```
1
2
3

        .REM !
```

IDENTIFICATION

PRODUCT CODE: AC-F416A-MC

PRODUCT NAME: CXMNAA0 MNCAD (A/D) MODULE

PRODUCT DATE: SEPTEMBER 1978

MAINTAINER: DEC/X11 SUPPORT GROUP

1.0 ABSTRACT

MNA IS AN IOMOD THAT EXERCISES THE MNCAD (A/D) ANALOG MODULE. THIS MODULE REQUIRES ONLY AN ANALOG GROUND ON CHANNEL ZERO IN ORDER TO BE RUN, HOWEVER; WITH SPECIAL SETUP, MORE OPTIONS CAN BE CHOSEN. ONE OPTION IS THE USE OF THE MNCKW (REAL TIME CLOCK) DEVICE. THIS OPTION ALLOWS EXERCISING THE MNCAD ASYNCHRONOUSLY WITH THE LSI-11 CPU, THAT IS, MNCAD CONVERSIONS WILL BE STARTED AT RANDOM TIMES TO ALLOW FOR MAXIMUM BUS NOISE DURING THE CONVERSION. IF THIS OPTION IS SELECTED, YOU MUST DESELECT MNC FROM A DEC/X11 RUN AND ONLY ONE MNCAD (A/D) CAN BE RUN.

WITH NORMAL OPERATION, RMS NOISE AND PEAK NOISE ARE SAMPLED ON CHANNEL ZERO AND COMPARED AGAINST A LIMIT. WITH THE SECOND OPERATION OPTION, MORE CHANNELS MAY BE SPECIFIED TO RUN THE NOISE TESTS ON. THE THIRD OPTION ALLOWS FOR SAMPLING OF ONE TO ALL THE CHANNELS OF THE MNCAD. A CHECK IS MADE TO SEE THAT THE INPUT VOLTAGE REMAINS STABLE WITHIN AN ALLOWED TOLERANCE. LOCATIONS WITHIN THIS MODULE ARE PROVIDED TO CHANGE ANY LIMIT, OR TO FORCE TYPEOUT OF ANY VALUE. UP TO FOUR MNCAD (A/D)'S CAN BE EXERCISED WITH THIS MODULE. WHEN SR1 BITS 1 OR 2 ARE SET AND ADDITIONAL MNCAD'S ARE ENABLED, THE SAME NUMBER OF CHANNELS ON EACH MNCAD WILL BE TESTED. IF SR1 BIT 0 IS SET, ONLY ONE MNCAD CAN BE BE TESTED.

2.0 REQUIREMENTS

HARDWARE: ONE MNCAD (A/D)
ONE MNCKW (CLOCK) <OPTIONAL>

STORAGE: MNA REQUIRES:
DECIMAL WORDS: 901
OCTAL WORDS: 1605
OCTAL BYTES: 3412

3.0 PASS DEFINITION

ONE PASS OF THE MNA MODULE CONSISTS OF GENERATING 8244(DECIMAL) INTERRUPTS (CONVERSIONS).

4.0 EXECUTION TIME

ONE PASS OF THE MNA MODULE RUNNING ALONE WITH SR1 = 0, 1 OR 2, WILL TAKE APPROXIMATELY ONE MINUTE. THE TIME FOR A PASS WITH SR1 = 4 WILL DEPEND UPON THE NUMBER OF CHANNELS SPECIFIED IN LOCATION "NLSTCH".

5.0 CONFIGURATION REQUIREMENTS

DEFAULT PARAMETERS:

DEVADR: 171000, VECTOR: 400, BR1: 4

DEVCNT: 1, SR1: 0

REQUIRED PARAMETERS:

NONE IF SR1=0

IF SR1 BITS 0, 1 OR 2 = 1 THEN SEE OPERATION OPTIONS
IF SR1 BIT 0 IS SET, ONLY 1 MNCAD WILL BE TESTED.

6.0 DEVICE/OPTION SETUP

AN ANALOG GROUND MUST BE PUT ON CH. 0
(BY SET CH0 FRONT PANEL SWITCH TO "TEST").

