495      ;LOAD LOOP COUNT
2496 007356 L:
2497 007356 104401 030623      TYPE     ,M27          ;QUERY FOR NUMBER OF ITERATIONS
2498 007362 104410      RDOCT
2499 007364 012637 002662      MOV      (SP)+,PASCNT ;:READ KEYBOARD
2500 007370 000137 007240      JMP      MONIT        ;:POP STACK INTO PASCNT
2501      ;RETURN TO MONITOR
2502
2503
2504
2505      ;CLEAR THE SYSTEM
2506 007374 CLEAR:
2507 007374 004737 014310      JSR      PC,KLEER     ;CLEAR THE SYSTEM
2508 007400 000137 007240      JMP      MONIT        ;RETURN TO MONITOR
2509
2510      ;CHANGE THE BASE ADDRESS
2511
2512
2513 007404 B:
2514 007404 012737 007574 000004      MOV      #5$,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
2520 007432 021627 160000      CMP      (SP),#160000 ;GET INPUT
2521 007436 103003      BHIS    1$           ;ADDRESS OF I/O PAGE ?
2522 007440 104401 001202      TYPE     ,$QUES       ;YES,BRANCH
2523 007444 000757      BR      B            ;TRY AGAIN
2524
2525 007446 012637 002660 1$:      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,$BDDAT
2531
2532 007510 104401 035105      TYPE     ,MASS64      ;CURRENT VECTOR ADDRESS
2533 007514 013746 002650      MOV      VECT0,-(SP)
2534 007520 104402      TYPOC                ;TYPE ADDRESS
2535
2536 007522 104401 035137 2$:      TYPE     ,MASS65      ;NEW VECTOR ADDRESS
2537 007526 104410      RDOCT
2538 007530 021627 001000      CMP      (SP),#1000  ;GET INPUT
2539 007534 103401      BLO     3$           ;LESS THAN LARGEST FLOATING ADDRESS
2540 007536 000771      BR      2$          ;YES,BRANCH
2541      ;TRY AGAIN
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

```

```
2546 007550 012637 002650      4$:  MOV    (SP)+,VECTO    ;SET NEW VECTOR ADDRESS
2547 007554 013737 C02650 002652  MOV    VECTO,VECTOA
2548 007562 062737 000002 002652  ADD    #2,VECTOA
2549
2550 007570 000137 007240      JMP    MONIT          ;GO BACK TO MONITOR
2551
2552 007574      5$:  EMT    3
      007574 104003      TYPE   ,MASS66        ;TRY AGAIN TURKEY
2553 007576 104401 035175      CMP    (SP)+,(SP)+    ;STORE STACK
2554 007602 022626
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
2563 007616 000000          HALT      ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
2564
2565
2566                      ;THIS ROUTINE ZEROS VARIABLES FROM LOCATION
2567                      ;VARSTART TO VAREND
2568
2569 007620          CLEARVAR:
2570
2571 007620 010046          MOV      RO,-(SP)      ;SAVE RO
2572 007622 012700 003404      MOV      #VARSTART,RO  ;START ADDRESS OF TABLE
2573 007626 005020          CLR      (RO)+          ;CLEAR MEMORY LOCATION
2574 007630 022700 004166      CMP      #VAREND,RO    ;END OF TABLE ?
2575 007634 001374          BNE     1$              ;NO,BRANCH
2576 007636 012600          MOV      (SP)+,RO      ;RESTORE RO
2577 007640 000207          RTS      PC              ;RETURN
2578
```

2580  
2581  
2582  
2583  
2584  
2585  
2586  
2587  
2588  
2589  
2590  
2591  
2592  
2593  
2594  
2595  
2596  
2597  
2598  
2599  
2600  
2601  
2602  
2603  
2604  
2605  
2606  
2607  
2608  
2609  
2610  
2611  
2612  
2613  
2614  
2615  
2616  
2617  
2618  
2619  
2620  
2621  
2622  
2623  
2624  
2625  
2626  
2627  
2628  
2629

```

::: THIS MODULE CONSISTS OF TWO ROUTINES.
::: MAKE ROUTINE === BUILDS THE CURRENT CATAGORY TABLE . CATAGORY
:::                   NUMBERS ARE STORED IN 'CAT' TABLE.
::: CHECK ROUTINE === INSURES HARDWARE MODULES ARE GROUPED ACCORDING
:::                   TO THEIR CATAGORY NUMBER

:: BASE === LOWEST ADDRESS OF D-BUS
:: IAR === BASE + 376
:: BYTE[X] === TABLE OF NUMBER OF BYTES PER MODULE
:: CTABLE === CURRENT SYSTEM MODULE CATAGORY TABLE
:: TADDR === D-BUS ADDRESS
:: GCOD === GENERIC CODE OF HARDWARE MODULE
:: GENER[X] === TABLE OF ALL VALID GENERIC CODES
:: CAT[X] === TABLE OF VALID MODULE CATAGORIES
:: RO === INDEX FOR CTABLE I.E. CTABLE[RO]
:: R1 === INDEX FOR CAT AND GENERIC TABLE I.E. CAT [R1],GENER[R1]
:: TEMP=== SET TO -1 TO INDICATE A TIMEOUT HAS OCCURRED

.REM%
GROUP
MAKE ROUTINE
: SET UP TIMEOUT VECTOR ADDRESS
: CLEAR RO,R1
: SET G-BIT
: SET TADDR= BASE ADDRESS
210$: : REPEAT
: : CLEAR R1
220$: : : READ GCOD AT TADDR (TRAP TO 4 IF NO MODULE AT THIS ADDRESS)
: : CLEAR HIGH BYTE OF GCOD
: : CLEAR TEMP
230$: : : REPEAT
: : : IF GENER[R1] = GCOD
: : : THEN
: : : : CTABLE[RO] = CAT[R1]
: : : : RO = RO + 2
: : : : TADDR = TADDR + BYTE [R1]
240$: : : : ELSE
: : : : R1 = R1 + 2
250$: : : : UNTIL (GCOD = GENER[R1]) OR (GENER[R1]) = 0
: : : IF GENER[R1]=0
: : : THEN
: : : : PRINT 'UNKNOWN GENERIC CODE'
: : : : TADDR = TADDR + 1
270$: : UNTIL TADDR = IAR
END MAKE
```

```

2631 .SBTTL CHECK GROUPING
2632 CHECK CONFIGURATION ROUTINE
2633 275$: : CLEAR R0,R1
2634 280$: : REPEAT
2635 : : WHILE CTABLE[R0] = CTABLE[R0+2] DO
2636 : : : RO=R0+2
2637 : : : END WHILE
2638 : : : R1 = R0 + 4
2639 290$: : : REPEAT
2640 : : : : IF CTABLE[R0] = CTABLE[R1]
2641 : : : : : THEN
2642 : : : : : : PRINT "CONFIGURATION PROBLEM"
2643 300$: : : : : R1 = R1 + 2
2644 : : : : : UNTIL (CTABLE[R1+2]) = 0
2645 : : : : : RO = RO + 2
2646 : : : : : IF TABLE[R0]==-1
2647 : : : : : : THEN
2648 : : : : : : : RO=R0+2
2649 305$: : : UNTIL CTABLE[R0+2] = 0
2650 310$: : CLEAR R0
2651 311$: : WHILE (CTABLE[R0]<>0) OR (CTABLE[R0] <>8) DO
2652 : : : RO=R0+2
2653 312$: : : IF CTABLE[R0]=8
2654 : : : : THEN
2655 : : : : : REPEAT
2656 : : : : : : IF CTABLE[R0]=7
2657 : : : : : : : THEN
2658 : : : : : : : : PRINT "CONFIGURATION PROBLEM"
2659 313$: : : : : : RO=R0+2
2660 : : : : : : UNTIL CTABLE[R0]=0
2661 319$: : END CHECK
2662
2663 TRAP ROUTINE
2664 320$: : TADDR = TADDR + 1
2665 : : IF TEMP = 0
2666 : : : THEN
2667 : : : : TEMP = -1
2668 : : : : : CTABLE[R0] = -1
2669 : : : : : RO =RO + 2
2670 : : UPDATE RETURN ADDRESS
2671 : : END TRAP
2672
2673 007642 012737 010252 000004 % GROUP: MOV #320$,ERRVEC
2674 007650 005037 000006 CLR ERRVEC+2 ;CLEAR PSW
2675 007654 005000 CLR R0
2676 007656 005001 CLR R1
2677 007660 152777 000004 172756 BISB #GBIT,@CSR ;ENABLE G-BIT
2678 007666 013737 002660 003462 MOV BASE,TADDR ;STARTING ADDRESS
2679
2680 007674 005001 210$: CLR R1 ;ZERO REG 1
2681 007676 117737 173560 003440 220$: MOVB @TADDR,GCOD ;READ GCOD
2682 007704 042737 177400 003440 BIC #177400,GCOD ;CLEAR HI BYTE
2683 007712 005037 003502 CLR TEMP
2684
2685
2686 :***REPEAT UNTIL GENER(R1)=GCOD OR END OF GENER TABLE
2687 007716 230$:
  
```

```
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$:   EMT      57
2706 010014 062737 000001 003462      260$:   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                307$:   CLR      R0
```



```

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

```
2779          .SBTTL DIGITAL MODULE MONITOR
2780 010332 004737 C07620          A: JSR PC,CLEARVAR ;CLEAR VARIABLES
2781 010336 012777 015612 172360 MOV #KBINT,@KBVEC
      010344 152777 000100 170606 BISB #BIT6,@$TKS ;ENABLE KB INTERRUPT
2782 010352 104401 032230 TYPE ,MASS13 ;TYPE ADDRESS OF MUT
2783 010356 104410 RDOCT
2784 010360 012637 003462 MOV (SP)+,TADDR ;:POP STACK INTO TADDR
2785 010364 023737 003462 002660 CMP TADDR,BASE
2786 010372 103757 BLO A
2787 010374 013746 000004 MOV ERRVEC,-(SP) ;SAVE ERROR VECTOR
2788 010400 012737 010704 000004 MOV #5$,ERRVEC ;SET LOC 4
2789 010406 004737 014310 JSR PC,KLEER ;CLEAR THE IOCM
2790 010412 105777 172226 TSTB @CSR ;MAKE SURE IOCM IS OK
2791 010416 001404 BEQ 17$
2792 010420 104010 EMT 10
2793 010422 012637 000004 MOV (SP)+,ERRVEC ;:POP STACK INTO ERRVEC
2794 010426 000207 RTS PC
2795 010430 112777 000004 172206 17$: MOVB #GBIT,@CSR ;SET GENERIC BIT
2796 010436 117700 173020 MOVB @TADDR,R0 ;R0=GENERIC CODE OF MUT
2797 010442 042700 177400 BIC #177400,R0
2798 010446 012701 002546 MOV #MODUL,R1 ;ADDRESS OT TEST
2799 010452 012703 002466 MOV #MUT,R3
2800 010456 012702 002404 MOV #GENER,R2 ;TABLE OF GENERIC CODES
2801 010462 020022 3$: CMP R0,(R2)+
2802 010464 001054 BNE 1$
2803 010466 011337 003456 MOV (R3),$MUT
2804 010472 004737 014310 JSR PC,KLEER
2805 010476 011137 003472 MOV (R1),DMUT
2806 010502 005037 001214 9$: CLR $PASS
2807 010506 004777 172760 4$: JSR PC,@DMUT ;TEST MODULE
2808 010512 004737 015526 JSR PC,CNTRC ;CONTROL C?
2809 010516 005737 002662 TST PASCNT ;IF ZERO LOOP ON TEST
2810 010522 001771 BEQ 4$
2811 010524 005237 001214 INC $PASS
2812 010530 023737 002662 001214 CMP PASCNT,$PASS
2813 010536 001363 BNE 4$
2814 010540 032777 020000 170406 BIT #BIT13,@SWR ;INHIBIT PRINT?
2815 010546 001011 BNE 7$
2816 010550 104401 001203 TYPE , $CRLF
2817 010554 013746 002662 MOV PASCNT,-(SP) ;:SAVE PASCNT FOR TYPEOUT
      010560 104402 TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
2818 010562 104401 030040 TYPE ,M11 ;# OF PASSES
2819 010566 104401 001203 TYPE , $CRLF
2820 010572 032777 040000 170354 7$: BIT #BIT14,@SWR ;LOOP?
2821 010600 001340 BNE 9$
2822 010602 104401 032300 TYPE ,MASS15 ;END OF MODULE TEST
2823 010606 004737 014310 JSR PC,KLEER
2824 010612 000137 007240 JMP MONIT
2825 010616 062701 000002 1$: ADD #2,R1
2826 010622 062703 000002 ADD #2,R3
2827 010626 005712 TST (R2) ;END OF TABLE OF GENERIC CODES
2828 010630 001314 BNE 3$
2829 010632 132700 000004 BITB #BIT2,R0 ;IF CSS MODULE
2830 010636 001406 BEQ 13$ ;-- BIT2
2831 010640 132700 000010 BITB #BIT3,R0 ;--AND BIT3
2832 010644 001403 BEQ 13$ ;--WILL BE SET
2833 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	RO,-(SP)		::SAVE RO 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		

```
2846  
2847  
2848 010720 023727 001154 177570 T:    CMP    SWR,#DSWR    ;IS THERE A HARDWARE SWR ?  
2849 010726 001003          BNE    1$          ;NO  
2850 010730 104401 030564          TYPE   ,M23        ;"USE HARDWARE SWR"  
2851 010734 000407          BR     2$          ;RETURN  
2852 010736 104401 030403 1$:    TYPE   ,M22        ;SWICHES OPTIONS SW REG =  
2853 010742 104410          RDOCT  
2854 010744 012677 170204          MOV    (SP)+,@SWR  ;:POP STACK INTO @SWR  
2855 010750 104401 031400          TYPE   ,M22        ;CR,LF  
2856 010754 000137 007240 2$:    JMP    MONIT  
2857  
2858  
2859  
2860  
2861
```

```
2864 .SBTTL AUTO TEST MONITOR
2865
2866
2867 ;THIS PART OF A PROGRAM IS A MONITOR FOR AUTOMATIC TEST.
2868 ;FIRST IT TESTS THE IOCM, THEN IT GENERATES A TABLE AND B TABLE.
2869 ;THEN IT TESTS EACH INDIVIDUAL I/O MODULE CONNECTED TO DBUS.
2870 ;REGISTERS R0 & R1 SERVE AS THE POINTERS FOR MUT
2871
2872 010760 AUTO:
2873 010760 012737 010776 000004 MOV #40$,ERRVEC
2874 010766 052777 000100 171646 BIS #BIT6,@CLKADR ;ENABLE LINE CLOCK INTERRUPT (UNIBUS COMPUTERS)
2875 010774 000401 BR 41$
2876 010776 022626 40$: CMP (SP)+,(SP)+ ;TRAP HERE IF LSI11
2877 011000 012737 007606 000004 41$: MOV #TMOVEC,ERRVEC ;RESTORE ERROR VECTOR
2878 011006 004737 013230 JSR PC,TIOCM
2879 011012 004737 014310 JSR PC,KLEER ;CLEAR THE IOCM
2880 011016 105777 171622 TSTB @CSR ;MAKE SURE IOCM IS OK
2881 011022 001402 BEQ 7$
2882 011024 104010 EMT 10
2883 011026 000000 HALT ;STOP HERE ! BIG PROBLEMS
2884 011030 013746 000004 7$: MOV @#ERRVEC,-(SP) ;SAVE ERROR VECTOR
2885 011034 012737 011206 000004 MOV #1$,ERRVEC ;SET ERROR VECTOR
2886 011042 013700 002660 MOV BASE,R0 ;R0=171000
2887 011046 112777 000004 171570 MOVB #GBIT,@CSR ;SET GENERIC CODE ENABLE
2888 011054 012702 001456 MOV #ATABL,R2
2889 011060 012705 001716 MOV #BTABL,R5
2890 011064 111001 5$: MOVB (R0),R1 ;R1=GENERIC CODE,READ GENERIC CODE
2891 011066 005037 003436 CLR CSS ;CLEAR CSS FLAG
2892 011072 042701 177400 BIC #177400,R1
2893 011076 004737 015230 JSR PC,TABLE ;FIND WHAT I/O IT IS
2894 011102 005764 002240 TST BYTE(R4) ;VALID MODULE ?
2895 011106 001015 BNE 3$ ;YES,BRANCH
2896 011110 005200 INC R0 ;ADD 1 TO ADDRESS
2897 011112 132701 000004 BITB #BIT2,R1 ;IF CSS MODULE
2898 011116 001413 BEQ 13$ ;- BIT2
2899 011120 132701 000010 BITB #BIT3,R1 ;--AND BIT3
2900 011124 001410 BEQ 13$ ;---WILL BE SET
2901 011126 105737 003436 TSTB CSS ;PRINT CSS WARNING MESSAGE ?
2902 011132 001005 BNE 13$ ;NO,BRANCH
2903 011134 104401 027731 TYPE ,MESCSS ;TYPE MESSAGE
2904 011140 000402 BR 13$ ;BRANCH
2905 011142 066400 002240 3$: ADD BYTE(R4),R0 ;ADD # OF BYTES TO ADDRESS
2906 011146 020037 002646 13$: CMP R0,IAR
2907 011152 103744 BLO 5$
2908 011154 005012 CLR (R2)
2909 011156 022702 001456 CMP #ATABL,R2
2910 011162 001017 BNE 6$
2911 011164 012637 000004 MOV (SP)+,ERRVEC ;;POP STACK INTO ERRVEC
2912 011170 142777 000004 171446 BICB #GBIT,@CSR
2913 011176 104401 034307 TYPE ,MASS50 ;NO I/O
2914 011202 000137 011526 JMP 26$
2915 011206 022626 1$: CMP (SP)+,(SP)+
2916 011210 005200 INC R0
2917 011212 020037 002646 CMP R0,IAR
2918 011216 103722 BLO 5$
2919 011220 005015 CLR (R5)
2920 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 C13230          25$:   JSR      PC,TIOCM
2923 011240 005000          CLR     R0
2924 011242 005001          22$:   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          MOV     MXNUM,@STAT1
2959 011452 105777 172132          TST     @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 C01214 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    JSR      PC,KLEER     ;CLEAR IOCM
      24$:    TST      @XXDP        ;UNDER XXDP ?
2991 011644 005777 170770    BEQ      51$          ;NO
2992 011650 001403          DEC      $PASS        ;YES, FIX $PASS
2993 011652 005337 001214    BR       LOGIC        ;GO TO $EOP
2994 011656 000421          TYPE     ,MASS17      ;END OF SYSTEM TEST
2995 011660 104401 032327    51$:    CLR      AFLAG
2996 011664 005037 002664    JMP      MONIT
2997 011670 000137 007240
2998
2999
3000
3001 011674 152777 000001 170742 31$:    ;TIMEOUT TRAPS HERE WITH PC & PSW ON STACK
      BISB   #RBIT,@CSR
3002 011702 013716 003610    MOV      STAT1,(SP)   ;SAVE ADDRESS
3003 011706 042716 000003    BIC      #3,(SP)     ;MAKE MODULE BASE ADDRESS
3004 011712 005776 000000    TST      @(SP)       ;CLEAR THE MODULE
3005 011716 022626          CMP      (SP)+,(SP)+ ;CLEAN STACK
3006 011720 000660          BR       33$
```

3008 011722

011722  
011722 000004  
011724 005037 001116  
011730 005237 001214  
011734 042737 100000 001214  
011742 005327  
011744 000001  
011746 003013  
011750 012737  
011752 000001  
011754 011744  
011756 013700 000042  
011762 001405  
011764 000005  
011766 004710  
011770 000240  
011772 000240  
011774 000240  
011776  
011776 000137  
012000 010760

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  
\$EOP:  
SCOPE  
CLR \$STNM ;;ZERO THE TEST NUMBER  
INC \$PASS ;;INCREMENT THE PASS NUMBER  
BIC #100000,\$PASS ;;DON'T ALLOW A NEG. NUMBER  
DEC (PC)+ ;;LOOP?  
\$EOPCT: .WORD 1  
BGT \$DOAGN ;;YES  
MOV (PC)+,@(PC)+ ;;RESTORE COUNTER  
\$ENDCT: .WORD 1  
\$EOPCT  
\$GET42: MOV @#42,R0 ;;GET MONITOR ADDRESS  
BEQ \$DOAGN ;;BRANCH IF NO MONITOR  
RESET ;;CLEAR THE WORLD  
\$ENDAD: JSR PC,(R0) ;;GO TO MONITOR  
NOP ;;SAVE ROOM  
NOP ;;FOR  
NOP ;;ACT11  
\$DOAGN:  
JMP @(PC)+ ;;RETURN  
\$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  
:  
:  
3010  
3011  
3012  
3013  
3014  
3015  
3016 012002 104401 035306 W: TYPE ,MASS70 ;TYPE WARNING MESSAGE  
3017 012006 132737 000001 001226 BITB #BIT0,$ENV ;UNDER APT ?  
3018 012014 001404 BEQ 7$ ;NO,BRANCH  
3019 012016 104401 035615 TYPE ,LOCAL ;"LOCALMODE ONLY"  
3020 012022 000137 007240 JMP MONIT ;RETURN TO MONITOR  
3021 012026 104401 032543 7$: TYPE ,MASS23 ;CONNECT OUTPUT TO INPUT MODULE  
3022 012032 012777 015612 170664 MOV #KBINT,@KBVEC  
3023 012040 152777 000100 167112 BISB #BIT6,@$TKS ;ENABLE KB INTERRUPT  
3023 012046 104401 032464 TYPE ,MASS21 ;TYPE ADDRESS OF OUTPUT MUT  
3024 012052 104410 RDOCT  
3025 012054 012637 003462 MOV (SP)+,TADDR ;:POP STACK INTO TADDR  
3026 012060 104401 032514 TYPE ,MASS22 ;TYPE ADDRESS OF INPUT MUT  
3027 012064 104410 RDOCT  
3028 012066 012637 003510 MOV (SP)+,TEMP1 ;:POP STACK INTO TEMP1  
3029 012072 112777 000004 170544 MOVB #GBIT,@CSR ;SET GENERIC BIT  
3030 012100 117700 171356 MOVB @TADDR,RO ;GENERIC CODE OF OUTPUT MUT  
3031 012104 020027 000041 CMP RO,#41 ;IS IT M6010?  
3032 012110 001051 BNE 1$ ;NOT M6010, ERROR  
3033 012112 117701 171372 MOVB @TEMP1,R1 ;GENERIC CODE OF INPUT MUT  
3034 012116 020127 000141 CMP R1,#141 ;IS IT M5010?  
3035 012122 001403 BEQ 2$  
3036 012124 020127 000121 CMP R1,#121 ;IS IT M5011?  
3037 012130 001044 BNE 6$  
3038 012132 005037 001214 2$: CLR $PASS  
3039 012136 004737 016162 3$: JSR PC,LOPTST ;TEST SUBROUTINE  
3040 012142 004737 015410 JSR PC,CLRINT ;CLEAR ALL INTERRUPTS  
3041 012146 004737 015526 JSR PC,CNTRC ;CONTROL-C?  
3042 012152 005237 001214 INC $PASS  
3043 012156 023737 002662 001214 CMP PASCNT,$PASS ;  
3044 012164 001364 BNE 3$  
3045 012166 032777 020000 166760 BIT #BIT13,@SWR ;INHIBIT PRINTOUT  
3046 012174 001005 BNE 5$  
3047 012176 013746 002662 MOV PASCNT,-(SP) ;:SAVE PASCNT FOR TYPEOUT  
3048 012202 104402 TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)  
3048 012204 104401 030040 TYPE ,M11 ;PASSES COMPLETED  
3049 012210 032777 040000 166736 5$: BIT #BIT14,@SWR ;LOOP  
3050 012216 001345 BNE 2$  
3051 012220 104401 032606 TYPE ,MASS25 ;END OF LOOP TEST  
3052 012224 004737 014310 JSR PC,KLEER  
3053 012230 000137 007240 JMP MONIT  
3054 012234 104401 032633 1$: TYPE ,MASS26 ;WRONG MODULE -OUTPUT MUST BE M6010  
3055 012240 000402 BR 4$  
3056 012242 104401 032702 6$: TYPE ,MASS28 ;WRONG MODULE- INPUT MUST BE EITHER M5010 OR M5011  
3057 012246 004737 014310 4$: JSR PC,KLEER  
3058 012252 000137 007240 JMP MONIT
```

```
3060          .SBTTL  MODULE TEST
3061
3062          ;THIS TEST ENABLE FIELD ENGINEER TO SELECT AND OUTPUT ANY DATA
3063          ;PATTERN TO OUTPUT MODULES,MONITOR AND PRINT DATA FROM INPUT MODULES
3064          ;
3065          ;
3066 012256 132737 000001 001227 F:   BITB  #BIT0,$ENVM   ;UNDER APT ?
3067 012264 001404          BEQ    6$          ;NO BRANCH
3068 012266 104401 035615          TYPE  ,LOCAL      ;"LOCAL MODE ONLY"
3069 012272 000137 007240          JMP   MONIT      ;RETURN TO MONITOR
3070 012276 104401 035306          6$:   TYPE  ,MASS70  ;TYPE WARNING MESSAGE
3071 012302 104401 032230          TYPE  ,MASS13  ;TYPE ADDRESS OF MUT
3072 012306 012777 015612 170410  MOV   #KBINT,@KBVEC
3073 012314 152777 000100 166636  BISB  #BIT6,@$TKS ;ENABLE KB INTERRUPT
3074 012322 104410          RDOCT
3075 012324 012637 003462          MOV   (SP)+,TADDR ;:POP STACK INTO TADDR
3076 012330 013746 000004          MOV   ERRVEC,-(SP) ;SAVE ERROR VECTOR
3077 012334 012737 012456 000004  MOV   #5$,ERRVEC  ;SET LOC 4
3078 012342 112777 000004 170274  MOVB  #GBIT,@CSR  ;SET GENERIC BIT
3079 012350 117700 171106          MOVB  @TADDR,R0   ;GENERIC CODE OF MUT
3080 012354 042700 177400          BIC   #177400,R0
3081 012360 012701 002672          MOV   #MOD,R1    ;ADDRESS OF TEST
3082 012364 012702 002404          MOV   #GENER,R2  ;TABLE OF GENERIC CODES
3083 012370 020022          3$:   CMP   R0,(R2)+
3084 012372 001013          BNE   2$
3085 012374 004737 014310          JSR   PC,KLEER
3086 012400 011137 003472          MOV   (R1),DMUT
3087 012404 004777 171062          JSR   PC,@DMUT
3088 012410 104401 032423          TYPE  ,MASS20
3089 012414 004737 015526          1$:   JSR   PC,CNTRC ;TEST MODULE
3090 012420 000775          BR    1$         ;TYPE CONTROL-C TO RETURN TO MONITOR
3091 012422 062701 000002          2$:   ADD   #2,R1    ;CONTROL-C ?
3092 012426 062703 000002          ADD   #2,R3
3093 012432 005711          TST   (R1)
3094 012434 001355          BNE   3$
3095 012436 104401 035250          TYPE  ,MASS67   ;DIGITAL I/O ONLY
3096 012442 012637 000004          MOV   (SP)+,ERRVEC ;RESTORE ERRVEC
3097 012446 004737 014310          JSR   PC,KLEER
3098 012452 000137 007240          JMP   MONIT
3099 012456 022626          5$:   CMP   (SP)+,(SP)+ ;RESTORE STACK POINTER
3100 012460 104024          EMT   24
3101 012462 012637 000004          MOV   (SP)+,ERRVEC
3102 012466 000137 006360          JMP   START
```

3103  
3104  
3105  
3106  
3107  
3108  
3109  
3110  
3111  
3112  
3113  
3114  
3115  
3116  
3117  
3118  
3119  
3120  
3121  
3122  
3123  
3124  
3125  
3126  
3127  
3128  
3129  
3130  
3131  
3132  
3133  
3134  
3135  
3136  
3137  
3138  
3139  
3140  
3141  
3142  
3143  
3144  
3145  
3146  
3147  
3148  
3149  
3150  
3151  
3152  
3153  
3154  
3155  
3156  
3157

.SBTTL MAP OF DBUS

;THIS MAPPER WILL PRINT I.D. MESSAGES OF PCS MODULES  
;--CONNECTED TO THE D-BUS. A SPECIAL CSS MESSAGE WILL  
;--BE PRINTED INFORMING THE USER THAT THE MODULE IS  
;---NOT TESTED BY THIS SOFTWARE.

012472	013746	000004		M:	MOV	ERRVEC,-(SP)	;SAVE ERROR VECTOR
012476	012737	013136	000004		MOV	#130\$,ERRVEC	;SET ERROR VECTOR
012504	012777	015612	170212		MOV	#KBINT,@KBVEC	
012512	152777	000100	166440		BISB	#BIT6,@\$TKS	;ENABLE KB INTERRUPT
012520	013700	002660			MOV	BASE,R0	;BASE ADDRESS IOCM
012524	004737	014310			JSR	PC,KLEER	;CLEAR EVERYTHING FOP SURE
012530	004737	015410			JSR	PC,CLRINT	;CLEAR ALL INTERRUPTS
012534	112777	000004	170102		MOVSB	#GBIT,@CSR	;SET GENERIC CODE ENABLE
012542	104401	001203			TYPE	,\$CRLF	
012546	111001			5\$:	MOVSB	(R0),R1	;READ GENERIC CODE
012550	104401	031403			TYPE	,MASS2	;ADDRESS
012554	010046				MOV	R0,-(SP)	::SAVE R0 FOR TYPEOUT
012556	104402				TYPOC		::GO TYPE--OCTAL ASCII(ALL DIGITS)
012560	042701	177400			BIC	#177400,R1	;CLEAR HIGH BYTE
012564	005004				CLR	R4	;R4 WILL BE THE INDEX
012566	012703	002404			MOV	#GENER,R3	;STARTING ADDRESS GENERIC CODE TABLE
012572	022301			10\$:	CMP	(R3)+,R1	;GENERIC CODE MATCH ?
012574	001431				BEQ	20\$	;YES,BRANCH
012576	062704	000002			ADD	#2,R4	;BUMP INDEX
012602	005713				TST	(R3)	;END OF TABLE ?
012604	001372				BNE	10\$	;NO,BRANCH
012606	132701	000004			BITB	#BIT2,R1	;IF CSS MODULE
012612	001413				BEQ	15\$	;-BIT2 AND
012614	132701	000010			BITB	#BIT3,R1	;-BIT3
012620	001410				BEQ	15\$	;-WILL BE SET
012622	105737	003436			TSTB	CSS	;PRINT MESSAGE ?
012626	001114				BNE	40\$	;NO,BRANCH
012630	104401	027731			TYPE	,MESCSS	;CSS MODULE I.D. MESSAGE
012634	005237	003436			INC	CSS	;SET PRINT ONLY ONCE FLAG
012640	000507				BR	40\$	;BUMP D-BUS BY 1
012642	104401	032135		15\$:	TYPE	,MASS11	;UNKNOWN GENERIC CODE
012646	010146				MOV	R1,-(SP)	;GENERIC CODE OF MODULE
012650	104402				TYPOC		
012652	104401	001203			TYPE	,\$CRLF	
012656	000500				BR	40\$	;BUMP D-BUS ADDRESS BY 1
012660	016437	002156	012670	20\$:	MOV	MESTBL(R4),25\$	;MODULE I.D. MESSAGE
012666	104401				TYPE		;PRINT IT
012670	000000			25\$:	.WORD	0	
012672	005037	003436			CLR	CSS	;CLEAR CSS FLAG
012676	022701	000321			CMP	#321,R1	;A014 SINGLE ENDED ?
012702	001404				BEQ	30\$	;YES,BRANCH

```
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$:  MOV     (SP)+,ERRVEC ;RESTORE ORIGINAL VECTOR
3208 013106 004737 007642      JSR    PC,GROUP   ;CHECK GROUPINGS
3209 013112 004737 014310      JSR    PC,KLEER   ;CLEAR D-BUS
3210 013116 132737 000001 001226  BITB   #BIT0,$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
```

IOCM TEST

3230  
3231  
3232  
3233  
3234  
3235  
3236  
3237  
3238  
3239  
3240  
3241  
3242  
3243  
3244  
3245  
3246  
3247  
3248  
3249  
3250  
3251  
3252  
3253  
3254  
3255  
3256  
3257  
3258  
3259  
3260  
3261  
3262  
3263  
3264  
3265

013166  
013166 013746 000004  
013172 012777 015612 167524  
013200 152777 000100 165752  
013206 004737 017306  
013212 013230  
013214 012637 000004  
013220 004737 014310  
013224 000137 007240  
  
013230 000004  
013232 112737 000001 001212  
013240 012737 013310 000004  
013246 112777 000074 167370  
013254 004737 014310  
013260 005037 001142  
013264 005037 001140  
013270 117737 167350 001142  
013276 105737 001142  
013302 001405  
013304 104001  
013306 000207  
013310 104003  
013312 022626  
013314 000207  
013316

.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

I:

MOV ERRVEC,-(SP) ;:PUSH ERRVEC ON STACK  
MOV #KBINT,@KBVEC  
BISB #BIT6,@\$TKS ;:ENABLE KB INTERRUPT  
JSR PC,SWLOOP  
TIOCM ;:DO IOCM TEST  
MOV (SP)+,ERRVEC ;:POP STACK INTO ERRVEC  
JSR PC,KLEER  
JMP MONIT

;THIS TEST CHECKS IF EACH BIT OF CSR  
;IS CLEARED BY THE C BIT. IT SETS THE CSR TO 74, SETS THE  
;C BIT AND VERIFIES THE CSR IS NOW SET TO 0  
;

TIOCM:

::\*\*\*\*\*

TST1:

SCOPE  
MOVB #1,\$TESTN  
MOV #10\$,ERRVEC  
MOVB #74,@CSR ;:SET ALL BITS  
JSR PC,KLEER ;:SET CBIT  
CLR \$BDDAT  
CLR \$GDDAT  
MOVB @CSR,\$BDDAT ;:STORE CONTENTS OF CSR IN BDDAT  
TSTB \$BDDAT ;:IS CSR CLEAR  
BEQ 2\$ ;:YES  
EMT 1  
RTS PC ;:FATAL ERROR RETURN TO MONITOR

10\$:

EMT 3  
CMP (SP)+,(SP)+ ;:RESTORE STACK  
RTS PC

2\$:

```
3267                                     ;THIS TEST CHECKS E BIT
3268                                     ;*****
TST2: SCOPE
3269 013316 000004                                MOV  #2,$TESTN
3270 013320 112737 000002 001212                MOV  #EBIT,$GDDAT ;LOAD $GDDAT WITH BIT TO BE SET IN CSR
3271 013326 012737 000100 001140                MOV  M7,EM2X      ;SET EBIT IN ERROR MESSAGE
3272 013334 113737 030034 022724                JSR  PC,BITSET    ;GO LOAD CSR AND CHK FOR BITS SET
3273 013342 004737 015260                        EMT  2
3274 013346 104002                                CLR  $GDDAT       ;RETURN HERE IF NO ERROR
3275 013350 005037 001140                        JSR  PC,BITSET
3276 013354 004737 015260                        EMT  2
3277
3278                                     ;THIS TEST CHECKS M BIT
3279                                     ;*****
TST3: SCOPE
3280 013362 000004                                ;SETTING THE M BIT CAUSES THE F BIT TO BE SER
3281
3282 013364 112737 000003 001212                MOV  #3,$TESTN
3283 013372 012737 000240 001140                MOV  #MBIT!FBIT,$GDDAT;LOAD WITH BITS TO RE SET IN CSR
3284 013400 113737 030032 022724                MOV  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  $GDDAT       ;RETURN HERE IF NO ERROR AND CLR DATA TO BE SET
3288 013420 004737 015260                        JSR  PC,BITSET    ;RESET CSR TO 0 AND VERIFY RESULTS
3289 013424 104002                                EMT  2
3290
3291                                     ;THIS TEST CHECKS THE D BIT
3292                                     ;*****
3293                                     ;*****
TST4: SCOPE
3294 013426 000004                                ;CLEARING THE DBIT CAUSES A CLEAR WHICH COULD CAUSE THE
3295                                     ;F BIT TO BE SET
3296 013430 112737 000004 001212                MOV  #4,$TESTN
3297 013436 012737 000020 001140                MOV  #DBIT,$GDDAT
3298 013444 113737 030030 022724                MOV  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  $GDDAT       ;INITIALIZE EXPECTED DATA.
3305 013476 112777 000074 167140                MOV  #74,@CSR    ;SET FEW BITS AT CSR
3306 013504 142777 000020 167132                BICB #DBIT,@CSR  ;CLEAR DBIT
3307 013512 013737 002636 003446                MOV  K,ZLOOP
3308 013520 005037 003444 65$: CLR  YLOOP ;WAIT
3309 013524 005237 003444 64$: INC  YLOOP
3310 013530 023727 003444 000007                CMP  YLOOP,#7
3311 013536 001372                                BNE  64$
3312 013540 005337 003446                        DEC  ZLOOP
3313 013544 001365                                BNE  65$
3314 013546 117737 167072 001142                MOV  @CSR,$BDDAT
3315 013554 132737 000177 001142                BITB #177,$BDDAT ;CSR = 0 ?
3316 013562 001401                                BEQ  1$           ;BR IF YES - ALL OK
3317 013564 104016                                EMT  16
3318 013566 004737 014310 1$: JSR  PC,KLEER     ;SET CLEAR IN THE CSR
```

```
3314                                     ;THIS TEST CHECKS T BIT
3315 :*****
3316 013572 000004                                TST5: SCOPE
3317 013574 112737 000005 001212                MOVB #5,$TESTN
3318 013602 012737 000010 001140                MOV #TBIT,$GDDAT ;LOAD $GDDAT WITH DATA TO BE SET IN CSR
3319 013610 113737 030026 022724                MOVB M4,EM2X ;SET ERROR MESSAGE
3320 013616 004737 015260                JSR PC,BITSET ;SET AND CHECK T BIT
3321 013622 104002                EMT 2
3322 013624 005037 001140                CLR $GDDAT ;LOAD $GDDAT WITH 0 TO BE SET IN CSR
3323 013630 004737 015260                JSR PC,BITSET ;ZERO CSR AND VERIFY RESULTS
3324 013634 104002                EMT 2
3325 013636 004737 015410                JSR PC,CLRINT ;CLEAR ALL INTERRUPT CAUSED BY T BIT
3326
3327 :*****
3328 013642 000004                                TST6: SCOPE
3329 013644 112737 000006 001212                MOVB #6,$TESTN
3330 013652 012737 000004 001140                MOV #GBIT,$GDDAT ;LOAD $GDDAT WITH DATA TO BE SET IN CSR
3331 013660 113737 030024 022724                MOVB M3,EM2X ;FIX ERROR MESSAGE
3332 013666 004737 015260                JSR PC,BITSET ;SET AND CHECK G BIT
3333 013672 104002                EMT 2
3334 013674 005037 001140                CLR $GDDAT ;LOAD $GDDAT WITH 0 TO BE SET IN CSR
3335 013700 004737 015260                JSR PC,BITSET ;ZERO CSR AND VERIFY RESULTS
3336 013704 104002                EMT 2
3337
3338 :*****
3339 :THIS TEST CHECKS RBIT
3340 :*****
3341 013706 000004                                TST7: SCOPE
3342 013710 112737 000007 001212                MOVB #7,$TESTN
3343 013716 012737 000001 001140                MOV #RBIT,$GDDAT ;LOAD $GDDAT WITH DATA TO BE SET IN CSR
3344 013724 113737 030022 022724                MOVB M2,EM2X ;SET ERROR MESSAGE
3345 013732 004737 015260                JSR PC,BITSET ;SET BIT IN CSR AND VERIFY RESULTS
3346 013736 104002                EMT 2
3347 013740 005037 001140                CLR $GDDAT ;LOAD $GDDAT WITH 0 TO BE LOADED INTO CSR
3348 013744 004737 015260                JSR PC,BITSET ;ZERO CSR AND VERIFY RESULTS
3349 013750 104002                EMT 2
```



3349  
3350  
3351  
3352  
3353  
3354  
3355

: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  
3356 013754 112737 000010 001212  
3357 013762 112777 000040 166654  
3358 013770 013700 002660  
3359 013774 005001  
3360 013776 005037 001140  
3361 014002 005037 001142  
3362 014006 111001  
3363 014010 042701 177400  
3364 014014 120001  
3365 014016 001405  
3366 014020 110037 001140  
3367 014024 110137 001142  
3368 014030 104004  
3369 014032 005200  
3370 014034 122700 000376  
3371 014040 001362  
3372 014042 004737 014310

\*\*\*\*\*  
TST10: SCOPE  
MOV #10,\$TESTN  
MOV #MBIT,@CSR ;SET MAINTENANCE MODE  
MOV BASE,R0  
CLR R1  
CLR \$GDDAT  
CLR \$BDDAT  
1\$: MOV (R0),R1 ;READ FIRST ADDRESS  
BIC #177400,R1  
CMPB R0,R1 ;CHECK IF DBUS=ADDRESS  
BEQ 2\$  
MOV R0,\$GDDAT  
MOV R1,\$BDDAT  
EMT 4  
2\$: INC R0  
CMPB #376,R0 ;IS IT THE LAST ADDRESS  
BNE 1\$  
JSR PC,KLEER ;CLEAR IOCM

```

3374
3375
3376
3377
3378
3379
3380
3381 014046 000004
3382 014050 112737 000011 001212
3383 014056 012777 014166 166564
3384 014064 012777 000340 166560
3385 014072 013746 000004
3386 014076 012737 014302 000004
3387 014104 012746 000000
3388 014110 012746 014116
3389 014114 000002
3390 014116 000240
3391 014120 112777 000140 166516
3392 014126 013737 002636 003446
3393 014134 005037 003444
3394 014140 005237 003444
3395 014144 023727 003444 177777
3396 014152 001372
3397 014154 005337 003446
3398 014160 001365
3399 014162 104014
3400 014164 000433
3401 014166 022626
3402 014170 117737 166452 003502
3403 014176 122737 000377 003502
3404 014204 001411
3405 014206 012737 000377 001140
3406 014214 005037 001142
3407 014220 113737 003502 001142
3408 014226 104006
3409 014230 152777 000001 166406
3410 014236 105777 166402
3411 014242 132777 000200 166374
3412 014250 001401
3413 014252 104015
3414 014254 004737 014310
3415 014260 012637 000004
3416 014264 012746 000000
3417 014270 012746 014276
3418 014274 000002
3419 014276 000240
3420 014300 000207
3421 3408
3422 3409 014302 022626
3423 3410 014304 104022
3424 3411 014306 000762

```

```

: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
MOV #11,$TESTN
MOV #5$,@VECTO ;SET VECTOR ADDRESS
MOV #PR7,@VECTOA ;SET VECTOR+2 ADDRESS
MOV ERRVEC,-(SP) ;:PUSH ERRVEC ON STACK
MOV #8$,ERRVEC
MOV #PRO,-(SP) ;SET PSW TO PRIORITY 0
MOV #64$,-(SP)
RTI
64$: NOP
1$: MOV #EBIT!MBIT,@CSR;START INTERRUPT
MOV K,ZLOOP
66$: CLR YLOOP ;WAIT
65$: INC YLOOP
CMP YLOOP,#-1
BNE 65$
DEC ZLOOP
BNE 66$
EMT 14
BR 6$
5$: CMP (SP)+,(SP)+ ;INTERRUPT WORKED
MOV @IAR,TEMP ;READ ADDRESS OF INTERRUPTING MODULE
CMPB #377,TEMP ;IAR SHOULD HAVE ALL ONES
BEQ 7$
MOV #377,$GDDAT
CLR $BDDAT ;ASSURE HIGH BYTE IS CLEAR
MOV TEMP,$BDDAT ;PREPARE FOR ERROR TYPEOUT
EMT 6
7$: BISB #RBIT,@CSR
TSTB @CSR ;CLEAR INTERRUPT
BITB #FBIT,@CSR ;CHECK IF INTERRUPT CLEAR
BEQ 6$
EMT 15
6$: JSR PC,KLEER ;CLEAR CSR
MOV (SP)+,ERRVEC ;:POP STACK INTO ERRVEC
MOV #PRO,-(SP) ;SET PSW TO PRIORITY 0
MOV #67$,-(SP)
RTI
67$: NOP
RTS PC
8$: CMP (SP)+,(SP)+
EMT 22
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  
3422  
3423  
3424  
3425  
3426

:\*\*\*\*\*  
: THIS ROUTINE IS USED TO DETERMINE THE VALUE OF K.  
: THIS CONSTANT K WILL BE USED IN THE DELAY MACRO.  
: THE INNER LOOP OF THE DELAY MACRO WILL BE DONE K TIMES.  
: EACH INNER LOOP IN THE DELAY MACRO WILL TAKE ~ 32 MICROSEC.  
:\*\*\*\*\*

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 C14560  TYPE     ,65$      ;;TYPE ASCIZ STRING
      014556 000414      BR       64$      ;;GET OVER THE ASCIZ
      ;;65$: .ASCIZ <CR><LF>/TURN ON LINE CLOCK./<CR><LF>
      64$:      BR       6$
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 003460 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,RO ;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
```

```
3498
3499
3500
3501
3502
3503
3504 015022
      015022 013746 003460
3505 015026 013746 003462
3506 015032 117737 166424 001142 2$:
3507 015040 123737 001140 001142
3508 015046 001404
3509 015050 013737 003462 003466
3510 015056 104007
3511 015060 005237 003462 1$:
3512 015064 005337 003460
3513 015070 001360
3514 015072 012637 003462
3515 015076 012637 003460
3516 015102 000207
3517
3518 015104
      015104 013746 003460
3519 015110 013746 003462
3520 015114 117737 166342 001142 2$:
3521 015122 123737 001140 001142
3522 015130 001404
3523 015132 013737 003462 003466
3524 015140 104011
3525 015142 005237 003462 1$:
3526 015146 005337 003460
3527 015152 001360
3528 015154 012637 003462
3529 015160 012637 003460
3530 015164 000207
```

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

TSTBYT: MOV BYTNUM,-(SP) ;;PUSH BYTNUM ON STACK  
MOV TADDR,-(SP) ;;PUSH TADDR ON STACK  
MOVB @TADDR,\$BDDAT ;MOV DATA TO BDDAT  
CMPB \$GDDAT,\$BDDAT ;IS DATA OK  
BEQ 1\$  
MOV TADDR,TBADDR  
EMT 7  
INC TADDR ;NEXT BYTE  
DEC BYTNUM  
BNE 2\$  
MOV (SP)+,TADDR ;;POP STACK INTO TADDR  
MOV (SP)+,BYTNUM ;;POP STACK INTO BYTNUM  
RTS PC

TSTONE: MOV BYTNUM,-(SP) ;;PUSH BYTNUM ON STACK  
MOV TADDR,-(SP) ;;PUSH TADDR ON STACK  
MOVB @TADDR,\$BDDAT ;MOV DATA TO BDDAT  
CMPB \$GDDAT,\$BDDAT ;IS DATA OK  
BEQ 1\$  
MOV TADDR,TBADDR  
EMT 11  
INC TADDR ;NEXT BYTE  
DEC BYTNUM  
BNE 2\$  
MOV (SP)+,TADDR ;;POP STACK INTO TADDR  
MOV (SP)+,BYTNUM ;;POP STACK INTO BYTNUM  
RTS PC

```
3532                                     ;THIS SUBROUTINE IS USED TO SET AS MANY
3533                                     ;BYTES AS SPECIFIED BY BYTNUM IN MUT
3534
3535 015166                               SETBYT:
      015166 013746 003460               MOV     BYTNUM,-(SP)      ;;PUSH BYTNUM ON STACK
3536 015172 013746 003462               MOV     TADDR,-(SP)     ;;PUSH TADDR ON STACK
3537 015176 113777 001140 166256 1$:   MOVB   $GDDAT,@TADDR    ;;SET DATA IN MUT
3538 015204 005237 003462               INC     TADDR
3539 015210 005337 003460               DEC     BYTNUM
3540 015214 001370                       BNE    1$
3541 015216 012637 003462               MOV     (SP)+,TADDR    ;;POP STACK INTO TADDR
3542 015222 012637 003460               MOV     (SP)+,BYTNUM   ;;POP STACK INTO BYTNUM
3543 015226 000207                       RTS     PC
```

3545  
3546  
3547 015230 005004  
3548 015232 012703 002404  
3549 015236 022301  
3550 015240 001404  
3551 015242 062704 000002  
3552 015246 005713  
3553 015250 001372  
3554 015252 010022  
3555 015254 010125  
3556 015256 000207

:: THIS ROUTINE IS USED IN S-COMMAND  
TABLE: CLR R4  
MOV #GENER,R3  
3\$: CMP (R3)+,R1  
BEQ 1\$  
ADD #2,R4  
TST (R3)  
BNE 3\$  
1\$: MOV R0,(R2)+  
MOV R1,(R5)+  
RTS PC



```
3558 ;THIS SUBROUTINE LOADS THE CSR AND VERIFIES THAT THE VALUE SET
3559 ;GETS LOADED PROPERLY. IF THE TEST CALLING THIS ROUTINE IS
3560 ;TEST 4 OR TEST 5, THE ROUTINE CLEARS THE FLAG BIT BEFORE CHECKING
3561 ;TAKES PLACE
3562
3563 ;PRELOADS:
3564 ; $GDDAT <== VALUE TO BE SET IN CSR
3565 ; $TESTN <== THE TEST NUMBER
3566
3567 ;RETURNS:
3568 ; $BDDAT DATA READ FROM CSR
3569 ; IF ERROR - THE ROUTINE RETURNS
3570 ; IF NO ERROR THE ROUTINE RETURNS+2
3571
3572
3573
3574 015260 113777 001140 165356 BITSET: MOVB $GDDAT,@CSR ;LOAD TESTED BIT
3575 015266 005037 001142 CLR $BDDAT
3576 015272 013737 002636 003446 MOV K,ZLOOP
3577 015300 005037 003444 65$: CLR YLOOP ;WAIT
3578 015304 005237 003444 64$: INC YLOOP
3579 015310 023727 003444 000007 CMP YLOOP,#7
3580 015316 001372 BNE 64$
3581 015320 005337 003446 DEC ZLOOP
3582 015324 001365 BNE 65$
3583 015326 117737 165312 001142 MOVB @CSR,$BDDAT ;READ CSR
3584 015334 023727 001212 000005 CMP $TESTN,#5 ;IS THIS TEST 5
3585 015342 001403 BEQ 3$ ;BRANCH IF YES
3586 015344 023727 001212 000004 CMP $TESTN,#4 ;IS THIS TEST 4
3587 015352 001003 3$: BNE 2$ ;BR IF NO
3588 015354 142737 000200 001142 BICB #FBIT,$BDDAT ;EXECUTE IF TEST 4 OR TEST 5
3589 015362 123737 001140 001142 2$: CMPB $GDDAT,$BDDAT ;COMPARE THEM
3590 015370 001002 BNE 1$
3591 015372 062716 000002 ADD #2,(SP) ;NO ERROR
3592 015376 000207 1$: RTS PC
3593
3594 ;THIS IS INTERRUPT ROUTINE FOR LINE CLOCK
3595 ;WHEN DIAGNOSTIC DOES NOT USE IT
3596
3597 NOCLK: RTI
3598
3599 ;THIS IS INTERRUPT ROUTINE FOR CLOCK TO COUNT TICKS
3600
3601 COUNT: INC CLK
3602 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 #TMGVEC,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
```

```

3640                                     ;THIS SUBROUTINE IS USED IN THE LOOP TEST
3641                                     ;AND IT IS SIMILAR TO THE TSTBYT ROUTINE
3642
3643 015620                                CHKBYT:
3644 015620 013746 003510                MOV     TEMP1,-(SP)      ;;PUSH TEMP1 ON STACK
3645 015624 013746 003460                MOV     BYTNUM,-(SP)   ;;PUSH BYTNUM ON STACK
3646 015630 117737 165654 001142 2$:    MOV     @TEMP1,$BDDAT  ;READ DATA FROM INPUT MODULE
3647 015636 123737 001140 001142        CMP     $GDDAT,$BDDAT
3648 015644 001410                        BEQ     1$
3649 015646 013737 003462 003466        MOV     TADDR,TBADDR
3650 015654 012637 003460                MOV     (SP)+,BYTNUM   ;;POP STACK INTO BYTNUM
3651 015660 012637 003510                MOV     (SP)+,TEMP1   ;;POP STACK INTO TEMP1
3652 015664 000207                        RTS     PC             ;ERROR
3653 015666 005237 003510 1$:          INC     TEMP1          ;GO TO NEXT BYTE
3654 015672 005337 003460                DEC     BYTNUM
3655 015676 001354                        BNE     2$
3656 015700 012637 003460                MOV     (SP)+,BYTNUM   ;;POP STACK INTO BYTNUM
3657 015704 012637 003510                MOV     (SP)+,TEMP1   ;;POP STACK INTO TEMP1
3658 015710 062716 000002                ADD     #2,(R6)       ;BYPASS ERROR IN MAINLINE CODE
3659 015714 000207                        RTS     PC
3660
3661
3662                                     ;THIS SUBROUTINE STARTS CONVERSION
3663                                     ;THIS SUBROUTINE WAITS FOR A <CR> AND
3664                                     ;MONITORS CNTRLC
3665 015716 104406                                CRTST: RDCHR
3666 015720 012637 003566                MOV     (SP)+,ANSW    ;SAVE ANSWER
3667 015724 004737 015526                JSR     PC,CNTRC     ;CONTROL C ?
3668 015730 122737 000015 003566        CMP     #15,ANSW     ;A <CR> ?
3669 015736 001367                        BNE     CRTST        ;WRONG CHARACTER, JUST WAIT
3670 015740 000207                        RTS     PC
3671

```

```

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
3706 016100 104402          TYPOC ;GO TYPE--OCTAL ASCII(ALL DIGITS)
3707 016102 104401 032677          TYPE ,MASS27        ;COLON & TAB
3708 016106 013746 001140          MOV    $GDDAT,-(SP)  ;;SAVE $GDDAT FOR TYPEOUT
3709 016112 104402          TYPOC ;GO TYPE--OCTAL ASCII(ALL DIGITS)
3710 016114 104401 031400          TYPE ,MASS0         ;CR & LF
3711 016120 000404          BR    3$
3712 016122 123760 001140 003452 4$:   CMPB   $GDDAT,DBUFF(R0) ;ANY CHANGE IN DATA ?
3713 016130 001356          BNE   2$          ;YES
3714 016132 005237 003462          3$:   INC    TADDR        ;GO TO NEXT BYTE IN MODULE
3715 016136 005700          TST    R0          ;LAST BYTE IN MODULE ?
3716 016140 001344          BNE   5$          ;NO
3717 016142 163737 003460 003462 SUB    BYTNUM,TADDR    ;RESTORE ADDRESS OF MUT
3718 016150 004737 015526          JSR   PC,CNTRC       ;TYPE CONTROL C TO RETURN TO MONITOR
3719 016154 012701 000001          MOV    #1,R1        ;NEW PASS
3720 016160 000732          BR    6$

```

IOCM TEST

```

3720
3721
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

```

3740 ;THIS SUBROUTINE SETS UP ADDRESSES FOR A/D A014
3741
3742 017100 013737 003462 003614 ADADDR: MOV TADDR,LBYTE
3743 017106 013746 003462 MOV TADDR,-(SP) ;;PUSH TADDR ON STACK
3744 017112 013737 003462 003616 MOV TADDR,HBYTE
3745 017120 005237 003616 INC HBYTE
3746 017124 013737 003616 003610 MOV HBYTE,STAT1
3747 017132 005237 003610 INC STAT1
3748 017136 013737 003610 003612 MOV STAT1,STAT2
3749 017144 005237 003612 INC STAT2
3750 017150 013737 003462 003466 MOV TADDR,TBADDR
3751 017156 013746 000004 MOV ERRVEC,-(SP) ;;PUSH ERRVEC ON STACK
3752 017162 012737 017236 000004 MOV #1$,ERRVEC
3753 017170 105777 164420 TSTB @LBYTE
3754 017174 005237 003462 INC TADDR
3755 017200 105777 164412 TSTB @HBYTE
3756 017204 005237 003462 INC TADDR
3757 017210 105777 164374 TSTB @STAT1
3758 017214 005237 003462 INC TADDR
3759 017220 105777 164366 TSTB @STAT2
3760 017224 2$:
3761 017230 012637 000004 MOV (SP)+,ERRVEC ;;POP STACK INTO ERRVEC
3762 017234 012637 003462 MOV (SP)+,TADDR ;;POP STACK INTO TADDR
3763 017234 000207 RTS PC
3764 017236 022626 1$: CMP (SP)+,(SP)+
3765 017240 104026 EMT 26
3766 017242 012737 007606 000004 MOV #TMOVEC,ERRVEC
3767 017250 005205 INC R5
3768 017252 000764 BR 2$
3769
3770
3771 017254 152777 000004 163362 GCODE: BISB #GBIT,@CSR ;;LOAD GBITE WITH GENERIC CODE
3772 017262 117737 164322 003606 MOVB @STAT1,GBITE
3773 017270 042737 177400 003606 BIC #177400,GBITE
3774 017276 142777 000004 163340 BICB #GBIT,@CSR
3775 017304 000207 RTS PC
3776 ;THIS SUBROUTINE RUNS A TEST AND CHECKS SW13 AND SW14
3777 017306 005037 001214 SWLOOP: CLR $PASS
3778 017312 011637 003506 MOV (SP),XXX ;;GET SUBR ADDR
3779 017316 017737 164164 003506 MOV @XXX,XXX ;;ANOTHER LEVEL OF DEFERRED
3780 017324 004777 164156 1$: JSR PC,@XXX
3781 017330 005237 001214 INC $PASS
3782 017334 023737 001214 002662 CMP $PASS,PASCNT
3783 017342 001370 BNE 1$
3784 017344 032777 020000 161602 BIT #BIT13,@SWR
3785 017352 001007 BNE 2$
3786 017354 104401 001203 TYPE , $CRLF
3787 017360 013746 002662 MOV PASCNT,-(SP) ;;SAVE PASCNT FOR TYPEOUT
3788 017366 104401 030040 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
3789 017372 032777 040000 161554 2$: BIT #BIT14,@SWR
3790 017400 001342 BNE SWLOOP
3791 017402 062716 000002 ADD #2,(R6) ;ADVANCE RETURN PC
3792 017406 000207 RTS PC
3793

```



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
$SCOPE:
:#####START OF CODE FOR THE XOR TESTER#####
$XTSTR: BR      6$          ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                                ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
017410 000416          MOV      @#ERRVEC, -(SP)    ;;SAVE THE CONTENTS OF THE ERROR VECTOR
017412 013746 000004          MOV      #5$, @#ERRVEC    ;;SET FOR TIMEOUT
017416 012737 017436 000004          TST      @#177060        ;;TIME OUT ON XOR?
017424 005737 177060          MOV      (SP)+, @#ERRVEC  ;;RESTORE THE ERROR VECTOR
017430 012637 000004          BR       $$SVLAD        ;;GO TO THE NEXT TEST
017434 000446          5$:    CMP      (SP)+, (SP)+    ;;CLEAR THE STACK AFTER A TIME OUT
017436 022626          MOV      (SP)+, @#ERRVEC  ;;RESTORE THE ERROR VECTOR
017440 012637 000004          BR       7$            ;;LOOP ON THE PRESENT TEST
017444 000434          6$:    ;#####END OF CODE FOR THE XOR TESTER#####
017446 032777 000400 161500          BIT      #BIT08, @SWR    ;;LOOP ON SPEC. TEST?
017454 001421          BEQ      2$            ;;BR IF NO
017456 005046          CLR      -(SP)        ;;CLEAR A TEMP. LOCATION
017460 117716 161470          MOVB    @SWR, (SP)     ;;PICKUP THE DESIRED TEST NUMBER
017464 001414          BEQ      8$            ;;BRANCH IF BAD TEST NUMBER IN SWR
017466 022716 000011          CMP      #11, (SP)    ;;CHECK THE NUMBER IN THE SWR
017472 002411          BLT      8$            ;;BRANCH IF TEST NUMBER IS OUT OF RANGE
017474 011637 001116          MOV      (SP), $TSTNM  ;;UPDATE THE TEST NUMBER
017500 005316          DEC      (SP)        ;;BACKUP BY ONE
017502 006316          ASL      (SP)        ;;SCALE THE TEST NUMBER AS AN INDEX
017504 062716 017622          ADD      #$$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      $$SVLAD      ;;BR IF NO
017526 032777 001000 161420          BIT      #BIT09, @SWR  ;;LOOP ON ERROR?
017534 001404          BEQ      4$            ;;BR IF NO
017536 013737 001124 001122          7$:    MOV      $LPERR, $LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
017544 000420          BR       $OVER
017546 105037 001117          4$:    CLRB    $ERFLG    ;;ZERO THE ERROR FLAG
017552 105237 001116          $$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:
          .REPT   $TN-1
          .WORD  TST1+2    ;;STARTING ADDRESS OF TEST 1
          .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 MULTIPLIER, -(SP)  
\* MOV MULTIPLICAND, -(SP)  
\* JSR PC, @#\$MULT  
\* RETURN ;: PRODUCT IS ON THE STACK

\* STACK PRODUCT

\* -----

\* TOP LSB'S  
\* +2 MSB'S

017644		\$MULT:	MOV	R0, -(SP)	:: PUSH R0 ON STACK
017644	010046		MOV	R1, -(SP)	:: PUSH R1 ON STACK
017646	010146		MOV	R2, -(SP)	:: PUSH R2 ON STACK
017650	010246		CLR	-(SP)	:: CLEAR THE SIGN KEY
017652	005046		MOV	12(SP), R1	:: GET THE MULTIPLICAND
017654	016601	000012	BPL	1\$	:: BR IF PLUS
017660	100002		INC	(SP)	:: SET THE SIGN KEY
017662	005216		NEG	R1	:: MAKE THE MULTIPLICAND POSTIVE
017664	005401		MOV	14(SP), R2	:: GET THE MULTIPLIER
017666	016602	000014	BPL	2\$	:: BR IF PLUS
017672	100002		DEC	(SP)	:: UPDATE THE SIGN KEY
017674	005316		NEG	R2	:: MAKE THE MULTIPLIER POSTIVE
017676	005402		MOV	#17., -(SP)	:: SET THE LOOP COUNT
017700	012746	000021	CLR	R0	:: SETUP FOR THE MULTIPLY LOOP
017704	005000		BCC	4\$	:: DON'T ADD IF MULTIPLICAND = 0
017706	103001		ADD	R2, R0	
017710	060200		ROR	R0	:: POSITION THE PARITIAL PRODUCT AND
017712	006000		ROR	R1	:: THE MULTIPLICAND
017714	006001		DEC	(SP)	:: HAS ALL BITS OF THE MULTIPLICAND BEEN DONE?
017716	005316		BNE	3\$	:: BR IF NO
017720	001372		CMP	(SP)+, (SP)	:: SHOULD PRODUCT BE NEGATIVE?
017722	022616		BEQ	5\$	:: GO TO EXIT IF NO
017724	001403		NEG	R0	:: YES--SO MAKE IT SO
017726	005400		NEG	R1	
017730	005401		SBC	R0	
017732	005600		TST	(SP)+	:: CLEAR SIGN INFO. OFF OF STACK
017734	005726		MOV	R0, 12(SP)	:: PUT THE PRODUCT ON THE STACK (MSB'S)
017736	010066	000012	MOV	R1, 10(SP)	:: LSB'S
017742	010166	000010	MOV	(SP)+, R2	:: POP STACK INTO R2
017746	012602		MOV	(SP)+, R1	:: POP STACK INTO R1
017750	012601		MOV	(SP)+, R0	:: POP STACK INTO R0
017752	012600		RTS	PC	
017754	000207				

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      # $APTBUF,$AAPTTR ;; 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 001603          BNE      3$          ;;SKIP IF NOT
020222 104401  C01202  4$:      TYPE    , $QUES      ;;TYPE A '?'
020226 000752          BR       1$          ;;CLEAR THE BUFFER AND LOOP
020230 111337  020300  3$:      MOV     (R3), 9$      ;;ECHO THE CHARACTER
020234 104401  020300          TYPE    , 9$
020240 122723  000015          CMP     #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          .BLKB  8.          ;;RESERVE 8 BYTES FOR TTY INPUT
020312          136          125          015  $TTYIN: .ASCIZ  /^U/<15><12>  ;;CONTROL 'U'
020315          012          000          .CNTLU: .ASCIZ  /^G/<15><12>  ;;CONTROL 'G'
020317          136          107          015  $CNTLG: .ASCIZ  /^G/<15><12>
020322          012          000          $MSWR: .ASCIZ  <15><12>/SWR = /
020324          015          012          123  $MNEW: .ASCIZ  / NEW = /
020327          127          122          040
020332          075          040          000
020335          040          040          116
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
$RDOCT: MOV     (SP), -(SP)  ;;PROVIDE SPACE FOR THE
MOV     4(SP), 2(SP)      ;;INPUT NUMBER
MOV     R0, -(SP)        ;;PUSH R0 ON STACK
MOV     R1, -(SP)        ;;PUSH R1 ON STACK
MOV     R2, -(SP)        ;;PUSH R2 ON STACK
1$:    RDLIN          ;;READ AN ASCIZ LINE
MOV     (SP)+, R0        ;;GET ADDRESS OF 1ST CHARACTER
MOV     R0, 5$          ;;AND SAVE IT
CLR     R1              ;;CLEAR DATA WORD
CLR     R2
2$:    MOV     (R0)+, -(SP)  ;;PICKUP THIS CHARACTER
BEQ     3$              ;;IF ZERO GET OUT
CMP     #'0, (SP)        ;;MAKE SURE THIS CHARACTER
BGT     4$              ;;IS AN OCTAL DIGIT
CMP     #'7, (SP)
BLT     4$
ASL     R1              ;;*2
ROL     R2

```

```

020346 011646          000004  000002
020350 016666
020356 010046
020360 010146
020362 010246
020364 104407          020474
020366 012600
020370 010037
020374 005001
020376 005002
020400 112046
020402 001420          000060
020404 122716
020410 003026          000067
020412 122716
020416 002423
020420 006301
020422 006102

```

3799

```
020424 006501 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$: TST (SP)+ ;;CLEAN PARTIAL FROM STACK
020470 105010 CLR (R0) ;;SET A TERMINATOR
020472 104401 TYPE ;;TYPE UP THRU THE BAD CHAR.
020474 000000 5$: .WORD 0
020476 104401 001202 TYPE ,SQUES ;;'"?' 'CR' & 'LF'
020502 000730 BR 1$ ;;TRY AGAIN
020504 000000 $HIOCT: .WORD 0 ;;HIGH ORDER BITS GO HERE

.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
$ERROR:
7$: INCB $ERFLG ;;SET THE ERROR FLAG
BEQ 7$ ;;DON'T LET THE FLAG GO TO ZERO
MOV $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
BIT #BIT10,@SWR ;;BELL ON ERROR?
BEQ 1$ ;;NO - SKIP
TYPE ,SBELL ;;RING BELL
1$: INC $ERTTL ;;COUNT THE NUMBER OF ERRORS
MOV (SP),$ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
SUB #2,$ERRPC
MOVB @ $ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
BIT #BIT13,@SWR ;;SKIP TYPEOUT IF SET
BNE 20$ ;;SKIP TYPEOUTS
JSR PC,$ERRTYP ;;GO TO USER ERROR ROUTINE
TYPE ,SCRLF
20$: CMPB #APTENV,$ENV ;;RUNNING IN APT MODE
BNE 2$ ;;NO,SKIP APT ERROR REPORT
BITB #APTPOOL,$ENVM ;;SPOOL ENTIRE ERROR TO HOST?
BNE 10$ ;;YES
MOVB $ITEMB,21$ ;;SET ITEM NUMBER AS ERROR NUMBER
JSR PC,$ATY4 ;;REPORT FATAL ERROR TO APT
21$: .BYTE 0
.BYTE 0
```

```
020636 000777 22$: BR 22$ ;;APT ERROR LOOP
020640 004737 C20716 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 3800 000002 6$: RTI ;;RETURN
.SBTTL ERROR MESSAGE TYPEOUT ROUTINE
;*****
;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($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.
$ERRTYP:
020716 104401 001203 TYPE ,$CRLF ;;"CARRIAGE RETURN" & "LINE FEED"
020722 010046 MOV RO,-(SP) ;;SAVE RO
020724 005000 CLR RO ;;PICKUP THE ITEM INDEX
020726 153700 001130 BISB @#$ITEMB,RO
020732 001004 BNE 1$ ;;IF ITEM NUMBER IS ZERO, JUST
;;TYPE THE PC OF THE ERROR
020734 013746 001132 MOV $ERRPC,-(SP) ;;SAVE $ERRPC FOR TYPEOUT
;;ERROR ADDRESS
020740 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
020742 000426 BR 6$ ;;GET OUT
020744 005300 1$: DEC RO ;;ADJUST THE INDEX SO THAT IT WILL
020746 006300 ASL RO ;; WORK FOR THE ERROR TABLE
020750 006300 ASL RO
020752 006300 ASL RO
020754 062700 004170 ADD #$ERRTB,RO ;;FORM TABLE POINTER
020760 012037 020770 MOV (RO)+,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$: MOV (RO)+,4$ ;;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$: MOV (RO),RO ;;PICKUP "DATA TABLE" POINTER
021016 001004 BNE 7$ ;;GO TYPE THE DATA
021020 012600 6$: MOV (SP)+,RO ;;RESTORE RO
021022 104401 001203 TYPE ,$CRLF ;;"CARRIAGE RETURN" & "LINE FEED"
021026 000207 RTS PC ;;RETURN
021030 7$:
021030 013046 MOV @ (RO)+,-(SP) ;;SAVE @ (RO)+ FOR TYPEOUT
021032 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
021034 005710 TST (RO) ;;IS THERE ANOTHER NUMBER?
```

3801

```
021036 00170          BEQ      6$          ;;BR IF NO
021040 104401 C21046   TYPE     8$          ;;TYPE TWO(2) SPACES
021044 000771          BR       7$          ;;LOOP
021046 040 040 000 8$: .ASCIZ / /          ;;TWO(2) SPACES
                        .EVEN
                        .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$: CMPB   #HT,(SP)          ;;BRANCH IF <HT>
021156 001430          BEQ     8$          ;;BRANCH IF NOT <CRLF>
021160 122716 000200   CMPB   #CRLF,(SP)
021164 001006          BNE     5$          ;;POP <CR><LF> EQUIV
021166 005726          TST    (SP)+          ;;TYPE A CR AND LF
021170 104401          TYPE
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
```

3802

```
021226 004737 021264 JSR PC,$TYPEC ;;GO TYPE A NULL
021232 105337 C21330 DECB $CHARCNT ;;DO NOT COUNT AS A COUNT
021236 000770 BR 7$ ;;LOOP
;HORIZONTAL TAB PROCESSOR
8$: MOVB #' ,(SP) ;;REPLACE TAB WITH SPACE
9$: JSR PC,$TYPEC ;;TYPE A SPACE
BITB #7,$CHARCNT ;;BRANCH IF NOT AT
BNE 9$ ;;TAB STOP
TST (SP)+ ;;POP SPACE OFF STACK
BR 2$ ;;GET NEXT CHARACTER
$TYPEC: TSTB @$TPS ;;WAIT UNTIL PRINTER IS READY
BPL $TYPEC
MOV 2(SP),@$TPB ;;LOAD CHAR TO BE TYPED INTO DATA REG.
CMPB #CR,2(SP) ;;IS CHARACTER A CARRIAGE RETURN?
BNE 1$ ;;BRANCH IF NO
CLRB $CHARCNT ;;YES--CLEAR CHARACTER COUNT
BR $TYPEX ;;EXIT
1$: CMPB #LF,2(SP) ;;IS CHARACTER A LINE FEED?
BEQ $TYPEX ;;BRANCH IF YES
INCB (PC)+ ;;COUNT THE CHARACTER
$CHARCNT: .WORD 0 ;;CHARACTER COUNT STORAGE
$TYPEX: RTS PC
.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
$TYPDS:
MOV R0,-(SP) ;;PUSH R0 ON STACK
MOV R1,-(SP) ;;PUSH R1 ON STACK
MOV R2,-(SP) ;;PUSH R2 ON STACK
MOV R3,-(SP) ;;PUSH R3 ON STACK
MOV R5,-(SP) ;;PUSH R5 ON STACK
MOV #20200,-(SP) ;;SET BLANK SWITCH AND SIGN
MOV 20(SP),R5 ;;GET THE INPUT NUMBER
BPL 1$ ;;BR IF INPUT IS POS.
NEG R5 ;;MAKE THE BINARY NUMBER POS.
MOVB #'-,1(SP) ;;MAKE THE ASCII NUMBER NEG.
1$: CLR R0 ;;ZERO THE CONSTANTS INDEX
MOV #DBLK,R3 ;;SETUP THE OUTPUT POINTER
MOVB #' ,(R3)+ ;;SET THE FIRST CHARACTER TO A BLANK
2$: CLR R2 ;;CLEAR THE BCD NUMBER
MOV $DTBL(R0),R1 ;;GET THE CONSTANT
3$: SUB R1,R5 ;;FORM THIS BCD DIGIT
BLT 4$ ;;BR IF DONE
INC R2 ;;INCREASE THE BCD DIGIT BY 1
BR 3$
4$: ADD R1,R5 ;;ADD BACK THE CONSTANT
TST R2 ;;CHECK IF BCD DIGIT=0
BNE 5$ ;;FALL THROUGH IF 0
TSTB (SP) ;;STILL DOING LEADING 0'S?
BMI 7$ ;;BR IF YES
```



```
021432 106516          5$: ASLB (SP)          ;;MSD?
021434 103003          BCC 6$           ;;BR IF NO
021436 116663 000001 177777 MOV 1(SP),-1(R3)  ;;YES--SET THE SIGN
021444 052702 000060 7$: BIS #'0,R2     ;;MAKE THE BCD DIGIT ASCII
021450 052702 000040 7$: BIS #' ,R2     ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
021454 110223          MOV  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  2$         ;;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 MOV -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 ,SDBLK  ;;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          $SDBLK: .BLKW 4
3804          .SBTTL APT COMMUNICATIONS ROUTINE
*****
021560 112737 000001 022024 $ATY1: MOV  #1,$FFLG  ;;TO REPORT FATAL ERROR
021566 112737 000001 022022 $ATY3: MOV  #1,$MFLG  ;;TO TYPE A MESSAGE
021574 000403          BR   $ATYC
021576 112737 000001 022024 $ATY4: MOV  #1,$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 #APTPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
021634 001425          BEQ  3$         ;;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$         ;;SUB START OF MESSAGE
021666 163700 001222          SUB  $MSGAD,R0   ;;GET MESSAGE LNTH IN WORDS
021672 006200          ASR  R0         ;;PUT LENGTH IN MAILBOX
021674 010037 001224          MOV  R0,$MSGGLT ;;TELL APT TO TAKE MSG.
021700 012737 000004 001206 MOV  #4,$MSGTYPE
021706 000413          BR   5$
```

```
021710 017037 000004 021734 3$: MOV @4(SP),4$ ;;PUT MSG ADDR IN JSR LINKAGE
021716 062766 C00002 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  
000001  
000100  
000040

3805

APTSIZE=200  
APTENV=001  
APTSPOOL=100  
APTCSUP=040

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

\*\*\*\*\*  
\*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT  
\*OCTAL (ASCII) NUMBER AND TYPE IT.  
\*\$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE

\*CALL:  
\* MOV NUM,-(SP) ;;NUMBER TO BE TYPED  
\* TYPOS ;;CALL FOR TYPEOUT  
\* .BYTE N ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE  
\* .BYTE M ;;M=1 OR 0  
\* ;;1=TYPE LEADING ZEROS  
\* ;;0=SUPPRESS LEADING ZEROS

\*\$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST  
\*\$TYPOS OR \$TYPOC

\*CALL:  
\* MOV NUM,-(SP) ;;NUMBER TO BE TYPED  
\* TYPON ;;CALL FOR TYPEOUT

\*\$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER

\*CALL:  
\* MOV NUM,-(SP) ;;NUMBER TO BE TYPED  
\* TYPOC ;;CALL FOR TYPEOUT

```
022026 017646 000000 022267 $TYPOS: MOV @4(SP),-(SP) ;;PICKUP THE MODE
022032 116637 000001 MOV 1(SP),$OFILL ;;LOAD ZERO FILL SWITCH
022040 112637 022271 MOV (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
$SBTTL SAVE AND RESTORE R0-R5 ROUTINES
*****
;*SAVE R0-R5
;*CALL:
;* SAVREG
;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:

```

3806

```

: *
: *TOP---(+16)
: * +2---(+18)
: * +4---R5
: * +6---R4
: * +8---R3
: * +10---R2
: * +12---R1
: * +14---R0
$SAVREG:
022272          MOV      R0,-(SP)          ;;PUSH R0 ON STACK
022272 010046   MOV      R1,-(SP)          ;;PUSH R1 ON STACK
022274 010146   MOV      R2,-(SP)          ;;PUSH R2 ON STACK
022276 010246   MOV      R3,-(SP)          ;;PUSH R3 ON STACK
022300 010346   MOV      R4,-(SP)          ;;PUSH R4 ON STACK
022302 010446   MOV      R5,-(SP)          ;;PUSH R5 ON STACK
022304 010546   MOV      22(SP),-(SP)        ;;SAVE PS OF MAIN FLOW
022306 016646 000022  MOV      22(SP),-(SP)        ;;SAVE PC OF MAIN FLOW
022312 016646 000022  MOV      22(SP),-(SP)        ;;SAVE PS OF CALL
022316 016646 000022  MOV      22(SP),-(SP)        ;;SAVE PC OF CALL
022322 016646 000022  RTI
022326 000002

: *RESTORE R0-R5
: *CALL:
: * RESREG
$RESREG:
022330          MOV      (SP)+,22(SP)        ;;RESTORE PC OF CALL
022330 012666 000022  MOV      (SP)+,22(SP)        ;;RESTORE PS OF CALL
022334 012666 000022  MOV      (SP)+,22(SP)        ;;RESTORE PC OF MAIN FLOW
022340 012666 000022  MOV      (SP)+,22(SP)        ;;RESTORE PS OF MAIN FLOW
022344 012666 000022  MOV      (SP)+,R5           ;;POP STACK INTO R5
022350 012605   MOV      (SP)+,R4           ;;POP STACK INTO R4
022352 012604   MOV      (SP)+,R3           ;;POP STACK INTO R3
022354 012603   MOV      (SP)+,R2           ;;POP STACK INTO R2
022356 012602   MOV      (SP)+,R1           ;;POP STACK INTO R1
022360 012601   MOV      (SP)+,R0           ;;POP STACK INTO R0
022362 012600   RTI
022364 000002

.SBTTL TRAP DECODER
: *****
: *THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE 'TRAP' INSTRUCTION
: *AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
: *OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
: *GO TO THAT ROUTINE.
$TRAP: MOV      R0,-(SP)          ;;SAVE R0
      MOV      2(SP),R0          ;;GET TRAP ADDRESS
      TST      -(R0)            ;;BACKUP BY 2
      MOV      (R0),R0          ;;GET RIGHT BYTE OF TRAP
      ASL      R0              ;;POSITION FOR INDEXING
      MOV      $TRPAD(R0),R0    ;;INDEX TO TABLE
      RTS      R0              ;;GO TO ROUTINE
: THIS IS USE TO HANDLE THE 'GETPRI' MACRO
$TRAP2: MOV      (SP),-(SP)      ;;MOVE THE PC DOWN
      MOV      4(SP),2(SP)      ;;MOVE THE PSW DOWN
      RTI                      ;;RESTORE THE PSW
.SBTTL TRAP TABLE
: *THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
: *BY THE 'TRAP' INSTRUCTION.
: ROUTINE

```

3807

3808

```

022422 022410          $TRPAD:  -----
022424 021052          $TRAP2  .WORD  $TRAP2
022426 022052          $TYPE   ;;CALL=TYPE   TRAP+1(104401) TTY TYPEOUT ROUTINE
022430 022026          $TYPOC  ;;CALL=TYPOC  TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
022432 022066          $TYPOS  ;;CALL=TYPOS  TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
022434 021334          $TYPON  ;;CALL=TYPON  TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
022436 017756          $TYPDS  ;;CALL=TYPDS  TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
022440 020152          $RDCHR  ;;CALL=RDCHR  TRAP+6(104406) TTY TYPEIN CHARACTER ROUTINE
022442 020346          $RDLIN  ;;CALL=RDLIN  TRAP+7(104407) TTY TYPEIN STRING ROUTINE
022444 022272          $RDOCT  ;;CALL=RDOCT  TRAP+10(104410) READ AN OCTAL NUMBER FROM TTY
022446 022330          $SAVREG ;;CALL=SAVREG TRAP+11(104411) SAVE R0-R5 ROUTINE
                                $RESREG ;;CALL=RESREG TRAP+12(104412) RESTORE R0-R5 ROUTINE

.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  $POWER      ;;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: .ASCIZ /IOCM TEST/<15><12>
3814 022670 103 123 122 EM2Y: .ASCIZ /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: .ASCIZ <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>
```

.MAIN. MACRO M1113 12-AUG-82 09:59 PAGE 39-1  
POWER DOWN AND UP ROUTINES

3868	026207	124	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 - DIFFFERENTIAL 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 TEST 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: .ASCIZ <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  
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,
```



1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16

.SBTTL DIGITAL MODULE TEST, M5010

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

M5010:

\*\*\*\*\*

TST12: SCOPE  
MOV #12,\$TESTN ;:SET TEST NUMBER IN APT MAIL BOX  
MOVB #0,\$GDDAT ;:SET WHAT DATA SHOULD BE  
MOV #4,BYTNUM ;:SET NUMBER OF BYTES  
JSR PC,TSTBYT ;:CHECK IF DATA IS CORRECT  
  
BISB #TBIT,@CSR  
MOVB #377,\$GDDAT ;:SET WHAT DATA SHOULD BE  
MOV #4,BYTNUM ;:SET NUMBER OF BYTES  
JSR PC,TSTBYT ;:CHECK IF DATA IS COPRECT  
  
BICB #TBIT,@CSR  
JSR PC,CLRINT  
RTS PC

040752  
040752 000004  
040754 012737 000012 001212  
040762 112737 000000 001140  
040770 012737 000004 003460  
040776 004737 015022  
  
041002 152777 000010 141634  
041010 112737 000377 001140  
041016 012737 000004 003460  
041024 004737 015022  
041030 142777 000010 141606  
041036 004737 015410  
041042 000207

.SBTTL DIGITAL MODULE TEST ,M5013

;THIS TEST CHECKS M5013 MODULE  
;8 BIT AC SENSE

18  
19  
20  
21  
22  
23  
24  
25  
26 041044  
27  
041044 000004  
041046 012737 000013 001212  
28 041054 152777 000020 141562  
29 041062 112737 000000 001140  
041070 012737 000001 003460  
041076 004737 015022  
30  
31 041102 152777 000010 141534  
32 041110 004737 015410  
33 041114 112737 000377 001140  
041122 012737 000001 003460  
041130 004737 015022  
34 041134 142777 000010 141502  
35 041142 004737 015410  
36  
37  
38 041146 012746 000000  
041152 012746 041160  
041156 000002  
041160 000240  
39 041162 012777 041316 141460  
40 041170 012777 000340 141454  
41 041176 152777 000010 141440  
42 041204 152777 000100 141432  
43 041212 013737 002636 003446  
041220 005037 003444  
041224 005237 003444  
041230 023727 003444 177777  
041236 001372  
041240 005337 003446  
041244 001365  
44 041246 132737 000001 001226  
45 041254 001403  
46 041256 005737 001214  
47 041262 001001  
48 041264  
041264 104005  
49 041266 142777 000100 141350  
50 041274 142777 000010 141342  
51 041302 004737 015410  
52 041306 012737 007606 000004  
53 041314 000207  
54  
55 041316 022626  
56 041320 005037 003510  
57 041324 113737 002661 003503  
58 041332 012737 041640 000004

M5013:

\*\*\*\*\*  
TST13: SCOPE  
MOV #13,\$TESTN ;;SET TEST NUMBER IN APT MAIL BOX  
BISB #DBIT,@CSR ;SET DISABLE BIT  
MOVB #0,\$GDDAT ;SET WHAT DATA SHOULD BE  
MOV #1,BYTNUM ;SET NUMBER OF BYTES  
JSR PC,TSTBYT ;CHECK IF DATA IS CORRECT  
  
BISB #TBIT,@CSR ;SET COMPLIM BIT  
JSR PC,CLRINT  
MOVB #377,\$GDDAT ;SET WHAT DATA SHOULD BE  
MOV #1,BYTNUM ;SET NUMBER OF BYTES  
JSR PC,TSTBYT ;CHECK IF DATA IS CORRECT  
BICB #TBIT,@CSR  
JSR PC,CLRINT  
  
MOV #PRO,-(SP) ;SET PSW TO PRIORITY 0  
MOV #64\$,-(SP)  
RTI  
64\$: NOP  
MOV #5@,\$@VECTO ;SET INTERRUPT VECTOR ADDRESS  
MOV #PR7,\$@VECTOA  
BISB #TBIT,@CSR ;START INTERRUPT  
BISB #EBIT,@CSR ;ENABLE INTERRUPT  
MOV K,ZLOOP  
66\$: CLR YLOOP ;WAIT  
65\$: INC YLOOP  
CMP YLOOP,#-1  
BNE 65\$  
DEC ZLOOP  
BNE 66\$  
BITB #BIT0,\$ENV ;UNDER APT ?  
BEQ 50\$ ;NO  
TST \$PASS ;FIRST PASS (\$PASS = 0) ?  
BNE 7\$ ;NO  
50\$: EMT 5  
7\$: BICB #EBIT,@CSR ;CLEAR CSR  
BICB #TBIT,@CSR  
JSR PC,CLRINT ;CLEAR ALL INTERRUPT  
MOV #TMOVEC,ERRVEC ;RETURN TO CALLING PROGRAM D  
RTS PC  
5\$: CMP (SP)+,(SP)+ ;RESET STACK  
CLR TEMP1  
MOVB BASE+1,TEMP+1 ;INITIALIZE TEMP  
MOV #4\$,ERRVEC

```
59 041340      6$:      MOV      K,ZLOOP
    041340 013737 C02636 003446      CLR      YLOOP      ;WAIT
    041346 005037 003444      68$:      INC      YLOOP
    041352 005237 003444      67$:      CMP      YLOOP,#1
    041356 023727 003444 000001      BNE     67$
    041364 001372      DEC      ZLOOP
    041366 005337 003446      BNE     68$
    041372 001365      MOVB     @IAR,TEMP      ;CHECK WHICH I/O INTERRUPTED
60 041374 117737 141246 003502      BISB     #RBIT,@CSR      ;HERE IF INTERRUPT
61 041402 152777 000001 141234      CMPB     TEMP,TADDR      ;IS IT MUT
62 041410 123737 003502 003462      BEQ     2$
63 041416 001414      TSTB     @TEMP
64 041420 105777 142056      INC      TEMP1      ;LOOP NO MORE THAN 400 TIMES
65 041424 005237 003510      CMP      #400,TEMP1
66 041430 022737 000400 003510      BNE     6$
67 041436 001340      TST      $PASS      ;FIRST PASS ($PASS = 0) ?
68 041440 005737 001214      BNE     2$      ;NO
69 041444 001001      EMT      5
70 041446 104005      2$:      TSTB     @TADDR      ;CLEAR INTERRUPT
71 041450 105777 142006      MOV      K,ZLOOP
72 041454 013737 002636 003446      70$:      CLR      YLOOP      ;WAIT
    041462 005037 003444      69$:      INC      YLOOP
    041466 005237 003444      CMP      YLOOP,#1
    041472 023727 003444 000001      BNE     69$
    041500 001372      DEC      ZLOOP
    041502 005337 003446      BNE     70$
    041506 001365      BITB     #FBIT,@CSR      ;CHECK IF INTERRUPT CLEAR
73 041510 132777 000200 141126      BEQ     8$
74 041516 001413      MOVB     @IAR,TEMP
75 041520 117737 141122 003502      CMP      TEMP,TADDR
76 041526 023737 003502 003462      BNE     9$
77 041534 001002      EMT      15
78 041536 104015      BR       7$
79 041540 000652      9$:      JSR      PC,CLRINT
80 041542 004737 015410      8$:      BITB     #TBIT,@CSR
81 041546 132777 000010 141070      BEQ     7$
82 041554 001644      MOV      #PRO,-(SP)      ;SET PSW TO PRIORITY 0
83 041556 012746 000000      MOV      #71$,-(SP)
    041562 012746 041570      RTI
    041566 000002      71$:      NOP
    041570 000240      BICB     #TBIT,@CSR      ;CHECK IF CLEARING TBIT CAUSE INTERRUPT
84 041572 142777 000010 141044      MOV      K,ZLOOP
85 041600 013737 002636 003446      73$:      CLR      YLOOP      ;WAIT
    041606 005037 003444      72$:      INC      YLOOP
    041612 005237 003444      CMP      YLOOP,#-1
    041616 023727 003444 177777      BNE     72$
    041624 001372      DEC      ZLOOP
    041626 005337 003446      BNE     73$
    041632 001365      EMT      5
86 041634 104005      BR       7$
87 041636 000613      4$:      CMP      (SP)+,(SP)+
88 041640 022626      EMT      5
89 041642 104005      JMP      2$
90 041644 000137 041450
```

.SBTTL DIGITAL MODULE TEST, M6010, M6010YA

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

```
92
93
94
95
96
97
98
99 041650
100 041650 000004
    041652 012737 000014 001212
101 041660 152777 000020 140756
102 041666 013700 003462
103 041672 105020
104 041674 105020
105 041676 105020
106 041700 105020
107 041702 005000
    041704 005001
    041706 005037 003452
    041712 005037 003454
    041716 116160 002630 003452
    041724 013737 003462 003502
    041732 060037 003502
    041736 116177 002630 141536
    041744 005002
    041746 013737 003462 003510
    041754 060237 003510
    041760 117737 141524 001142
    041766 123762 001142 003452
    041774 001404
    041776 116237 003452 001140
    042004 104017
    042006 005202
    042010 022702 000004
    042014 001354
    042016 005201
    042020 022701 000004
    042024 001334
    042026 005200
    042030 022700 000004
    042034 001323
108 042036 000207

M6010:
*****
TST14: SCOPE
MOV #14,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
BISB #DBIT,@CSR ;SET D BIT
MOV TADDR,R0
CLRB (R0)+ ;CLEAR ALL FOUR BYTES OF I/O REGISTER
CLRB (R0)+
CLRB (R0)+
CLRB (R0)+
CLR R0
4$: CLR R1
CLR DBUFF ;CLEAR IMAGE OF I/O REGISRERS
CLR DBUFF+2
3$: MOVB PATT(R1),DBUFF(R0);SET DATA PATTERN IN IMIGE
MOV TADDR,TEMP
ADD R0,TEMP
MOVB PATT(R1),@TEMP ;SET DATA IN I/O REG
CLR R2
2$: MOV TADDR,TEMP1
ADD R2,TEMP1
MOVB @TEMP1,$BDDAT ;READ I/O REGISTER
CMPB $BDDAT,DBUFF(R2);IS DATA OK?
BEQ 1$
MOVB DBUFF(R2),$GDDAT
EMT 17
1$: INC R2
CMP #4,R2
BNE 2$ ;TEST IF OTHER BYTES ARE OK
INC R1
CMP #4,R1 ;LAST PATTERN?
BNE 3$ ;TEST NEXT DATA PATTERN
INC R0
CMP #4,R0 ;LAST BYTE?
BNE 4$ ;NO,TEST NEXT BYTE
RTS PC
```

```
110 .SBTTL DIGITAL MODULE TEST, M6012, M6013
111
112
113 ;THIS TEST CHECKS MODULES M6012 AND M6013
114 ;8 BIT AC AND DC OUT
115
116
117
118 042040 M6012:
119 042040 M6013:
120 :*****
120 042040 000004 TST15: SCOPE
121 042042 012737 000015 001212 MOV #15,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
122 042050 152777 000020 140566 BISB #DBIT,@CSR
122 042056 012737 000125 001140 MOV #125,$GDDAT ;SET DATA PATTERN
122 042064 012737 000001 003460 MOV #1,BYTNUM ;SET NUMBER OF BYTES
122 042072 004737 015166 JSR PC,SETBYT ;SET DATA IN I/O MODULE
122 042076 012737 000001 003460 MOV #1,BYTNUM
122 042104 004737 015022 JSR PC,TSTBYT ;TEST IF DATA IS SET CORRECTLY
123
124
125 042110 012737 000252 001140 MOV #252,$GDDAT ;SET DATA PATTERN
125 042116 012737 000001 003460 MOV #1,BYTNUM ;SET NUMBER OF BYTES
125 042124 004737 015166 JSR PC,SETBYT ;SET DATA IN I/O MODULE
125 042130 012737 000001 003460 MOV #1,BYTNUM
125 042136 004737 015022 JSR PC,TSTBYT ;TEST IF DATA IS SET CORRECTLY
126
127 042142 012737 000377 001140 MOV #377,$GDDAT ;SET DATA PATTERN
127 042150 012737 000001 003460 MOV #1,BYTNUM ;SET NUMBER OF BYTES
127 042156 004737 015166 JSR PC,SETBYT ;SET DATA IN I/O MODULE
127 042162 012737 000001 003460 MOV #1,BYTNUM
127 042170 004737 015022 JSR PC,TSTBYT ;TEST IF DATA IS SET CORRECTLY
128
129
130 042174 012737 000000 001140 MOV #0,$GDDAT ;SET DATA PATTERN
130 042202 012737 000001 003460 MOV #1,BYTNUM ;SET NUMBER OF BYTES
130 042210 004737 015166 JSR PC,SETBYT ;SET DATA IN I/O MODULE
130 042214 012737 000001 003460 MOV #1,BYTNUM
130 042222 004737 015022 JSR PC,TSTBYT ;TEST IF DATA IS SET CORRECTLY
131 042226 000207 RTS PC
```



```

133 .SBTTL DIGITAL MODULE TEST, M6011
134
135 ;THIS TEST CHECKS MODULE M6011
136 ;ONE SHOT DC OUT
137
138
139 042230 M6011:
    ;*****
    TST16: SCOPE
    042230 000004 MOV #16,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
    042232 012737 000016 001212 MOV TADDR,TADDR1
    140 042240 013737 003462 003464 MOV TADDR1
    141 042246 005237 003464 INC TADDR1
    142 042252 152777 000020 140364 BISB #DBIT,@CSR
    143 042260 012777 015402 140400 MOV #COUNT,@CLKVC ;SET LOC 100 - CLOCK VECTOR
    144 042266 012777 000340 140374 MOV #PR7,@CLKVCA ;SET LOC 102
    145 042274 012746 000000 MOV #PRO,-(SP) ;SET PSW TO PRIORITY 0
    042300 012746 042306 MOV #64$,-(SP)
    042304 000002 RTI
    042306 000240 64$: NOP
    146 042310 012737 000252 042550 MOV #252,TSTDAT ;FIRST DATA PATTERN
    147
    148 042316 005037 003476 4$: CLR CLK
    149 042322 012737 000377 001140 MOV #377,$GDDAT ;SET DATA PATTERN
    150 042330 012737 000002 003460 MOV #2,BYTNUM ;SET NUMBER OF BYTES
    151 042336 004737 015166 JSR PC,SETBYT ;SET DATA IN I/O MODULE
    152 042342 004737 015022 JSR PC,TSTBYT ;TEST IF DATA IS SET CORRECTLY
    153 042346 005001 CLR R1 ;TEST IF LINE CLOCK IS INTERRUPTING
    154 042350 005737 003476 1$: TST CLK
    155 042354 001004 BNE 2$
    156 042356 005201 INC R1
    157 042360 001373 BNE 1$
    158 042362 104025 EMT 25
    159 042364 000465 BR M6011X ;EXIT TEST
    160
    161 042366 105777 141070 2$: TSTB @TADDR ;TEST OUTPUT TO LOCATE ASYNCHRONOUS CLOCK
    162 042372 001003 BNE 3$
    163 042374 105777 141064 TSTB @TADDR1
    164 042400 001411 BEQ 5$ ;WHEN OUTPUT CLEARS START TEST
    165 042402 023727 003476 001224 3$: CMP CLK,#660. ;WAIT 11 SEC MAX
    166 042410 002766 BLT 2$
    167 042412 005037 001140 CLR $GDDAT ;PATTERN SHOULD BE CLEAR
    168 042416 004737 015104 JSR PC,TSTONE
    169 042422 000446 BR M6011X ;EXIT TEST
    170
    171 042424 005037 003476 5$: CLR CLK ;CLEAR CLOCK COUNTER
    172 042430 013737 042550 001140 MOV TSTDAT,$GDDAT ;SET DATA PATTERN
    173 042436 004737 015166 JSR PC,SETBYT ;SET DATA IN I/O MODULE
    174 042442 004737 015022 JSR PC,TSTBYT ;TEST IF DATA IS SET CORRECTLY
    175 042446 105777 141010 6$: TSTB @TADDR ;TEST OUTPUT
    176 042452 001003 BNE 7$
    177 042454 105777 141004 TSTB @TADDR1
    178 042460 001404 BEQ 10$
    179 042462 022737 001224 003476 7$: CMP #660.,CLK ;WAIT 11 SEC MAX
    180 042470 002766 BLT 6$
    181 042472 013737 003476 042552 10$: MOV CLK,CLKSTP ;GET TIME
    182 042500 023727 042552 001142 CMP CLKSTP,#610. ;LESS THAN 10 SEC?
    183 042506 003404 BLE 11$ ;YES, GOOD
    
```

```

184 042510 005037 001140          CLR      $GDDAT          ;NO, SHOULD BE CLEAR
185 042514 004737 C15104          12$:   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

```

198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208

.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

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 COMPLI'N 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	#PRO,-(SP)	;SET PSW TO PRIORITY 0
042664	012746	042672		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
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
229 043006	142777	000010	137630	BICB	#TBIT,@CSR	;CLEAR 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)+
235 043032	005037	003510		CLR	TEMP1	;ADJUST STACK POINTER
236 043036	113737	002661	003503	MOVB	BASE+1,TEMP+1	;INITIALIZE TEMP
237 043044	012737	043530	000004	MOV	#10\$,ERRVEC	

238	043052				7\$:	MOV	K,ZLOOP	
	043052	013737	C02636	003446		CLR	YLOOP	;WAIT
	043060	005037	003444		68\$:	INC	YLOOP	
	043064	005237	003444		67\$:	CMP	YLOOP,#1	
	043070	023727	003444	000001		BNE	67\$	
	043076	001372				DEC	ZLOOP	
	043100	005337	003446			BNE	68\$	
	043104	001365				MOVB	@IAR,TEMP	;FIND WHICH I/O INTERRUPTED
239	043106	117737	137534	003502		CMPB	TEMP,TADDR	;IS IT MUT
240	043114	123737	003502	003462		BEQ	8\$	;YES
241	043122	001416				BISB	#RBIT,@CSR	;CLEAR INTERRUPT
242	043124	152777	000001	137512		TSTB	@TEMP	
243	043132	105777	140344			INC	TEMP1	
244	043136	005237	003510			CMP	#400,TEMP1	
245	043142	022737	000400	003510		BNE	7\$	
246	043150	001340				EMT	5	
247	043152	104005				JMP	2\$	
248	043154	000137	043000			BICB	#EBIT,@CSR	
249	043160	142777	000100	137456	8\$:	MOVB	#377,\$GDDAT	
250	043166	112737	000377	001140		MOVB	@COSADR,\$BDDAT	;TEST COS REGISTER
251	043174	117737	140270	001142		CMPB	\$GDDAT,\$BDDAT	;IS IT ALL ONES
252	043202	123737	001140	001142		BEQ	3\$	
253	043210	001401				EMT	20	
254	043212	104020				BISB	#RBIT,@CSR	
255	043214	152777	000001	137422	3\$:	TSTB	@COSADR	;CLEAR INTERRUPT
256	043222	105777	140242			CLR	\$GDDAT	
257	043226	005037	001140			MOVB	@COSADR,\$BDDAT	
258	043232	117737	140232	001142		BEQ	4\$	
259	043240	001401				EMT	13	
260	043242	104013				INC	COSADR	
261	043244	005237	003470		4\$:	INC	TADDR	;TEST NEXT BYTE
262	043250	005237	003462			MOV	K,ZLOOP	
263	043254	013737	002636	003446		CLR	YLOOP	;WAIT
	043262	005037	003444		70\$:	INC	YLOOP	
	043266	005237	003444		69\$:	CMP	YLOOP,#1	
	043272	023727	003444	000001		BNE	69\$	
	043300	001372				DEC	ZLOOP	
	043302	005337	003446			BNE	70\$	
	043306	001365				MOVB	@IAR,TEMP	
264	043310	117737	137332	003502		CMPB	TEMP,TADDR	;CHECK IF IT INTERRUPTED
265	043316	123737	003502	003462		BEQ	11\$	
266	043324	001403				EMT	5	
267	043326	104005				JMP	2\$	
268	043330	000137	043000			MOVB	#377,\$GDDAT	
269	043334	112737	000377	001140	11\$:	MOVB	@COSADR,\$BDDAT	;CHECK IF COS REGISTER IS ALL ONES
270	043342	117737	140122	001142		CMPB	\$GDDAT,\$BDDAT	
271	043350	123737	001140	001142		BEQ	5\$	
272	043356	001401				EMT	20	
273	043360	104020				BISB	#RBIT,@CSR	
274	043362	152777	000001	137254	5\$:	TSTB	@COSADR	;CLEAR INTERRUPT
275	043370	105777	140074			CLR	\$GDDAT	
276	043374	005037	001140			MOVB	@COSADR,\$BDDAT	
277	043400	117737	140064	001142		BEQ	6\$	
278	043406	001401				EMT	13	
279	043410	104013				DEC	COSADR	
280	043412	005337	003470		6\$:	DEC	TADDR	;RESET ADDRESS
281	043416	005337	003462					

282	043422	004,37	015410		JSR	PC,CLRINT		;CLEAR REMAINING INTERRUPTS
283	043426	132777	C00010	137210	BITB	#TBIT,@CSR		;CHECK IF CLEARING T BIT CAUSE INTERRUPT
284	043434	001433			BEQ	9\$		
285	043436	012746	000000		MOV	#PRO,-(SP)		;SET PSW TO PRIORITY 0
	043442	012746	043450		MOV	#71\$,-(SP)		
	043446	000002			RTI			
	043450	000240			NOP			
286	043452	142777	000010	137164	71\$: 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

296  
297  
298  
299  
300  
301  
302  
303 043540  
304  
043540 000004  
043542 012737 000020 001212  
305 043550 152777 000020 137066  
306 043556 112737 000000 001140  
043564 012737 000002 003460  
043572 004737 015022  
307  
308 043576 152777 000010 137040  
309 043604 112737 000377 001140  
043612 012737 000002 003460  
043620 004737 015022  
310 043624 142777 000010 137012  
311 043632 004737 015410  
312 043636 012746 000000  
043642 012746 043650  
043646 000002  
043650 000240  
313 043652 012777 044006 136770  
314 043660 012777 000340 136764  
315 043666 152777 000100 136750  
316 043674 152777 000010 136742  
317 043702 013737 002636 003446  
043710 005037 003444  
043714 005237 003444  
043720 023727 003444 177777  
043726 001372  
043730 005337 003446  
043734 001365  
318 043736 132737 000001 001226  
319 043744 001403  
320 043746 005737 001214  
321 043752 001001  
322 043754  
043754 104005  
323 043756 142777 000100 136660  
324 043764 142777 000010 136652  
325 043772 012737 007606 000004  
326 044000 004737 015410  
327 044004 000207  
328  
329 044006  
330  
331 044006 022626  
332 044010 005037 003510  
333 044014 113737 002661 003503  
334 044022 012737 044342 000004  
335 044030  
044030 013737 002636 003446

M5012:

:\*\*\*\*\*

TST20: SCOPE  
MOV #20,\$TESTN ;;SET TEST NUMBER IN APT MAIL BOX  
BISB #DBIT,@CSR  
MOVB #0,\$GDDAT ;SET WHAT DATA SHOULD BE  
MOV #2,BYTNUM ;SET NUMBER OF BYTES  
JSR PC,TSTBYT ;CHECK IF DATA IS CORRECT  
BISB #TBIT,@CSR  
MOVB #377,\$GDDAT ;SET WHAT DATA SHOULD BE  
MOV #2,BYTNUM ;SET NUMBER OF BYTES  
JSR PC,TSTBYT ;CHECK IF DATA IS CORRECT  
BICB #TBIT,@CSR  
JSR PC,CLRINT  
MOV #PR0,-(SP) ;SET PSW TO PRIORITY 0  
MOV #64\$,-(SP)  
RTI  
64\$: NOP  
MOV #6\$,@VECTO ;SET INTERRUPT VECTOR  
MOV #PR7,@VECTOA  
BISB #EBIT,@CSR ;ENABLE INTERRUPT  
BISB #TBIT,@CSR ;START INTERRUPT  
MOV K,ZLOOP  
66\$: CLR YLOOP ;WAIT  
65\$: INC YLOOP  
CMP YLOOP,#-1  
BNE 65\$  
DEC ZLOOP  
BNE 66\$  
BITB #BIT0,\$ENV ;UNDER APT ?  
BEQ 50\$ ;NO  
TST \$PASS ;FIRST PASS (\$PASS = 0) ?  
BNE 7\$ ;NO  
50\$: EMT 5  
7\$: BICB #EBIT,@CSR ;CLEAR CSR  
BICB #TBIT,@CSR  
MOV #TMOVEC,ERRVEC  
JSR PC,CLRINT ;CLEAR ALL INTERRUPT  
RTS PC ;RETURN TO MONITOR  
6\$: ; INTERRUPT ROUTINE  
CMP (SP)+,(SP)+ ;RESET STACK POINTER  
CLR TEMP1  
MOVB BASE+1,TEMP+1 ;INITIALIZE TEMP  
MOV #4\$,ERRVEC  
3\$: 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 #PRO,-(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
```

```
373 .SBTTL TESTING DIGITAL MODULE M5014
374 ;M5014 DUAL 16-BIT INPUT COUNTER
375
376 044346 M5014:
          044346 000004
          044350 012737 000021 001212 TST21: SCOPE
          ;*****
          MOV #21,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
377
378 044356 104411 SAVREG
379 044360 012746 000340 MOV #PR7,-(SP) ;:SET PSW TO PRIORITY 7
          044364 012746 044372 MOV #64$,-(SP)
          044370 000002 RTI
          044372 000240 64$: NOP
380 ;ADDRESS INITIALIZATION
381 044374 013737 003462 003626 MOV TADDR,ACNTL ;:INITIALIZE COUNTER A LOW-BYTE
382 044402 013737 003462 003624 MOV TADDR,ACNTH ;:INITIALIZE COUNTER A HIGH-BYTE
383 044410 062737 000001 003624 ADD #1,ACNTH ;:MAKE COUNTER A ADDRESS
384 044416 013737 003462 003632 MOV TADDR,BCNTL ;:INITIALIZE COUNTER B LOW-BYTE
385 044424 062737 000002 003632 ADD #2,BCNTL ;:MAKE COUNTER B ADDRESS
386 044432 013737 003462 003630 MOV TADDR,BCNTH ;:INITIALIZE COUNTER B HIGH-BYTE
387 044440 062737 000003 003630 ADD #3,BCNTH ;:MAKE COUNTER B ADDRESS
388
389 ;:START WITH A KNOWN CONDITION -- CLEAR THE MODULE
390
391 044446 004737 014310 JSR PC,KLEER ;:CLEAR MODULE.
392 044452 152777 000020 136164 BISB #DBIT,@CSR ;:SHUT OFF WORLD
393
394 ;:STATUS OF M5014 AT THIS TIME SHOULD BE --
395
396 ; COUNTER A = 0 COUNTER B = 0
397
398 ;:CHECK THAT COUNTERS ARE CLEAR
399
400 044460 005037 001140 CLR $GDDAT ;:INITIALIZE EXPECTED DATA.
401 044464 013737 003626 003462 MOV ACNTL,TADDR ;:TEST COUNTER A
402 044472 013737 003626 003636 MOV ACNTL,CTRSL
403 044500 013737 003624 003634 MOV ACNTH,CTRSH
404
405 044506 117737 137122 001143 1$: MOVB @CTRSH,$BDDAT+1 ;:READ HIGH BYTE OF COUNTER.
406 044514 117737 137116 001142 MOVB @CTRSL,$BDDAT ;:READ LOW BYTE OF COUNTER.
407 044522 023737 001140 001142 CMP $GDDAT,$BDDAT ;:IF DATA IS AS EXPECTED
408 044530 001401 BEQ 2$ ;:-THEN SKIP ERROR
409
410 044532 104075 EMT 75
411
412 044534 023737 003632 003462 2$: CMP BCNTL,TADDR ;:IF DONE CHECKING CLEAR
413 044542 001412 BEQ TSTPT2 ;:-THEN DO PATTERN TEST
414 044544 013737 003632 003462 MOV BCNTL,TADDR ;:-ELSE TEST COUNTER B
415 044552 013737 003632 003636 MOV BCNTL,CTRSL
416 044560 013737 003630 003634 MOV BCNTH,CTRSH
417 044566 000747 BR 1$
```



```
419                                     ;CHECK THE READ/WRITE CAPABILITIES OF THE COUNTERS
420
421 044570 005000          TSTPT2: CLR    R0          ;INITIALIZE PATTERN POINTER.
422 044572 142777 000010 136044 BICB   #TBIT,@CSR   ;ASSURE THAT TBIT IS OFF
423
424 044600 013737 003626 003462 1$:  MOV   ACNTL,TADDR   ;TEST COUNTER A
425 044606 013737 003626 003636      MOV   ACNTL,CTRSL
426 044614 013737 003624 003634      MOV   ACNTH,CTRSH
427 044622 116037 002630 001140      MOVB  PATT(R0),$GDDAT ;GET EXPECTED PATTERN
428 044630 116037 002630 001141      MOVB  PATT(R0),$GDDAT+1
429
430 044636 152777 000010 136000 2$:  BISB   #TBIT,@CSR   ;INITIALIZE BOARD
431 044644 113777 001141 136762      MOVB  $GDDAT+1,@CTRSH ;WRITE INTO HIGH BYTE OF COUNTER
432 044652 113777 001140 136756      MOVB  $GDDAT,@CTRSL  ;WRITE PATTERN INTO LOW BYTE
433 044660 117737 136750 001143      MOVB  @CTRSH,$BDDAT+1 ;READ COUNTERS HIGH BYTE
434 044666 117737 136744 001142      MOVB  @CTRSL,$BDDAT  ;READ LOW BYTE OF COUNTER
435 044674 023737 001140 001142      CMP   $GDDAT,$BDDAT ;IF PATTERN IS AS EXPECTED
436 044702 001401          BEQ   3$          ;--THEN SKIP ERROR
437
438 044704 104076          EMT   76
439
440 044706 142777 000010 135730 3$:  BICB   #TBIT,@CSR   ;STOP COUNTING
441 044714 023737 003632 003462      CMP   BCNTL,TADDR   ;IF COUNTER B JUST TESTED
442 044722 001414          BEQ   4$          ;--THEN DO NEXT PATTERN
443
444 044724 005137 001140          COM   $GDDAT          ;--ELSE PRODUCE EXPECTED PATTERN
445 044730 013737 003632 003462      MOV   BCNTL,TADDR   ; TEST COUNTER B
446 044736 013737 003632 003636      MOV   BCNTL,CTRSL
447 044744 013737 003630 003634      MOV   BCNTH,CTRSH
448 044752 000731          BR    2$
449
450 044754 005200          4$:  INC   R0          ;MOVE PATTERN POINTER
451 044756 122760 000123 002630      CMPB  #123,PATT(R0) ;IF NOT AT END OF PATTERN TABLE
452 044764 001305          BNE  1$          ;--THEN DO NEW PATTERN
```

```
454                                     ;CHECK THAT TBIT INITIALIZES MODULE
455
456 044766 005037 001140 TINIMD: CLR $GDDAT ;INITIALIZE EXPECTED DATA.
457 044772 112777 177777 136624 MOVB #-1,@ACNTH ;LOAD THE HIGH BYTE BUFFER.
458 045000 112777 177777 136620 MOVB #-1,@ACNTL ;LOAD COUNTER A WITH ALL 1'S.
459 045006 112777 177777 136616 MOVB #-1,@BCNTL ;LOAD COUNTER B WITH ALL 1'S.
460 045014 112777 177777 136602 MOVB #-1,@ACNTH ;LOAD THE HIGH BYTE
461 045022 112777 177777 136600 MOVB #-1,@BCNTH ;LOAD COUNTER B HIGH BYTE
462 045030 112777 000020 135606 MOVB #DBIT,@CSR ;ASSURE CSR IS CLEAR.
463
464 045036 142777 000010 135600 BICB #TBIT,@CSR ;INITIALIZE BOARD.
465 045044 152777 000010 135572 BISB #TBIT,@CSR
466 045052 013737 003626 003462 MOV ACNTL,TADDR ;INITIALIZE FIRST TEST ADDRESS
467 045060 013737 003626 003636 MOV ACNTL,CTRSL
468 045066 013737 003624 003634 MOV ACNTH,CTRSH
469
470 045074 117737 136534 001143 1$: MOVB @CTRSH,$BDDAT+1 ;READ HIGH BYTE OF TEST COUNTER
471 045102 117737 136530 001142 MOVB @CTRSL,$BDDAT ;READ LOW BYTE OF TEST COUNTER
472 045110 023737 001140 001142 CMP $GDDAT,$BDDAT ;IF DATA IS AS EXPECTED
473 045116 001401 BEQ 2$ ;-THEN SKIP ERROR
474
475 045120 104077 EMT 77
476
477 045122 023737 003632 003462 2$: CMP BCNTL,TADDR ;IF DONE CHECKING INIT
478 045130 001412 BEQ INTHL2 ;-THEN CHECK INTERRUPTS
479 045132 013737 003632 003462 MOV BCNTL,TADDR ;-ELSE TEST COUNTER B.
480 045140 013737 003632 003636 MOV BCNTL,CTRSL
481 045146 013737 003630 003634 MOV BCNTH,CTRSH
482 045154 000747 BR 1$
483
484 045156 112777 000020 135460 INTHL2: MOVB #DBIT,@CSR ;START INTERNAL FREQ SOURCE
485 045164 152777 000010 135452 BISB #TBIT,@CSR
486 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$
487 045226 004737 015410 JSR PC,CLRINT ;CLEAR ANY INTERRUPTS CAUSED BY TBIT.
488
489 045232 013737 003626 003462 1$: MOV ACNTL,TADDR ;INITIALIZE FIRST COUNTER TO TEST
490 045240 013737 003626 003636 MOV ACNTL,CTRSL
491 045246 013737 003624 003634 MOV ACNTH,CTRSH
492 045254 012777 045456 135366 MOV #2$,@VECTO ;INITIALIZE INTERRUPT VECTOR
493 045262 012777 000340 135362 MOV #PR7,@VECTOA ;INITIALIZE INTERRUPT PRIORITY
494 045270 012737 177776 003642 MOV #-2,NUM ;INITIALIZE A COUNT
495
```

```

497                                     ;ALLOW AN INTERRUPT AND WAIT
498
499 045276                               8$:
045276 012746 000000                   MOV    #PRO,-(SP)      ;SET PSW TO PRIORITY 0
045302 012746 045310                   MOV    #66$,-(SP)
045306 000002                           RTI
045310 000240                           66$:  NOP
500 045312 152777 000100 135324       BISB   #EBIT,@CSR     ;ALLOW THE INTERRUPT
501 045320 113777 003643 136306       MOVB   NUM+1,@CTRSB  ;CLEAR HIGH BYTE BUFFER
502 045326 113777 003642 136302       MOVB   NUM,@CTRSL   ;LOAD COUNTER
503 045334 013737 002636 003446       MOV    K,ZLOOP
045342 005037 003444                   68$:  CLR    YLOOP        ;WAIT
045346 005237 003444                   67$:  INC    YLOOP
045352 023727 003444 177777         CMP    YLOOP,#-1
045360 001372                           BNE   67$
045362 005337 003444                   DEC    ZLOOP
045366 001365                           BNE   68$
504 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 003444                   DEC    ZLOOP
045422 001365                           BNE   70$
505
506 045424 142777 000100 135212       BICB   #EBIT,@CSR   ;SHUT OFF INTERRUPTS
507 045432 012746 000340               MOV    #PR7,-(SP)   ;SET PSW TO PRIORITY 7
045436 012746 045444                   MOV    #71$,-(SP)
045442 000002                           RTI
045444 000240                           71$:  NOP
508
509 045446 104103                       EMT    103
510
511 045450 000532                       BR     7$           ;SKIP INTERRUPT ROUTINE

```

```

513                                     ;TIMEOUT HANDLER FOR INTERRUPT CHECK
514
515 045452 022626                       33$:  CMP      (SP)+,(SP)+      ;RESET STACK POINTER
516 045454 000452                       BR       38$              ;CONTINUE INTERRUPT CHECKS
517
518                                     ;ON INTERRUPT DO 2$
519
520 045456 022626                       2$:  CMP      (SP)+,(SP)+      ;RESET STACK POINTER
521 045460 013746 000004                 MOV      ERRVEC,-(SP)      ;SAVE TIMEOUT VECTOR
522 045464 013746 000006                 MOV      ERRVEC+2,-(SP)   ;SAVE TIMEOUT PRIORITY
523 045470 013737 045452 000004         MOV      33$,ERRVEC       ;ON TIME OUT GO TO 33$
524 045476 012737 000340 000006         MOV      #PR7,ERRVEC+2   ;ASSURE PSW REMAINS PRIORITY 7
525 045504 005037 003510                 CLR      TEMP1           ;PREPARE TO CONSTRUCT INTERRUPT ADDRESS
526 045510 013737 002660 003502         MOV      BASE,TEMP       ;INITIALIZE TEMP
527 045516 152777 000001 135120         3$:  BISB    #RBIT,@CSR    ;PREPARE TO CLEAR INTERRUPT
528
529                                     ;CHECK IF COUNTER INTERRUPTS
530
531 045524 117737 135116 003502         MOVB    @IAR,TEMP        ;GET LOW BYTE OF INTERRUPT ADDRESS
532
533 045532 023737 003626 003462         CMP     ACNTL,TADDR     ;IF COUNTER A IS BEING TESTED
534 045540 001406                       BEQ     35$             ;--THEN SEE IF COUNTER B INTERRUPTS
535
536 045542 023737 003626 003502         CMP     ACNTL,TEMP      ;--ELSE IF A DIDN'T INTERRUPT
537 045550 001006                       BNE    36$             ;--THEN SKIP ERROR
538
539 045552                               34$:
540 045552 104110                       EMT     110
541 045554 000410                       BR      37$            ;---CLEAR UNWANTED INTERRUPT
542
542 045556 023737 003632 003502         35$:  CMP     BCNTL,TEMP   ;IF COUNTER B INTERRUPTED
543 045564 001772                       BEQ     34$            ;--THEN IT WAS UNEXPECTED
544
545 045566 023737 003502 003462         36$:  CMP     TEMP,TADDR   ;IF COUNTER INTERRUPTED
546 045574 001411                       BEQ     4$             ;--THEN CLEAN UP.
547
548                                     ;CLEAR UNWANTED INTERRUPT
549
550 045576 105777 135700                 37$:  TSTB   @TEMP       ;CLEAR UNWANTED INTERRUPT
551 045602 005237 003510                 38$:  INC     TEMP1      ;INCREMENT INTERRUPT+TIMEOUT COUNT
552 045606 022737 000400 003510         CMP     #400,TEMP1     ;IF NOT EXCESSIVE COUNT NUMBER
553 045614 001340                       BNE    3$             ;--THEN CHECK AGAIN
554
555 045616 104103                       EMT     103
556
557 045620 012637 000004                 4$:  MOV     (SP)+,ERRVEC  ;RESTORE TIMEOUT VECTOR
558 045624 012637 000006                 MOV     (SP)+,ERRVEC+2  ;RESTORE TIMEOUT PRIORITY
559 045630 004737 015410                 JSR    PC,CLRINT       ;CLEAR ALL INTERRUPTS
560 045634 132777 000200 135002         BITB   #FBIT,@CSR     ;IF INTERRUPT CLEARS
561 045642 001401                       BEQ     5$             ;THEN SKIP ERROR.
562
563 045644 104074                       EMT     74
564
565 045646 012737 177777 001140         5$:  MOV     #-1,$GDDAT   ;INITIALIZE EXPECTED DATA
566 045654 013737 002636 003446         MOV     K,ZLOOP
567 045662 005037 003444                 73$:  CLR     YLOOP      ;WAIT
568 045666 005237 003444                 72$:  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$:    CMP      BCNTL,TADDR      ;IF FINISHED CHECKING INTERRUPTS
577 045744 001413                      BEQ      TRAMP2         ;-THEN TEST TRANSITION POINTS
578 045746 013737 003632 003462      MOV      BCNTL,TADDR    ;-ELSE CHECK COUNTER B
579 045754 013737 003632 003636      MOV      BCNTL,CTRSL
580 045762 013737 003630 003634      MOV      BCNTH,CTRSH
581 045770 000137 045276                      JMP      8$
```

```

583                                     ;TEST THE TRANSITION POINTS
584
585 045774 112777 000020 134642 TRANP2: MOVB   #DBIT,@CSR      ;CLEAR EXTRANEIOUS BITS AT CSR.
586 046002 152777 000010 134634      BISB   #TBIT,@CSR
587 046010 013737 003626 003462      MOV    ACNTL,TADDR      ;INITIALIZE COUNTER TEST ADDRESS
588 046016 013701 003624              MOV    ACNTH,R1         ;INITIALIZE HIGH BYTE ADDRESS
589 046022 013700 003626              MOV    ACNTL,R0        ;INITIALIZE LOW BYTE ADDRESS
590
591 046026 012737 100000 003642 1$:   MOV    #BIT15,NUM      ;INITIALIZE TRANSITION NUMBER
592
593 046034 000257              5$:   CCC                ;CLEAR ALL CONDITION CODES
594 046036 006037 003642          ROR    NUM              ;DETERMINE NEXT TRANSITION VALUE
595 046042 103014              BCC    2$              ;IF NOT FINISHED THIS COUNTER DO 2$.
596
597 046044 023737 003462 003632 6$:   CMP    TADDR,BCNTL     ;IF COUNTER B WAS CHECKED
598 046052 001453              BEQ    STSTI2          ;--THEN CHECK FREQUENCY
599 046054 013737 003632 003462      MOV    BCNTL,TADDR     ;--ELSE PREPARE TO TEST COUNTER B
600 046062 013701 003630              MOV    BCNTH,R1        ;--LOAD HIGH BYTE ADDRESS
601 046066 013700 003632              MOV    BCNTL,R0        ;--LOAD LOW BYTE ADDRESS
602 046072 000755              BR     1$              ;
603
604 046074 005005              2$:   CLR    R5              ;INITIALIZE LOOP COUNT
605 046076 013737 003642 001140      MOV    NUM,$GDDAT      ;COMPUTE EXPECTED DATA
606 046104 013737 003642 003502      MOV    NUM,TEMP        ;COMPUTE DATA TO LOAD
607 046112 005337 003502          DEC    TEMP
608
609 046116 113711 003503          MOVB   TEMP+1,(R1)     ;LOAD HIGH BYTE OF COUNTER
610 046122 113710 003502          MOVB   TEMP,(R0)      ;LOAD COUNTER
611 046126 000240              NOP
612 046130 000240              NOP
613 046132 000240              NOP
614
615 046134 111137 001143              3$:   MOVB   (R1),$BDDAT+1 ;READ HIGH BYTE
616 046140 111037 001142          MOVB   (R0),$BDDAT    ;READ LOW BYTE
617 046144 000240              NOP
618 046146 023737 003502 001142      CMP    TEMP,$BDDAT    ;IF COUNT CHANGED
619 046154 001004              BNE    4$              ;--THEN CHECK DATA
620 046156 005305              DEC    R5              ;--ELSE IF DELAY NOT UP
621 046160 001365              BNE    3$              ;--THEN LOOK FOR A COUNT
622
623 046162 104101              EMT    101
624
625 046164 000723              BR     5$
626
627 046166 023737 001140 001142 4$:   CMP    $GDDAT,$BDDAT ;IF COUNT WAS GOOD
628 046174 001717              BEQ    5$              ;--THEN SKIP ERROR
629
630 046176 104102              EMT    102
631
632 046200 000715              BR     5$              ;--THEN DO ANOTHER TRANSITION CHECK

```

```
634                                     ;TEST THAT COUNTERS COUNT TO WITHIN 20% OF 5KHZ
635
636 046202 004737 014310                STST12: JSR    PC,KLEER          ;CLEAR EVERYTHING
637 046206 112777 000020 134430        MOVB   #DBIT,@CSR          ;SHUT OFF WORLD
638 046214 152777 000010 134422        BISB   #TBIT,@CSR          ;ENABLE 5KHZ CLOCKING
639 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$
640 046256 004737 015410                JSR    PC,CLRINT          ;CLEAR ALL INTERRUPTS
641 046262 012777 015402 134376        MOV    #COUNT,@CLKVC     ;INITIALIZE CLOCK VECTOR
642 046270 012777 000340 134372        MOV    #PR7,@CLKVCA       ;SET UP PRIORITY FOR CLK INT
643 046276 012777 046712 134344        MOV    #4$,@VECTO         ;INITIALIZE INTERRUPT VECTOR
644 046304 012777 000340 134340        MOV    #PR7,@VECTOA       ;SET UP PRIORITY FOR INT
645
646 046312 012737 001750 003644        MOV    #1000.,VARI        ;INITIALIZE ALLOWED FRROR
647 046320 012737 165204 001140        MOV    #-5500.,$GDDAT     ;INITIALIZE EXPECTED DATA
648 046326 163737 003644 001140        SUB    VARI,$GDDAT        ;SUBTRACT FUDGE FACTOR
649 046334 005037 003642                CLR    NUM                ;INITIALIZE PROGRAM POSITION POINTER
650
651                                     ;ALLOW THE INTERRUPTS
652
653 046340 152777 000100 134276        BISB   #EBIT,@CSR          ;ENABLE INTERRUPT BIT AT CSR
654 046346 012746 000000                MOV    #PRO,-(SP)         ;SET PSW TO PRIORITY 0
        046352 012746 046360                MOV    #66$,-(SP)
        046356 000002                        RTI
        046360 000240                66$:   NOP
655
656 046362 113777 001141 135234        MOVB   $GDDAT+1,@ACNTH    ;LOAD HIGH BYTE BUFFER
657 046370 113777 001140 135230        MOVB   $GDDAT,@ACNTL     ;LOAD COUNTER A
658 046376 113777 001140 135226        MOVB   $GDDAT,@BCNTL     ;LOAD COUNTER B
659
660                                     ;TEST THAT THE CLOCK IS ON
661
662 046404 005037 003476                CLR    CLK                ;INITIALIZE CLOCK
663 046410 005037 003444                CLR    YLOOP              ;INITIALIZE LOOP COUNT
664 046414 005737 003476                11$:   TST    CLK                ;IF CLOCK IS COUNTING
665 046420 001005                        BNE    15$                ;-THEN CHECK THAT COUNTERS COUNT
666 046422 005237 003444                INC    YLOOP              ;-ELSE IF CLOCK MAY COUNT
667 046426 001372                        BNE    11$                ;--THEN CHECK FOR COUNT
668
669 046430 104025                        EMT    25
670 046432 000514                        BR     14$                ;--SKIP CLOCK DEPENDENT TESTS
671
672 046434 012737 000001 003642        15$:   MOV    #1,NUM           ;SET A CLOCK FLAG
673
674                                     ;WAIT UP TO ONE SECOND FOR A COUNT
675
676 046442 105777 135156                12$:   TSTB   @ACNTH           ;LOAD LOW BYTE BUFFER
677 046446 127737 135154 001140        CMPB   @ACNTL,$GDDAT      ;IF COUNTER A IS COUNTING
678 046454 001015                        BNE    13$                ;-THEN CHECK FREQUENCY
679 046456 105777 135146                TSTB   @BCNTH           ;-ELSE CHECK COUNTER B
680 046462 127737 135144 001140        CMPB   @BCNTL,$GDDAT     ;--IF COUNTER B IS COUNTING
681 046470 001007                        BNE    13$                ;---THEN CHECK FREQUENCY
```

```

682 046472 023727 003476 000074      CMP      CLK,#60.      ;---ELSE IF 1 SECOND NOT UP
683 046500 002760                      BLT      12$          ;----THEN WAIT FOR COUNT
684
685
686 046502 012737 000002 003642      MOV      #2,NUM       ;INDICATE PROGRAM MADE IT HERE
687
688 046510 005037 003476          13$:    CLR      CLK          ;INITIALIZE THE CLOCK
689
690 046514 023727 003476 000074      1$:    CMP      CLK,#60.  ;IF 1 SECOND NOT UP
691 046522 002774                      BLT      1$          ;--THEN WAIT
692
693 046524 142777 000110 134112      BICB    #EBIT!TBIT,@CSR ;DON'T LEAVE THESE BITS HANGING
694 046532 012746 000340          MOV      #PR7,-(SP)   ;SET PSW TO PRIORITY 7
      046536 012746 046544          MOV      #67$,-(SP)
      046542 000002          RTI
      046544 000240          67$:    NOP
695
696 046546 117737 135052 001143      MOVB    @ACNTH,$BDDAT+1 ;GET CONTENTS OF COUNTER A
697 046554 117737 135046 001142      MOVB    @ACNTL,$BDDAT
698 046562 117737 135042 003651      MOVB    @BCNTH,TEMPB+1 ;GET CONTENTS OF COUNTER B
699 046570 117737 135036 003650      MOVB    @BCNTL,TEMPB
700
701 046576 013737 003644 001140      MOV      VARI,$GDDAT  ;GET EXPECTED COUNT
702 046604 006337 003644          ASL     VARI          ;DOUBLE THE VARIANCE
703 046610 005437 003644          NEG     VARI          ;GET TWO'S COMPLIMENT
704
705 046614 023737 003644 001142      CMP      VARI,$BDDAT  ;IF COUNT IS WITHIN 20% OF 5KHZ
706 046622 100404                      BMI     2$          ;--THEN SKIP ERROR
707
708 046624 013737 003626 003462      MOV      ACNTL,TADDR  ;-ELSE PREPARE FOR ERROR
709 046632 104106                      EMT     106
710
711 046634 023737 003644 003650      2$:    CMP      VARI,TEMPB ;IF COUNT IS WITHIN 20% OF 5KHZ
712 046642 100407                      BMI     3$          ;--THEN SKIP ERROR
713
714 046644 013737 003632 003462      MOV      BCNTL,TADDR  ;-ELSE PREPARE FOR ERROR
715 046652 013737 003650 001142      MOV      TEMPB,$BDDAT ;--DITTO
716 046660 104106                      EMT     106
717
718 046662 000520          3$:    BR      TSECTB    ;CHECK TIME BASE
719
720 046664 142777 000110 133752      14$:    BICB    #EBIT!TBIT,@CSR ;DON'T LEAVE BITS HANGING
721 046672 012746 000340          MOV      #PR7,-(SP)   ;SET PSW TO PRIORITY 7
      046676 012746 046704          MOV      #68$,-(SP)
      046702 000002          RTI
      046704 000240          68$:    NOP
722 046706 000137 047546          JMP     CBITCM       ;SKIP CLOCK DEPENDENT TESTS

```



```
724 ;ON INTERRUPT DO 4$
725
726 046712 022626 4$: CMP (SP)+,(SP)+ ;RESTORE STACK POINTER
727 046714 013737 003476 003510 MOV CLK,TEMP1 ;GET TIME AT INTERRUPT
728 046722 013737 002660 003502 MOV BASE,TEMP ;INITIALIZE TEMP
729 046730 117737 133712 003502 MOVB @IAR,TEMP ;GET INTERRUPTING ADDRESS
730
731 046736 023737 003626 003502 CMP ACNTL,TEMP ;IF NOT COUNTER A INTERRUPTING
732 046744 001013 BNE 5$ ;-THEN SKIP ERROR
733
734 046746 117737 134652 001143 MOVB @ACNTH,$BDDAT+1
735 046754 117737 134646 001142 MOVB @ACNTL,$BDDAT
736 046762 013737 003626 003462 MOV ACNTL,TADDR ;-ELSE PREPARE FOR ERROR TYPEOUT
737 046770 104106 EMT 106
738 046772 000445 BR 7$ ;CLEAN UP
739
740 046774 023737 003632 003502 5$: CMP BCNTL,TEMP ;IF NOT COUNTER B INTERRUPTING
741 047002 001013 BNE 6$ ;-THEN SKIP ERROR
742
743 047004 117737 134620 001143 MOVB @BCNTH,$BDDAT+1
744 047012 117737 134614 001142 MOVB @BCNTL,$BDDAT
745 047020 013737 003632 003462 MOV BCNTL,TADDR ;-ELSE PREPARE FOR ERROR TYPEOUT
746 047026 104106 EMT 106
747
748 047030 000426 BR 7$ ;CLEAN UP
749
750 047032 152777 000001 133604 6$: BISB #RBIT,@CSR ;CLEAR UNWANTED INTERRUPT
751 047040 105777 134436 TSTB @TEMP
752 047044 012746 000000 MOV #PRO,-(SP) ;SET PSW TO PRIORITY 0
047050 012746 047056 MOV #69$,-(SP)
047054 000002 RTI
047056 000240
753 047060 005737 003642 69$: NOP
754 047064 001002 TST NUM ;IF NOT AT 11$ MARK
755 047066 000137 046414 BNE 16$ ;-THEN GO TO CORRECT MARK
756 047072 022737 000001 003642 16$: JMP 11$ ;-ELSE CONTINUE AT 11$ MARK
757 047100 001205 CMP #1,NUM ;IF AT THE 1$ MARK
758 047102 000137 046442 BNE 1$ ;-THEN GO TO IT
759 JMP 12$ ;-ELSE GO TO THE 12$ MARK
760 047106 004737 015410 7$: JSR PC,CLRINT ;CLEAR ALL INTERRUPTS
761 047112 132777 000200 133524 BITB #FBIT,@CSR ;IF INTERRUPT CLEARS
762 047120 001401 BEQ TSECTB ;-THEN SKIP ERROR
763 047122 104104 EMT 104
```

```

765                                     ;CHECK THAT A 10 SECOND TIME BASE IS CHOSEN
766
767 047124                                TSECTB:
    047124 012746 000340                MOV    #PR7,-(SP)      ;SET PSW TO PRIORITY 7
    047130 012746 047136                MOV    #64$,-(SP)
    047134 000002                        RTI
    047136 000240                        NOP
768 047140 012737 141520 001140        64$:  MOV    #50000,$GDDAT ;INITIALIZE EXPECTED DATA
769 047146 142777 000110 133470        BICB  #TBIT!EBIT,@CSR ;ASSURE TBIT AND EBIT ARE CLEAR
770 047154 012777 047470 133466        MOV    #4$,@VECTO    ;INITIALIZE INTERRUPT VECTOR
771 047162 012777 000340 133462        MOV    #PR7,@VECTOA ;INITIALIZE INTERRUPT PSW STATUS
772 047170 152777 000010 133446        BISB  #TBIT,@CSR    ;INITIALIZE MODULE
773 047176 004737 015410                JSR   PC,CLRINT     ;CLEAR ALL INTERRUPTS
774 047202 105077 134416                CLRB  @ACNTH        ;CLEAR HIGH BYTE BUFFER
775 047206 105077 134414                CLRB  @ACNTL        ;CLEAR COUNTER A
776 047212 105077 134414                CLRB  @BCNTL        ;CLEAR COUNTER B
777 047216 005037 003476                CLR   CLK           ;INITIALIZE CLOCK COUNT
778
779 047222 152777 000100 133414        BISB  #EBIT,@CSR    ;ALLOW INTERRUPTS
780 047230 012746 000000                MOV    #PRO,-(SP)   ;SET PSW TO PRIORITY 0
    047234 012746 047242                MOV    #65$,-(SP)
    047240 000002                        RTI
    047242 000240                        NOP
781 047244 013737 002636 003446        65$:  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$
782 047300 005737 003476                TST   CLK           ;IF CLOCK IS COUNTING
783 047304 001014                        BNE   1$            ;-THEN WAIT 11 SECS
784 047306 112777 000020 133330        MOVB  #DBIT,@CSR    ;-ELSE CLEAN UP THE CSR
785 047314 012746 000340                MOV    #PR7,-(SP)   ;SET PSW TO PRIORITY 7
    047320 012746 047326                MOV    #68$,-(SP)
    047324 000002                        RTI
    047326 000240                        68$:  NOP
786
787 047330 104025                        EMT   25
788 047332 000137 047546                JMP   CBITCM        ; AND CHECK THAT CBIT CLEARS MUT
789
790 047336 023727 003476 001332        1$:  CMP   CLK,#730.  ;IF A LITTLE MORE THAN 12 SECONDS NOT UP
791 047344 002774                        BLT   1$            ;-THEN WAIT
792
793 047346 012746 000340                MOV    #PR7,-(SP)   ;SET PSW TO PRIORITY 7
    047352 012746 047360                MOV    #69$,-(SP)
    047356 000002                        RTI
    047360 000240                        69$:  NOP
794 047362 112777 000020 133254        MOVB  #DBIT,@CSR    ;CLEAR EXTRANEIOUS BITS
795 047370 104103                        EMT   103
796
797 047372 013737 003624 003634        5$:  MOV    ACNTH,CTRSH ;INITIALIZE FIRST TEST ADDRESS
798 047400 013737 003626 003636        MOV    ACNTL,CTRSL
799
800 047406 013737 003636 003462        2$:  MOV    CTRSL,TADDR ;LOAD TEST ADDRESS
801 047414 117737 134214 001143        MOVB  @CTRSH,$BDDAT+1 ;READ HIGH BYTE OF TEST COUNTER
802 047422 117737 134210 001142        MOVB  @CTRSL,$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      #PRO,-(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.
```

```
857 .SBTTL TESTING DIGITAL MODULE M6014
858 ;M6014 DUAL 16-BIT OUTPUT COUNTER
859
860 047724 M6014:
      047724 000004
      047726 012737 000022 001212 TST2: SCOPE
861 ;*****
862 047734 012746 000340 MOV #22,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
      047740 012746 047746 MOV #PR7,-(SP) ;SET PSW TO PRIORITY 7
      047744 000002 MOV #64$,-(SP)
      047746 000240 RTI
863 64$: NOP
      047750 013737 003462 003626 ;ADDRESS INITIALIZATION
      047756 013737 003462 003624 MOV TADDR,ACNTL ;INITIALIZE COUNTER A LOW-BYTE
      047764 062737 000001 003624 MOV TADDR,ACNTH ;INITIALIZE COUNTER A HIGH-BYTE
      047772 013737 003462 003632 ADD #1,ACNTH ;MAKE COUNTER A ADDRESS
      050000 062737 000002 003632 MOV TADDR,BCNTL ;INITIALIZE COUNTER B LOW-BYTE
      050006 013737 003462 003630 ADD #2,BCNTL ;MAKE COUNTER B ADDRESS
      050014 062737 000003 003630 MOV TADDR,BCNTH ;INITIALIZE COUNTER B HIGH-BYTE
      ADD #3,BCNTH ;MAKE COUNTER B ADDRESS
871 ;START WITH A KNOWN CONDITION -- CLEAR THE MODULE
872
873 JSR PC,KLEER ;CLEAR MODULE.
874 050022 004737 014310 BISB #DBIT,@CSR ;SHUT OFF WORLD
875 050026 152777 000020 132610
876
877 ;STATUS OF M6014 AT THIS TIME SHOULD BE --
878
879 ; COUNTER A = 0 COUNTER B = 0
880
881 ;CHECK THAT COUNTERS ARE CLEAR
882
883 050034 005037 001140 CLR $GDDAT ;INITIALIZE EXPECTED DATA.
884 050040 013737 003626 003636 MOV ACNTL,CTRSL ;TEST COUNTER A
885 050046 013737 003624 003634 MOV ACNTH,CTRSH
886
887 050054 013737 003636 003462 1$: MOV CTRSL,TADDR ;INITIALIZE TEST ADDRESS
888 050062 117737 133546 001143 MOV @CTRSH,$BDDAT+1 ;READ HIGH BYTE OF COUNTER.
889 050070 117737 133542 001142 MOV @CTRSL,$BDDAT ;READ LOW BYTE OF COUNTER.
890 050076 023737 001140 001142 CMP $GDDAT,$BDDAT ;IF DATA IS AS EXPECTED
891 050104 001401 BEQ 2$ ;-THEN SKIP ERROR
892
893 050106 104075 EMT 75
894
895 050110 023737 003632 003462 2$: CMP BCNTL,TADDR ;IF DONE CHECKING CLEAR
896 050116 001407 BEQ TSTPT1 ;-THEN DO PATTERN TEST
897 050120 013737 003632 003636 MOV BCNTL,CTRSL ;-ELSE TEST COUNTER B
898 050126 013737 003630 003634 MOV BCNTH,CTRSH
899 050134 000747 BR 1$
900
```

```
902                                     ;CHECK THE READ/WRITE CAPABILITIES OF THE COUNTERS
903
904 050136 005000 TSTPT1: CLR      RO      ;INITIALIZE PATTERN POINTER.
905 050140 142777 000010 132476 BICB   #TBIT,@CSR ;ASSURE THAT TBIT IS OFF
906
907 050146 013737 003626 003636 1$:  MOV   ACNTL,CTRSL ;TEST COUNTER A
908 050154 013737 003624 003634     MOV   ACNTH,CTRSH
909 050162 116037 002630 001140     MOVB  PATT(RO),$GDDAT ;GET EXPECTED PATTERN
910 050170 116037 002630 001141     MOVB  PATT(RO),$GDDAT+1
911
912 050176 152777 000010 132440 2$:  BISB  #TBIT,@CSR ;INITIALIZE BOARD
913 050204 013737 003636 003462     MOV   CTRSL,TADDR ;INITIALIZE TEST ADDRESS
914 050212 113777 001141 133414     MOVB  $GDDAT+1,@CTRSH ;WRITE INTO HIGH BYTE OF COUNTER
915 050220 113777 001140 133410     MOVB  $GDDAT,@CTRSL ;WRITE PATTERN INTO LOW BYTE
916 050226 117737 133402 001143     MOVB  @CTRSH,$BDDAT+1 ;READ COUNTERS HIGH BYTE
917 050234 117737 133376 001142     MOVB  @CTRSL,$BDDAT ;READ LOW BYTE OF COUNTER
918 050242 023737 001140 001142     CMP   $GDDAT,$BDDAT ;IF PATTERN IS AS EXPECTED
919 050250 001401                      BEQ   3$ ;-THEN SKIP ERROR
920
921 050252 104076                      EMT   76
922
923 050254 142777 000010 132362 3$:  BICB  #TBIT,@CSR ;STOP COUNTING
924 050262 023737 003632 003462     CMP   BCNTL,TADDR ;IF COUNTER B JUST TESTED
925 050270 001411                      BEQ   4$ ;-THEN DO NEXT PATTERN
926
927 050272 005137 001140                      COM   $GDDAT ;-ELSE PRODUCE EXPECTED PATTERN
928 050276 013737 003632 003636     MOV   BCNTL,CTRSL ;TEST COUNTER B
929 050304 013737 003630 003634     MOV   BCNTH,CTRSH
930 050312 000731                      BR    2$
931
932 050314 005200 4$:  INC   RO      ;MOVE PATTERN POINTER
933 050316 122760 000123 002630     CMPB  #123,PATT(RO) ;IF NOT AT END OF PATTERN TABLE
934 050324 001310                      BNE   1$ ;-THEN DO NEW PATTERN
```

```
936
937
938 ;CHECK THAT TBIT INITIALIZES MODULE
939 050326 005037 001140 TINIMO: CLR $GDDAT ;INITIALIZE EXPECTED DATA.
940 050332 112777 177777 133264 MOVB #-1,@ACNTH ;LOAD THE HIGH BYTE BUFFER.
941 050340 112777 177777 133260 MOVB #-1,@ACNTL ;LOAD COUNTER A WITH ALL 1'S.
942 050346 112777 177777 133256 MOVB #-1,@BCNTL ;LOAD COUNTER B WITH ALL 1'S.
943 050354 112777 000020 132262 MOVE #DBIT,@CSR ;ASSURE CSR IS CLEAR.
944
945 050362 152777 000010 132254 BISB #TBIT,@CSR ;INITIALIZE BOARD.
946 050370 013737 003626 003636 MOV ACNTL,CTRSL ;INITIALIZE FIRST TEST ADDRESS
947 050376 013737 003624 003634 MOV ACNTH,CTRSH
948
949 050404 013737 003636 003462 1$: MOV CTRSL,TADDR ;INITIALIZE TEST ADDRESS
950 050412 117737 133216 001143 MOVB @CTRSH,$BDDAT+1 ;READ HIGH BYTE OF TEST COUNTER
951 050420 117737 133212 001142 MOVB @CTRSL,$BDDAT ;READ LOW BYTE OF TEST COUNTER
952 050426 023737 001140 001142 CMP $GDDAT,$BDDAT ;IF DATA IS AS EXPECTED
953 050434 001401 BEQ 2$ ;-THEN SKIP ERROR
954
955 050436 104077 EMT 77
956
957 050440 023737 003632 003462 2$: CMP BCNTL,TADDR ;IF DONE CHECKING INIT
958 050446 001407 BEQ INTHL1 ;-THEN CHECK MODULE INTERRUPTS
959 050450 013737 003632 003636 MOV BCNTL,CTRSL ;-ELSE TEST COUNTER B.
960 050456 013737 003630 003634 MOV BCNTH,CTRSH
961 050464 000747 BR 1$
962
963 ;CHECK THAT MODULE INTERRUPTS
964
965 050466 112777 000030 132150 INTHL1: MOVB #DBIT!TBIT,@CSR ;START INTERNAL FREQ SOURCE
966 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$
967 050530 004737 015410 JSR PC,CLRINT ;CLEAR ANY INTERRUPTS CAUSED BY TBIT.
968
969 050534 013737 003624 003462 1$: MOV ACNTH,TADDR ;INITIALIZE FIRST COUNTER TO TEST
970 050542 012777 050710 132100 MOV #2$,@VECTO ;INITIALIZE INTERRUPT VECTOR
971 050550 012777 000340 132074 MOV #PR7,@VECTOA ;INITIALIZE INTERRUPT PRIORITY
972 050556 012737 000002 003642 MOV #2,NUM ;INITIALIZE A COUNT
973
```

```
975                                     ;ALLOW AN INTERRUPT AND WAIT
976
977 050564                               8$:
050564 012746 000000                     MOV #PRO,-(SP) ;SET PSW TO PRIORITY 0
050570 012746 050576                     MOV #66$,-(SP)
050574 000002                             RTI
050576 000240                             66$: NOP
978 050600 152777 000100 132036         BISB #EBIT,@CSR ;ALLOW THE INTERRUPT
979 050606 113777 003643 132646         MOVB NUM+1,@ADDR ;CLEAR HIGH BYTE BUFFER
980 050614 142737 000001 003462         BICB #BIT0,TADDR ;PREPARE TO LOAD COUNTER
981 050622 113777 003642 132632         MOVB NUM,@TADDR ;LOAD COUNTER
982 050630 013737 002636 003446         MOV K,ZLOOP
050636 005037 003444                     68$: 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$
983
984 050664 104103                         EMT 103
985
986 050666 012746 000340                 MOV #PR7,-(SP) ;SET PSW TO PRIORITY 7
050672 012746 050700                 MOV #69$,-(SP)
050676 000002                             RTI
050700 000240                             69$: NOP
987 050702 000537                         BR 7$ ;SKIP INTERRUPT ROUTINE
```



```

989                                     ;TIMEOUT HANDLER FOR INTERRUPT CHECK
990
991 050704 022626                       33$:  CMP      (SP)+,(SP)+      ;RESET STACK POINTER
992 050706 000452                       BR       38$              ;CONTINUE INTERRUPT CHECKS
993
994                                     ;ON INTERRUPT DO 2$
995
996 050710 022626                       2$:  CMP      (SP)+,(SP)+      ;RESET STACK POINTER
997 050712 013746 000004                MOV      ERRVEC,-(SP)      ;SAVE TIMEOUT VECTOR
998 050716 013746 000006                MOV      ERRVEC+2,-(SP)   ;SAVE TIMEOUT PRIORITY
999 050722 013737 050704 000004        MOV      33$,ERRVEC       ;ON TIME OUT GO TO 33$
1000 050730 012737 000340 000006       MOV      #PR7,ERRVEC+2    ;ASSURE PSW REMAINS PRIORITY 7
1001 050736 005037 003510                CLR      TEMP1            ;PREPARE TO CONSTRUCT INTERRUPT ADDRESS
1002 050742 013737 002660 003502       MOV      BASE,TEMP        ;INITIALIZE TEMP
1003 050750 152777 000001 131666       3$:  BISB     #RBIT,@CSR     ;PREPARE TO CLEAR INTERRUPT
1004
1005                                     ;CHECK IF COUNTER INTERRUPTS
1006
1007 050756 117737 131664 003502        MOVB     @IAR,TEMP        ;GET LOW BYTE OF INTERRUPT ADDRESS
1008
1009 050764 023737 003626 003462        CMP      ACNTL,TADDR     ;IF COUNTER A IS BEING TESTED
1010 050772 001406                       BEQ      35$              ;--THEN SEE IF COUNTER B INTERRUPTS
1011
1012 050774 023737 003626 003502        CMP      ACNTL,TEMP      ;--ELSE IF A DIDN'T INTERRUPT
1013 051002 001006                       BNE     36$              ;--THEN SKIP ERROR
1014
1015 051004                               34$:
1016 051004 104110                       EMT      110
1017 051006 000410                       BR       37$              ;---CLEAR UNWANTED INTERRUPT
1018
1018 051010 023737 003632 003502       35$:  CMP      BCNTL,TEMP     ;IF COUNTER B INTERRUPTED
1019 051016 001772                       BEQ      34$              ;--THEN IT WAS UNEXPECTED
1020
1021 051020 023737 003502 003462       36$:  CMP      TEMP,TADDR     ;IF COUNTER INTERRUPTED
1022 051026 001411                       BEQ      4$              ;--THEN CLEAN UP.
1023
1024                                     ;CLEAR UNWANTED INTERRUPT
1025
1026 051030 105777 132446                37$:  TSTB     @TEMP        ;CLEAR UNWANTED INTERRUPT
1027 051034 005237 003510                38$:  INC      TEMP1        ;INCREMENT INTERRUPT+TIMEOUT COUNT
1028 051040 022737 000400 003510        CMP      #400,TEMP1      ;IF NOT EXCESSIVE COUNT NUMBER
1029 051046 001340                       BNE     3$              ;--THEN CHECK AGAIN
1030
1031 051050 104103                       EMT      103
1032
1033 051052 012637 000004                4$:  MOV      (SP)+,ERRVEC   ;RESTORE TIMEOUT VECTOR
1034 051056 012637 000006                MOV      (SP)+,ERRVEC+2  ;RESTORE TIMEOUT PRIORITY
1035 051062 004737 015410                JSR      PC,CLRINT       ;CLEAR ALL INTERRUPTS
1036 051066 132777 000200 131550        BITB     #FBIT,@CSR     ;IF INTERRUPT CLEARS
1037 051074 001401                       BEQ      5$              ;THEN SKIP ERROR.
1038
1039 051076 104074                       EMT      74
1040
1041 051100 005037 001140                5$:  CLR      $GDDAT        ;INITIALIZE EXPECTED DATA
1042 051104 013737 002636 003446        MOV      K,ZLOOP
      051112 005037 003444                71$:  CLR      YLOOP
      051116 005237 003444                70$:  INC      YLOOP
  
```

```
051122 023127 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 112777 000030 131412 TRANP1: MOVB #DBIT!TBIT,@CSR ;CLEAR EXTRANEIOUS BITS AT CSR.
1060 051232 013737 003624 003462 MOV ACNTH,TADDR ;INITIALIZE COUNTER TO TEST
1061
1062 051240 012737 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 1$ ;
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          STSTIH: CLR B   @ACNTH      ;LOAD THE HIGH BYTE BUFFER
1106 051436 105077 132164          CLR B   @ACNTL      ;CLEAR COUNTER A
1107 051442 105077 132164          CLR B   @BCNTL      ;CLEAR COUNTER B
1108 051446 004737 015410          JSR     PC,CLRINT   ;ASSURE PENDING INTERRUPTS ARE SERVICED
1109
1110 051452 142777 000010 131164    BIC B   #TBIT,@CSR  ;PREPARE TO INITIALIZE BOARD
1111 051460 112777 000030 131156    MOV B   #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,$GDDAT   ;ADD FUDGE FACTOR
1121
1122                                     ;ALLOW THE INTERRUPTS
1123
1124 051544 152777 000100 131072    BIS B   #EBIT,@CSR  ;ENABLE INTERRUPT BIT AT CSR
1125 051552 012746 000000          MOV     #PRO,-(SP)    ;SET PSW TO PRIORITY 0
      051556 012746 051564          MOV     #64$,-(SP)
      051562 000002
      051564 000240          64$:  NOP
1126
1127 051566 113777 001141 132030    MOV B   $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    MOV B   $GDDAT,@ACNTL ;LOAD COUNTER A
1131 051612 113777 001140 132012    MOV B   $GDDAT,@BCNTL ;LOAD COUNTER B
1132
1133 051620 005237 003646          8$:   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$:   TST     CLK
1140 051640 001767          BEQ     8$          ;IF CLOCK HASN'T COUNTED
      ;-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$,-(SP)
      051662 000002
      051664 000240          65$:  NOP
1146 051666 142777 000110 130750    BIC B   #EBIT!TBIT,@CSR ;DON'T LEAVE THESE BITS HANGING
1147
1148 051674 117737 131724 001143    MOV B   @ACNTH,$BDDAT+1 ;GET CONTENTS OF COUNTER A
1149 051702 117737 131720 001142    MOV B   @ACNTL,$BDDAT
1150 051710 117737 131714 003651    MOV B   @BCNTH,TEMPB+1 ;GET CONTENTS OF COUNTER B
1151 051716 117737 131710 003650    MOV B   @BCNTL,TEMPB
1152
1153 051724 006337 003644          ASL     VARI         ;DOUBLE THE VARIANCE

```

MAIN. MACRO M1113 12-AUG-82 09:59 PAGE 64-1  
 TESTING DIGITAL MODULE M6014

1154								
1155	051730	023737	03644	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$: BISB #RBIT,@CSR ;PREPARE TO CLEAR UNEXPECTED INTERRUPT
1193 052076 105777 131400 TSTB @TEMP ;CLEAR THE INTERRUPT
1194 052102 012746 000000 MOV #PRO,-(SP) ;SET PSW TO PRIORITY 0
052106 012746 052114 MOV #66$,-(SP)
052112 000002 RTI
052114 000240 NOP
1195 052116 000651 BR 1$ 66$: ;SEE IF TIME IS UP
1196
1197 052120 004737 015410 7$: 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
1200 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.

```

1230  
 1231  
 1232  
 1233  
 1234  
 1235  
 1236  
 1237  
 1238  
 1239  
 1240  
 1241  
 1242  
 1243  
 1244  
 1245  
 1246  
 1247  
 1248  
 1249  
 1250  
 1251  
 1252  
 1253  
 1254  
 1255  
 1256  
 1257  
 1258  
 1259  
 1260  
 1261  
 1262  
 1263  
 1264  
 1265  
 1266  
 1267  
 1268  
 1269  
 1270  
 1271

```
.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
```

```
SR5016: .WORD 171000 ;ADDRESS OF STATUS REGISTER
DR5016: .WORD 171001 ;ADDRESS OF DATA REGISTER

OVFS: .WORD BIT4 ;OVERFLOW BIT FOR COUNTER 0
      .WORD BIT5 ;OVERFLOW BIT FOR COUNTER 1
      .WORD BIT6 ;OVERFLOW BIT FOR COUNTER 2
      .WORD BIT7 ;OVERFLOW BIT FOR COUNTER 3

DSWCH: .WORD 0 ;COUNTER 0 DIP SWITCH SETTINGS
       .WORD 0 ;COUNTER 1 DIP SWITCH SETTINGS
       .WORD 0 ;COUNTER 2 DIP SWITCH SETTINGS
       .WORD 0 ;COUNTER 3 DIP SWITCH SETTINGS

CTASCII: .ASCII <12><15>/M5016/
CTN: .ASCIZ <12><15>/ CTN 1 2 3 4 5 6 7 8 9 10 ADDRESS = /
BLANK2: .ASCIZ / /
.EVEN
```

```
052276 171000
052300 171001
052302 000020
052304 000040
052306 000100
052310 000200
052312 000000
052314 000000
052316 000000
052320 000000
052322 012 015 115
052331 012 015 040
052413 040 040 000
```



1273 052416  
052416 000004  
052420 012737 000023 001212  
1274  
1275  
1276  
1277 052426 004737 014310  
1278 052432 152777 000020 130204  
1279 052440 013737 003462 052276  
1280 052446 013737 003462 052300  
1281 052454 052737 000001 052300  
1282 052462 104411  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296 052464 005037 001140  
1297  
1298  
1299  
1300 052470 117737 177602 001142  
1301 052476 123737 001140 001142  
1302 052504 001401  
1303  
1304 052506 104075

M5016:  
\*\*\*\*\*  
TST23: SCOPE  
MOV #23,\$TESTN ;;SET TEST NUMBER IN APT MAIL BOX  
;START WITH A KNOWN CONDITION -- CLEAR THE MODULE  
JSR PC,KLEER ;CLEAR MODULE.  
BISB #DBIT,@CSR ;SHUT OFF WORLD  
MOV TADDR,SR5016 ;INITIALIZE STATUS REG ADDRESS  
MOV TADDR,DR5016 ;INITIALIZE DATA REG ADDRESS  
BIS #BIT0,DR5016  
SAVREG ;SAVE ALL REGISTERS  
  
;STATUS OF M5016/PRESCALER AT THIS TIME SHOULD BE --  
; STATUS REGISTER = 0 DATA REGISTER = 0  
; -CONTENTS OF COUNTERS ARE-  
; COUNTER 0 CONTAINS 0  
; COUNTER 1 CONTAINS 0  
; COUNTER 2 CONTAINS 0  
; COUNTER 3 CONTAINS 0  
CLR \$GDDAT ;INITIALIZE EXPECTED DATA  
;CHECK THAT STATUS REGISTER IS CLEAR.  
MOVB @SR5016,\$BDDAT ;READ STATUS REGISTER  
CMPB \$GDDAT,\$BDDAT ;IF STATUS REGISTER IS CLEAR  
BEQ CK4CLX ;-THEN CHECK COUNTERS ARE CLEAR  
EMT 75

```
1306
1307
1308 ;CHECK THAT ALL COUNTERS ARE CLEAR
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 ;INITIALIZE 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 111
1349
1350 052644 000756 BR 1$ ; CONTINUE READ/WRITE TESTING.
1351
1352 ;--END 1$
1353
```

```
1355
1356 ;STATUS OF M5016/PRESCALER AT THIS TIME SHOULD BE --
1357 ;
1358 ; STATUS REGISTER = 4 DATA REGISTER = 0
1359 ;
1360 ; -CONTENTS OF COUNTERS ARE-
1361 ;
1362 ; COUNTER 0 CONTAINS 0
1363 ; COUNTER 1 CONTAINS 0
1364 ; COUNTER 2 CONTAINS 0
1365 ; COUNTER 3 CONTAINS 0
1366
1367 ;TEST THAT CLEAR ENABLE BIT SELF CLEARS
1368
1369 052646 005037 001140 CKSRCX: CLR $GDDAT ;PLACE EXPECTED DATA IN $GDDAT
1370 052652 004737 015526 JSR PC,CNTRC ;CONTROL C ???????
1371 052656 105777 177416 TSTB @DR5016 ;CLEAR STATUS REG CLEAR BIT
1372 052662 117737 177410 001142 MOVB @SR5016,$BDDAT ;READ STATUS REGISTER
1373 052670 023737 001140 001142 CMP $GDDAT,$BDDAT ;IF STAT REG CLEARS SELF
1374 052676 001401 BEQ CCNTCX ;-THEN CHECK THAT COUNTERS CLEAR
1375
1376 052700 104112 EMT 112
1377
1378 ;CHECK THAT STATUS REG CLEARS EACH COUNTER INDIVIDUALLY
1379
1380 052702 CCNTCX:
1381 052702 012746 000340 MOV #PR7,-(SP) ;SET PSW TO PRIORITY 7
1382 052706 012746 052714 MOV #64$,-(SP)
1383 052712 000002 RTI
1384 052714 000240 64$: NOP
1385 052716 005037 003640 CLR CNUM ;INITIALIZE COUNTER NUMBER
1386 052722 142777 000010 127714 BICB #TBIT,@CSR ;ASSURE TBIT IS CLEAR
1387 052730 152777 000010 127706 BISB #TBIT,@CSR ;TOGGLE THE COUNTERS
1388 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 C00340      MOV #PR7,-(SP)      ;SET PSW TO PRIORITY 7
      053054 012746 053062      MOV #64$,-(SP)
      053060 000002
      053062 000240
1414 053064 004737 015526      64$:  NOP
1415
1416 053070 012737 000010 003642      JSR PC,CNTRC      ;CONTROL C ??????
1417 053076 012737 177777 003640      MOV #10,NUM      ;INITIALIZE OVERFLOW POINTER
1418
1419
1420
      ;INITIALIZE COUNTER DIP SWITCH SETTINGS
1421 053104 005005
1422 053106 005065 052312      10$:  CLR R5          ;INITIALIZE POINTER REGISTER
1423 053112 005725
1424 053114 022705 000010      CLR DSWCH(R5)    ;CLEAR CONTENTS OF DIP TABLE
1425 053120 001372
      TST (R5)+     ;ADVANCE POINTER
      CMP #10,R5   ;IF NOT FINISHED
      BNE 10$      ;-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
      CLR $BDDAT+1 ;CLEAR EXTRANEIOUS DATA
1453 053250 023737 001140 001142      CMP $GDDAT,$BDDAT ;IF DATA IS AS EXPECTED
1454 053256 001750
      BEQ 2$      ;-THEN DO NEXT COUNT
1455
1456
      ;--END 2$
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$:  EMT 114
1464 053276 104114
      BR 1$      ;--DO NEXT COUNTER
```

```

1465
1466 ;DETERMINE IF LEGAL RADIX
1467
1468 053300 012737 000001 003502 4$: MOV #BIT0,TEMP ;INITIALIZE RADIX LOOP VARIANT
1469
1470 ;--DO 40$ WHILE ( NOT ALL LEGAL RADII TESTED )
1471
1472 053306 022737 000400 003502 40$: CMP #BIT8,TEMP ;IF ALL LEGAL RADII NOT TESTED
1473 053314 001002 BNE 5$ ;--THEN SKIP ERROR
1474
1475 053316 104117 EMT 117
1476 053320 000700 BR 1$ ;--DO NEXT COUNTER
1477
1478 053322 006337 003502 5$: ASL TEMP ;COMPUTE LEGAL RADIX TO TEST
1479 053326 023737 003502 001140 CMP TEMP,$GDDAT ;IF NOT CORRECT RADIX
1480 053334 001364 BNE 40$ ;--THEN SEE IF ALL RADII TESTED
1481
1482 ;--END 40$
1483
1484 053336 006237 003502 ASR TEMP ;--ELSE COMPUTE CORRECT SWITCH
1485 053342 013705 003640 MOV CNUM,R5 ;--INITIALIZE OFFSET
1486 053346 006305 ASL R5
1487 053350 053765 003502 052312 BIS TEMP,DSWCH(R5) ;--SET CORRESPONDING DIP SWITCH
1488
1489 ;CHECK THAT DATA IS RESET ON OVERFLOW
1490
1491 053356 117737 176716 001142 MOVB @DR5016,$BDDAT ;--IF COUNTER CNUM NOT ZERO
1492 053364 001003 BNE 6$ ;---THEN SKIP DIP SWITCH SET
1493
1494 053366 052765 001000 052312 BIS #BIT9,DSWCH(R5) ;---ELSE COUNTER RESETS ON OVF
1495
1496 ;CHECK THAT OVERFLOW CLEARS
1497
1498 053374 152777 000001 127242 6$: BISB #RBIT,@CSR
1499 053402 105777 176670 TSTB @SR5016 ;CLEAR THE OVERFLOW
1500 053406 005037 001142 CLR $BDDAT ;INITIALIZE DATA WORD
1501 053412 117737 176660 001142 MOVB @SR5016,$BDDAT ;READ STATUS REGISTER
1502 053420 005037 001140 CLR $GDDAT ;INITIALIZE EXPECTED DATA
1503 053424 033737 003642 001142 BIT NUM,$BDDAT ;IF DATA IS AS EXPECTED
1504 053432 001633 BEQ 1$ ;--THEN TEST NEXT COUNTER
1505
1506 053434 104115 EMT 115
1507 053436 000631 BR 1$ ;--TEST NEXT COUNTER
1508
1509 ;--END 1$
1510

```

```
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 EXTRANEIOUS 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 C00001 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$:   JSR   PC,CLRINT ;CLEAR ALL POSSIBLE INTERRUPTS
1579
1580                    ;FORCE AN INTERRUPT
1581
1582 053734 012746 000000          MOV    #PRO,-(SP)   ;SET PSW TO PRIORITY 0
      053740 012746 053746          MOV    #64$,-(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$.
1604 054100 142777 000010 126536    BICB   #TBIT,@CSR  ;ASSURE THAT TBIT IS CLEARED.
1605 054106 005037 003510          CLR    TEMP1       ;INITIALIZE INTERRUPT TALLY.
1606 054112 013737 002660 003502    MOV    BASE,TEMP   ;INITIALIZE TEMP
1607 054120 152777 000001 126516    7$:   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

```



```
1614
1615
1616
1617 054144 105777 127332
1618 054150 005237 003510
1619 054154 022737 000400 003510
1620 054162 001356
1621
1622 054164 142777 000100 126452
1623 054172 010437 000004
1624 054176 000137 053472
1625
1626 054202 052765 000400 052312
1627 054210 012737 000001 003650
1628
1629
1630
1631 054216 016537 052302 001140
1632 054224 117737 176046 001142
1633 054232 042737 177417 001142
1634 054240 023737 001140 001142
1635 054246 001401
1636
1637 054250 104114
1638
1639 054252 000137 053472
1640
1641
1642
1643 054256 005737 003650
1644 054262 001010
1645 054264 132737 000001 001226
1646 054272 001403
1647 054274 005737 001214
1648 054300 001075
1649 054302
      054302 104103

;CLEAR THE UNWANTED INTERRUPT
16$: TSTB @TEMP ;--ELSE CLEAR INTERRUPT.
      INC TEMP1 ;COUNT NUMBER OF INTERRUPTS.
      CMP #400,TEMP1 ;IF 400 INTERRUPTS NOT COUNTED
      BNE 7$ ;--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 EXTRANEIOUS DATA
      CMP $GDDAT,$BDDAT ;IF DATA IS AS EXPECTED
      BEQ 18$ ;--THEN SKIP OVERFLOW ERROR

      EMT 114

18$: JMP 1$ ;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      ,CTASCII ;PRINT A HEADING
1660 054322 013746 003462      MOV      TADDR,-(SP) ;:PUSH TADDR ON STACK
1661 054326 104402              TYPDC      ;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
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$:   CMP      #3,CNUM      ;IF COUNTER # IS NOT LESS THAN 3
1738 054544 002416          BLT      MAXRDX      ;--THEN GET MAXIMUM RADIX
1739 054546 113777 003640 175522  MOVB     CNUM,@SR5016 ;--ELSE LOAD STATUS REG WITH COUNTER #
1740 054554 117737 175520 001142  MOVB     @DR5016,$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

```

```
1752 ;GET MAXIMUM RADIX
1753
1754 054602 012737 000400 003502 MAXRDX: MOV #BIT8,TEMP ;INITIALIZE LOOP VARIANT
1755
1756 ;TRY EACH SWITCH SETTING UNTIL ONE AGREES STARTING AT 8
1757
1758 ;DO 3$ WHILE (TEMP IS NOT MAXIMUM OR NO SWITCHES SET )
1759 054610 006237 003502 3$: ASR TEMP ;GET BIT TO TEST
1760 054614 001463 BEQ 5$ ;ESCAPE IF NO SWITCHES SET
1761
1762 054616 012737 177777 003640 MOV #-1,CNUM ;INITIALIZE COUNTER VARIANT
1763
1764 ;SEE IF ANY COUNTERS ARE SET TO SWITCH SETTING TEMP
1765
1766 ;--DO 4$ WHILE ( COUNTERS NOT ALL DONE )
1767 054624 005237 003640 4$: INC CNUM ;ADVANCE COUNTER NUMBER
1768 054630 022737 000004 003640 CMP #4,CNUM ;IF DONE ALL COUNTERS
1769 054636 001764 BEQ 3$ ;--THEN SEE IF DONE SWITCHES
1770
1771 054640 013705 003640 MOV CNUM,R5 ;CONSTRUCT THE OFFSET
1772 054644 006305 ASL R5
1773 054646 033765 003502 052312 BIT TEMP,DSWCH(R5) ;IF A SWITCH IS DETECTED NOT SET
1774 054654 001763 BEQ 4$ ;--THEN TRY THE NEXT COUNTER
1775
1776 ;--END 4$
1777 ;--END 3$
1778
1779 054656 006337 003502 ASL TEMP ;GET MAXIMUM RADIX
1780
1781 054662 006337 003502 ASL TEMP ;GET TWICE MAXIMUM
1782 054666 005337 003502 DEC TEMP ;COMPUTE NUMBER TO MAX BIT SETS
1783
1784 ;CLEAR ALL THE COUNTERS BEFORE MAXING THE COUNTS
1785
1786 054672 005037 003642 CLR NUM ;INITIALIZE LOOP COUNT
1787
1788 ;DO 50$ WHILE ( NOT ALL COUNTERS HAVE BEEN CLEARED )
1789
1790 054676 022737 000004 003640 50$: CMP #4,CNUM ;IF ALL COUNTERS CLEARED
1791 054704 001425 BEQ 52$ ;--THEN FILL COUNTS TO MAXIMUM
1792
1793 054706 113777 003640 175362 MOVB CNUM,@SR5016 ;LOAD COUNTER NUMBER TO CLEAR
1794 054714 152777 000004 175354 BISB #4,@SR5016 ;PREPARE TO CLEAR COUNTER
1795 054722 105777 175352 TSTB @DR5016 ;CLEAR THE COUNTER
1796 054726 005037 001140 CLR $GDDAT ;INITIALIZE EXPECTED DATA
1797 054732 117737 175342 001142 MOVB @DR5016,$BDDAT ;READ COUNTER
1798 054740 023737 001140 001142 CMP $GDDAT,$BDDAT ;IF DATA IS AS EXPECTED
1799 054746 001401 BEQ 51$ ;--THEN SKIP ERROR
1800
1801 054750 104112 EMT 112
1802
1803 054752 005237 003640 51$: INC CNUM ;PREPARE TO DO NEXT COUNTER
1804 054756 000747 BR 50$ ;DO NEXT COUNTER
1805
1806 ;--END 50$
1807
1808 ;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$:   CMP      #3,CNUM      ;IF COUNTER # IS NOT LESS THAN 3
1847 055102 002416                                     ;--THEN CLEAN UP AND EXIT
1848 055104 113777 003640 175164      MOVB     CNUM,@SR5016    ;--ELSE LOAD STATUS REG WITH COUNTER #
1849 055112 117737 175162 001142      MOVB     @DR5016,$BDDAT ;READ DATA REG
1850 055120 123737 001140 001142      CMPB     $GDDAT,$BDDAT  ;IF COUNTER 'CNUM' IS CLEAR
1851 055126 001401                                     ;--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
```

```

1863 .SBTTL MONITOR TO SELECT DAC TESTS
1864 055144 005737 C02664 DCAMON: TST AFLAG ;UNDER AUTO MODE ?
1865 055150 001416 BEQ DACSTR ;NO,BRANCH TO MANUAL
1866 055152 012777 015612 125544 MOV #KBINT,@KBVEC
1866 055160 152777 000100 123772 BISB #BIT6,@$TKS ;ENABLE KB INTERRUPT
1867 055166 004737 055450 JSR PC,GTADRS ;SET UP ADDRESSES FOR DAC
1868 055172 004737 055544 JSR PC,TSTADR ;CHECK PRESENCE OF ADDRESSES
1869 055176 000207 RTS PC
1870 055200 004737 055512 JSR PC,DDBIT ;GO DO D-BIT TEST
1871 055204 000207 RTS PC ;RETURN TO AUTO MONITOR
1872
1873 055206 012737 055206 003652 DACSTR: MOV #DACSTR,RETURN ;SET RETURN FOR CONTROL A
1874 055214 004737 055450 DACMON: JSR PC,GTADRS ;SET UP ADDRESSES
1875 055220 004737 055544 JSR PC,TSTADR ;CHECK RESPONSE OF ADDRESSES
1876 055224 000207 DCOUT: RTS PC ;EXIT TO MONITOR ON ERROR
1877 055226 004737 060112 JSR PC,DAGTST ;CHECK THAT GENERIC CODE OF
1878 055232 104401 036516 TYPE ,DANGC ;THE DAC IS 261 IF IT IS GO TEST
1879 055236 000207 RTS PC ;ELSE RETURN TO MAIN MON$PASS
1880 055240 104401 036652 DCMN: TYPE ,DANSEL ;SELECT TEST A, T, D,
1881 055244 004737 014310 JSR PC,KLEER
1882 055250 104406 RDCHR ;--?--FOR MEANING OF SELECTION
1883 055252 012637 003566 MOV (SP)+,ANSW ;:POP STACK INTO ANSW
1884 055256 104401 003566 TYPE ,ANSW
1885 055262 104401 001203 TYPE ,$CRLF
1886 055266 122737 000110 003566 CMPB #'H,ANSW ;IF QUESTION THE ELIGHTEN
1887 055274 001003 BNE 1$ ;USER BY TYPING TEXT OF TEST
1888 055276 104401 036754 TYPE ,TEXT ;SELECTION DESCRIPTORS
1889 055302 000756 BR DCMN ;NOW SELEC TEST
1890 055304 122737 000101 003566 1$: CMPB #'A,ANSW ;SELECT DAC CALIBRATION TEST
1891 055312 001015 BNE 2$ ;IF NOT THIS TEST THEN CHECK NEXT
1892 055314 104401 030660 TYPE ,M30 ;DISCONNECT CUSTOMER WIRES
1893 055320 104406 5$: RDCHR
1894 055322 012637 003566 MOV (SP)+,ANSW ;:POP STACK INTO ANSW
1895 055326 122737 000015 003566 CMPB #15,ANSW
1896 055334 001371 BNE 5$
1897 055336 004737 056132 JSR PC,DACTST ;ELSE DO THIS TEST
1898 055342 000137 055240 JMP DCMN
1899 055346 122737 000124 003566 2$: CMPB #'T,ANSW ;SELECT TBIT (STATUS BITS) TEST
1900 055354 001016 BNE 3$ ;IF NOT THIS TEST THEN CHECK NEXT
1901 055356 104401 030660 TYPE ,M30
1902 055362 104406 6$: RDCHR
1903 055364 012637 003566 MOV (SP)+,ANSW ;:POP STACK INTO ANSW
1904 055370 122737 000015 003566 CMPB #15,ANSW
1905 055376 001371 BNE 6$
1906 055400 004737 017306 14$: JSR PC,SWLOOP
1907 055404 056672 INTCOM ;DO INTCOM TEST
1908 055406 000137 055240 JMP DCMN
1909 055412 122737 000104 003566 3$: CMPB #'D,ANSW ;SELECT DBIT TEST
1910 055420 001005 BNE 4$ ;IF NOT THIS TEST THEN CHECK NEXT
1911 055422 004737 017306 9$: JSR PC,SWLOOP ;LOOP HERE
1912 055426 055512 DDBIT ;DO DDBIT TEST
1913 055430 000137 055240 JMP DCMN
1914 055434 004737 015526 4$: JSR PC,CNTRC ;CHECK FOR EXIT BY CONTROL "C"
1915 055440 104401 030374 TYPE ,M20 ;TYPE ? IF NO TST SELECTION FOUND
1916 055444 000137 055240 JMP DCMN ;GO TO START OF DAC MONITOR
1917
1918

```



```
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
1943 055544 013703 C00004
1944 055550 012737 055602 000004
1945 055556 004737 014310
1946 055562 005000
1947 055564 005001
1948 055566 013700 003462
1949 055572 105720
1950 055574 001407
1951 055576 104027
1952 055600 000405
1953 055602
    055602 104026
1954 055604 004737 015526
1955 055610 022626
1956 055612 000406
1957 055614 005201
1958 055616 022701 000010
1959 055622 001363
1960 055624 062716 000002
1961 055630 010337 000004
1962 055634 000207
1963
1964 055636

.SBTTL REGISTER VERIFICATION
TSTADR: MOV ERRVEC,R3 ;SAVE PRESENT LOC 4
        MOV #2$,ERRVEC ;SET UP TIME OUT VECTOR
        JSR PC,KLEER ;INIT THE SYSTEM
        CLR R0 ;CLR ADDRESS LOC
        CLR R1 ;CLR ADDRESS COUNTER
        MOV TADDR, R0 ;GET FIRST REG ADDRESS
1$: TSTB (R0)+ ;IS IT THERE AND CLEAR
    BEQ 3$ ;IF IT IS GET NEXT ADDRESS
    EMT 27
    BR 3$ ;AND THEN GET NEXT ADDRESS

2$: EMT 26
    JSR PC,CNTRC ;CHECK FOR CONTROL "C" EXIT
    CMP (SP)+,(SP)+ ;ADJUST THE STACK AFTER TIME OUT
    BR 4$ ;RETURN MAIN MONITOR
3$: INC R1 ;UPDATE ADDRESS COUNTER
    CMP #10,R1 ;CHECK FOR LAST REG
    BNE 1$ ;IF IT IS THFN
    ADD #2,(SP) ;CONTINUE WITH TEST
4$: MOV R3,ERRVEC ;RESTORE LOC4
    RTS PC ;RETURN

TSTRGS:
*****
TST24: SCOPE
        MOV #24,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
        CLR ROTPAT ;CLEAR THE PATTRN SPOT
        CLR R0 ;CLEAR THE INDEX REGISTER
        MOV #7,KOUNT ;DO SEVEN FOR NOW
1$: JSR PC,ROTDAT ;NOW GO ROTATE A BIT
2$: MOVB ROTPAT,@LDATA0(R0) ;NOW GO WRITE IT
    JSR PC,UPREG ;NOW GET THE NEXT REGISTER
    BR 2$ ;AND WRITE IN (4 IN ALL)
3$: MOV ROTPAT,$GDDAT ;SAVE PATTERN FOR CHECKING
    MOVB @LDATA0(R0),$BDDAT ;READ THE RSGISTER
    MOV LDATA0(R0),TEMP ;AND STORE ITS ADDRESS
    CMPB $GDDAT,$BDDAT ;SEE IF IT IS GOOD
    BEQ 4$ ;IF GOOD CONTINUE
    EMT 30
4$: JSR PC,CNTRC ;CHECK FOR CONTROL "C" EXIT
    JSR PC,UPREG ;READ NEXT REGISTER
    BR 3$ ;UNTIL 4 DONE
    DEC KOUNT ;AND UNTIL7 ROTATES
    BPL 1$ ;NOW GO BACK AND DO IT
5$: CLR ROTPAT ;CLEAR PATTERN SPOT
    MOV #1,KOUNT ;DO ONLY 2 THIS TIME
6$: JSR PC,ROTDAT ;GO ROTATE A BIT NOW
    MOVB ROTPAT,@HDATA0(R0) ;GO WRITE IT NOW
    JSR PC,UPREG ;GET THE NEXT REGISTER
    BR 6$ ;AND WRITE IN IT (4 IN ALL)
7$: MOV ROTPAT,$GDDAT ;SAVE PATTERN FOR CHECKING
    MOVB #0,@LDATA0(R0) ;NOW LET IT TRANSFER
    MOVB @HDATA0(R0),$BDDAT ;READ THE REGISTER
    MOV HDATA0(R0),TEMP ;AND STORE ITS ADDRESS
    CMPB $GDDAT,$BDDAT ;SEE IF IT IS GOOD
    BEQ 8$ ;IF GOOD CONTINUE
  
```

1995	056050	104030			EMT	30	
1996	056052	004737	C15526		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			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 INDFX REG
2014	056124	062716	000002		ADD	#2,(SP)	:UPDATE THE STACK
2015	056130	000207			RTS	PC	:CONTINUE TEST
2016							
2017							
2018							

```
2020 .SBTTL DAC CALIBRATION TEST
2021
2022
2023 056132 004737 014310 DACTST: JSR PC,KLEER
2024
2025 056136 004737 060156 DATST: JSR PC,DECCHN ;GET CHANEL # AND DECODE IT
2026 056142 104401 037062 TYPE ,CTXTV ;DO REFER VOLTAGE CAL Y-N
2027 056146 104406 RDCHR ;GET Y-N ANSW
2028 056150 012637 003566 MOV (SP)+,ANSW ;:POP STACK INTO ANSW
2029 056154 104401 003566 TYPE ,ANSW ;ECHO IT
2030 056160 122737 000116 003566 CMPB #'N,ANSW ;IF IT IS A NO
2031 056166 001413 BEQ 5$ ;GO TO CHAN AND
2032 056170 122737 000131 003566 CMPB #'Y,ANSW ;IS IT A YES?
2033 056176 001403 BEQ 20$
2034 056200 104401 030374 TYPE ,M20 ;?
2035 056204 000754 BR DATST
2036 056206 004737 015526 20$: JSR PC,CNTRC ;PASS REFER VOLT CAL TST
2037 056212 004737 056230 JSR PC,DCTST
2038 056216 004737 015526 5$: JSR PC,CNTRC
2039 056222 104401 036331 TYPE ,M81
2040 056226 000411 BR DTST
2041
2042 056230 104401 037245 DCTST: TYPE ,CM1 ;ADJ R135 @ PIN 6 OF E82 FOR 10.24
2043 056234 004737 015716 JSR PC,CRTST ;USING SWITCHES AT E79 IF NEEDED
2044 ;WHEN DONE HIT <CR>
2045 056240 104401 037306 TYPE ,CM2 ;ADJ R134 @ PIN 60 OF E81 FOR 5.12
2046 056244 004737 015716 JSR PC,CRTST ;WHEN DONE HIT <CR>
2047 056250 000207 RTS PC
2048 056252 013700 003532 DTST: MOV XCHAN,RO ;GET REG INDEX TO CHAN
2049 056256 104401 037347 TYPE ,CM3 ;ADJ VOLTAGE OFFSET ON CHAN 1 TO 0.000
2050 056262 004737 015716 21$: JSR PC,CRTST ;WAIT FOR <CR>
2051 056266 112770 000003 002730 1$: MOVB #3,@HDATA0(RO) ;LOAD ALL ONES INTO
2052 056274 112770 000377 002726 MOVB #377,@LDATA0(RO) ;CHAN SELECTED
2053 056302 104401 037410 TYPE ,CM4 ;CHECK VOLTAGE LEVEL ON CHANNEL SELECTED
2054 056306 004737 015716 22$: JSR PC,CRTST ;FOR 10.23V + OR - 10 MIN
2055 056312 104401 036370 TYPE ,M82
2056 056316 104406 41$: RDCHR
2057 056320 012637 003566 MOV (SP)+,ANSW ;:POP STACK INTO ANSW
2058 056324 104401 003566 TYPE ,ANSW
2059 056330 004737 015526 JSR PC,CNTRC
2060 056334 122737 000116 003566 CMPB #'N,ANSW
2061 056342 001002 BNE 40$
2062 056344 000137 056660 JMP 7$
2063 056350 122737 000131 003566 40$: CMPB #'Y,ANSW
2064 056356 001357 BNE 41$
2065 056360 104401 040177 TYPE ,CM12 ;ASK IF 4 TO 20 MA RANGE WANTED
2066 056364 104406 23$: RDCHR ;READ ANSWER
2067 056366 012637 003566 MOV (SP)+,ANSW ;:POP STACK INTO ANSW
2068 056372 104401 003566 TYPE ,ANSW ;ECHO ANSWER
2069 056376 122737 000131 003566 CMPB #'Y,ANSW ;IF YES GO TO 4 TO 20 MA TEST
2070 056404 001455 BEQ 5$ ;ELSE DO 0 TO 20 MA TEST
2071 056406 122737 000116 003566 CMPB #'N,ANSW
2072 056414 001403 BEQ 4$
2073 056416 004737 015526 JSR PC,CNTRC
2074 056422 000760 BR 23$
2075 056424 112770 000003 002730 4$: MOVB #3,@HDATA0(RO) ;RELOAD UPPER BYTE OF CHANEL
2076 056432 112770 000377 002726 MOVB #377,@LDATA0(RO) ;SELECTED AND LET IT GO (CONV)
```

```
2077 056440 104401 037624          TYPE      ,CM9                ;ADJ CURRENT GAIN TO 10.000V
2078 056444 004737 015716          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          JSR        PC,CRTST          ;OR 19.5 MICRO-AMPS
2084
2085 056474 104401 040132          TYPE      ,CM11             ;ASK TO REVERIFY
2086 056500 104406          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$
2095 056540 112770 000003 002730  MOVB      #3,@HDATA0(R0)    ;EXIT AFTER ALINEMENT
2096 056546 112770 000377 002726  MOVB      #377,@LDATA0(R0) ;LOAD ALL ONFS
2097 056554 104401 037624          TYPE      ,CM9                ;INTO SELECTED CHANEL
2098 056560 104406          RDCHR          ;ADJ CURRENT GAIN TO
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          JSR        PC,CRTST          ;FOR 2.00V OR 4.00MA
2106 056616 104401 040132          TYPE      ,CM11
2107 056622 104406          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          TYPE      ,M83
2116 056664 004737 014310          JSR        PC,KLEER
2117 056670 000207          RTS        PC                ;END OF CALIBRATION
```

2119  
2120  
2121 056672  
  
056672 000004  
056674 012737 000025 001212  
2122 056702 152777 000010 123734  
2123 056710 112777 000000 124010  
2124 056716 112777 000100 124006  
2125 056724 112777 000200 124004  
2126 056732 112777 000300 124002  
2127 056740 004737 014416  
2128  
2129 056744 013700 002730  
2130 056750 111001  
2131 056752 042701 177703  
2132 056756 022701 000074  
2133 056762 001417  
2134  
2135 056764 010137 003502  
2136 056770 006237 003502  
2137 056774 006237 003502  
2138 057000 012737 000017 003600  
2139 057006 104031  
2140 057010 004737 014310  
2141 057014 000207  
2142 057016 004737 015526  
2143 057022 062700 000002  
2144 057026 023700 002744  
2145 057032 003746  
2146 057034 005000  
2147 057036 004737 014310  
2148 057042 152777 000010 123574  
2149 057050 112777 000000 123650  
2150 057056 112777 000004 123646  
2151 057064 112777 000010 123644  
2152 057072 112777 000014 123642  
2153 057100 004737 014416  
2154 057104 117702 123620  
2155 057110 042702 177703  
2156 057114 022702 000074  
2157 057120 001415  
2158 057122 010237 003502  
2159 057126 006237 003502  
2160 057132 006237 003502  
2161 057136 012737 000017 003600  
2162 057144 104031  
2163 057146 004737 014310  
2164 057152 000207  
2165 057154 004737 015526  
2166 057160 004737 014310  
2167 057164 152777 000010 123452  
2168 057172 112777 000014 123526  
2169 057200 112777 000010 123524  
2170 057206 112777 000004 123522  
2171 057214 112777 000000 123520  
2172 057222 004737 014416

.SBTTL INTERNAL COMPARATOR CHECK

INTCOM:

\*\*\*\*\*

1ST25: SCOPE  
MOV #25,\$TESTN ;:SET TEST NUMBER IN APT MAIL BOX  
BISB #TBIT,@CSR ;:SET THE TBIT INTO THE CSR  
MOVB #0,@LDATA0 ;:START CH 0 CONVERSIONS  
MOVB #100,@LDATA1 ;:START CH 1 CONVERSIONS  
MOVB #200,@LDATA2 ;:START CH 2 CONVERSIONS  
MOVB #300,@LDATA3 ;:START CH 3 CONVERSIONS  
JSR PC,DALLY ;:WAIT FOR ALL CHANELS TO FINISH  
  
4\$: MOV HDATA0,R0 ;:GET FIRST REG TO CHECK  
MOVB (R0),R1 ;:GET HIGH DATA BYTE  
BIC #177703,R1 ;:CLEAR UNWANTED BITS  
CMP #74,R1 ;:IF ALL BIT ARE SET  
BEQ 5\$ ;:CONTINUE WITH TEST  
  
MOV R1,TEMP  
ASR TEMP  
ASR TEMP  
MOV #17,TEMP3  
EMT 31  
JSR PC,KLEER  
RTS PC  
5\$: JSR PC,CNTRC ;:CHECK FOR CONTROL "C" EXIT  
ADD #2,R0 ;:UPDATE TO NEXT HIGH DATA REG  
CMP HDATA3,R0 ;:AND CHECK TO SEE IF ALL STATUS  
BLE 4\$ ;:BITS ARE SET WHEN DONE  
CLR R0 ;:CLEAR R0 AND SYSTEM  
JSR PC,KLEER ;:RIGHT HERE  
BISB #TBIT,@CSR ;:RELOAD THE TBIT  
MOVB #0,@LDATA0 ;:CHECK ALL STATUS  
MOVB #4,@LDATA1 ;:BITS AT THE 3-LSB  
MOVB #10,@LDATA2 ;:DIFFERENCE MARGIN  
MOVB #14,@LDATA3  
JSR PC,DALLY ;:WAIT FOR IT TO HAPPEN  
MOVB @HDATA0,R2 ;:GET THE STATUS INFO  
BIC #177703,R2 ;:CLEAR UNWANTED BITS  
CMP #74,R2 ;:ALL FOUR STATUS BITS  
BEQ 6\$ ;:SHOULD BE SET ELSE ERROR  
MOV R2,TEMP  
ASR TEMP  
ASR TEMP  
MOV #17,TEMP3  
EMT 31  
JSR PC,KLEER  
RTS PC  
6\$: JSR PC,CNTRC ;:CHECK FOR CONTROL "C" EXIT  
JSR PC,KLEER ;:CLEAR THE SYSTEM  
BISB #TBIT,@CSR ;:RELOAD THE TBIT  
MOVB #14,@LDATA0 ;:CHECK ALL STATUS BIT AT  
MOVB #10,@LDATA1 ;:THE DESCENDING 3-LSB  
MOVB #4,@LDATA2 ;:DIFFERENCE MARGIN  
MOVB #0,@LDATA3  
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      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      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      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$            ;
```

```

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 WHICH 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    1$                ;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$:   RTS      PC                ;NOW USE DATA
2280
2281 060112 004737 014310          DAGTST: JSR    PC,KLEER          ;CLEAR IT ALL
2282 060116 152777 000004 122520      BISB   #GBIT,@CSR       ;SET THE GENERIC BIT IN THE CSR
2283 060124 117700 122576          MOVB   @LDATA0,RO        ;READ THE GENERIC CODE IN DAC
2284 060130 042700 177400          BIC    #177400,RO        ;CLEAR THE UPPER BYTE
2285 060134 022700 000261          CMP    #261,RO          ;CHECK THAT GENERIC CODE IS 261
2286 060140 001002          BNE    1$                ;IF NOT RETURN ERR MESS AND EXIT

```



```
2287 060142 012716 055240          MOV      #DCMN, (SP)          ;ELSE CONTINUE TO TEST
2288 060146 142777 C00004 122470 1$: BICB    #GBIT, @CSR          ;BUT FIRST CLEAR GBIT
2289 060154 000207          RTS      PC
2290
2291
2292 060156 104401 036714          DECCHN: TYPE      ,DACHN          ;ASK FOR CHANEL NUMBER (0,1,2,3)
2293 060162 104410          RDOCT
2294 060164 012637 003566          MOV      (SP)+,ANSW          ;GET CHANEL NUMBER
2295 060170 022737 000004 003566    CMP      #4,ANSW          ;;POP STACK INTO ANSW
2296 060176 003003          BGT      1$                ;CHECK FOR RIGHT CHAN
2297 060200 104401 030374          TYPE      ,M20                ;?
2298 060204 000764          BR
2299 060206 013737 003566 003530 1$: MOV      ANSW,DCHAN          ;GET DAC CHANEL SELECTED
2300 060214 000241          CLC
2301 060216 006337 003566          ASL      ANSW                ;NO END AROUND CARRY WANTED
2302 060222 013737 003566 003532    MOV      ANSW,XCHAN          ;CONVERT TO VCHAN SELECTION
2303 060230 006337 003532          ASL      XCHAN                ;GET ANSW TO INDEX
2304 060234 013737 003566 003534    MOV      ANSW,VCHAN          ;MULTIPLY BY 2 AGAIN
2305 060242 062737 000001 003566    ADD      #1,ANSW            ;STORE SELECTED VCHAN
2306 060250 013737 003566 003536    MOV      ANSW,ICHAN          ;CONVERT TO ICHAN SELECTED
2307 060256 000207          RTS      PC                ;STORE SELECTE ICHAN
2308                                     ;RETURN
```

```

2310
2311
2312
2313
2314      .SBTTL A/D MONITOR
2315
2316      ;THIS IS MONITOR FOR FOR AUTO MODE FOR A/D
2317 060260 005737 002664      ADTST:  TST      AFLAG      ;UNDER AUTO MODE ?
2318 060264 001431              BEQ      ATOD      ;NO,BRANCH TO MANUAL TEST
2319 060266 012777 015612 122430  MOV      #KBINT,@KBVEC
2319 060274 152777 000100 120656  BISB     #BIT6,@$TKS      ;ENABLE KB INTERRUPT
2320 060302 005005              CLR      R5
2321 060304 004737 017100      JSR     PC,ADADDR      ;SET ADDRESSES AND CHECK REG.
2322 060310 005705              TST     R5              ;ADDRESS ERROR?
2323 060312 001015              BNE     1$
2324 060314 152777 000001 122322  BISB     #RBIT,@CSR
2325 060322 105077 123262      CLRB    @STAT1
2326 060326 105077 123260      CLRB    @STAT2
2327 060332 004737 017254      JSR     PC,GCODE      ;FIND IF SE/DIFF MODF
2328 060336 004737 061620      JSR     PC,ADLOG      ;TEST LOGIC
2329 060342 004737 063244      JSR     PC,RUMP      ;TEST MONOTINICITY
2330 060346 000207      1$:     RTS      PC
2331
2332      ;THIS IS MONITOR FOR OPERATOR INTERVENTION TESTS
2333
2334 060350      ATOD:
2334 060350 012777 015612 122346  MOV      #KBINT,@KBVEC
2334 060356 152777 000100 120574  BISB     #BIT6,@$TKS      ;ENABLE KB INTERRUPT
2335 060364 004737 014310      JSR     PC,KLEER
2336 060370 012737 060376 003652  MOV      #ADRET,RETURN      ;SET RETURN ADDRESS FOR CONTROL A
2337 060376 005005      ADRET:  CLR      R5
2338 060400 004737 017100      JSR     PC,ADADDR      ;SET ADDRESSES
2339 060404 005705              TST     R5              ;ADDRESS ERROR?
2340 060406 001360              BNE     ATOD
2341 060410 152777 000001 122226  BISB     #RBIT,@CSR
2342 060416 105077 123166      CLRB    @STAT1      ;SET CHANNEL ZERO-NO MAX
2343 060422 004737 017254      JSR     PC,GCODE      ;FIND GENERIC CODE
2344
2345      ADMON:
2346 060426 104401 034132      TYPE    ,MASS46      ;WHICH TEST TO RUN?
2347 060432 012737 000014 003456  MOV      #14,$MUT      ;SET MODULE TYPE FOR ERRORS
2348 060440 105077 123144      CLRB    @STAT1
2349 060444 105077 123142      CLRB    @STAT2
2350 060450 104406      RDCHR
2351 060452 012637 003566      MOV     (SP)+,ANSW      ;:POP STACK INTO ANSW
2352 060456 104401 003566      TYPE    ,ANSW      ;:ECHO
2353 060462 104401 031400      TYPE    ,MASS0      ;:CR,LF
2354 060466 004737 015526      JSR     PC,CNTRC      ;CHECK FOR CONTROL C
2355 060472 122737 000103 003566  CMPB    #'C,ANSW      ;IS IT CALIBRATION-C
2356
2357 060500 001003              BNE     1$              ;NO
2358 060502 004737 060712      JSR     PC,ADCALB      ;EXECUTE CALIBRATION
2359 060506 000747      BR      ADMON
2360
2361
2362 060510 122737 000104 003566  1$:     CMPB    #'D,ANSW      ;IS IT LOGIC TEST
2363 060516 001005              BNE     2$              ;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 C33417          ADCALB: TYPE ,MASS37          ;WHICH MUX NUMBER YOU WANT TO
2409 060716 004737 017254          JSR PC,GCODE
2410 060722 004737 015526          JSR PC,CNTRC
2411
2412 060726 013746 000004          MOV ERRVEC,-(SP)          ;TEST-ZERO FOR A/D
2413 060732 012737 061570 000004  MOV #20$,ERRVEC          ;;PUSH ERRVEC ON STACK
2414 060740 104410
2415 060742 012637 003502          RDOCT
2416 060746 122737 000007 003502  MOV (SP)+,TEMP          ;;POP STACK INTO TEMP
2417 060754 002003          CMPB #7,TEMP
2418 060756 104401 030374          BGE 1$
2419 060762 000753          TYPE ,M20
2420
2421 060764 006337 003502          BR ADCALB
2422 060770 006337 003502          1$: ASL TEMP          ;SHIFT IT 5 TIMES TO SET MUX #
2423 060774 006337 003502          ASL TEMP
2424 061000 006337 003502          ASL TEMP
2425 061004 006337 003502          ASL TEMP
2426 061010 104401 033463          5$: 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$: CLR R0          ;SET GAIN =1
2457 061162 122737 000322 003576  CMPB #322,TEMP2          ;IS IT A156 DIFF
2458 061170 001007          BNE 7$
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
2471 061236 104401 033655          9$:  TYPE  ,MASS41 ;HERE ONLY IF A157
2472 061242 104410          RDOCT ;WHICH GAIN?
2473 061244 012637 003602          MOV  (SP)+,TEMP4 ;;POP STACK INTO TEMP4
2474 061250 005000          CLR  R0
2475 061252 023760 003602 003330 11$:  CMP  TEMP4,GAINB(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  GAINB(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  GAINB(R0),-(SP) ;;SAVE GAINB(R0) FOR TYPEOUT
2492 061326 104402          TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2493 061330 153777 003600 122252 4$:  BISB  TEMP3,@STAT1
2494 061336 104401 033754          TYPE  ,MASS43 ;CONNECT VOLTAGE SOURCE, TYPE CR
2495 061342 004737 015716          JSR  PC,CRTST ;WAIT FOR CR
2496 061346 104401 034011          TYPE  ,MASS44 ;TYPE HEADER -- OCTAL VOLTAGE--
2497 061352 004737 065502          14$:  JSR  PC,CONV7 ;DO 7 CONV & AVERAGE THEM
2498 061356 012701 000002          MOV  #2,R1
2499 061362          13$:
2500 061362 013737 002636 003446          MOV  K,ZLOOP
2501 061370 005037 003444          65$:  CLR  YLOOP ;WAIT
2502 061374 005237 003444          64$:  INC  YLOOP
2503 061400 023727 003444 177777          CMP  YLOOP,#-1
2504 061406 001372          BNE  64$
2505 061410 005337 003446          DEC  ZLOOP
2506 061414 001365          BNE  65$
2507 061416 005301          DEC  R1
2508 061420 001360          BNE  13$
2509 061422 032737 000004 003654          BIT  #BIT2,AVRSAV
2510 061430 001410          BEQ  40$
2511 061432 005237 003512          INC  INTDAT
2512 061436 013746 003512          MOV  INTDAT,-(SP) ;;SAVE INTDAT FOR TYPEOUT
2513 061442 104402          TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2514 061444 005337 003512          DEC  INTDAT
2515 061450 000403          BR   41$
2516
2517 061452          40$:
2518 061452 013746 003512          MOV  INTDAT,-(SP) ;;SAVE INTDAT FOR TYPEOUT
2519 061456 104402          TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2520 061460 162737 004000 003512 41$:  SUB  #4000,INTDAT ;CALCULATE VOLTAGE IN MILLIVOLTS
    
```

```
2510 061466 013737 003512 001140      MOV      INTDAT,$GDDAT
2511 061474 063737 C01140 001140      ADD      $GDDAT,$GDDAT      ;MULTIPLY BY 5
2512 061502 063737 001140 001140      ADD      $GDDAT,$GDDAT
2513 061510 063737 003512 001140      ADD      INTDAT,$GDDAT
2514 061516 042737 177770 003654      BIC      #177770,AVRSAV
2515 061524 013704 003654      MOV      AVRSAV,R4
2516 061530 116404 003374      MOV      AVRTBL(R4),R4
2517 061534 060437 001140      ADD      R4,$GDDAT
2518
2519 061540 104401 030036      TYPE    ,M10                ;TAB
2520 061544 104401 030036      TYPE    ,M10
2521 061550 013746 001140      MOV      $GDDAT,-(SP)      ;;SAVE $GDDAT FOR TYPEOUT
2522 061554 104405      TYPDS   ;;GO TYPE--DECIMAL ASCII WITH SIGN
2523 061556 104401 031400      TYPE    ,MASSO             ;CR,LF
2524 061562 004737 015526      JSR     PC,CNTRC           ;CONTROL C?
2525 061570 022626      BR      14$               ;DO IT AGAIN.
2526 061572 104401 031011      CMP     (SP)+,(SP)+
2527 061576 012637 000004      TYPE    ,M84              ;SELECTED MUX DID NOT RESPOND
2528 061602 152777 000001 121034      MOV     (SP)+,ERRVEC      ;;POP STACK INTO ERRVEC
2529 061610 105777 122000      BISB   #RBIT,@CSR        ;RESET TIME-OUT
2530 061614 000137 060712      TSTB   @LBYTE
      JMP   ADCALB
```

```
2532 .SBTTL LOGIC TEST OF A014
2533
2534 061620 ADLOG:
          061620 000004
          061622 012737 000026 001212 TST26: SCOPE
          *****
          061620 004737 014310 JSR PC,KLEER ;CLEAR THE IOCM WORLD
          061630 012746 000000 MOV #PRO,-(SP) ;SET PSW TO PRIORITY 0
          061634 012746 061646 MOV #64$,-(SP)
          061640 000002 RTI
          061644 000240 64$: NOP
          061650 152777 000001 120766 BISB #RBIT , @CSR ;CLEAR RANDOM INTERUPT
          061656 005037 001142 CLR $BDDAT
          061662 117737 121724 001142 MOVB @STAT2,$BDDAT ;STORE DATA IN $BDAT
          061670 132777 000001 121714 BITB #BIT0 , @STAT2 ;CHECK BUSY BIT CLEAR
          061676 001401 BEQ 1$ ;IF NO ERROR CONTINUE
          061700 104035 EMT 35
          061702 132777 000002 121702 1$: BITB #BIT1,@STAT2 ;CHECK CONV DONE IS CLEAR
          061710 001401 BEQ 2$ ;IF NO ERROR CONTINUE
          061712 104035 EMT 35
          061714 132777 000004 121670 2$: BITB #BIT2,@STAT2 ;CHECK ERROR BIT IS CLEAR
          061722 001401 BEQ 3$ ;IF NO ERROR CONTINUE
          061724 104035 EMT 35
          061726 132777 000010 121656 3$: BITB #BIT3,@STAT2 ;CHECK MUX TIME OUT IS CLEAR
          061734 001401 BEQ 4$ ;IF NO ERROR CONTINUE
          061736 104035 EMT 35
          061740 132777 000360 121644 4$: BITB #360,@STAT2 ;CHECK GAIN BITS
          061746 001401 BEQ BSYCON ;IF NO ERROR CONTINUE
          061750 104035 EMT 35
          061752 012777 062116 120670 BSYCON: MOV #10$,@VECTO
          061760 012777 000340 120664 MOV #PR7,@VECTOA
          061766 152777 000001 120650 BISB #RBIT,@CSR ;CLEAR INTERRUPTS
          061774 105777 121610 TSTB @STAT1
          062000 012700 177777 MOV #-1 , RO ;INITIALIZE COUNTER
          062004 152777 000001 121600 BISB #BIT0 , @STAT2 ;START CONVERSION
          062012 005200 1$: INC RO ;SRATR COUNTER
          062014 022700 000300 CMP #300 , RO ;CHECK FOR COUNT OF 100
          062020 001002 BNE 8$ ;STILL WAIT FOR CONVERSION
          062022 104040 EMT 40
          062024 000474 BR 7$
          062026 132777 000002 121556 8$: BITB #BIT1 , @STAT2 ;CHECK FOR CONVERSION DONE
          062034 001766 BEQ 1$ ;IF NOT DONE THEN LOOP
          062036 132777 000001 121546 2$: BITB #BIT0 , @STAT2 ;VERIFY BUSY CLEAR
          062044 001402 BEQ 3$ ;IF CLEAR CONTINUE
          062046 104036 EMT 36
          062050 000462 BR 7$
          062052 132777 000004 121532 3$: BITB #BIT2,@STAT2 ;CHECK ERROR CLEAR
          062060 001402 BEQ 4$ ;CONTINUE IF CLEAR
          062062 104037 EMT 37
          062064 000454 BR 7$
          062066 105777 120552 4$: TSTB @CSR ;TST INTERRUPT FLAG
          062072 100402 BMI 5$ ;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$:
2608 062216 152777 000001 120420 7$:    BISB   #RBIT,@CSR      ;CLEAR INTERRUPTS
2609 062224 105777 121364          TSTB   @LBYTE
2610 062230 004737 015410          JSR    PC,CLRINT
2611 062234 012746 000000          MOV     #PRO,-(SP)     ;SET PSW TO PRIORITY 0
      062240 012746 062246          MOV
      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 C00007 3$: CMP #7 , RO ;ELSE CHECK 7 DIFF CHANELS
2638 062360 001350 BNE 1$ ;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 , RO ;START WITH CH10 IN DIFF MODE
2652 062432 000402 BR 2$ ;AND START CHECK
2653 062434 012700 000020 1$: MOV #20 , RO ;START WITH CH20 IN SING END MODE
2654 062440 152777 000001 120176 2$: BISB #RBIT , @CSR ;CLEAR ANY INTERRUPTS
2655 062446 105777 121142 TSTB @LBYTE ;CLEAR RIF BIT
2656 062452 110077 121132 4$: MOVB RO , @STAT1 ;LOAD NON-EXISTANT CHANEL
2657 062456 132777 000004 121126 BITB #BIT2 , @STAT2 ;CHECK ERROR BIT SET
2658 062464 001002 BNE 5$ ;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$: INC RO ;UPDATE TO NEXT CHANEL
2667 062510 022700 000040 CMP #40 , RO ;CHECK FOR LAST CHANEL
2668 062514 001351 BNE 2$ ;CONTINUE IF NOT DONE
2669
2670 062516 012700 000020 ADGAN: MOV #20,RO ;GET FIRST GAIN VALUE
2671 062522 152777 000001 120114 1$: BISB #RBIT,@CSR ;CLEAR INTERRUPTS
2672 062530 105077 121054 CLRB @STAT1
2673 062534 105777 121054 TSTB @LBYTE
2674 062540 110077 121046 MOVB RO,@STAT2 ;WRITE GAIN
2675 062544 132777 000004 121040 BITB #BIT2,@STAT2 ;CHECK FOR ERROR
2676 062552 001002 BNE 2$ ;BRANCH IF ERROR
2677 062554 104045 EMT 45
2678 062556 000405 BR 3$
2679 062560 062700 000020 2$: ADD #20,RO ;GET NEXT GAIN
2680 062564 022700 000400 CMP #400,RO ;CHECK FOR LAST
2681 062570 001354 BNE 1$ ;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$
```



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								

2775  
2776  
2777  
2778  
2779  
2780 063244  
2781 063244 000004  
063246 012737 000027 001212  
2782 063254 012746 000000  
063260 012746 063266  
063264 000002  
063266 000240  
2783 063270 005737 003500  
2784 063274 001407  
2785 063276 005737 002664  
2786 063302 001003  
2787 063304 104056  
2788 063306 104401 031200  
2789 063312 000207  
2790 063314  
2791 063314 152777 000001 117322  
2792 063322 105777 120264  
2793 063326 152777 000010 117310  
2794 063334 004737 015410  
2795 063340 012777 064022 117302  
2796 063346 012777 000340 117276  
2797 063354 152777 000100 117262  
2798 063362 005004  
2799 063364 004737 065452  
2800 063370 005037 003442  
2801 063374 152777 000001 120210  
2802 063402 132737 000001 003442  
2803 063410 001774  
2804 063412 117737 120176 003572  
2805 063420 117737 120172 003573  
2806 063426 005737 003572  
2807 063432 001356  
2808 063434 005037 003556  
2809 063440 005037 003560  
2810 063444 005037 003562  
2811 063450 005037 003442  
2812 063454 152777 000001 120130  
2813 063462 132737 000001 003442  
2814 063470 001774  
2815 063472 117737 120116 003572  
2816 063500 117737 120112 003573  
2817 063506 013737 003572 003560  
2818 063514 013737 003572 001142  
2819 063522 023737 003556 003560  
2820 063530 001431  
2821 063532 005337 003560  
2822 063536 023737 003556 003560  
2823 063544 001416  
2824 063546 062737 000002 003560  
2825 063554 023737 003556 003560  
2826 063562 001401

```
.SBTTL A/D MONOTONICITY TEST
:*****
: MONTONICITY TEST
:*****
RUMP:
:*****
TST27: SCOPE
MOV #27,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
MCMV #PRO,-(SP) ;;SET PSW TO PRIORITY C
MOV #64$,-(SP)
RTI
64$: NOP
TST NORAMP
BEQ 1$
TST AFLAG
BNE 2$
EMT 56
TYPE ,MASS51
2$: RTS
1$: PC
BISB #RBIT,@CSR ;CLEAR DONE FLAG
TSTB @STAT2
BISB #TBIT,@CSR ;SET T BIT
JSR PC,CLRINT
RAMPST: MOV #INTR14,@VECTO ;SET UP INTERRUPT VECTOR
MOV #340,@VECTOA ;SET UP PRIORITY
BISB #EBIT,@CSR ;ENABLE INTERRUPTS
CLR R4 ;CLEAR ERROR POINTER
JSR PC,SETRAM ;SET RAMP
CLR IFLAG ;RESET INTERRUPT FLAG
1$: BISB #BIT0,@STAT2 ;START CONVERSION
2$: BITB #BIT0,IFLAG ;WAIT FOR INTERRUPT
BEQ 2$
MOV @LBYTE,DATALO ;WAIT FOR -10.240 VOLTS
MOV @HBYTE,DATALO+1
TST DATALO
BNE 1$
BEG: CLR BLAST ;STORE FIRST CONVERSION
CLR LAST
CLR LASTCN
STCO: CLR IFLAG ;RESET INTERRUPT FLAG
BISB #BIT0,@STAT2 ;START CONVERSION
13$: BITB #BIT0,IFLAG ;WAIT FOR INTERRUPT
BEQ 13$
MOV @LBYTE,DATALO
MOV @HBYTE,DATALO+1
MOV DATALO,LAST
MOV DATALO,$BDDAT
CMP BLAST,LAST
BEQ 3$
DEC LAST
2822: CMP BLAST,LAST ;COMPARE BLAST+1 WITH LAST
BEQ 2$
ADD #2,LAST
2825: CMP BLAST,LAST
BEQ 1$
```

```
2827 063564 000431          BR      4$          ;ERROR MESSAGE
2828 063566 005737 C03562    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$:    TST      LAST      ;ARE WE STILL AT MINUS 10 VOLTS ?
2839 063636 001304          BNE     STCO
2840 063640 005737 003562    TST     LASTCN    ;IS RAMP GOING UP ?
2841 063644 001701          BEQ     STCO
2842 063646 000431          BR      27$
2843 063650 013737 003556 001140 4$:    MOV     BLAST, $GDDAT ;CHECK FOR ERRORS
2844 063656 005737 003562    TST     LASTCN    ;GOOD DATA FOR ERROR MESSAGE
2845 063662 001003          BNE     7$
2846 063664 062737 000002 003556  ADD     #2, BLAST  ;IS RAMP GOING UP ?
2847 063672 005337 003556    7$:    DEC     BLAST      ;YES, DECREMENT LAST CONVERSION
2848 063676 013764 001140 002746  MOV     $GDDAT,RAMP1(R4) ;RAMP GOING DOWN
2849 063704 013764 001142 003066  MOV     $BDDAT,RAMP2(R4) ;ADJUST BLAST AFTER ERROR
2850 063712 013764 003562 003206  MOV     LASTCN,RAMP3(R4)
2851 063720 062704 000002    ADD     #2,R4      ;STORE RAMP SLOPE IN TABLE
2852 063724 022704 000050    CMP     #40.,R4   ;INCREMENT TABLE POINTER
2853 063730 001331          BNE     3$
2854 063732 005704          27$:   TST     R4        ;IS TABLE FULL
2855 063734 001425          BEQ     OUT       ;NO
2856 063736 104401 031047    TYPE   ,MASS55   ;ANY ERRORS ?
2857 063742 005003          CLR     R3        ;NO,EXIT
2858 063744 016337 002746 001140 28$:   MOV     RAMP1(R3),$GDDAT ;OUT OF RANGE BY TWO BITS
2859 063752 016337 003066 001142  MOV     RAMP2(R3),$BDDAT ;POINTER FOR ERROR TABLE
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$
2865 064000 020427 000050    CMP     R4,#40.  ;NO
2866 064004 001001          BNE     OUT      ;TOO MANY ERRORS ?
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 INTERRUPT HANDLER ***
2874 064022          INTR14:
2875 064022 010046          MOV     R0,-(SP)  ;SAVE R0
2876 064024 013737 002636 003446  MOV     K,ZLOOP
2877 064032 005037 003444 65$:   CLR     YLOOP    ;WAIT
2878 064036 005237 003444 64$:   INC     YLOOP
2879 064042 023727 003444 000001  CMP     YLOOP,#1
2880 064050 001372          BNE     64$
2881 064052 005337 003446    DEC     ZLOOP
2882 064056 001365          BNE     65$
2877 064060 117700 116562  MOV     @IAR,R0   ;GET ADDRESS OF MODULE
```

```

2878 064064 04200 177400      BIC    #177400,R0      ;CLEAR HIGH BYTE
2879 064070 063700 C02660      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$: .ASCIZ <CR><LF>/A014 RAMP TEST/
      66$:
2883 064132 104401 064140      TYPE   ,69$           ;;TYPE ASCIZ STRING
      064136 000424                BR     68$           ;;GET OVER THE ASCIZ
      ;;69$: .ASCIZ <CR><LF>/UNEXPECTED INTERRUPT FROM MODULE AT /
      68$:
2884 064210 010046                MOV    R0,-(SP)       ;ADDRESS
2885 064212 104402                TYPOC                ;TYPE IT
2886 064214 104401 001203      TYPE   ,SCLF
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$:  BISB   #RBIT,@CSR     ;CLEAR INTERRUPT
2892 064242 105777 117344      TSTB   @STAT2        ;THIS MODULE
2893 064246 013737 002636 003446      MOV    K,ZLOOP
      064254 005037 003444      CLR    YLOOP         ;WAIT
      064260 005237 003444      INC    YLOOP
      064264 023727 003444 000001      CMP    YLOOP,#1
      064272 001372                BNE   70$
      064274 005337 003446      DEC    ZLOOP
      064300 001365                BNE   71$
2894 064302 005237 003442      INC    IFLAG         ;SET INTERRUPT FLAG
2895 064306 012600                MOV    (SP)+,R0       ;RESTORE R0
2896 064310 000002                RTI

```

```

2898
2899
2900
2901 064312
2902
    064312 000004
    064314 012737 000030 001212
2903 064322 012737 000156 003456
2904 064330 013737 003610 003466
2905 064336 105077 117250
2906 064342 005037 001142
2907 064346 013746 000004
2908 064352 012737 065176 000004
2909 064360 113777 003450 117222
2910 064366 013737 003450 001140
2911 064374 117737 117210 001142
2912 064402 123737 001140 001142
2913 064410 001403
2914 064412 104060
2915 064414 000137 065164
2916 064420 113777 003450 117162 1$:
2917 064426 004737 017254
2918 064432 122737 000342 003606
2919 064440 001455
2920 064442 122737 000322 003606
2921 064450 001412
2922 064452 122737 000323 003606
2923 064460 001406
2924 064462 013737 003606 003600
2925 064470 104061
2926 064472 000137 065164
2927 064476 012705 000017 20$:
2928 064502 005004
2929 064504 012737 000040 001140
2930 064512 113777 003450 117070
2931 064520 113777 001140 117064 43$:
2932 064526 132777 000004 117056
2933 064534 001001
2934 064536 104062
2935 064540 004737 015526 116072 42$:
2936 064544 152777 000001
2937 064552 105777 117032
2938 064556 005237 001140
2939 064562 005204
2940 064564 022704 000020
2941 064570 001353
2942 064572 000402
2943 064574 012705 000037 2$:
2944 064600 010537 001140 21$:
2945 064604 053737 003450 001140
2946 064612 113777 001140 116770 4$:
2947 064620 117737 116764 001142
2948 064626 123737 001140 001142
2949 064634 001403
2950 064636 104063
2951 064640 004737 015526
2952 064644 005337 001140 3$:
    
```

.SBTTL LOGIC TEST OF A156 AND A157

MUX1:

\*\*\*\*\*

```

TST30: SCOPE
MOV #30,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
MOV #156,$MUT
MOV STAT1,TBADDR
CLRB @STAT2
CLR $BDDAT
MOV ERRVEC,-(SP) ;;PUSH ERRVEC ON STACK
MOV #30$,ERRVEC
MOVB MXNUM,@STAT1 ;SET MUX #
MCV MXNUM,$GDDAT
MOVB @STAT1,$BDDAT ;CHECK IF MUX RESPONDS
CMPB $GDDAT,$BDDAT
BEQ 1$
EMT 60
JMP 13$
MOV MXNUM,@STAT1 1$:
JSR PC,GCODE
CMPB #342,GBITE ;SE GENERIC CODE?
BEQ 2$
CMPB #322,GBITE ;DIF MODE A156
BEQ 20$
CMPB #323,GBITE ;A157?
BEQ 20$ ;YES, DIFFERENTIAL MODE
MOV GBITE,TEMP3
EMT 61
JMP 13$
MOV #17,R5 20$:
CLR R4
MOV #40,$GDDAT
MOVB MXNUM,@STAT1
MOV $GDDAT,@STAT2 43$:
BITB #BIT2,@STAT2 ;SET TOO HIGH CHANNEL
BNE 42$ ;CHECK IF ERROR SET
EMT 62
JSR PC,CNTRC
BISB #RBIT,@CSR ;CLEAR ERROR BIT
TSTB @STAT1
INC $GDDAT
INC R4
CMP #20,R4
BNE 43$
BR 21$
MOV #37,R5 2$:
MOV R5,$GDDAT ;CHECK ALL CHANNELS
BIS MXNUM,$GDDAT
MOVB $GDDAT,@STAT1 4$:
MOVB @STAT1,$BDDAT
CMPB $GDDAT,$BDDAT
BEQ 3$
EMT 63
JSR PC,CNTRC
DEC $GDDAT 3$:
    
```

```

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                    CLR      R1
2973 064756 156177 003352 116626 7$:      BISB    GAIN(R1),@STAT2
2974 064764 152777 000010 115652 22$:      BISB    #TBIT,@CSR
2975 064772 113777 003450 116610      MOVB    MXNUM,@STAT1
2976 065000 152777 000001 116604      BISB    #BIT0,@STAT2        ;START CONVERSION
2977 065006 005200                    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	022026	30\$:	CMP	(SP)+,(SP)+
3011	065200	104065		EMT	65
3012	065202	000770		BR	13\$

```
3014 .SBTTL DIGITAL MODULE TEST M6015
3015 ;THIS TEST CHECKS M6015 MODULE
3016 ;16-BIT RETENTIVE ISOLATED DC OUTPUT
3017
3018 065204 M6015:
3019 ;:*****
065204 000004 TST31: SCOPE
065206 012737 000031 001212 MOV #31,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
3020 065214 152777 000020 115422 BISB #DBIT,@CSR
3021 065222 013700 003462 MOV TADDR,RO
3022 065226 105020 CLR (RO)+
3023 065230 105010 CLR (RO) ;CLEAR TWO BYTES
3024 065232 005000 CLR RO
3025 065234 005001 CLR R1
3026 065236 005000 CLR RO
065240 005001 4$: CLR R1
065242 005037 003452 CLR DBUFF ;CLEAR IMAGE OF I/O REGISRERS
065246 005037 003454 CLR DBUFF+2
065252 116160 002630 003452 3$: MOVB PATT(R1),DBUFF(RO);SET DATA PATTERN 'N IMIGE
065260 013737 003462 003502 MOV TADDR,TEMP
065266 060037 003502 ADD RO,TEMP
065272 116177 002630 116202 MOVB PATT(R1),@TEMP ;SET DATA IN :/O REG
065300 005002 CLR R2
065302 013737 003462 003510 2$: 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 RO
065364 022700 000002 CMP #2,RO ;LAST BYTE?
065370 001323 BNE 4$ ;NO,TEST NEXT BYTE
3027 065372 013700 003462 MOV TADDR,RO ;REGISTER ADDRESS
3028 065376 012737 177777 001140 MOV #-1,$GDDAT ;ALL 1'S
3029 065404 012737 000002 003460 MOV #2,BYTNUM ;WRITE 2 BYTES
3030 065412 004737 015166 JSR PC,SETBYT ;GO WRITE IT
3031 065416 005037 001140 CLR $GDDAT ;CLEAR GDDAT
3032 065422 004737 014310 JSR PC,KLEER ;SET C BIT
3033 065426 112037 001142 MOVB (RO)+,$BDDAT ;LOW BYTE
3034 065432 111037 001142 MOVB (RO),$BDDAT ;HI BYTE
3035 065436 023737 001140 001142 CMP $GDDAT,$BDDAT ;ALL 0'S READ ?
3036 065444 001401 BEQ 5$ ;YES,BRANCH
3037 065446 104075 EMT 75 ;C-BIT BROKE
3038 065450 5$:
3039 065450 000207 RTS PC
3040
```

```
3042          .SBTTL  SUBROUTINES
3043
3044
3045
3046
3047 065452 022737 000321 003606 SETRAM: CMP      #321,GBITE
3048 065460 001004          BNE      1$
3049 065462 112777 000004 116120          MOVB   #4,@STAT1
3050 065470 000403          BR      2$
3051 065472 112777 000002 116110 1$:      MOVB   #2,@STAT1
3052 065500 000207          2$:      RTS      PC
3053
3054          ;      THIS SUBROUTINE DOES 8 CONVERSIONS AND STORES AVERAGE IN INTDAT
3055 065502          ;CONV7:
3056 065502 117737 116102 003620 3$:      MOVB   @STAT1,ST1SAV          ;SAVE STAT1
3057 065510 117737 116076 003622          MOVB   @STAT2,ST2SAV          ;SAVE STAT2
3058 065516 005037 003512          CLR      INTDAT
3059 065522 005037 003604          CLR      TEMP5
3060 065526 152777 000001 116056 2$:      BISB   #BIT0,@STAT2          ;START CONVERSION
3061 065534 132777 000002 116050 1$:      BITB   #BIT1,@STAT2          ;WAIT FOR CONVERSION DONE
3062 065542 001774          BEQ      1$
3063 065544 152777 000001 115072          BISB   #RBIT,@CSR
3064 065552 117737 116036 003502          MOVB   @LBYTE,TEMP          ;ASSEMBLE DATA
3065 065560 117737 116032 003503          MOVB   @HBYTE,TEMP+1
3066 065566 042737 170000 003502          BIC      #170000,TEMP
3067 065574 063737 003502 003512          ADD     TEMP,INTDAT          ;ADD TO PREVIOUS DATA
3068 065602 005237 003604          INC     TEMP5
3069 065606 113777 003620 115774          MOVB   ST1SAV,@STAT1
3070 065614 113777 003622 115770          MOVB   ST2SAV,@STAT2
3071 065622 022737 000010 003604          CMP     #10,TEMP5          ;IS IT LAST ONE
3072 065630 001336          BNE     2$          ;NO
3073 065632 013737 003512 003654          MOV     INTDAT,AVRSAV          ;SAVE TOTAL
3074 065640 006237 003512          ASR     INTDAT          ;FIND AVERAGE
3075 065644 006237 003512          ASR     INTDAT
3076 065650 006237 003512          ASR     INTDAT
3077 065654 042737 170000 003512          BIC     #170000,INTDAT
3078 065662 013737 003512 001142          MOV     INTDAT,$BDDAT
3079 065670 000207          RTS      PC
3080
```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61

.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

```
A631: JSR PC,A631AD ;SET UP ADDRESSES  
TST AFLAG ;UNDER AUTO MODE?  
BEQ A631MN ;NO,BRANCH  
JSR PC,KLEER ;CLEAR D-BUS  
JSR PC,A631L ;DO LOGIC TESTS  
RTS PC ;RETURN  
  
A631MN: MOV #1100,SP ;SET STACK  
TYPTXT <<CR><LF>/A631 12-BIT DAC MONITOR (H-HELP)/<CR><LF>>  
MOV #A631MN,RETURN ;ADDRESS FOR CONTROL 'A'  
  
1$: TYPTXT </A631:/>  
RDCHR ;GET COMMAND  
MOV (SP)+,200$ ;SAVE COMMAND  
CMP #3,200$ ;CONTROL 'C' ?  
BNE 3$ ;NO,BRANCH  
JMP MONIT ;RETURN TO MONITOR  
3$: CLR RO ;CLEAR RO USED AS INDEX  
;***** FIND COMMAND *****  
4$: CMP 300$(RO),200$ ;A MATCH ?  
BEQ 5$ ;YES, BRANCH  
ADD #2,RO ;ADVANCE POINTER  
TST 300$(RO) ;END OF TABLE ?  
BNE 4$ ;NO, BRANCH  
TYPTXT <<15><12> /ILLEGAL COMMAND/<7>>  
BR 1$ ;GO TRY AGAIN  
  
;***** EXECUTE COMMAND *****  
5$: MOV 500$(RO),6$ ;GET READY TO TYPE  
TYPE ;THE COMMAND  
6$: 0 ;THE MESSAGE TO TYPE  
TYPE , $CRLF ;TYPE A CR THEN LF  
CLR $PASS ;CLEAR PASS COUNT  
7$: MOV RO,6$ ;SAVE INDEX FOR LOOP ON TEST  
9$: JSR PC,@400$(RO) ;GO DO THE TEST  
INC $PASS ;INCREMENT PASS COUNT  
MOV 6$,RO ;RESTORE INDEX
```

```

62 066164 023;37 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
76 066234 000000                      200$:  ;***** VARIABLE FOR ECHOING *****
                                           .WORD  0
                                           ;

```

```
78
79
80 066236 000110      300$:
81 066240 000114      .WORD 'H           ;HELP FILE
82 066242 000123      .WORD 'L           ;DAC LOGIC TEST
83 066244 000103      .WORD 'S           ;DAC-SPOPE ROUTINE
84 066246 000115      .WORD 'C           ;CALIBRATION ROUTINE
85 066250 000000      .WORD 'M           ;MANUAL FIELD OPERATION
86                                     .WORD 0            ;END OF 300$ TABLE
87
88 066252 066446      400$:
89 066254 066636      .WORD A631H        ;HELP FILE
90 066256 067766      .WORD A631L        ;DAC LOGIC TEST
91 066260 071526      .WORD A631S        ;DAC-SCOPE
92 066262 070600      .WORD A631C        ;CALIBRATION ROUTINE
93                                     .WORD A631M        ;MANUAL FIELD OPERATION
94
95 066264 066300      500$:
96 066266 066315      .WORD 600$         ;HELP FILE
97 066270 066337      .WORD 606$         ;DAC LOGIC TEST
98 066272 066364      .WORD 617$         ;DAC-SCOPE ROUTINE
99 066274 066413      .WORD 609$         ;CALIBRATION ROUTINE
100 066276 000000     .WORD 616$         ;MANUAL FIELD OPERATION
101                                     .WORD 0            ;END OF TABLE 500$ MARKER
102
103 066300      040      110      105 600$:
104 066315      040      104      101 606$:
105 066337      040      104      101 617$:
106 066364      040      103      101 609$:
107 066413      040      115      101 616$:
108                                     .ASCIZ / HELP FILE/<15><12>
109                                     .ASCIZ / DAC LOGIC TEST/<15><12>
110                                     .ASCIZ / DAC-SCOPE ROUTINE/<15><12>
111                                     .ASCIZ / CALIBRATION ROUTINE/<15><12>
112                                     .ASCIZ / MANUAL FIELD OPERATION/<15><12>
113                                     .EVEN
114
115 066446 104401 066460      A631H:
116 066452 005726
117 066454 000137 065716
118 066460      050      114      051 1$:
119 066504      050      115      051
120 066532      050      123      051
121 066555      050      103      051
122                                     .ASCIZ /((L)-DAC LOGIC TEST/<CR><LF>
123                                     .ASCIZ /((M)-MANUAL OPERATION/<CR><LF>
124                                     .ASCIZ /((S)-SCOPE ROUTINE/<CR><LF>
125                                     .ASCIZ /((C)-CALIBRATION PROCEDURE/<CR><LF>
126                                     .EVEN
```

119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166

```
.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
:   ADO-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 $BDADR=ADO TO AD7 DO
:   : READ [$BDADR]
: END (FOR)
END (DAC12)
DAC12 - TIMOUT HANDLER
%

TMOT12: MOV     (SP)+,BTEMP     ;SAVE PC
        MOV     (SP)+,CTEMP     ;SAVE PSW
        EMT     214
        MOV     CTEMP,-(SP)     ;RESTORE PSW
        MOV     BTEMP,-(SP)     ;RESTORE PC
        RTI

A631L:
DAC12:
*****
TST32: SCOPE
        MOV     #32,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
        BISB    #DBIT,@CSR     ;SET DBIT
        JSR     PC,CNTRC       ;EXIT IF CONTROL C
        MOV     #TMOT12,ERRVEC ;SET UP TIMEOUT VECTOR
        MOV     #340,ERRVEC+2
        MOV     ADO,$BDADR     ;SET UP FOR LOOP
        DEC     $BDADR
211$:  INC     $BDADR           ;ADVANCE ADDRESS
        CMP     AD7,$BDADR     ;DONE LOOP ?
        BLO     DAC13         ;YES,_BRANCH
        JSR     PC,CNTRC       ;EXIT IF CONTROL C
        TSTB    @$BDADR        ;READ THE ADDRESS
        BR      211$          ;END (FOR)
*****
```

```
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189 066734      101      066      063      EM215: .ASCIZ /A631 ERROR -TIMEOUT WHEN WRITING ADDRESS/
193
194
195 067006      000000   000000   000000   DF1:    .WORD 0,0,0,0,0,0,0,0,
196
197 067030      012637   003412   TMOT13: MOV      (SP)+,BTEMP      ;SAVE PC
198 067034      012637   003414   MOV      (SP)+,CTEMP      ;SAVE PSW
199 067040      104215   EMT      215
200 067042      013746   003414   MOV      CTEMP,-(SP)      ;RESTORE PSW
201 067046      013746   003412   MOV      BTEMP,-(SP)      ;RESTORE PC
202 067052      000002   RTI
203
204 067054
DAC13:
*****
TST33: SCOPE
067054 000004
067056 012737 000033 001212 MOV      #33,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
205 067064 152777 000020 113552 BISB     #DBIT,@CSR          ;SET DBIT
206 067072 004737 015526 JSR      PC,CNTRC         ;EXIT IF CONTROL C
207 067076 012737 067030 000004 MOV      #TMOT13,ERRVEC  ;SET UP TIMEOUT VECTOR
208 067104 012737 000340 000006 MOV      #340,ERRVEC+2   ;
209 067112 013737 003416 001136 MOV      ADO,$BDADR      ;SET UP FOR LOOP
210 067120 005337 001136 DEC      $BDADR          ;
211 067124 005237 001136 211$: INC      $BDADR          ;ADVANCE ADDRESS
212 067130 023737 003434 001136 CMP      AD7,$BDADR      ;DONE LOOP ?
213 067136 103425 BLO     DAC14            ;YES, BRANCH
214 067140 004737 015526 JSR      PC,CNTRC         ;EXIT IF CONTROL C
215 067144 105077 111766 CLR      @BDADR          ;WRITE THE ADDRESS
216 067150 000765 BR      211$            ;END (FOR)
217
```



219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
266  
267  
268

```
*****  
: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)  
: ADO-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  
: : : [ADO]=M[0]  
: : : [AD1]=M[1]  
: : : [AD2]=M[2]  
: : : [AD3]=M[3]  
: : : [AD4]=M[4]  
: : : [AD5]=M[5]  
: : : [AD6]=M[6]  
: : : [AD7]=M[7]  
230$: : : FOR X=0 TO 7 DO  
: : : : IF [ADO+X]<>M[X] THEN REPORT ERROR  
: : : : END (FOR)  
240$: : : 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)  
%  
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,
```

270  
271 067212  
067212 000004  
067214 012737 000034 001212  
272 067222 152777 000020 113414  
273 067230 012737 076146 000004  
274 067236 005037 067542  
275 067242 005037 067544  
276 067246 005037 067546  
277 067252 005037 067550  
278 067256 012737 000001 067542  
279 067264 005002  
280 067266 005001  
281  
282 067270 004737 015526  
283 067274 113777 067542 114114  
284 067302 113777 067543 114110  
285 067310 113777 067544 114104  
286 067316 113777 067545 114100  
287 067324 113777 067546 114074  
288 067332 113777 067547 114070  
289 067340 113777 067550 114064  
290 067346 113777 067551 114060  
291  
292 067354 012700 177777  
293 067360 005200  
294 067362 022700 000007  
295 067366 103426  
296 067370 004737 015526  
297 067374 013737 003416 003410  
298 067402 060037 003410  
299 067406 117737 113776 001142  
300 067414 123760 001142 067542  
301 067422 001756  
302 067424 113737 003410 001136  
303 067432 116037 067542 001140  
304 067440 104216  
305 067442 000746  
306  
307 067444 000241  
308 067446 006137 067542  
309 067452 006137 067544  
310 067456 006137 067546  
311 067462 006137 067550  
312 067466 103003  
313 067470 052737 000001 067542  
314 067476 005201  
315 067500 022701 000100  
316 067504 001271  
317 067506 005202  
318  
319 067510 005137 067542  
320 067514 005137 067544  
321 067520 005137 067546  
322 067524 005137 067550  
323 067530 022702 000002

;DAC14 - RAM MEMORY TEST

DAC14:

\*\*\*\*\*

TST34: SCOPE  
MOV #34,\$TESTN ;;SET TEST NUMBER IN APT MAIL BOX  
BISB #DBIT,@CSR ;;SET DBIT  
MOV #TIMOUT,ERRVEC ;;RESET TIMEOUT VECTOR  
CLR 10\$ ;;-CLEAR 4 WORDS  
CLR 10\$+2  
CLR 10\$+4  
CLR 10\$+6  
MOV #1,10\$ ;--M[0]=1  
CLR R2 ;--KEY=0  
210\$: CLR R1 ;--COUNT=0  
220\$: JSR PC,CNTRC ;EXIT IF CONTROL C  
MOVB 10\$,@AD0 ;---[AD0]=M[0]  
MOVB 10\$+1,@AD1 ;---[AD1]=M[1]  
MOVB 10\$+2,@AD2 ;---[AD2]=M[2]  
MOVB 10\$+3,@AD3 ;---[AD3]=M[3]  
MOVB 10\$+4,@AD4 ;---[AD4]=M[4]  
MOVB 10\$+5,@AD5 ;---[AD5]=M[5]  
MOVB 10\$+6,@AD6 ;---[AD6]=M[6]  
MOVB 10\$+7,@AD7 ;---[AD7]=M[7]  
MOV #-1,R0 ;---FOR X=0 TO 7 DO  
231\$: INC R0  
CMP #7,R0 ;END OF LOOP ?  
BLO 240\$ ;YES, BRANCH  
JSR PC,CNTRC ;EXIT IF CONTROL C  
MOV AD0,ATEMP ;---IF [AD0+X]<>M[X] THEN REPORT ERROR  
ADD R0,ATEMP  
MOVB @ATEMP,\$BDDAT ;SAVE DATA  
CMPB \$BDDAT,10\$(R0) ;CORRECT DATA ?  
BEQ 231\$ ;YES, BRANCH  
MOVB ATEMP,\$BDADR  
MOVB 10\$(R0),\$GDDAT  
EMT 216  
BR 231\$ ;---END (FOR)  
240\$: CLC ;---ROTATE NUMBER BY ONE  
ROL 10\$  
ROL 10\$+2  
ROL 10\$+4  
ROL 10\$+6  
BCC 241\$ ;WRAP AROUND BIT ?  
313: BIS #BIT0,10\$ ;YES, DO IT  
241\$: INC R1 ;---COUNT=COUNT+1  
CMP #64.,R1 ;COUNT=64. ?  
BNE 220\$ ;NO, BRANCH  
INC R2 ;--KEY=KEY+1  
319: COM 10\$ ;---COMPLIMENT 64. BIT WORD  
320: COM 10\$+2  
321: COM 10\$+4  
322: COM 10\$+6  
323: CMP #2,R2 ;KEY=2 ?

324 067534 001254  
325 067536 000137 C37616  
326  
327 067542

BNE 210\$  
JMP DAC15

10\$: .BLKW 4

;NO, BRANCH  
;DO NEXT TEST

; 'M' THE 64 BIT WORD

329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380

```
*****  
: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.  
:   ADO-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=ADO TO AD7 DO  
:   $BDDAT=[$BDADR]  
:   IF $BDDAT<>#262 THEN REPORT ERROR  
: END (FOR)  
: CLR GBIT  
END (DAC15 - GENERIC CODE TEST)  
%  
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,
```

```
DAC15:  
:*****  
TST35: SCOPE  
MOV #35,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX  
BISB #DBIT,@CSR ;SET DBIT  
JSR PC,CNTRC ;EXIT IF CONTROL C  
MOVB #CBIT,@CSR ;CLEAR THE SUBSYSTEM  
MOV #1000,ATEMP ;SET UP COUNT  
201$: DEC ATEMP  
: BNE 201$  
BISB #GBIT,@CSR ;SET THE GBIT  
MOV ADO,$BDADR ;SET UP FOR  
DEC $BDADR  
CLR $BDDAT  
211$: INC $BDADR ;MAKE NEXT ADDRESS  
CMP AD7,$BDADR ;END OF LOOP ?  
BLO 220$ ;YES, BRANCH  
JSR PC,CNTRC ;EXIT IF CONTROL C  
MOVB @ $BDADR,$BDDAT ;SAVE THE CODE  
CMPB #262,$BDDAT ;CORRECT CODE ?  
BEQ 211$ ;YES, BRANCH  
MOV #262,$GDDAT  
EMT 217  
BR 211$ ;END (FOR)  
220$: BICB #GBIT,@CSR ;CLEAR THE GBIT  
221$: RTS PC
```

382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424

.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

\*\*\*\*\*

```
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
  
A631S: .REM %  
: 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*)  
%
```

455  
456  
457  
458  
459  
460  
461  
462  
463  
468 067766 005001  
469 067770  
470 070004 010146  
471 070006 006216  
472 070010 104403  
473 070012 001  
474 070013 000  
475 070014  
476 070044 104406  
477 070046 122726 000131  
478 070052 001067  
479 070054  
480 070066 104401 001203  
481 070072  
482 070124 104410  
483 070126 012661 070540  
484 070132  
485 070164 104410  
486 070166 012661 070550  
487 070172  
488 070222 104410  
489 070224 012661 070560  
490 070230 000405  
491  
492  
493  
494 070232  
495 070244 004737 015526  
496 070250 062701 000002  
497 070254 020127 000006  
498 070260 101002  
499 070262 000137 067770  
500

```
:-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]  
A631S: CLR R1  
210$: TYPTXT <<15><12> /DAC/>  
MOV R1, -(SP) ;TYPE NUMBER  
ASR (SP) ;DIVIDE BY 2  
TYPOS ;ONE DIGIT  
.BYTE 1  
.BYTE 0 ;SUPPRESS ZEROS  
TYPTXT <% PARAMETERSE[Y/N]:%>  
RDCHR ;GET RESPONSE  
CMPB #'Y, (SP)+ ;IS IT YES ?  
BNE 220$ ;NO, BRANCH  
TYPTXT </YES/>  
TYPE , $CRLF  
TYPTXT </LOWER LIMIT(OCTAL):/>  
RDOCT  
MOV (SP)+, SCLL(R1) ;STORE VALUE  
TYPTXT </UPPER LIMIT(OCTAL):/>  
RDOCT  
MOV (SP)+, SCHL(R1) ;STORE VALUE  
TYPTXT </STEP SIZE(OCTAL):/>  
RDOCT  
MOV (SP)+, SCSTP(R1);STORE VALUE  
BR 221$  
:---R1=R1+2  
:--UNTIL (R1>6)  
220$: TYPTXT </NO/>  
221$: JSR PC, CNTRC ;EXIT IF CONTROL C  
ADD #2, R1 ;ADVANCE THE DAC #'S  
CMP R1, #6 ;R1>6 ?  
BHI 230$ ;YES, BRANCH  
JMP 210$ ;NO, JUMP
```

```
502 :--R1=0
503 :--EXECUTE[Y/N] ?
504 :-UNTIL (YES)
505 070266 005001 230$: CLR R1 ;RESET TO ZERO
506 070270 TYPTXT <<15><12> %EXECUTE [Y/N]:%>
507 070320 104406 RDCHR
508 070322 122726 000131 CMPB #'Y,(SP)+ ;YES ?
509 070326 001413 BEQ 240$ ;YES, BRANCH
510 070330 004737 015526 JSR PC,CNTRC ;EXIT IF CONTROL C
511 070334 TYPTXT </NO/>
512 070346 104401 001203 TYPE $CRLF
513 070352 000137 067770 JMP 210$ ;
514
515 :-INITAILIZE THE CURRENT COUNT
516 070356 013737 070540 070570 240$: MOV SCLL+0,SCCUR+0 ;DAC0 START
517 070364 013737 070542 070572 MOV SCLL+2,SCCUR+2 ;DAC1 START
518 070372 013737 070544 070574 MOV SCLL+4,SCCUR+4 ;DAC2 START
519 070400 013737 070546 070576 MOV SCLL+6,SCCUR+6 ;DAC3 START
520
521 :-REPEAT
522 :--FOR R1=0 TO 6 BY 2 DO
523 :---SCCUR[R1]=SCCUR[R1]+SCSTP[R1]
524 :---IF SCCUR[R1]>SCHL[R1]
525 :----THEN (*ADJUST VALUE*)
526 :-----SCCUR[R1]=SCLL[R1]
527 :---LOAD DAC[R1]
528 :---END (*FOR*)
529 :-UNTIL (^C) OR (^A)
530 070406 250$: TYPTXT </YES/>
531 070420 104401 001203 TYPE $CRLF ;
532 070424 013737 003416 003410 260$: MOV ADO,ATEMP ;SET UP ADDRESS
533 070432 013737 003420 003412 MOV AD1,BTEMP ;SET UP ADDRESS
534 070440 012701 177776 MOV #-2,R1 ;SET UP COUNT
535 070444 062701 000002 261$: ADD #2,R1 ;
536 070450 022701 000006 CMP #6,R1 ;END OF LOOP ?
537 070454 103763 BLO 260$ ;YES, BRANCH
538 070456 116177 070571 112726 MOVB SCCUR+1(R1),@BTEMP;LOAD DAC(R1)
539 070464 116177 070570 112716 MOVB SCCUR(R1),@ATEMP ;
540 070472 066161 070560 070570 ADD SCSTP(R1),SCCUR(R1);SET UP VALUE
541 070500 026161 070570 070550 CMP SCCUR(R1),SCHL(R1); ABOVE LIMIT ?
542 070506 101403 BLOS 262$ ;NO, BRANCH
543 070510 016161 070540 070570 MOV SCLL(R1),SCCUR(R1);RESET VALUE
544 070516 062737 000002 003410 262$: ADD #2,ATEMP ;MAKE NEXT ADDRESS
545 070524 062737 000002 003412 ADD #2,BTEMP ;MAKE NEXT ADDRESS
546 070532 004737 015526 JSR PC,CNTRC ;EXIT IF CONTROL C
547 070536 000742 BR 261$
548
549
550 070540 SCLL: .BLKW 4 ;LOW LIMIT FOR EACH DAC
551 070550 SCHL: .BLKW 4 ;UPPER LIMIT FOR EACH DAC
552 070560 SCSTP: .BLKW 4 ;STEP SIZE FOR EACH DAC
553 070570 SCCUR: .BLKW 4 ;CURRENT VALUE FOR EACH DAC
```



555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610

```
.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 REPOSE 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(CCHARACTER)
:       :       :       : IF CNTRLC THEN EXIT
270$:   :       :       : WHILE (CH=OCTAL_VALUE) DO
:       :       :       :   : FLAG=1
:       :       :       :   : NUMB=CH+(NUMB * 8)
:       :       :       :   : GET(CCHARACTER)
:       :       :       :   : IF CNTRLC THEN EXIT
:       :       :       :   : END (WHILE)
280$:   :       :       : 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)
%
```

```
612                                     :-REPEAT
613                                     :--FOR DACX=0 TO 3 DO
614                                     :---REPEAT
615                                     :----CASE DACX OF
616 070600 000400 A631M: BR 202$ ;DON'T INCREMENT YET
617 070602 200$:
618 070602 012701 177777 202$: MOV #-1,R1 ;SET UP FOR LOOP
619 070606 005201 201$: INC R1 ;ADVANCE COUNT
620 070610 022701 000003 201$: CMP #3,R1 ;EXIT LOOP ?
621 070614 103772 BLO 200$ ;YES, BRANCH
622 070616 022701 000000 210$: CMP #0,R1 ;DACX=0 ?
623 070622 001413 BEQ 220$ ;YES, BRANCH
624 070624 022701 000001 CMP #1,R1 ;DACX=1 ?
625 070630 001432 BEQ 230$ ;YES, BRANCH
626 070632 022701 000002 CMP #2,R1 ;DACX=2 ?
627 070636 001451 BEQ 240$ ;YES, BRANCH
628 070640 022701 000003 CMP #3,R1 ;DACX=3 ?
629 070644 001470 BEQ 250$ ;YES, BRANCH
630 070646 000000 HALT ;R1 CAN'T BE THAT HIGH
631 070650 000776 BR -2 ;FIX IT
632 070652 220$: TYPTXT <<15><12> /DAC0 [OCTAL]: />
633 070702 117746 112510 MOVB @AD0,-(SP) ;
634 070706 117766 112506 000001 MOVB @AD1,1(SP) ;
635 070714 000465 BR 251$ ;
636 070716 230$: TYPTXT <<15><12> /DAC1 [OCTAL]: />
637 070746 117746 112450 MOVB @AD2,-(SP) ;
638 070752 117766 112446 000001 MOVB @AD3,1(SP) ;
639 070760 000443 BR 251$ ;
640 070762 240$: TYPTXT <<15><12> /DAC2 [OCTAL]: />
641 071012 117746 112410 MOVB @AD4,-(SP) ;
642 071016 117766 112406 000001 MOVB @AD5,1(SP) ;
643 071024 000421 BR 251$ ;
644 071026 250$: TYPTXT <<15><12> /DAC3 [OCTAL]: />
645 071056 117746 112350 MOVB @AD6,-(SP) ;
646 071062 117766 112346 000001 MOVB @AD7,1(SP) ;
647 071070 104403 251$: TYPOS
648 071072 004 .BYTE 4 ;FOUR DIGITS
649 071073 001 .BYTE 1 ;DON'T SUPPRESS
650 071074 TYPTXT </ />
```

```
652  
653  
654  
655  
656  
657 071104 005002  
658 071106 005000  
659 071110 104406  
660 071112 012603  
661 071114 042703 000200  
662 071120 004737 015526  
663 071124 110337 071136  
664 071130 104401 071136  
665 071134 000401  
666 071136 000000  
667  
668  
669  
670  
671  
672  
673  
674 071140 022703 000060  
675 071144 101030  
676 071146 022703 000067  
677 071152 103425  
678 071154 042703 177770  
679 071160 012700 000001  
680 071164 006302  
681 071166 006302  
682 071170 006302  
683 071172 060302  
684 071174 104406  
685 071176 012603  
686 071200 042703 000200  
687 071204 004737 015526  
688 071210 110337 071222  
689 071214 104401 071222  
690 071220 000401  
691 071222 000000  
692 071224 000745  
693  
694  
695 071226 020227 007777  
696 071232 101423  
697 071234  
698 071276 000137 070616
```

```
      :---NUMB=0  
      :---FLAG=0  
      :---GET(CCHARACTER)  
      :---IF CNTRLC THEN EXIT  
260$: CLR R2 ;  
      CLR R0 ;  
      RDCHR ;  
      MOV (SP)+,R3 ;SAVE CHARACTER  
      BIC #BIT7,R3 ;STRIP PARITY BIT  
      JSR PC,CNTRC ;EXIT IF CONTROL C  
      MOV R3,262$ ;STORE CHAR  
      TYPE ,262$ ;  
      BR 270$ ;  
262$: .WORD 0 ;CHAR STORAGE  
  
      :---WHILE (CH=OCTAL_VALUE) DO  
      :---FLAG=1  
      :---NUMB=CH+(NUMB * 8)  
      :---GET(CCHARACTER)  
      :---IF CNTRLC THEN EXIT  
      :---END (WHILE)  
270$: CMP #60,R3 ;>59 ?  
      BHI 280$ ;NO, BRANCH  
      CMP #67,R3 ;<70 ?  
      BLO 280$ ;NO, BRANCH  
      BIC #177770,R3 ;STRIP UNNECESSARY GARBAGE  
      MOV #1,R0 ;SET THE FLAG  
      ASL R2 ;X8  
      ASL R2 ;  
      ASL R2 ;  
      ADD R3,R2 ;CH+  
      RDCHR ;  
      MOV (SP)+,R3 ;  
      BIC #BIT7,R3 ;STRIP PARITY BIT  
      JSR PC,CNTRC ;EXIT IS CONTROL C  
      MOV R3,272$ ;PREPARE TO PRINT  
      TYPE ,272$ ;  
      BR 273$ ;  
272$: .WORD 0 ;USED TO EHCO CHAR  
273$: BR 270$ ;  
  
      :---UNTIL (NUMB <= 7777)  
280$: CMP R2,#7777 ;R2<=7777 ?  
      BLOS 281$ ;YES, BRANCH  
      TYPTXT <<15><12> /*** NUMBER TOO LARGE ***/>  
      JMP 210$ ;TRY AGAIN
```

```
700
701
702      :----IF FLAG=1
703      :----THEN /LOAD THE DAC/
704      :----SET UP DAC WORD
705      :----CASE DACX OF
706      :--UNTIL (CONTROL C)
706 071302 010237 071456      281$: MOV R2,20$      ;FOR BYTE OPERATIONS
707 071306 005700      TST R0      ;FLAG=0 ?
708 071310 001453      BEQ 321$      ;YES, BRANCH
709 071312 005000      CLR R0      ;CLEAR IT
710 071314 022701 000000      CMP #0,R1      ;DAC0 ?
711 071320 001413      BEQ 290$      ;YES, BRANCH
712 071322 022701 000001      CMP #1,R1      ;DAC1 ?
713 071326 001420      BEQ 300$      ;YES, BRANCH
714 071330 022701 000002      CMP #2,R1      ;DAC2 ?
715 071334 001424      BEQ 310$      ;YES, BRANCH
716 071336 022701 000003      CMP #3,R1      ;DAC3 ?
717 071342 001430      BEQ 320$      ;YES, BRANCH
718 071344 000000      HALT      ;R1 CAN'T BE THAT HIGH
719 071346 000776      BR -2      ;FIX IT
720 071350 113777 071457 112042 290$: MOVB 20$+1,@AD1      ;SET UP HIGH BYTE
721 071356 113777 071456 112032      MOVB 20$,@AD0      ;LOAD THE WORD
722 071364 000137 071440      JMP 321$      ;
723 071370 113777 071457 112026 300$: MOVB 20$+1,@AD3      ;SET UP HIGH BYTE
724 071376 113777 071456 112016      MOVB 20$,@AD2      ;LOAD THE WORD
725 071404 000415      BR 321$      ;
726 071406 113777 071457 112014 310$: MOVB 20$+1,@AD5      ;SET UP HIGH BYTE
727 071414 113777 071456 112004      MOVB 20$,@AD4      ;LOAD THE WORD
728 071422 000406      BR 321$      ;
729 071424 113777 071457 112002 320$: MOVB 20$+1,@AD7      ;SET UP HIGH BYTE
730 071432 113777 071456 111772      MOVB 20$,@AD6      ;LOAD THE WORD
731 071440 022701 000003      321$: CMP #3,R1      ;DAC3 ?
732 071444 001002      BNE 322$      ;NO, BRANCH
733 071446 104401 001203      TYPE ,SRLF      ;
734 071452 000137 070606      322$: JMP 201$      ;
735
736
737 071456 000000      20$:      ;***** VARIABLE STORAGE *****
      .WORD 0      ;TEMPORARY STORAGE
```

739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766

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

```
A631AD: CLR      R1  
        MOV      TADDR,R0      ;A631 ADDRESS  
1$:     CMP      R1,#16.      ;LAST ADDRESS ?  
        BEQ      2$          ;YES,BRANCH  
        MOV      R0,ADC(R1)    ;SET ADDRESS  
        INC      R0          ;ADD 1 TO ADDRESS  
        INC      R1  
        INC      R1  
        BR       1$          ;DO IT AGAIN  
2$:     RTS      PC
```

```
*****  
:THIS ROUTINE WAITS FOR A KEYBOARD RESPONSE.  
*****
```

```
WAIT:   TSTB    @$TKS      ;KEY STRUCK ?  
        BPL     WAIT      ;NO,BRANCH  
        JSR    PC,CNTRC  
        RTS    PC
```

768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812

.SBTTL A631 CALIBRATION ROUTINE

```
*****  
: A631 12 BIT DAC CALIBRATION ROUTINE  
:*****
```

```
.REM *  
A631C: A631 CALIBRATION ROUTINE  
: ASK "VOLTAGE OR CURRENT?"  
: IF ANSWER = VOLTAGE  
: : THEN  
: : : PRINT "SET E66 TO 11010100"  
: : : FOR RO = 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 [RO] = 0000  
: : : : REPEAT UNTIL KEYSTRUCK  
: : : : PRINT "SET EDC TO 10.2375"  
: : : : PRINT "ADJUST POT R XX UNTIL METER NULLS"  
: : : : LOAD DAC [RO] WITH 7777  
: : : : REPEAT UNTIL KEYSTRUCK  
: : : : END (FOR)  
: IF ANSWER = CURRENT  
: : THEN  
: : : PRINT "SET E66 TO 10101010"  
: : : PRINT "SET EDC TO NULL MODE"  
: : : FOR RO = 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[RO] 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[RO] WITH 7777  
: : : : REPEAT UNTIL KEYSTRUCK  
: : : : END (FOR)  
: END (A631 CALIBRATION)  
*
```

```
814  
815  
816  
817  
818  
819  
820 071526  
821 071606 104401 001203  
822  
823 071612 000240  
824 071614  
825 071660 104406  
826 071662 012600  
827 071664 020027 000126  
828 071670 001410  
829 071672 020027 000103  
830 071676 001002  
831 071700 000137 072626  
832 071704 004737 015526  
833 071710 000741  
834  
835 071712  
836 071732 104401 073622  
837 071736 104401 073703  
838 071742 104401 073747  
839 071746 104406  
840 071750 005726  
841 071752 104401 001203  
842 071756 104401 073410  
843 071762 104401 001203  
844 071766 012700 177777  
845 071772 005200  
846 071774 020027 000003  
847 072000 101305  
848 072002  
849 072046 010046  
850 072050 104403  
851 072052 001  
852 072053 000  
853 072054 104401 001203  
854 072060  
855 072112 104401 001203  
856 072116  
857 072164 010001  
858 072166 006301  
859 072170 016137 074412 072200  
860 072176 104401  
861 072200 000000  
862 072202 104401 001203  
863 072206 104401 073747  
864 072212 104406  
865 072214 005726  
866 072216 104401 001203  
867 072222 104401 001203  
868 072226  
869 072324  
870 072414
```

```
*****  
:A631 FIELD SERVICE CALIBRATION ROUTINE  
*****  
A631C: TYPTXT </A631 FIELD SERVICE CALIBRATION PROCEDURE />  
TYPE , $CRLF  
5$: NOP  
S5: TYPTXT <<CR><LF>*VOLTAGE OR CURRENT [V/C] ?*>  
RDCHR ;GET RESPONSE  
MOV (SP)+,R0 ;SAVE IT  
CMP R0,#'V ;VOLTAGE ?  
BEQ 200$ ;YES, BRANCH  
CMP R0,#'C ;CURRENT ?  
BNE 8$ ;NO, BRANCH  
JMP 300$ ;DO CURRENT CALIBRATION  
8$: JSR PC,CNTRC ;CHECK FOR EXIT  
BR S5 ;TRY AGAIN  
200$: TYPTXT </VOLTAGE/<CR><LF>>  
TYPE ,411$ ;"SET E66 TO VOLTAGE MODE  
TYPE ,412$ ;-  
TYPE ,414$ ;"HIT A KEY WHEN READY"  
RDCHR  
TST (SP)+ ;CLEAR STACK  
TYPE , $CRLF  
TYPE ,410$ ;SET EDC TO NULL MODE  
TYPE , $CRLF  
MOV #-1,R0 ;SET UP FOR LOOP  
205$: INC R0 ;BUMP FOR LOOP  
CMP R0,#3 ;LAST DAC ?  
BHI S5 ;YES, BRANCH  
TYPTXT <<CR><LF><LF>/ VOLTAGE CALIBRATION DAC />  
MOV R0,-(SP)  
TYPOS  
.BYTE 1  
.BYTE 0  
TYPE , $CRLF  
TYPTXT <<CR><LF>/SET EDC TO 0.0000 />  
TYPE , $CRLF  
TYPTXT </CONNECT EDC TO SCREW TERMINALS />  
MOV R0,R1  
ASL R1 ;OFFSET TO TERMINAL TABLE  
MOV 30$(R1),7$ ;TERMINAL NUMBERS  
TYPE  
7$: .WORD 0  
TYPE , $CRLF  
TYPE ,414$ ;"HIT A KEY WHEN READY"  
RDCHR ;WAIT FOR REPOSE  
TST (SP)+ ;CLEAR STACK  
TYPE , $CRLF  
TYPE , $CRLF  
TYPTXT </VOLTAGE OFFSET SHOULD BE LESS THAN (+ OR -) 2.5 MVOLT./>  
TYPTXT <<CR><LF>/ NOTE: THERE IS NO VOLTAGE OFFSET ADJUSTMENT./>  
TYPTXT <<CR><LF>/ IF NOT WITHIN ABOVE LIMITS - REPLACE MODULE./>
```

871	072512	104401	001203		TYPE	,\$CRLF	
872	072516	012746	C30000		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			.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							



```

895
896 072626          300$:  TYPTXT </CURRENT/<<CR><LF>>
897 072646 104401 073622  TYPE      ,411$           ;SET E66
898 072652 104401 073725  TYPE      ,413$           ; -TO CURRENT MODE
899 072656 104401 073747  TYPE      ,414$           ;HIT ANY KEY WHEN READY
900 072662 104406          RDCHR          ;WAIT FOR RESPONSE
901 072664 005726          TST      (SP)+          ;CLEAR STACK
902 072666 104401 073410  TYPE      ,410$           ;"SET EDC TO NULL MODE"
903
904 072672 012700 177777          MOV      #-1,R0          ;SET UP FOR LOOP
905 072676 005200          305$:  INC      R0              ;BUMP R0
906 072700 020027 000003          CMP      R0,#3          ;LAST DAC ?
907 072704 101402          BLOS     9$             ;NO BRANCH
908 072706 000137 071614          JMP      S5             ;JUMP
909 072712          9$:  TYPTXT <<CR><LF>/      CURRENT CALIBRATION DAC />
910 072754 010046          MOV      R0,-(SP)
911 072756 104403          TYPOS
912 072760 001          .BYTE 1
913 072761 000          .BYTE 0
914 072762 104401 001203          TYPE      ,SCLRF
915 072766 104401 001203          TYPE      ,SCLRF
916 072772          TYPTXT </CONNECT 500 OHM .01% RESISTOR TO />
917 073044          TYPTXT <<CR><LF>/      SCREW TERMINALS />
918 073100 010001          MOV      R0,R1          ;GET INDEX
919 073102 006301          ASL      R1              ;* 2
920 073104 016137 074112 073114          MOV      10$(R1),2$     ;ADDRESS OF MESSAGE
921 073112 104401          TYPE
922 073114 000000          2$:  .WORD 0
923 073116          TYPTXT <<CR><LF>/SET EDC TO 2.5 MVOLT./>
924 073154          TYPTXT <<CR><LF>/CONNECT EDC ACROSS RESISTOR./<<CR><LF>>
925 073224 104401 073747          TYPE      ,414$           ;HIT A KEY WHEN READY
926 073230 104406          RDCHR          ;WAIT FOR RESPONSE
927 073232 005726          TST      (SP)+          ;CLEAR STACK
928 073234 104401 073770          TYPE      ,415$
929 073240 010001          MOV      R0,R1          ;GET DAC #
930 073242 006301          ASL      R1              ;* 2
931 073244 016137 074346 073254          MOV      15$(R1),3$     ;ADDRESS OF MESSAGE
932 073252 104401          TYPE
933 073254 000000          3$:  .WORD 0              ;POT #
934 073256 104401 074007          TYPE      ,416$           ;"UNTIL EDC NULLS"
935 073262 012746 000001          MOV      #1,-(SP)       ;VALUE TO LOAD
936 073266 004737 073362          JSR      PC,400$        ;LOAD IT
937 073272 104406          RDCHR          ;WAIT FOR RESPONSE
938 073274 005726          TST      (SP)+          ;CLEAR STACK
939 073276 104401 001203          TYPE      ,SCLRF
940 073302 104401 074056          TYPE      ,417$           ;"SET EDC TO 10.2375 V"
941 073306 104401 073770          TYPE      ,415$           ;"ADJUST RXX"
942 073312 010001          MOV      R0,R1          ;DAC #
943 073314 006301          ASL      R1              ;* 2
944 073316 016137 074370 073326          MOV      20$(R1),4$     ;ADDRESS OF MESSAGE
945 073324 104401          TYPE
946 073326 000000          4$:  .WORD 0
947 073330 104401 074007          TYPE      ,416$           ;"UNTIL METER NULLS"
948 073334 012746 007777          MOV      #7777,-(SP)    ;VALUE TO LOAD
949 073340 004737 073362          JSR      PC,400$        ;GO LOAD IT
950 073344 104406          RDCHR          ;WAIT FOR REPNSE
951 073346 005726          TST      (SP)+          ;CLEAR STACK

```

952 073350 152777 000002 107266  
953 073356 000137 C72676

BISB #CBIT,@CSR  
JMP 305\$

:CLEAR DACS  
:DO IT AGAIN

```
955 :*****
956 :      CALL:   MOV     DAC#,RO
957 :             MOV     #VALUE,-(SP)
958 :             JSR     PC,400$
959 :*****
960
961
962 073362 010001      400$:  MOV     RO,R1           ;DON'T DESTROY RO
963 073364 006301      ASL     R1             ;* 2
964 073366 063701 003420  ADD     AD1,R1         ;HIGH BYTE ADDRESS
965 073372 116611 000003  MOVB   3(SP),@R1      ;LOAD HIGH BYTE
966 073376 005301      DEC     R1             ;LOW BYTE ADDRESS
967 073400 116611 000002  MOVB   2(SP),@R1      ;LOAD LOW BYTE
968 073404 012616      MOV     (SP)+,(SP)    ;ADJUST STACK
969 073406 000207      RTS     PC
970
971
972 073410      015      012      075  410$:  .ASCII <CR><LF>/==> SET EDC TO NULL MODE <==/<CR><LF>
973 073450      075      075      076  .ASCII *==>CALIBRATIONS ARE TO BE WITHIN (+ OR -)1.25MVOLT (1/2LSB)<==*
974 073546      015      012      075  .ASCIZ <CR><LF>/==>SET METER SENSITIVITY ACCORDINGLY<==/<CR><LF>
975 073622      123      105      124  411$:  .ASCII /SET E66 AS FOLLOWS 1=ON 0=OFF/<CR><LF>
976 073661      061      040      062  .ASCIZ /1 2 3 4 5 6 7 8/<CR><LF>
977 073703      061      040      061  412$:  .ASCIZ /1 1 0 1 0 1 0 0/<CR><LF>           ;VOLTAGE MODE
978 073725      061      040      060  413$:  .ASCIZ /1 0 1 0 1 0 1 0/<CR><LF>           ;CURRENT MODE
979 073747      074      103      122  414$:  .ASCIZ /<CR> TO CONTINUE/
980 073770      015      012      101  415$:  .ASCIZ <CR><LF>/ADJUST POT R/
981 074007      040      125      116  416$:  .ASCIZ / UNTIL EDC NULLS. <CR> TO CONTINUE/<CR><LF>
982 074056      123      105      124  417$:  .ASCIZ /SET EDC TO 10.2375 VOLTS./<CR><LF>
983 .EVEN
984
985 :*****
986 :      CURRENT SCREW TERMINALS.
987 :*****
988
989 074112 074122      10$:  11$      ;DAC 0
990 074114 074164      12$      ;DAC 1
991 074116 074232      13$      ;DAC 2
992 074120 074300      14$      ;DAC 3
993 074122      040      050      053  11$:  .ASCIZ / (+)7 (-)8 OR BERG PINS (+)7 (-)8/
994 074164      040      050      053  12$:  .ASCIZ / (+)15 (-)16 OR BERG PINS (+)15 (-)16/
995 074232      040      050      053  13$:  .ASCIZ / (+)23 (-)24 OR BERG PINS (+)49 (-)50/
996 074300      040      050      053  14$:  .ASCIZ / (+)33 (-)34 OR BERG PINS (+)39 (-)40/
997 .EVEN
998
999
1000 :*****
1001 :      CURRENT OFFSET TRIMPOT
1002 :*****
1003
1004 074346 074356      15$:  16$      ;DAC 0
1005 074350 074361      17$      ;DAC 1
1006 074352 074363      18$      ;DAC 2
1007 074354 074365      19$      ;DAC 3
1008 074356      061      061      000  16$:  .ASCIZ /11/           ;R11
1009 074361      070      000      17$:  .ASCIZ /8/           ;R8
1010 074363      066      000      18$:  .ASCIZ /6/           ;R6
1011 074365      063      000      19$:  .ASCIZ /3/           ;R3
```

```
1012          .EVEN
1013
1014
1015          ;*****
1016          ;CURRENT AND VOLTAGE GAIN TRIMPOT
1017          ;*****
1018
1019 074370 074400      20$: 21$
1020 074372 074403      22$
1021 074374 074405      23$
1022 074376 074407      24$
1023 074400      061 060 000 21$: .ASCIZ /10/          ;R10
1024 074403      067 000      22$: .ASCIZ /7/          ;R7
1025 074405      064 000      23$: .ASCIZ /4/          ;R4
1026 074407      062 000      24$: .ASCIZ /2/          ;R2
1027          .EVEN
1028
1029
1030          ;*****
1031          ;VOLTAGE DAC OUTPUT SCREW TERMINALS
1032
1033 074412 074422      30$: 31$      ;DAC 0
1034 074414 074464      32$      ;DAC 1
1035 074416 074532      33$      ;DAC 2
1036 074420 074600      34$      ;DAC 3
1037 074422      040 050 053 31$: .ASCIZ / (+)5 (-)6 OR BERG PINS (+)5 (-)6/
1038 074464      040 050 053 32$: .ASCIZ / (+)13 (-)14 OR BERG PINS (+)13 (-)14/
1039 074532      040 050 053 33$: .ASCIZ / (+)21 (-)22 OR BERG PINS (+)47 (-)48/
1040 074600      040 050 053 34$: .ASCIZ / (+)31 (-)32 OR BERG PINS (+)37 (-)38/
```

```
1049
1050
1051
1052
1053
1054
1055
1056 074646 005737 002664
1057 074652 001407
1058 074654 004737 014310
1059 074660 004737 075466
1060 074664 004737 075534
1061 074670 000207
1062
1063
1064 074672
1065 074672 012737 076146 000004
1066 074700 012737 000340 000006
1067 074706 012706 001100
1068 074712 012777 000340 105732 1$:
1069 074720 004737 014310
1070 074724 005037 001214
1071 074730 104401 075227
1072 074734 004737 112646
1073 074740 105777 106452
1074 074744 104406
1075 074746 112637 075200
1076 074752 004737 015526
1077 074756 113737 075200 075174
1078 074764 104401 075174
1079 074770 104401 001203
1080 074774 042737 100200 075200
1081 075002 005000
1082
1083
1084 075004 026037 075326 075200 4$:
1085 075012 001410
1086 075014 062700 000002
1087 075020 005760 075326
1088 075024 001367
1089 075026 104401 075202
1090 075032 000727
1091
```

```

.SBTTL A020 A/D CONVERTER MONITOR
:*****
:* A020 MONITOR - THE MONITOR GETS YOU THERE *
:*****

:*****AUTOMATIC TEST *****
AUTO20: TST AFLAG ;UNDER AUTO MODE ?
        BEQ A20SRT ;NO BRANCH
        JSR PC,KLEER ;CLEAR D BUS
        JSR PC,ADSET ;SET UP ADO-AD3
        JSR PC,A20TE1 ;DO THE TEST
        RTS PC

:***** THE INTERROGATOR *****
A20SRT:
A20MON: MOV #TIMOUT,ERRVEC ;SET UP TIMEOUT VECTOR
        MOV #340,ERRVEC+2 ;SET UP PRIORITY
        MOV #1100,SP ;RESET STACK POINTER
1$:     MOV #340,@VECTOA ;SET PRIORITY = 7
        JSR PC,KLEER ;CLEAR THE D BUS
        CLR $PASS ;CLEAR PASS COUNT
        TYPE ,220$ ;TYPE HEADER
        JSR PC,PRI7 ;SET PRIORITY 7
        TSTB @ADO ;CHECK FOR TIMEOUT
        RDCHR ;READ FIRST CHARACTER
        MOVB (SP)+,200$ ;SAVE FIRST LETTER
        JSR PC,CNTRC ;CONTROL C ?
        MOVB 200$,100$ ;SET UP ECHO
        TYPE ,100$ ;
        TYPE , $CRLF ;TYPE CR,LF
        BIC #100200,200$ ;STRIP BIT7 OF CHARACTERS
        CLR RO ;RESET COMMAND POINTER

:***** FIND COMMAND *****
CMP 300$(RO),200$ ;A MATCH ?
BEQ 5$ ;YES, BRANCH
ADD #2,RO ;ADVANCE POINTER
TST 300$(RO) ;END OF TABLE ?
BNE 4$ ;NO, BRANCH
TYPE ,210$ ;COMMAND NOT FOUND
BR 1$ ;GO TRY AGAIN
```

```
1093 ;***** EXECUTE COMMAND *****
1094 075034 004737 C75466 5$: JSR PC,ADSET ;SET THE ADDRESS
1095 075040 016037 075344 075050 MOV 500$(RO),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 , $CRLF ;TYPE A CR THEN LF
1099 075056 016037 075336 003652 MOV 400$(RO),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$: CLR RO ;SET UP RO
1116 075146 016037 075344 075156 8$: MOV 500$(RO),9$ ;GET READY TO TYPE
1117 075154 104401 TYPE ;THE COMMAND
1118 075156 000000 9$: 0 ;THE MESSAGE TO TYPE
1119 075160 062700 000002 ADD #2,RO ;ADVANCE POINTER
1120 075164 005760 075344 TST 500$(RO) ;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
```

```
1137  
1138 075326 000114 300$: ;***** COMMAND TABLES *****  
1139 075330 000110 .WORD 'L ;LOGIC TEST  
1140 075332 000103 .WORD 'H ;HELP FILE  
1141 075334 000000 .WORD 'C ;CALIBRATION  
1142 .WORD 0 ;END OF 300$ TABLE  
1143  
1144 075336 075534 400$: ;***** TEST POINTERS *****  
1145 075340 075144 .WORD A20TE1 ;LOGIC TEST  
1146 075342 105426 .WORD 7$ ;HELP FILE  
1147 .WORD A20CAL ;CALIBRATION  
1148  
1149 075344 075354 500$: ;***** HEADER POINTERS *****  
1150 075346 075402 .WORD 600$ ;LOGIC TEST  
1151 075350 075427 .WORD 601$ ;HELP FILE  
1152 075352 000000 .WORD 602$ ;CALIBRATION  
1153 .WORD 0 ;END OF TABLE 500$ MARKER  
1154  
1155 075354 050 114 051 600$: ;***** HEADER MESSAGES *****  
1156 075402 050 110 051 601$: .ASCIZ /(L)-A020 LOGIC TEST/<15><12>  
1157 075427 050 103 051 602$: .ASCIZ /(H)-A020 HELP FILE/<15><12>  
1158 .ASCIZ #(C)-A020 CALIBRATION/MANUAL #<15><12>  
1159 .EVEN  
1160  
1161 075466 013737 003462 003416 ADSET: ;***** SET UP ADDRESSES *****  
1162 075474 013737 003416 003420 MOV TADDR,ADO ;SET BASE ADDRESS  
1163 075502 005237 003420 MOV ADO,AD1 ;  
1164 075506 013737 003420 003422 INC AD1 ;MAKE SECOND ADDRESS  
1165 075514 005237 003422 MOV AD1,AD2 ;  
1166 075520 013737 003422 003424 INC AD2 ;MAKE THIRD ADDRESS  
1167 075526 005237 003424 MOV AD2,AD3 ;  
1168 075532 000207 INC AD3 ;MAKE FOURTH ADDRESS  
1169 RTS PC
```

1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1180  
1181  
1182 C75534  
  
075534 000004  
075536 012737 000036 001212  
1183 075544 004737 015526  
1184 075550 013737 003416 001136  
1185 075556 012737 075640 000004  
1186  
1187  
1188 075564 105777 103346  
1189 075570 005237 001136  
1190 075574 004737 015526  
1191 075600 105777 103332  
1192 075604 005237 001136  
1193 075610 004737 015526  
1194 075614 105777 103316  
1195 075620 005237 001136  
1196 075624 004737 015526  
1197 075630 105777 103302  
1198 075634 000410  
1199 075636 000736  
1200  
1201  
1202 075640 012637 001136  
1203 075644 012746 075652  
1204 075650 000002  
1205 075652  
075652 104120  
1206 075654 000207

```
.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.
:*****

A20TE1:
:*****
TST36: SCOPE
MOV #36,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
JSR PC,CNTRC ;;GOTO MONITOR IF CNTRLC
MOV A00,$BDADR ;;$BDADR = 171000
MOV #TMOUT,ERRVEC ;;ESTABLISH TIMEOUT RETURN

:***** CHECK FOR NO TIMEOUT *****
TSTB @$BDADR ;171000 EXIST ?
INC $BDADR ;MAKE NEXT ADDRESS
JSR PC,CNTRC ;GOTO MONITOR IF CNTRC
TSTB @$BDADR ;171001 EXIST ?
INC $BDADR ;MAKE NEXT ADDRESS
JSR PC,CNTRC ;GOTO MONITOR IF CNTRC
TSTB @$BDADR ;171002 EXIST ?
INC $BDADR ;MAKE NEXT ADDRESS
JSR PC,CNTRC ;GOTO MONITOR IF CNTRC
TSTB @$BDADR ;171003 EXIST ?
BR A20TE2
BR A20TE1 ;REPLACE ABOVE BRANCH WITH NOP TO LOOP

:*****TIMEOUT TRAPS HERE *****
TMOUT: MOV (SP)+,$BDADR ;SAVE OLD PC
MOV #1$,-(SP) ;TO LOAD PC
RTI ;THIS ONE TOO

1$: EMT 120
RTS PC
```



```
1208
1209
1210
1211
1212
1213
1214 075656
      075656 000004
      075660 012737 000037 001212
1215 075666 004737 015526
1216 075672 012737 076004 000004
1217
1218
1219 075700 013737 003416 001136 1$:
1220 075706 105077 103224
1221 075712 004737 015526
1222 075716 104121
1223 075720 013737 003420 001136
1224 075726 105077 103204
1225 075732 004737 015526
1226 075736 104121
1227 075740 013737 003424 001136
1228 075746 105077 103164
1229 075752 004737 015526
1230 075756 104121
1231
1232
1233 075760 012737 075640 000004
1234 075766 013737 003422 001136
1235 075774 105077 103136
1236 076000 000404
1237 076002 000725
1238
1239
1240 076004 062716 000014 2$:
1241 076010 000002
```

```
*****
: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.
*****

A20TE2:
*****
TST37: SCOPE
MOV #37,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
JSR PC,CNTRC ;GOTO MONITOR IF CNTRC
MOV #2$,ERRVEC ;SET UP TIMEOUT RETURN

***** CHECK FOR TIMEOUT *****
MOV ADO,$BDADR ;SET UP 171000
CLRB @$BDADR ;TRY WRITING
JSR PC,CNTRC ;GOTO MONITOR IF CNTRC
EMT 121
MOV AD1,$BDADR ;SET UP 171001
CLRB @$BDADR ;TRY WRITING
JSR PC,CNTRC ;GOTO MONITOR IF CNTRC
EMT 121
MOV AD3,$BDADR ;SET UP 171003
CLRB @$BDADR ;TRY WRITING
JSR PC,CNTRC ;GOTO MONITOR IF CNTRC
EMT 121

***** CHECK FOR NO TIMEOUT *****
MOV #TMOUT,ERRVEC ;SET UP TIMEOUT RETURN
MOV AD2,$BDADR ;SET UP 171002
CLRB @$BDADR ;TRY WRITING
BR A20TE3
BR A20TE2 ;REPLACE ABOVE BRANCH WITH NOP TO LOOP

***** WRITE TIMEOUT RETURN *****
ADD #14,(SP) ;ADJUST STACK
RTI ;RETURN TO PC+16
```

1243  
1244  
1245  
1246  
1247  
1248 076012  
  
076012 000004  
076014 012737 000040 001212  
1249 076022 005037 001142  
1250 076026 004737 015526  
1251 076032 012737 076146 000004  
1252  
1253  
1254 076040 112777 000002 104576  
1255 076046 012700 001000  
1256 076052 005300  
1257 076054 001376  
1258 076056 004737 015526  
1259 076062 105777 105336  
1260 076066 001411  
1261 076070 005037 001140  
1262 076074 013737 003424 001136  
1263 076102 117737 103030 001142  
1264 076110 104135  
1265  
1266  
1267 076112 105777 104526  
1268 076116 001411  
1269 076120 005037 001140  
1270 076124 013737 002644 001136  
1271 076132 117737 104506 001142  
1272 076140 104136  
1273  
1274 076142 000412  
1275 076144 000722  
1276  
1277  
1278 076146 012637 001136  
1279 076152 012746 076160  
1280 076156 000002  
1281 076160 004737 015526  
1282 076164 104134  
1283 076166 000207

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

A20TE3:

\*\*\*\*\*  
TST40: SCOPE  
MOV #40,\$TESTN ;;SET TEST NUMBER IN APT MAIL BOX  
CLR \$BDDAT ;RESET THE DATA  
JSR PC,CNTRC ;GOTO MONITOR IF CNTRC  
MOV #TIMOUT,ERRVEC ;SET UP TIMEOUT FOR ALL

\*\*\*\*\* CHECK THAT AD3 = 0 \*\*\*\*\*  
1\$: MOVB #CBIT,@CSR ;CLEAR THE THING  
MOV #1000,R0 ;SET UP DELAY  
DEC R0 ;DECREASE BY ONE  
BNE 1\$ ;LOOP TILL = 0  
JSR PC,CNTRC ;GOTO MONITOR IF CNTRC  
TSTB @AD3 ;171003 = 0 ?  
BEQ 2\$ ;YES, BRANCH  
CLR \$GDDAT ;SET UP GOOD DATA  
MOV AD3,\$BDADR ;SET UP 171003  
MOVB @BDADR,\$BDDAT ;SET UP BAD DATA  
EMT 135

\*\*\*\*\* CHECK THAT CSR = 0 \*\*\*\*\*  
2\$: TSTB @CSR ;CSR = 0 ?  
BEQ 3\$ ;YES, BRANCH  
CLR \$GDDAT ;SET UP GOOD DATA  
MOV CSR,\$BDADR ;SET UP TYPEOUT  
MOVB @CSR,\$BDDAT ;SET UP TYPEOUT  
EMT 136

3\$: BR A20TE4  
BR A20TE3 ;REPLACE ABOVE BRANCH WITH NOP TO LOOP

\*\*\*\*\* GENERAL TIMEOUT ROUTINE \*\*\*\*\*  
TIMOUT: MOV (SP)+,\$BDADR ;SAVE PC  
MOV #1\$,-(SP) ;TO LOAD PC  
RTI ;THIS ONE TOO !  
1\$: JSR PC,CNTRC ;GOTO MONITOR IF CNTRC  
EMT 134  
RTS PC

```
1285 :*****
1286 :TST4-THE G BIT IS SET AND EACH OF THE FOUR ADDRESSES ARE READ. THE GENERIC CODE
1287 :OF 324 OR 304 IS EXPECTED FROM EACH ADDRESS.
1288 :*****
1289
1290 076170 A20TE4:
      076170 000004
      076172 012737 000041 001212 TST41: SCOPE
1291 076200 005037 001142      MOV #41,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
1292 076204 004737 015526      CLR $BDDAT ;;CLEAR THIS WORD
1293      JSR PC,CNTRC ;;GOTO MONITOR IF CNTRC
1294      ;***** ESTABLISH CORRECT CODE *****
1295 076210 112777 000002 104426      MOVB #CBIT,@CSR ;;CLEAR THE THING
1296 076216 012700 001000      MOV #1000,RO ;;SET UP DELAY
1297 076222 005300      1$: DEC RO ;;DECREASE BY ONE
1298 076224 001376      BNE 1$ ;;LOOP TILL = 0
1299 076226 112777 000004 104410      MOVB #GBIT,@CSR ;;SET UP TO READ GENERIC CODE
1300 076234 013737 003416 001136      MOV ADO,$BDADR ;;SET UP 171000
1301 076242 013737 003416 076456      MOV ADO,200$ ;;SAVE ADDRESS
1302 076250 062737 000004 076456      ADD #4,200$ ;;EXIT ADDRESS
1303 076256 012737 000324 001140      2$: MOV #324,$GDDAT ;;PREPARE TYPEOUT
1304 076264 004737 015526      JSR PC,CNTRC ;;GOTO MONITOR IF CNTRC
1305 076270 127737 102642 001140      CMPB @BDADR,$GDDAT ;;CORRECT GENERIC CODE ?
1306 076276 001414      BEQ 3$ ;;YES, BRANCH
1307 076300 012737 000304 001140      MOV #304,$GDDAT ;;PREPARE TYPEOUT
1308 076306 127737 102624 001140      CMPB @BDADR,$GDDAT ;;OTHER GENERIC CODE CORRECT ?
1309 076314 001405      BEQ 3$ ;;YES, BRANCH
1310 076316 117737 102614 001142      MOVB @BDADR,$BDDAT ;;SET UP TYPEOUT
1311 076324 104123      EMT 123
1312 076326 000207      RTS PC
1313
1314      ;***** CHECK LOCATIONS FOR CORRECT CODE *****
1315 076330 013737 001140 003606      3$: MOV $GDDAT,GBITE ;;STORE FOR SAFE KEEPING
1316 076336 005237 001136      INC $BDADR ;;ADVANCE POINTER
1317 076342 023737 076456 001136      CMP 200$,$BDADR ;;MORE ADDRESSES TO GO ?
1318 076350 001413      BEQ 4$ ;;NO, BRANCH
1319 076352 004737 015526      JSR PC,CNTRC ;;GOTO MONITOR IF CNTRC
1320 076356 127737 102554 003606      CMPB @BDADR,GBITE ;;CORRECT CODE ?
1321 076364 001761      BEQ 3$ ;;YES, BRANCH
1322 076366 117737 102544 001142      MOVB @BDADR,$BDDAT ;;SET UP TYPEOUT
1323 076374 104123      EMT 123
1324 076376 000207      RTS PC
1325
1326      ;***** CHECK C BIT CLEARS AD3 *****
1327 076400 112777 000002 104236      4$: MOVB #CBIT,@CSR ;;CLEAR THE THING
1328 076406 012700 001000      MOV #1000,RO ;;SET UP DELAY
1329 076412 005300      5$: DEC RO ;;DECREASE BY ONE
1330 076414 001376      BNE 5$ ;;LOOP TILL = 0
1331 076416 004737 015526      JSR PC,CNTRC ;;GOTO MONITOR IF CNTRC
1332 076422 105777 104776      TSTB @AD3 ;;171003 = 0 ?
1333 076426 001411      BEQ 6$ ;;YES, BRANCH
1334 076430 005037 001140      CLR $GDDAT ;;SET UP GOOD DATA
1335 076434 013737 003424 001136      MOV AD3,$BDADR ;;SET UP 171003
1336 076442 117737 102470 001142      MOVB @BDADR,$BDDAT ;;SET UP BAD DATA
1337 076450 104135      EMT 135
1338 076452 000402      6$: BR A20TE5
```

.MAIN. MACRO M1113 12-AUG-82 09:59 PAGE 117-1  
A20 LOGIC TEST

1339 076454 000o45  
1340  
1341  
1342 076456 000000

BR A20TE4 ;REPLACE ABOVE BRANCH WITH NOP TO LOOP  
;\*\*\*\*\* VARIABLES \*\*\*\*\*  
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:
          :*****
          TST42: SCOPE
076460 000004
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  
1399 076742 000000

200\$: ;\*\*\*\*\* CONSTANTS \*\*\*\*\*  
;WORD 0 ;DATA PATTERN STROAGE

1401  
1402  
1403  
1404  
1405  
1406  
1407 076744  
  
076744 000004  
076746 012737 000043 001212  
1408 076754 004737 112556  
1409 076760 005037 001142  
1410 076764 004737 015526  
1411 076770 012777 077062 103652  
1412  
1413  
1414 076776 112777 000002 103640  
1415 077004 012700 001000  
1416 077010 005300  
1417 077012 001376  
1418 077014 013737 003422 001136  
1419 077022 112777 000200 104372  
1420 077030 112777 000100 103606  
1421 077036 000240  
1422 077040 000240  
1423 077042 142777 000100 103574  
1424 077050 004737 015526  
1425 077054 000240  
1426 077056 104125  
1427 077060 000465  
1428  
1429 077062 005066 000002 2\$:  
1430 077066 012716 077074  
1431 077072 000002  
1432 077074 3\$:  
077074 013737 002636 003446  
077102 005037 003444 65\$:  
077106 005237 003444 64\$:  
077112 023727 003444 000001  
077120 001372  
077122 005337 003446  
077126 001365  
1433 077130 132777 000200 103506  
1434 077136 001001  
1435 077140 000422  
1436 077142 005037 003502 7\$:  
1437 077146 117737 103474 003502  
1438 077154 123737 003502 003416  
1439 077162 001424  
1440 077164 152777 000001 103452  
1441 077172 062737 002660 003502  
1442 077200 105777 104276  
1443 077204 000733  
1444 077206 004737 015526 8\$:  
1445 077212 013737 003502 001136  
1446 077220 005037 001140  
1447 077224 113737 003502 001142

```
*****  
:TST6-WITH THE ERROR BIT IS SET THE INTERRUPT SYSTEM IS TURNED ON. AN INTERRUPT  
:SHOULD OCCUR, CAUSING THE IAR (171376) TO CONTAIN THE DEVICE ADDRESS. THE C BIT  
:IS THEN SET AND THE ERROR AND THE INTERRUPTS ARE CHECKED TO BE CLEARED.  
*****  
A20TE6:  
*****  
TST43: SCOPE  
MOV #43,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX  
JSR PC,PRI0 ;SET PRIORITY ZERO  
CLR $BDDAT ;CLEAR THIS WORD  
JSR PC,CNTRC ;EXIT IF ^C  
MOV #2$,@VECTO ;SET UP NEW VECTOR  
  
;***** CHECK INTERRUPT ON ERROR *****  
MOV #CBIT,@CSR ;SET CBIT  
MOV #1000,R0 ;SET UP DELAY  
1$: DEC R0 ;DECREASE BY ONE  
BNE 1$ ;LOOP TILL = 0  
MOV AD2,$BDADR ;IN CASE OF TIMEOUT  
MOV #200,@AD2 ;SET 171002 = 200 (ILLEGAL)  
MOV #EBIT,@CSR ;ENABLE INTERRUPTS  
NOP ;WAIT FOR INTERRUPTS  
NOP ;WAIT FOR INTERRUPTS  
BICB #EBIT,@CSR ;DISABLE INTERRUPTS  
JSR PC,CNTRC ;GOTO MONITOR IF CNTRC  
NOP ;DALLY A BIT  
EMT 125  
BR 4$  
  
2$: CLR 2(SP) ;LSI11 EXTRA CODE  
MOV #3$, (SP) ;TO LOAD PSW  
RTI ;THIS ONE TOO !  
  
3$: MOV K,ZLOOP  
65$: CLR YLOOP ;WAIT  
64$: INC YLOOP  
CMP YLOOP,#1  
BNE 64$  
DEC ZLOOP  
BNE 65$  
BITB #FBIT,@CSR ;ANY INTERRUPTS ?  
BNE 7$ ;YES, BRANCH  
BR 8$ ;ERROR CONDITION  
7$: CLR TEMP ;START FROM SRATCH  
MOV @IAR,TEMP ;STORE ADDRESS  
CMPB TEMP,ADO ;CORRECT INTERRUPT ADDRESS ?  
BEQ 4$ ;YES, BRANCH  
BISB #RBIT,@CSR ;PREPARE TO CLEAR  
ADD #BASE,TEMP ;MAKE AN ADDRESS  
TSTB @TEMP ;DO IT  
BR 3$ ;TRY AGAIN  
8$: JSR PC,CNTRC ;GOTO MONITOR IF CNTRC  
MOV TEMP,$BDADR ;SET UP ADDRESS  
CLR $GDDAT ;SET UP GOOD DATA  
MOV TEMP,$BDDAT ;SET UP BAD DATA
```

```
1448 077232 104140 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
```



.MAIN. MACRO M1113 12-AUG-82 09:59 PAGE 120  
A20 LOGIC TEST

1473			
1474	077342	012637	C01136
1475	077346	012746	077354
1476	077352	000002	
1477	077354	004737	015526
1478	077360	104126	
1479	077362	000207	

```

;***** UNEXPECTED INTERRUPT HANDLER *****
IOTRAP: MOV      (SP)+,$BDADR      ;SAVE PC
          MOV      #1$,-(SP)       ;TO LOAD PC
          RTI                          ;THIS ONE TOO !
1$:      JSR      PC,CNTRC          ;GOTO MONITOR IF CNTRC
          EMT      126
          RTS      PC

```

1481  
1482  
1483  
1484  
1485  
1486 077364  
  
077364 000004  
077366 012737 000044 001212  
1487 077374 004737 015526  
1488 077400 005037 001142  
1489 077404 005037 001140  
1490 077410 012777 077342 103232  
1491 077416 013737 003424 001136  
1492 077424 012777 000340 103220  
1493  
1494  
1495 077432 112777 000002 103204  
1496 077440 012700 001000  
1497 077444 005300  
1498 077446 001376  
1499 077450 112777 000200 103744  
1500 077456 112777 000001 103160  
1501 077464 013737 003424 001136  
1502 077472 105777 101440  
1503 077476 004737 015526  
1504 077502 105777 101430  
1505 077506 001401  
1506 077510 104141  
1507  
1508  
1509 077512 005077 103134  
1510 077516 012777 077570 103124  
1511 077524 112777 000200 103670  
1512 077532 112777 000001 103104  
1513 077540 105777 103652  
1514 077544 112777 000100 103072  
1515 077552 000240  
1516 077554 000240  
1517 077556 142777 000100 103060  
1518 077564 000240  
1519 077566 000453  
1520  
1521 077570  
077570 013737 002636 003446  
077576 005037 003444  
077602 005237 003444  
077606 023727 003444 000001  
077614 001372  
077616 005337 003446  
077622 001365  
1522 077624 132777 000200 103012  
1523 077632 001001  
1524 077634 000002  
1525 077636 005037 003502  
1526 077642 117737 103000 003502  
1527 077650 123737 003502 003416

\*\*\*\*\*  
: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:

\*\*\*\*\*

TST44: SCOPE  
MOV #44,\$TESTN ;;SET TEST NUMBER IN APT MAIL BOX  
JSR PC,CNTRC ;GOTO MONITOR IF CNTRC  
CLR \$BDDAT ;SET UP BAD DATA  
CLR \$GDDAT ;SET UP GOOD DATA  
MOV #IOTRAP,@VECTO ;SET UP NEW VECTOR  
MOV AD3,\$BDADR ;SET UP ADDRESS  
MOV #340,@VECTOA ;SET PRIORITY SEVEN

\*\*\*\*\* CHECK THAT R BIT CLEARS FLAG \*\*\*\*\*

1\$:  
MOV #CBIT,@CSR ;SET C BIT  
MOV #1000,R0 ;SET UP DELAY  
DEC R0 ;DECREASE BY ONE  
BNE 1\$ ;LOOP TILL = 0  
MOV #200,@AD2 ;SET 171002 = 200 (ILLEGAL)  
MOV #RBIT,@CSR ;GET READY TO CLEAR FLAG  
MOV AD3,\$BDADR ;SET UP DATA  
TSTB @BDADR ;CLEAR THIS DEVICE BY READING  
JSR PC,CNTRC ;GOTO MONITOR IF CNTRC  
TSTB @BDADR ;ERROR FLAG RESET ?  
BEQ 2\$ ;YES, BRANCH  
EMT 141

\*\*\*\*\* CHECK THAT INTERRUPT IS CLEARED \*\*\*\*\*

2\$:  
CLR @VECTOA ;SET TRAP PRIORITY = 0  
MOV #3\$,@VECTO ;SET UP VECTOR  
MOV #200,@AD2 ;SET UP ILLEGAL CONDITION  
MOV #RBIT,@CSR ;CLEAR THE CONDITION  
TSTB @ADO ;THIS DEVICE ONLY  
MOV #EBIT,@CSR ;ENABLE INTERRUPTS  
NOP ;WAIT FOR INTERRUPT  
NOP ;WAIT FOR INTERRUPT  
BICB #EBIT,@CSR ;DISABLE INTERRUPTS  
NOP ;WAIT FOR INTERRUPT  
BR 4\$

3\$:

MOV K,ZLOOP

65\$: CLR YLOOP ;WAIT

64\$:

INC YLOOP

CMP YLOOP,#1

BNE 64\$

DEC ZLOOP

BNE 65\$

BITB #FBIT,@CSR ;ANY INTERRUPTS ?

BNE 7\$ ;YES, BRANCH

7\$:

CLR TEMP ;START FROM SRATCH

MOV @IAR,TEMP ;STORE ADDRESS

CMPB TEMP,ADO ;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
```

```
1554 :*****
1555 :TST10-THIS TEST CHECKS THAT THE CHANNEL NUMBER JUST WRITTEN CAN BE READ BACK
1556 :IMMEDIATELY. THE CHANNEL IS LOADED INTO BITS 0-3 AND READ BACK FROM BITS 0-3 OF
1557 :ADDRESS TWO.
1558 : R1=DATA PATTERN
1559 :*****
1560
1561 100002 A20T10:
      100002 000004
      100004 012737 000045 001212 TST45: SCOPE
1562 100012 012701 177777          MOV #45,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
1563 100016 005037 001140          MOV #-1,R1 ;;SET UP PATTERN
1564 100022 005037 001142          CLR $GDDAT ;;SET UP GOOD DATA
1565 100026 112777 000002 102610 CLR $BDDAT ;;SET UP BAD DATA
1566 100034 012700 001000          MOVB #CBIT,@CSR ;;SET CBIT
1567 100040 005300          MOV #1000,R0 ;;SET UP DELAY
1568 100042 001376          1$: DEC R0 ;;DECREASE THE COUNT
      BNE 1$ ;;LOOP TILL=0
1569
1570 100044 004737 015526          2$: JSR PC,CNTRC ;;EXIT IF ^C
1571 100050 005201          INC R1 ;;MAKE NEXT PATTERN
1572 100052 013737 003422 001136 MOV AD2,$BDADR ;;SET UP ADDRESS
1573 100060 110177 101052          MOVB R1,@$BDADR ;;LOAD THE CHANNEL NUMBER
1574 100064 117737 101046 001142 MOVB @$BDADR,$BDDAT ;;GET DATA FROM A20
1575 100072 042737 000360 001142 BIC #360,$BDDAT ;;STRIP BITS TO IGNORE
1576 100100 010137 001140          MOV R1,$GDDAT ;;SET UP GOOD DATA
1577 100104 023737 001142 001140 CMP $BDDAT,$GDDAT ;;DATA OK?
1578 100112 001401          BEQ 4$ ;;YES, BRANCH
1579 100114 104143          EMT 143
1580
1581 100116 112777 000002 102520 4$: MOVB #CBIT,@CSR ;;SET THE CBIT
1582 100124 012700 001000          MOV #1000,R0 ;;SET UP DELAY
1583 100130 005300          5$: DEC R0
1584 100132 001376          BNE 5$ ;;LOOP TILL = 0
1585 100134 013737 002644 001136 MOV CSR,$BDADR ;;GET CSR
1586 100142 105777 100770          TSTB @$BDADR ;;CSR CLEAR?
1587 100146 001406          BEQ 6$ ;;YES, BRANCH
1588 100150 117737 100762 001142 MOVB @$BDADR,$BDDAT ;;SET UP BAD DATA
1589 100156 005037 001140          CLR $GDDAT ;;SET UP GOOD DATA
1590 100162 104144          EMT 144
1591
1592 100164 022701 000017          6$: CMP #17,R1 ;;DO MORE PATTERNS ?
1593 100170 001407          BEQ 7$ ;;NO, BRANCH
1594 100172 022737 000324 003606 CMP #324,GBITE ;;16 CHANNELS ?
1595 100200 001721          BEQ 2$ ;;YES, BRANCH
1596 100202 022701 000007          CMP #7,R1 ;;DO MORE PATTERNS ?
1597 100206 001316          BNE 2$ ;;YES, BRANCH
1598
1599 :***** CHECK C BIT CLEARS AD3 *****
1600 100210 112777 000002 102426 7$: MOVB #CBIT,@CSR ;;CLEAR THE THING
1601 100216 012700 001000          MOV #1000,R0 ;;SET UP DELAY
1602 100222 005300          8$: DEC R0 ;;DECREASE BY ONE
1603 100224 001376          BNE 8$ ;;LOOP TILL = 0
1604 100226 004737 015526          JSR PC,CNTRC ;;GOTO MONITOR IF CNTRC
1605 100232 105777 103166          TSTB @AD3 ;;171003 = 0 ?
1606 100236 001411          BEQ 9$ ;;YES, BRANCH
1607 100240 005037 001140          CLR $GDDAT ;;SET UP GOOD DATA
```

.MAIN. MACRO M1113 12-AUG-82 09:59 PAGE 122-1  
A20 LOGIC TEST

1608	100244	013,37	003424	001136
1609	100252	117737	100660	001142
1610	100260	104135		
1611				
1612	100262	000401		
1613	100264	000646		

MOV	AD3,\$BDADR	:SET UP 171003
MOVB	@\$BDADR,\$BDDAT	:SET UP BAD DATA
EMT	135	
9\$:	BR	A20T11
	BR	A20T10 ;REPLACE ABOVE BRANCH WITH A NOP TO LOOP

```
1615 :*****
1616 :TST11-THIS TEST OVERLOADS THE NUMBER OF CHANNELS INPUT AND CHECKS FOR THE ERROR
1617 :BIT BEING SET. THE C BIT IS THEN SET, AND THE OVERLOAD CONDITION IS CHECKED TO
1618 :BE CLEARED.
1619 :*****
1620
1621 100266 A20T11:
      :*****
      TST46: SCOPE
1622 100266 000004
1623 100270 012737 000046 001212 MOV #46,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
1624 100276 005037 001142 CLR $BDDAT ;CLEAR THIS WORD
1625 100302 004737 015526 JSR PC,CNTRC ;EXIT IF ^C
1626 100306 013737 003424 001136 MOV AD3,$BDADR ;SET UP ERROR ADDRESS
1627 100314 112777 000002 102322 :***** ESTABLISH SYNC WITH EOC *****
1628 100322 012700 001000 MOVB #CBIT,@CSR ;CLEAR THE WORLD
1629 100326 005300 11$: DEC R0 ;SET UP DELAY
1630 100330 001376 BNE 11$ ;DECREASE THE COUNT
1631 100332 112777 000001 103062 MOVB #1,@AD2 ;LOOP TILL = 0
1632 100340 005000 CLR R0 ;LOAD A CHANNEL
1633 100342 132777 000002 103054 12$: BITB #BIT1,@AD3 ;SET UP DELAY
1634 100350 001013 BNE 14$ ;DONE BIT SET ?
1635 100352 005300 DEC R0 ;YES, BRANCH
1636 100354 001372 BNE 12$ ;DECREASE DELAY
1637 100356 117737 103042 001142 MOVB @AD3,$BDDAT ;LOOP TILL ZERO
1638 100364 012737 000002 001140 MOV #2,$GDDAT ;
1639 100372 104146 EMT 146 ;
1640 100374 000137 101060 JMP A20T12 ;
1641
1642 :***** LOAD FIRST CHANNEL *****
1643 100400 112777 000002 102236 14$: MOVB #CBIT,@CSR ;CLEAR THE WORLD
1644 100406 012700 001000 MOV #1000,R0 ;SET UP DELAY
1645 100412 005300 1$: DEC R0 ;DECREASE THE COUNT
1646 100414 001376 BNE 1$ ;LOOP TILL = 0
1647 100416 012737 000001 001140 MOV #1,$GDDAT ;EXPECTED DATA
1648 100424 112777 000007 102770 MOVB #7,@AD2 ;LOAD CHANNEL
1649 100432 117737 100500 001142 MOVB @BDADR,$BDDAT ;GET STATUS
1650 100440 122737 000001 001142 CMPB #1,$BDDAT ;ONLY ACTIVE BIT?
1651 100446 001410 BEQ 2$ ;YES, BRANCH
1652 100450 005037 001140 CLR $GDDAT ;SET UP GOOD DATA
1653 100454 105737 001142 TSTB $BDDAT ;IS ALL CLEAR ?
1654 100460 001403 BEQ 2$ ;YES, BRANCH
1655 100462 104122 EMT 122 ;
1656 100464 000137 101060 JMP A20T12 ;GO TO NEXT TEST
1657
```

.MAIN. MACRO M1113 12-AUG-82 09:59 PAGE 124  
A20 LOGIC TEST

```

1659
1660 100470 112777 C00007 102724 2$:
1661 100476 117737 100434 001142
1662 100504 122737 000201 001142
1663 100512 001412
1664 100514 012737 000200 001140
1665 100522 122737 000200 001142
1666 100530 001403
1667 100532 104124
1668 100534 000137 101060
1669
1670
1671 100540 112777 000007 102654 3$:
1672 100546 117737 100364 001142
1673 100554 122737 000205 001142
1674 100562 001412
1675 100564 012737 000204 001140
1676 100572 122737 000204 001142
1677 100600 001403
1678 100602 104130
1679 100604 000137 101060
1680
1681
1682 100610 112777 000007 102604 4$:
1683 100616 117737 100314 001142
1684 100624 122737 000205 001142
1685 100632 001412
1686 100634 012737 000204 001140
1687 100642 122737 000204 001142
1688 100650 001403
1689 100652 104131
1690 100654 000137 101060
1691
1692
1693 100660 112777 000002 101756 5$:
1694 100666 012700 001000
1695 100672 005300 6$:
1696 100674 001376
1697 100676 105777 102522
1698 100702 001413
1699 100704 013737 003424 001136
1700 100712 117737 102506 001142
1701 100720 005037 001140
1702 100724 104132
1703 100726 000137 101060

;***** LOAD SECOND CHANNEL *****
MOV  #7,@AD2 ;LOAD CHANNEL
MOV  @BDADR,$BDDAT ;GET STATUS
CMP  #201,$BDDAT ;ACTIVE & BUSY
BEQ  3$ ;YES, BRANCH
MOV  #200,$GDDAT ;SET UP GOOD DATA
CMP  #200,$BDDAT ;ONLY BUSY ?
BEQ  3$ ;YES, BRANCH
EMT  124
JMP  A20T12 ;GO TO NEXT TEST

;***** LOAD THIRD CHANNEL *****
MOV  #7,@AD2 ;LOAD CHANNEL
MOV  @BDADR,$BDDAT ;GET STATUS
CMP  #205,$BDDAT ;ACTIVE,BUSY,& ERROR ONLY?
BEQ  4$ ;YES, BRANCH
MOV  #204,$GDDAT ;SET UP GOOD DATA
CMP  #204,$BDDAT ;BUSY & ERROR ?
BEQ  4$ ;YES, BRANCH
EMT  130
JMP  A20T12 ;GO TO NEXT TEST

;***** LOAD FOURTH CHANNEL *****
MOV  #7,@AD2 ;LOAD CHANNEL
MOV  @BDADR,$BDDAT ;GET STATUS
CMP  #205,$BDDAT ;ACTIVE,BUSY & ERROR ONLY?
BEQ  5$ ;YES, BRANCH
MOV  #204,$GDDAT ;SET UP GOOD DATA
CMP  #204,$BDDAT ;BUSY & ERROR ?
BEQ  5$ ;YES, BRANCH
EMT  131
JMP  A20T12 ;GO TO NEXT TEST

;***** CHECK THAT C BIT REALLY DOES CLEAR *****
MOV  #CBIT,@CSR ;CLEAR EVERYTHING
MOV  #1000,R0 ;SET UP DELAY
DEC  R0 ;DECREASE COUNT
BNE  6$ ;LOOP TILL = 0
TSTB @AD3 ;STATUS REG CLEAR ?
BEQ  7$ ;YES, BRANCH
MOV  AD3,$BDADR ;SET UP ERROR TYPEOUT
MOV  @AD3,$BDDAT ;SET UP ERROR TYPEOUT
CLR  $GDDAT ;SET UP ERROR TYPEOUT
EMT  132
JMP  A20T12 ;GO TO NEXT TEST

```

```
1705
1706 100732 112777 C00007 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
```



```
1731 :*****
1732 :TST12-THIS TEST CHECKS THAT THE CHANNEL NUMBER WRITTEN CAN BE READ IN THE
1733 :CONVERTED DATA CHANNEL BITS (4-7) OF ADDRESS THREE AFTER A CONVERSION USING
1734 :DUMB MODE.
1735 : R1=DATA PATTERN
1736 : R2=EXPECTED DATA
1737 :*****
1738
1739 101060 A20T12:
      :*****
      TST47: SCOPE
101060 000004          MOV #47,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
101062 012737 000047 001212 CLR $BDDAT          ;;SET THIS TO 0
1740 101070 005037 001142 CLR R1              ;;SET UP DATA PATTERN
1741 101074 005001          CLR R2              ;;SET UP EXPECTED DATA PATTERN
1742 101076 005002
1743
1744 101100 004737 015526 1$: 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 ADDRESS
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  $BDDAT,ADO ;CORRECT INTERRUPT ADDRESS ?
1797 101412 001417              BEQ   13$         ;YES, BRANCH
1798 101414 063737 002660 001142  ADD   BASE,$BDDAT ;MAKE NEXT ADDRESS
1799 101422 152777 000001 101214  BISB  #RBIT,@CSR  ;PREPARE TO CLEAR
1800 101430 105777 077506              TSTB  @SBDDAT     ;DO IT
1801 101434 000752              BR    7$          ;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
```

.MAIN. MACRO M1113 12-AUG-82 09:59 PAGE 127-1  
A20 LOGIC TEST

1841	101630	013;37	003424	001136	MOV	AD3,\$BDADR	;SET UP 171003
1842	101636	117737	C77274	001142	MOVB	@\$BDADR,\$BDDAT	;SET UP BAD DATA
1843	101644	104135			EMT	135	
1844							
1845	101646	000402			BR	A20T13	
1846	101650	000137	101060	12\$:	JMP	A20T12	;REPLACE ABOVE BRANCH WITH NOP TO LOOP

1848  
1849  
1850  
1851  
1852  
1853  
1854  
1855  
1856  
1857 101654  
  
101654 000004  
101656 012737 000050 001212  
1858 101664 005037 001140  
1859 101670 005037 001142  
1860 101674 004737 015526  
1861  
1862  
1863 101700 112777 000002 100736  
1864 101706 012700 001000  
1865 101712 005300  
1866 101714 001376  
1867 101716 012777 077342 100724  
1868 101724 012777 000340 100720  
1869 101732 004737 112556  
1870 101736 112777 000100 100700  
1871  
1872  
1873 101744 005037 003442  
1874 101750 112777 000001 101444  
1875 101756 112777 000002 101436  
1876 101764 117737 101434 001142  
1877 101772 123727 001142 000201  
1878 102000 001414  
1879 102002 142777 000100 100634  
1880 102010 012737 000201 001140  
1881 102016 013737 003424 001136  
1882 102024 104151  
1883 102026 000137 103546  
1884  
1885  
1886 102032 012777 103410 100610  
1887 102040 005004  
1888 102042 005737 003442  
1889 102046 001022  
1890 102050 162704 000001  
1891 102054 001372  
1892 102056 142777 000100 100560  
1893 102064 013737 002644 001136  
1894 102072 117737 100546 001142  
1895 102100 012737 000001 003542  
1896 102106 104205  
1897 102110 000137 103546

```
*****  
: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:

\*\*\*\*\*

```
TST50: SCOPE  
MOV #50,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX  
CLR $GDDAT ;;SET UP GOOD DATA  
CLR $BDDAT ;;SET UP BAD DATA  
JSR PC,CNTRC ;;EXIT IF ^C
```

\*\*\*\*\* CLEAR EVERYTHING, SETUP TRAPS, ENABLE INTERRUPTS \*\*\*\*\*

```
1$: MOVB #CBIT,@CSR ;;CLEAR THE THING  
MOV #1000,R0 ;;SET UP DELAY  
2$: DEC R0 ;;DECREASE COUNT  
BNE 2$ ;;SCOPE TILL = 0  
MOV #IOTRAP,@VECTO ;;SET UP UNEXPECTED INTERRUPT  
MOV #340,@VECTOA ;;SET TRAP PRIORITY TO 7  
JSR PC,PRIO ;;SET CPU PRIORITY TO ZERO  
MOVB #EBIT,@CSR ;;ENABLE INTERRUPTS
```

\*\*\*\*\* LOAD 2 CHANNELS, CHECK STATUS(AD3) \*\*\*\*\*

```
CLR IFLAG ;;SET INTERRUPT FLAG = 0  
MOVB #1,@AD2 ;;LOAD 1ST CHANNEL #  
MOVB #2,@AD2 ;;LOAD 2CD CHANNEL #  
MOVB @AD3,$BDDAT ;;GET THE BYTE  
CMPB $BDDAT,#201 ;;BUSY, ACTIVE ONLY?  
BEQ 3$ ;;YES, BRANCH  
BICB #EBIT,@CSR ;;SHUT OFF INTERRUPTS  
MOV #201,$GDDAT ;;SET UP GOOD DATA  
MOV AD3,$BDADR ;;LOAD ADDRESS NUMBER  
EMT 151  
JMP A20T14 ;;ABORT TEST
```

\*\*\*\*\* WAIT FOR INTERRUPT #1 \*\*\*\*\*

```
3$: MOV #400$,@VECTO ;;SET UP TRAP VECTOR  
CLR R4 ;;SET UP DELAY  
4$: TST IFLAG ;;ANY INTERRUPTS OCCUR ?  
BNE 5$ ;;YES, BRANCH  
SUB #1,R4 ;;DECREASE THE COUNT  
BNE 4$ ;;LOOP TILL = 0  
BICB #EBIT,@CSR ;;DISABLE INTERRUPTS  
MOV CSR,$BDADR ;;SET UP ADDRESS  
MOVB @CSR,$BDDAT ;;SAVE CSR  
MOV #1,ACOUNT ;;INTERRUPT NUMBER  
EMT 205  
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
```

```
1948 ;***** LOAD A CHANNEL, CHECK STATUS(AD3) *****
1949 102404 005037 C03442 20$: CLR IFLAG ;RESET INTERRUPT FLAG
1950 102410 112777 000005 101004 MOV #5,@AD2 ;LOAD 5TH CHANNEL #
1951 102416 117737 101002 001142 MOV @AD3,$BDDAT ;GET THE BYTE
1952 102424 123727 001142 000201 CMPB $BDDAT,#201 ;BUSY, ACTIVE ONLY?
1953 102432 001414 BEQ 21$ ;YES, BRANCH
1954 102434 142777 000100 100202 BICB #EBIT,@CSR ;SHUT OFF INTERRUPTS
1955 102442 012737 000201 001140 MOV #201,$GDDAT ;SET UP GOOD DATA
1956 102450 013737 003424 001136 MOV AD3,$BDADR ;LOAD ADDRESS NUMBER
1957 102456 104155 EMT 155
1958 102460 000137 103546 JMP A20T14 ;ABORT TEST
1959
1960 ;***** WAIT FOR INTERRUPT #3 *****
1961 102464 012777 103410 100156 21$: MOV #400$,@VECTO ;SET UP TRAP VECTOR
1962 102472 005004 CLR R4 ;SET UP DELAY
1963 102474 005737 003442 22$: TST IFLAG ;ANY INTERRUPTS OCCUR ?
1964 102500 001022 BNE 23$ ;YES, BRANCH
1965 102502 162704 000001 SUB #1,R4 ;DECREASE THE COUNT
1966 102506 001372 BNE 22$ ;LOOP TILL = 0
1967 102510 142777 000100 100126 BICB #EBIT,@CSR ;DISABLE INTERRUPTS
1968 102516 013737 002644 001136 MOV CSR,$BDADR ;SET UP ADDRESS
1969 102524 117737 100114 001142 MOVB @CSR,$BDDAT ;SAVE CSR
1970 102532 012737 000003 003542 MOV #3,ACOUNT ;INTERRUPT NUMBER
1971 102540 104205 EMT 205
1972 102542 000137 103546 JMP A20T14 ;ABORT TEST
1973
1974 ;***** CHECK RECV'D DATA *****
1975 102546 012777 077342 100074 23$: MOV #IOTRAP,@VECTO ;SET UP UNEXPECTED INTERRUPT
1976 102554 122703 000065 CMPB #65,R3 ;CHANNEL DATA OK ?
1977 102560 001414 BEQ 30$ ;YES, BRANCH
1978 102562 142777 000100 100054 BICB #EBIT,@CSR ;SHUT OFF INTERRUPTS
1979 102570 010337 001142 MOV R3,$BDDAT ;BAD DATA
1980 102574 012737 000065 001140 MOV #65,$GDDAT ;SAVE GOOD DATA
1981 102602 013737 003422 001136 MOV AD2,$BDADR ;SET UP ADDRESS
1982 102610 104156 EMT 156
1983 102612 000137 103546 JMP A20T14 ;ABORT TEST
1984
1985 ;***** CHECK STATUS(AD3) *****
1986 102616 005037 003442 30$: CLR IFLAG ;RESET INTERRUPT FLAG
1987 102622 117737 100576 001142 MOVB @AD3,$BDDAT ;GET THE BYTE
1988 102630 123727 001142 000001 CMPB $BDDAT,#1 ;ACTIVE ONLY?
1989 102636 001414 BEQ 31$ ;YES, BRANCH
1990 102640 142777 000100 077776 BICB #EBIT,@CSR ;SHUT OFF INTERRUPTS
1991 102646 012737 000001 001140 MOV #1,$GDDAT ;SET UP GOOD DATA
1992 102654 013737 003424 001136 MOV AD3,$BDADR ;LOAD ADDRESS NUMBER
1993 102662 104157 EMT 157
1994 102664 000137 103546 JMP A20T14 ;ABORT TEST
1995
```

```
1997  
1998 102670 012777 103410 077752 31$: ;***** WAIT FOR INTERRUPT #4 *****  
1999 102676 005004 ;MOV #400$,@VECTO ;SET UP TRAP VECTOR  
2000 102700 005737 003442 32$: CLR R4 ;SET UP DELAY  
2001 102704 001022 TST IFLAG ;ANY INTERRUPTS OCCUR ?  
2002 102706 162704 000001 BNE 33$ ;YES, BRANCH  
2003 102712 001372 SUB #1,R4 ;DECREASE THE COUNT  
2004 102714 142777 000100 077722 BNE 32$ ;LOOP TILL = 0  
2005 102722 013737 002644 001136 BICB #EBIT,@CSR ;DISABLE INTERRUPTS  
2006 102730 117737 077710 001142 MOV CSR,$BDADR ;SET UP ADDRESS  
2007 102736 012737 000004 003542 MOVB @CSR,$BDDAT ;SAVE CSR  
2008 102744 104205 EMT #4,ACOUNT ;INTERRUPT NUMBER  
2009 102746 000137 103546 JMP A20T14 ;ABORT TEST  
2010  
2011 ;***** CHECK RECV'D DATA *****  
2012 102752 012777 077342 077670 33$: MOV #IOTRAP,@VECTO ;SET UP UNEXPECTED INTERRUPT  
2013 102760 122703 000105 CMPB #105,R3 ;CHANNEL DATA OK ?  
2014 102764 001416 BEQ 40$ ;YES, BRANCH  
2015 102766 142777 000100 077650 BICB #EBIT,@CSR ;SHUT OFF INTERRUPTS  
2016 102774 010337 001142 MOV R3,$BDDAT ;BAD DATA  
2017 103000 012737 000105 001140 MOV #105,$GDDAT ;SAVE GOOD DATA  
2018 103006 013737 003422 001136 MOV AD2,$BDADR ;SET UP ADDRESS  
2019 103014 104160 EMT 160  
2020 103016 000137 103546 JMP A20T14 ;ABORT TEST  
2021  
2022 ;***** CHECK STATUS(AD3) *****  
2023 103022 005037 003442 40$: CLR IFLAG ;RESET INTERRUPT FLAG  
2024 103026 117737 100372 001142 MOVB @AD3,$BDDAT ;SAVE THE DATA  
2025 103034 123727 001142 000001 CMPB $BDDAT,#1 ; ACTIVE ONLY?  
2026 103042 001414 BEQ 41$ ;YES, BRANCH  
2027 103044 142777 000100 077572 BICB #EBIT,@CSR ;SHUT OFF INTERRUPTS  
2028 103052 012737 000001 001140 MOV #1,$GDDAT ;  
2029 103060 013737 003424 001136 MOV AD3,$BDADR ;SET UP ADDRESS NUMBER  
2030 103066 104161 EMT 161  
2031 103070 000137 103546 JMP A20T14 ;ABORT TEST  
2032  
2033 ;***** WAIT FOR INTERRUPT #5 *****  
2034 103074 012777 103410 077546 41$: MOV #400$,@VECTO ;SET UP TRAP VECTOR  
2035 103102 005004 CLR R4 ;SET UP DELAY  
2036 103104 005737 003442 42$: TST IFLAG ;ANY INTERRUPTS OCCUR ?  
2037 103110 001022 BNE 43$ ;YES, BRANCH  
2038 103112 162704 000001 SUB #1,R4 ;DECREASE THE COUNT  
2039 103116 001372 BNE 42$ ;LOOP TILL = 0  
2040 103120 142777 000100 077516 BICB #EBIT,@CSR ;DISABLE INTERRUPTS  
2041 103126 013737 002644 001136 MOV CSR,$BDADR ;SET UP ADDRESS  
2042 103134 117737 077504 001142 MOVB @CSR,$BDDAT ;SAVE CSR  
2043 103142 012737 000005 003542 MOV #5,ACOUNT ;INTERRUPT NUMBER  
2044 103150 104203 EMT 203  
2045 103152 000137 103546 JMP A20T14 ;ABORT TEST  
2046
```

```

2048
2049 103156 012777 C77342 077464 43$:
2050 103164 122703 000125
2051 103170 001413
2052 103172 010337 001142
2053 103176 012737 000125 001140
2054 103204 013737 003422 001136
2055 103212 104162
2056 103214 000137 103546
2057
2058
2059 103220 005037 003442 001142 50$:
2060 103224 117737 100174
2061 103232 005737 001142
2062 103236 001434
2063 103240 022737 000001 001142
2064 103246 001410
2065 103250 005037 001140
2066 103254 013737 003424 001136
2067 103262 104163
2068 103264 000137 103546
2069
2070
2071 103270 005004 001142 55$:
2072 103272 117737 100126 001142 56$:
2073 103300 005737 001142
2074 103304 001411
2075 103306 162704 000001
2076 103312 001367
2077 103314 005037 001140
2078 103320 013737 003424 001136
2079 103326 104204
2080
2081
2082 103330 112777 000002 077306 60$:
2083 103336 012700 001000
2084 103342 005300 61$:
2085 103344 001376
2086 103346 004737 015526
2087 103352 105777 100046
2088 103356 001411
2089 103360 005037 001140
2090 103364 013737 003424 001136
2091 103372 117737 075540 001142
2092 103400 104016
2093
2094 103402 000461 62$:
2095 103404 000137 101654

```

```

:***** CHECK RECVD DATA *****
MOV #IOTRAP,@VECTO ;SET UP UNEXPECTED INTERRUPT
CMPB #125,R3 ;CHANNEL DATA OK ?
BEQ 50$ ;YES, BRANCH
MOV R3,$BDDAT ;BAD DATA
MOV #125,$GDDAT ;SAVE GOOD DATA
MOV AD2,$BDADR ;SET UP ADDRESS
EMT 162
JMP A20T14 ;ABORT TEST

:***** CHECK STATUS(AD3) *****
CLR IFLAG ;RESET INTERRUPT FLAG
MOVB @AD3,$BDDAT ;SAVE THE DATA
TST $BDDAT ;ALL CLEAR ?
BEQ 60$ ;YES, BRANCH
CMP #1,$BDDAT ;ACTIVE SET ?
BEQ 55$ ;YES, BRANCH
CLR $GDDAT ;
MOV AD3,$BDADR ;SET UP ADDRESS NUMBFR
EMT 163
JMP A20T14 ;ABORT TEST

:***** CHECK ACTIVE CLEARS AFTER DELAY *****
CLR R4 ;SET UP DELAY
MOVB @AD3,$BDDAT ;SAVE STATUS
TST $BDDAT ;STATUS NOW CLEAR ?
BEQ 60$ ;YES, BRANCH
SUB #1,R4 ;COUNT DOWN
BNE 56$ ;LOOP TILL ZERO
CLR $GDDAT ;
MOV AD3,$BDADR ;SET ADDRESS
EMT 204

:***** CHECK C BIT CLEARS AD3 *****
MOVB #CBIT,@CSR ;CLEAR THE THING
MOV #1000,R0 ;SET UP DELAY
DEC R0 ;DECREASE BY ONE
BNE 61$ ;LOOP TILL = 0
JSR PC,CNTRC ;GOTO MONITOR IF CNTRC
TSTB @AD3 ;171003 = 0 ?
BEQ 62$ ;YES, BRANCH
CLR $GDDAT ;SET UP GOOD DATA
MOV AD3,$BDADR ;SET UP 171003
MOVB @$BDADR,$BDDAT ;SET UP BAD DATA
EMT 16

BR A20T14
JMP A20T13 ;REPLACE ABOVE BRANCH WITH NOP TO LOOP

```



```
2097  
2098 103410 005237 C03442 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$: CLR TEMP ;START FROM SRATCH  
2104 103466 117737 077154 003502 MOVB @IAR,TEMP ;STORE ADDRESS  
2105 103474 123737 003502 003416 CMPB TEMP,ADO ;CORRECT INTERRUPT ADDRESS ?  
2106 103502 001411 BEQ 403$ ;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  
2119  
2120  
2121  
2122 103546  
2123 103546 000004  
2124 103550 012737 000051 001212  
2125 103556 005037 001142  
2126 103562 004737 015526  
2127 103566 112777 000002 077050  
2128 103574 012700 001000  
2129 103600 005300  
2130 103602 001376  
2131 103604 112777 000001 077610  
2132 103612 112777 000002 077602  
2133 103620 005000  
2134 103622 132777 000002 077574  
2135 103630 001024  
2136 103632 005300  
2137 103634 001372  
2138 103636 004737 015526  
2139 103642 012737 000002 001140  
2140 103650 117737 077550 001142  
2141 103656 013737 003424 001136  
2142 103664 042737 177775 001142  
2143 103672 104164  
2144  
2145 103674 004737 015526  
2146 103700 005000  
2147 103702 117737 077516 001142  
2148 103710 022737 000012 001142  
2149 103716 001411  
2150 103720 005300  
2151 103722 001367  
2152 103724 013737 003424 001136  
2153 103732 012737 000012 001140  
2154 103740 104165  
2155  
2156 103742 117737 077454 001142  
2157 103750 122737 000022 001142  
2158 103756 001407  
2159 103760 013737 003422 001136  
2160 103766 012737 000022 001140  
2161 103774 104166
```

```
*****  
: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.  
*****  
A20T14:  
*****  
TST51: SCOPE  
MOV #51,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX  
CLR $BDDAT ;CLEAR THIS LOCATION  
JSR PC,CNTRC ;EXIT IF ^C  
  
MOV #CBIT,@CSR ;CLEAR THE WORLD  
MOV #1000,R0 ;SET UP DELAY  
1$: DEC R0 ;COUNT DOWN  
BNE 1$ ;LOOP TILL = 0  
  
MOV #1,@AD2 ;LOAD 1ST CHANNEL  
MOV #2,@AD2 ;LOAD 2ND CHANNEL  
CLR R0 ;SET COUNT = 0  
2$: BIT #BIT1,@AD3 ;DONE BIT SET ?  
BNE 4$ ;YES, BRANCH  
DEC R0 ;DECREASE COUNT  
BNE 2$ ;LOOP TILL = 0  
JSR PC,CNTRC ;GOTO MONITOR IF CNTRC  
MOV #2,$GDDAT  
MOV @AD3,$BDDAT  
MOV AD3,$BDADR  
BIC #177775,$BDDAT ;IGNORE OTHER BITS  
EMT 164  
  
JSR PC,CNTRC ;GOTO MONITOR IF CNTRC  
CLR R0 ;RESET THE COUNT  
4$: MOV @AD3,$BDDAT ;GET THE DATA  
CMP #12,$BDDAT ;DATA OVER AND DONE  
BEQ 5$ ;YES, BRANCH  
DEC R0 ;DECREASE THE COUNT  
BNE 4$ ;KEEP TRYING  
MOV AD3,$BDADR  
MOV #12,$GDDAT  
EMT 165  
  
5$: MOV @AD2,$BDDAT ;SAVE THE DATA  
CMP #22,$BDDAT ;CHANNEL #'S INTACT?  
BEQ 6$ ;YES, BRANCH  
MOV AD2,$BDADR  
MOV #22,$GDDAT  
EMT 166
```

```
2163 103776 112777 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
```

2189  
2190  
2191  
2192  
2193  
2194  
2195  
2196  
2197  
2198  
2199  
2200  
2201  
2202  
2203  
2204  
2205  
2206  
2207  
2208  
2209  
2210  
2211  
2212  
2213  
2214  
2215  
2216  
2217  
2218  
2219  
2220  
2221  
2222  
2223  
2224  
2225  
2226  
2227  
2228  
2229  
2230  
2231  
2232  
2233  
2234  
2235  
2236  
2237  
2238  
2239  
2240  
2241  
2242  
2243  
2244  
2245

```
.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 APPLYING A VOLTAGE DIRECTLY TO TP1&TP2. ALWAYS REMOVE 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
2283								
2284					CAL TAB:	37200.	1.	1\$ :51.200V
2285	104124	037200	000001	104440		37200.	1.	2\$ :50.000V
2286	104132	037200	000001	104455		6200.	1.	3\$ :51.200V W/O FS PRECISION SOURCE
2287	104140	006200	000001	104472		6315.	1.	4\$ :50.000V W/O FS PRECISION SOURCE
2288	104146	006315	000001	104507		37200.	1.	33\$ :25.600V
2289	104154	037200	000001	105245		37200.	1.	34\$ :25.000V
2290	104162	037200	000001	105262		14400.	1.	35\$ :25.600V W/O FS PRECISION SOURCE
2291	104170	014400	000001	105277		14632.	1.	36\$ :25.000V W/O FS PRECISION SOURCE
2292	104176	014632	000001	105314		37200.	1.	5\$ :20.480V
2293	104204	037200	000001	104524		37200.	1.	6\$ :20.000V
2294	104212	037200	000001	104541		17500.	1.	7\$ :20.480V W/O FS PRECISION SOURCE
2295	104220	017500	000001	104556		20000.	1.	10\$ :20.000V W/O FS PRECISION SOURCE
2296	104226	020000	000001	104573		37200.	1.	11\$ :10.240V
2297	104234	037200	000001	104610		37200.	1.	12\$ :10.000V
2298	104242	037200	000001	104625		37200.	1.	13\$ :5.1200V
2299	104250	037200	000001	104642		37200.	1.	14\$ :5.0000V
2300	104256	037200	000001	104657		37200.	1.	37\$ :2.5600V
2301	104264	037200	000001	105331		37200.	1.	40\$ :2.5000V
2302	104272	037200	000001	105346		37200.	1.	15\$ :2.0480V
2303	104300	037200	000001	104674		37200.	1.	16\$ :2.0000V
2304	104306	037200	000001	104711		37200.	1.	17\$ :1.0240V
2305	104314	037200	000001	104726		37200.	1.	20\$ :1.0000V
2306	104322	037200	000001	104743		37200.	1.	41\$ :500.0MV
2307	104330	037200	000001	105363		37200.	1.	42\$ :482.8MV
2308	104336	037200	000001	105404		37200.	1.	21\$ :204.80MV
2309	104344	037200	000001	104761		37200.	1.	22\$ :200.00MV
2310	104352	037200	000001	105003		37200.	1.	23\$ :102.40MV
2311	104360	037200	000001	105025		37200.	1.	24\$ :100.00MV
2312	104366	037200	000001	105047		37200.	1.	25\$ :51.200MV
2313	104374	037200	000001	105071		37200.	1.	26\$ :50.000MV
2314	104402	037200	000001	105113		37200.	2.	27\$ :20.480MV
2315	104410	037200	000002	105135		37200.	2.	30\$ :20.000MV
2316	104416	037200	000002	105157		37200.	4.	31\$ :10.240MV
2317	104424	037200	000004	105201		37200.	4.	32\$ :10.000MV
2318	104432	037200	000004	105223				

						:***** ASCII MESSAGES *****		
2319								
2320								
2321	104440	065	060	056	1\$:	.ASCIZ	/50.000	VOLTS/
2322	104455	064	070	056	2\$:	.ASCIZ	/48.828	VOLTS/
2323	104472	061	060	056	3\$:	.ASCIZ	/10.000	VOLTS/
2324	104507	061	060	056	4\$:	.ASCIZ	/10.000	VOLTS/
2325	104524	062	060	056	5\$:	.ASCIZ	/20.000	VOLTS/
2326	104541	061	071	056	6\$:	.ASCIZ	/19.531	VOLTS/
2327	104556	061	060	056	7\$:	.ASCIZ	/10.000	VOLTS/
2328	104573	061	060	056	10\$:	.ASCIZ	/10.000	VOLTS/
2329	104610	061	060	056	11\$:	.ASCIZ	/10.000	VOLTS/
2330	104625	071	056	067	12\$:	.ASCIZ	/9.7656	VOLTS/
2331	104642	065	056	060	13\$:	.ASCIZ	/5.0000	VOLTS/
2332	104657	064	056	070	14\$:	.ASCIZ	/4.8828	VOLTS/
2333	104674	062	056	060	15\$:	.ASCIZ	/2.0000	VOLTS/
2334	104711	061	056	071	16\$:	.ASCIZ	/1.9531	VOLTS/
2335	104726	061	056	060	17\$:	.ASCIZ	/1.0000	VOLTS/
2336	104743	060	056	071	20\$:	.ASCIZ	/0.97656	VOLTS/
2337	104761	062	060	060	21\$:	.ASCIZ	/200.00	MILLIVOLTS/
2338	105003	061	071	065	22\$:	.ASCIZ	/195.31	MILLIVOLTS/
2339	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		

```

2357 ;***** FIELD CALIBRATION ROUTINE *****
2358 ;--SET CBIT TO CLEAR WORLD
2359 ;--SET UP TIMEOUT VECTOR
2360 ;--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

```



```
2408 ;--IF CALIBRATE GAIN
2409 ;--THEN
2410 ;---CALL 500$ (GET CHANNEL #)
2411 ;---TYPE SELECT VOLTAGE TO CALIBRATE FROM TABLE AND ENTER NUMBER
2412 ;---TYPE REMOVE CUSTOMER CABLE AND PRESS A KEY WHEN READY
2413 105616 004737 015526 40$: JSR PC,CNTRC ;IS CONTROL C
2414 105622 104401 122576 TYPE ,NOS ;TYPE 'NO'
2415 105626 104401 106335 41$: TYPE ,210$ ;DO YOU WISH TO CALIBRATE GAIN?
2416 105632 104406 RDCHR ;GET A CHARACTER
2417 105634 022726 000131 CMP #'Y,(SP)+ ;A YES RESPONSE
2418 105640 001160 BNE 50$ ;NO,BRANCH
2419 105642 104401 122572 TYPE ,YES$ ;TYPE 'YES'
2420 105646 004737 110702 JSR PC,500$ ;GET A CHANNEL NUMBER
2421 105652 104401 106465 TYPE ,240$ ;DISCONNECT AND PRESS KEY
2422 105656 104406 RDCHR ;
2423 105660 005726 TST (SP)+ ;
2424 105662 104401 001203 TYPE ,$CRLF ;
2425
2426
2427 ;--- GET #X
2428 ;---OFFSET=(6*X)-6
2429 ;---VALUE=CALTAB(OFFSET)
2430 ;---TOL=CALTAB+2(OFFSET)
2431 ;---TYPE MESSAGE CALTAB+4(OFFSET)
2432 ;---CALL CALRTN (CONVERSION ROUTINE)
2433 105666 104401 106600 46$: TYPE ,245$ ;'PRINT CALIBRATION TABLE ?'
2434 105672 104406 RDCHR ;GET A CHARACTER
2435 105674 022726 000131 CMP #'Y,(SP)+ ;A YES RESPONSE
2436 105700 001405 BEQ 47$ ;YES, BRANCH
2437 105702 004737 015526 JSR PC,CNTRC ;CONTROL C ?
2438 105706 104401 122576 TYPE ,NOS ;NO
2439 105712 000404 BR 42$
2440 105714 104401 122572 47$: TYPE ,YES$ ;TYPE 'YES'
2441 105720 104401 106652 TYPE ,241$ ;PRINT TABLE
2442 105724 104401 110310 42$: TYPE ,243$ ;'ENTER NUMBER FROM TABLE'
2443 105730 104410 RDOCT ;
2444 105732 021627 000042 CMP (SP),#42 ;TOO LARGE ?
2445 105736 101404 BLOS 43$ ;NO,BRANCH
2446 105740 005726 TST (SP)+ ;CLEAR CHARACTER
2447 105742 104401 110360 TYPE ,242$ ;TOO LARGE !
2448 105746 000747 BR 46$ ;TRY AGAIN
2449 105750 022716 000001 43$: CMP #1,(SP) ;BELOW 1?
2450 105754 101404 BLOS 45$ ;NO,BRANCH
2451 105756 005726 TST (SP)+ ;CLEAR CHARACTER
2452 105760 104401 110360 TYPE ,242$ ;ERROR TOO LOW
2453 105764 000740 BR 46$ ;
2454 105766 012746 000006 45$: MOV #6,-(SP) ;MULTIPLIER
2455 105772 004737 017644 JSR PC,@#$MULT ;MULTIPLY
2456 105776 012600 MOV (SP)+,R0 ;THE OFFSET
2457 106000 005726 TST (SP)+ ;IGNORE MSB
2458 106002 162700 000006 SUB #6,R0 ;ADJUST FOR START OF 1
2459 106006 016037 104124 003576 MOV CALTAB(R0),TEMP2;
2460 106014 016037 104126 111220 MOV CALTAB+2(R0),CALTOL;
2461 106022 016037 104130 106052 MOV CALTAB+4(R0),44$;PREPARE TO TYPE MESSAGE
2462 106030 104401 106036 TYPE ,65$ ;:TYPE ASCIZ STRING
106034 000405 BR 64$ ;:GET OVER THE ASCIZ
;:65$: .ASCIZ <CR><LF>/INPUT /
```

```
2463 106050 104401 64$: TYPE
2464 106052 000000 44$: 0 ;MESSAGE ADDRESS
2465 106054 104401 106062 66$: TYPE ,67$ ;:TYPE ASCIZ STRING
106060 000415 BR 66$ ;:GET OVER THE ASCIZ
66$: .ASCIZ / FOR AN AVERAGE VALUE OF /
2466 106114 013746 003576 68$: MOV TEMP2,-(SP) ;:SAVE TEMP2 FOR TYPEOUT
106120 104402 TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
2467 106122 104401 106130 68$: TYPE ,69$ ;:TYPE ASCIZ STRING
106126 000411 BR 68$ ;:GET OVER THE ASCIZ
68$: .ASCIZ <CR><LF>%TOLERANCE +/- %
2468 106152 013746 111220 70$: MOV CALTOL,-(SP)
2469 106156 104403 TYPOS
2470 106160 001 .BYTE 1
2471 106161 000 .BYTE 0
2472 106162 104401 106170 70$: TYPE ,71$ ;:TYPE ASCIZ STRING
106166 000403 BR 70$ ;:GET OVER THE ASCIZ
70$: .ASCIZ / LSB/
2473 106176 004737 111014 70$: JSR PC,300$ ;
```



2494  
2495 106260 015 012 101 200\$:  
2496 106314 015 012 115 201\$:  
2497 106335 015 012 103 210\$:  
2498 106366 015 012 122 230\$:  
2499 106465 015 012 122 240\$:  
2500 106547 054 040 120  
2501 106600 015 012 120 245\$:  
2502 106652 015 012 012 241\$:  
2503 106672 011 011 011  
2504 106715 015 012 055  
2505 106736 011 011 011  
2506 106763 015 012 061  
2507 106776 011 011 011  
2508 107016 015 012 062  
2509 107031 011 011 011  
2510 107051 015 012 063  
2511 107124 011 011 062  
2512 107140 015 012 064  
2513 107213 011 011 062  
2514 107227 015 012 065  
2515 107242 011 011 011  
2516 107262 015 012 066  
2517 107275 011 011 011  
2518 107315 015 012 067  
2519 107370 011 011 063  
2520 107405 015 012 061  
2521 107461 011 011 063  
2522 107476 015 012 061  
2523 107512 011 011 011  
2524 107533 015 012 061  
2525 107547 011 011 011  
2526 107570 015 012 061  
2527 107644 011 011 063  
2528 107661 015 012 061  
2529 107735 011 011 063  
2530 107752 015 012 061  
2531 107766 011 011 011  
2532 110007 015 012 061  
2533 110023 011 011 011  
2534 110044 015 012 061  
2535 110060 011 011 011  
2536 110101 015 012 062  
2537 110115 011 011 011  
2538 110136 015 012 062  
2539 110152 015 012 062  
2540 110166 015 012 012  
2541 110233 015 012 115  
2542 110310 015 012 111 243\$:  
2543 110360 015 012 116 242\$:  
2544 110411 015 012 124 260\$:  
2545 110440 015 012 117 270\$:  
2546 110462 015 012 105 280\$:  
2547 110514 015 012 052 289\$:  
2548 110544 015 012 127 290\$:  
2549 110605 015 012 103 291\$:  
2550

```
***** MESSAGES *****  
.ASCIIZ <15><12> #ADJUST THE OFFSET [Y/N] ?#  
.ASCIIZ <15><12> #MANUAL [Y/N] ?#  
.ASCIIZ <15><12> #CALIBRATE GAIN [Y/N] ?#  
.ASCIIZ <15><12> /REMOVE CUSTOMER CABLE AND INPUT 0.0V, PRESS A KEY WHEN READY/  
.ASCII <15><12> /REMOVE CUSTOMER CABLE AND CONNECT VOLTAGE SOURCE/  
.ASCIIZ /, PRESS A KEY WHEN READY/  
.ASCIIZ <15><12> #PRINT CALIBRATION VOLTAGE TABLE [Y/N] ?#  
.ASCII <15><12><12> /VALUE VOLTAGE/  
.ASCII / VALUE VOLTAGE/  
.ASCII <15><12> /-----/  
.ASCII /-----/  
.ASCII <15><12> /1 51.200V/ 23 2.0480V/  
.ASCII / 24 2.0000V/  
.ASCII <15><12> /2 50.000V/ 25 1.0240V/  
.ASCII / 26 1.0000V/  
.ASCII <15><12> %3 51.200V W/O FULL SCALE PRECISION SOURCE%  
.ASCII / 27 512.0MV/  
.ASCII <15><12> %4 50.000V W/O FULL SCALE PRECISION SOURCE%  
.ASCII / 30 500.0MV/  
.ASCII <15><12> /5 25.600V/ 31 204.80MV/  
.ASCII / 32 200.00MV/  
.ASCII <15><12> %7 25.600V W/O FULL SCALE PRECISION SOURCE%  
.ASCII / 33 102.40MV/  
.ASCII <15><12> %10 25.000V W/O FULL SCALE PRECISION SOURCE%  
.ASCII / 34 100.00MV/  
.ASCII <15><12> /11 20.480V/ 35 51.200MV/  
.ASCII / 36 50.000MV/  
.ASCII <15><12> %13 20.480V W/O FULL SCALE PRECISION SOURCE%  
.ASCII / 37 20.480MV/  
.ASCII <15><12> %14 20.000V W/O FULL SCALE PRECISION SOURCE%  
.ASCII / 40 20.000MV/  
.ASCII <15><12> /15 10.240V/ 41 10.240MV/  
.ASCII / 42 10.000MV/  
.ASCII <15><12> /16 10.000V/  
.ASCII <15><12> /17 5.1200V/  
.ASCII <15><12> /20 5.0000V/  
.ASCII <15><12> /22 2.5000V/  
.ASCII <15><12> /21 2.5600V/  
.ASCII <15><12><12> /INPUT A VALUE FROM THE TABLE WHICH/  
.ASCIIZ <15><12> /MATCHES THE VOLTAGE YOU ARE CALIBRATING TO/  
.ASCIIZ <15><12> /INPUT VALUE FROM TABLE [1-42 OCTAL]: /  
.ASCIIZ <15><12> /NUMBER NOT IN RANGE !!/
```

```
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 ;
```

.MAIN. MACRO M1113 12-AUG-82 09:59 PAGE 142-1  
A020 CALIBRATION/MANUAL ROUTINE

2609 111214 000.37 006360  
2610  
2611  
2612 111220 000000  
2613 111222 000000  
2614 111224 000000

JMP START

:\*\*\*\*\* VARIABLE \*\*\*\*\*  
CALTOL: 0 ;HOLDS TOLERANCE  
CALUP: 0 ;UPPER LIMIT  
CALLO: 0 ;LOWER LIMIT

```
2616
2617
2618
2619
2620
2621 111226 032766 100000 000002 TYPERR: BIT #BIT15,2(SP) ;ERROR BIT SET ?
2622 111234 001402 BEQ 1$ ;NO, BRANCH
2623 111236 104401 111312 TYPE ,20$
2624 111242 032766 040000 000002 1$: BIT #BIT14,2(SP) ;OVER RANGE BIT SET ?
2625 111250 001402 BEQ 2$ ;NO, BRANCH
2626 111252 104401 111343 TYPE ,21$
2627 111256 032766 020000 000002 2$: BIT #BIT13,2(SP) ;DATA OVERRUN SET ?
2628 111264 001402 BEQ 3$ ;NO, BRANCH
2629 111266 104401 111402 TYPE ,22$
2630 111272 032766 004000 000002 3$: BIT #BIT11,2(SP) ;TOO MANY CONVERSIONS ?
2631 111300 001402 BEQ 4$ ;NO, BRANCH
2632 111302 104401 111440 TYPE ,23$
2633 111306 012616 4$: MOV (SP)+,(SP) ;MOVE THE RETURN PC
2634 111310 000207 RTS PC
2635 111312 015 012 105 20$: .ASCIZ <15><12> /ERROR-- ERROR BIT SET /
2636 111343 015 012 105 21$: .ASCIZ <15><12> /ERROR-- OVER RANGE BIT SET /
2637 111402 015 012 105 22$: .ASCIZ <15><12> /ERROR-- DATA OVER RUN SET /
2638 111440 015 012 105 23$: .ASCIZ <15><12> /ERROR--RECEIVED INCORRECT # OF CONVERSION /
2639 .EVEN
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651 111516 017746 071126 LDATABL: MOV @VECTO,-(SP) ;SAVE OLD VECTOR
2652 111522 012777 111732 071120 MOV #19$,@VECTO ;SET UP NEW VECTOR
2653 111530 004737 112556 JSR PC,PRI0
2654 111534 010046 MOV R0,-(SP) ;SAVE R0
2655 111536 010146 MOV R1,-(SP) ;SAVE R1
2656 111540 005000 CLR R0 ;SET=0
2657 111542 005001 CLR R1 ;SET=0
2658
2659
2660 111544 005060 112170 1$: ;***** INITIALIZE THE TABLE *****
2661 111550 062700 000002 CLR TABLE1(R0) ;CLEAR A ELEMENT
2662 111554 022700 000100 ADD #2,R0 ;INCREASE THE OFFSET
2663 111560 001371 CMP #100,R0 ;END OF TABLE ?
2664 111562 005000 BNE 1$ ;NO, BRANCH
2665 111564 152777 000001 071052 CLR R0 ;RESET R0
2666 111572 105777 071626 BISB #RBIT,@CSR ;CLEAR THE UNIT
2667 111576 152777 000100 071040 TSTB @AD3 ;DO IT
2668 111604 116677 000010 071610 2$: MOVB 10(SP),@AD2 ;ENABLE INTERRUPTS
2669 111612 132777 000004 071604 BITB #BIT2,@AD3 ;LOAD CHANNEL TO BE CONVERTED
2670 111620 001407 BEQ 3$ ;IS ERROR BIT SET ?
2671 111622 004737 112646 JSR PC,PRI7 ;NO, BRANCH
2672 111626 052766 100000 000010 BIS #BIT15,10(SP) ;SET PRIORITY SEVEN
;SET COUNT ERROR
```

```
2673 111634 000137 111712          JMP      5$                ;AND LEAVE
2674 111640 105777 C71560          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$:  BICB   #EBIT,@CSR       ;DISABLE INTERRUPTS
2686 111720 012601          MOV    (SP)+,R1         ;RESTORE R1
2687 111722 012600          MOV    (SP)+,R0         ;RESTORE R0
2688 111724 012677 070720          MOV    (SP)+,@VECTO     ;RESTORE VECTO
2689 111730 000207          RTS     PC
2690
2691
2692 111732          ;*****INTERRUPT HANDLER *****
111732 013737 002636 003446 19$:  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$:  CLR    TEMP             ;START FROM SRATCH
2697 112004 117737 070636 003502          MOVB   @IAR,TEMP        ;STORE ADDRESS
2698 112012 063737 002660 003502          ADD    BASE,TEMP        ;MAKE NEXT ADDRESS
2699 112020 023737 003502 003416          CMP    TEMP,ADO         ;CORRECT INTERRUPT ADDRESS ?
2700 112026 001406          BEQ    24$              ;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$:  MOVB   @ADO,TABLE1(R1)  ;GET DATA
2705 112052 117761 071342 112171          MOVB   @AD1,TABLE1+1(R1);GET DATA
2706 112060 032761 100000 112170          BIT    #BIT15,TABLE1(R1);OVER RANGE?
2707 112066 001403          BEQ    21$              ;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  
2718  
2719

TABLE1: .BLKW 42 ;SAVE 32(D) LOCATIONS AND MORE

```
2721 :*****
2722 :THIS ROUTINE WILL FIND THE AVERAGE OF THE DATA IN THE TABLE. THE NEW AVERAGE
2723 :IS RETURNED ON THE STACK.
2724 :CALL:
2725 :      JSR      PC,AVERG
2726 :      MOV      (SP)+,AVERAGE
2727 :*****
2728
2729 AVERG: MOV      R0,-(SP)      ;SAVE R0
2730      CLR      5$          ;CLEAR EXTRA BITS
2731      CLR      6$          ;CLEAR LSB'S
2732      CLR      7$          ;CLEAR MSB'S
2733      CLR      R0          ;SET UP OFFSET
2734      BIC      #BIT15,TABLE1(R0);STRIP OVERRANGE BIT
2735      BIT      #BIT14,TABLE1(R0);NEG VALUE ?
2736      BEQ      10$         ;NO, BRANCH
2737      BIC      #BIT14,TABLE1(R0);CLEAR SIGN BIT
2738      TST      TABLE1(R0) ;NEG ZERO ?
2739      BEQ      10$         ;YES, BRANCH
2740      NEG      TABLE1(R0) ;MAKE NEGATIVE
2741      ADD      #177777,7$  ;ADD IN SIGN BITS
2742      ADD      TABLE1(R0),6$ ;ANY OVERFLOW ?
2743      ADC      7$          ;ADD TO MSB'S
2744      ADD      #2,R0       ;STEP OFFSET
2745      CMP      #100,R0     ;END OF TABLE ?
2746      BNE      1$         ;NO, BRANCH
2747
2748      MOV      (SP)+,R0     ;RESTORE R0
2749      MOV      #5,4$       ;SET UP DIV BY 8
2750      CLC          ;
2751      ROR      7$          ;DIVIDE
2752      ROR      6$          ;BY
2753      ROR      5$          ;2
2754      DEC      4$          ;DECREASE THE COUNT
2755      BNE      2$         ;NOT DONE YET, BRANCH
2756      TST      6$          ;NEG VALUE ?
2757      BPL      11$        ;NO, BRANCH
2758      COM      6$          ;START 2'S COMPL
2759      COM      5$          ;MAKE 1'S COM
2760      ADD      #20000,5$   ;INC 5$
2761      ADC      6$          ;GET CARRY
2762      BIS      #BIT14,6$   ;INDICATE NEGATIVE
2763      TST      5$          ;MSB OF EXTRA SET ?
2764      BPL      3$         ;NO, BRANCH
2765      MOV      6$,5$       ;SAVE VALUE
2766      BIC      #140000,5$  ;STRIP SIGN BIT
2767      CMP      #37777,5$  ;MAX COUNT ?
2768      BEQ      3$         ;NO, BRANCH
2769      INC      6$          ;ADD HALF A BIT
2770      MOV      (SP),-(SP)  ;MOVE RETURN PC
2771      MOV      6$,2(SP)    ;PUT AVERAGE ON STACK
2772      RTS      PC
2773
2774      4$: .WORD 0          ;COUNT STORAGE
2775      5$: .WORD 0          ;EXTRA BITS STORAGE
2776      6$: .WORD 0          ;LSB'S STORAGE
2777      7$: .WORD 0          ;MSB'S STORAGE
```

```
2778  
2779  
2780 ;***** SET PRIORITY *****  
2781 112556 012737 000000 112666 PRI0: MOV #0,PRI0 ;SAVE PRIORITY  
2782 112564 000433 BR PRIOR  
2783 112566 012737 000040 112666 PRI1: MOV #40,PRI0 ;SAVE PRIORITY  
2784 112574 000427 BR PRIOR  
2785 112576 012737 000100 112666 PRI2: MOV #100,PRI0 ;SAVE PRIORITY  
2786 112604 000423 BR PRIOR  
2787 112606 012737 000140 112666 PRI3: MOV #140,PRI0 ;SAVE PRIORITY  
2788 112614 000417 BR PRIOR  
2789 112616 012737 000200 112666 PRI4: MOV #200,PRI0 ;SAVE PRIORITY  
2790 112624 000413 BR PRIOR  
2791 112626 012737 000240 112666 PRI5: MOV #240,PRI0 ;SAVE PRIORITY  
2792 112634 000407 BR PRIOR  
2793 112636 012737 000300 112666 PRI6: MOV #300,PRI0 ;SAVE PRIORITY  
2794 112644 000403 BR PRIOR  
2795 112646 012737 000340 112666 PRI7: MOV #340,PRI0 ;SAVE PRIORITY  
2796 112654 011646 PRIOR: MOV (SP),-(SP) ;SAVE RETURN PC  
2797 112656 013766 112666 000002 MOV PRIO,2(SP) ;SET RETURN PS  
2798 112664 000002 RTI  
2799 112666 000000 PRI0: 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
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 NOS: .ASCIZ /NO/

```

.MAIN. MACRO M1113 12-AUG-82 09:59 PAGE 147  
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.  
:\*\*\*\*\*

1  
2  
3  
4  
5  
6  
7  
8  
9

006360

.END START

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

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	NOS	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	PRI0	112666	STSTIH	051432	TMOUT	075640	TYPDSW	054316
MBIT =	000040	PRIOR	112654	STSTI2	046202	TMOVEC	007606	TYPE =	104401
MESCSS	027731	PRI0	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	SWO =	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
MO	030016	PR1 =	000040	SW03 =	000010	TSTONE	015104	VECTO	002650
M1	030012	PR2 =	000100	SW04 =	000020	TSTPT1	050136	VECTOA	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	\$BADDR	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