SR1 BIT0=1 USE THE MNCKW OPTION FOR A TO D STARTS
(THE MNCKW <MNC> MODULE MUST BE DESELECTED)

SR1 BIT1=1 ALL CHANNELS SPECIFIED MUST HAVE
AN ANALOG GROUND.

SR1 BIT2=1 SELECT ADDITIONAL CHANNELS FOR NOISE TESTING.

7.0 MODULE OPERATION

1. (START) (RESTRT) INITLIZE POINTERS AND TYPE-OUT VALUES.

2. (LOOP) BIT EXERCISE CSR

3. (ADRMS1) PREFORM RMS NOISE CHECK ON SPECIFIED CHANNEL. WE FIRST USE SAR TO FIND THE DAC VALUE THAT PRODUCES A 16/84 SPLIT FOR THE LEFT BOUNDARY OF NOISE. THEN WE USE SAR TO FIND THE DAC VALUE THAT PRODUCES A 84/16 SPLIT FOR THE RIGHT VALUE. WE THEN SUBTRACT THE TWO DAC VALUES AND WE HAVE A VALUE FOR THE 68% AREA OF NOISE (RMS). IT IS THEN COMARED AGAINST THE ALLOWED LIMIT TO SEE IF EXCESSIVE NOISE IS ON THE CHANNEL.

4. (ADPK1) PREFORM PEAK NOISE CHECK ON SPECIFIED CHANNEL, WE FIRST USE SAR TO FIND THE DAC VALUE THAT PRODUCES A .6% SPLIT FOR THE LEFT BOUNDARY OF NOISE. THEN WE USE SAR TO FIND THE DAC VALUE THAT PRODUCES A .6% SPLIT FOR THE RIGHT BOUNDARY. WE THEN SUBTRACT THE TWO DAC VALUES AND WE HAVE A VALUE OF 98% AREA OF NOISE (PEAK). IT IS THEN COMPARED AGAINST THE ALLOWED LIMIT TO SEE IF EXCESSIVE NOISE IS ON THE CHANNEL.

5. IF MULTIPLE CHANNELS ARE SELECTED FOR NOISE TESTING TEST NEXT CHANNEL, IF SINGULAR RETEST CHAN. 0.

6. IF MULTI-CHANNEL SAMPLING IS SELECTED, TAKE SAMPLES ON EACH CHANNEL SPECIFIED, AND COMpare THE AVERAGE OF THE SAMPLES AGAINST THE OLD AVERAGE FOR THE CHANNEL. IF THE DIFFERENCE IS GREATER THAN THE TOLERANCE, REPORT THE DATA ON THAT CHANNEL.

7. REPORT END PASS IF LAST UNIT, BRANCH TO "LOOP" IF MORE UNITS.

8. (SAR) SAR Is A SUCCESSIVE APPROXIMATION ROUTINE. IT IS USED TO FIND A DAC VALUE THAT PRODUCES A DESIRED SPLIT. IT DOES THIS BY TRYING A DAC VALUE AND TAKING 512 CONVERSIONS ON THE A/D. IF THE AMOUNT OF THE SAMPLES IS LOWER THEN SPECIFIED IT INCREASES THE DAC VALUE, IF THE AMOUNT OF SAMPLES IS HIGHER THEN SPECIFIED, IT DECREASES THE DAC VALUE. IF THE END DAC VALUE IS EITHER 000 OR 377 WE HAVE A "WARPAROUND" ERROR. THIS OCCURS WHEN WE ARE UNABLE TO ADJUST THE DAC TO PRODUCE A DESIRED SPLIT, AND INDICATES EXCESSIVE NOISE ON A CHANNEL.

9. (RANDY) THIS IS A RANDOM NUMBER GENERATOR. IF THE MNCKW CLOCK OPTION IS SELECTED WE GET THE NUMBER THAT WE PUT INTO THE CLOCK PRESET REGISTER FROM THIS ROUTINE.

8.0 OPERATION OPTIONS

1. VALID SR1 VALUES

SR1 BIT	ENABLE/DISABLE	FUNCTION
0	0	INHIBIT USE OF MNCKW OPTION.
0	1	ENABLE USE OF MNCKW OPTION. NOTE: IF ENABLED, YOU MUST DESELECT "MNC" FROM DEC/X11 RUN AND ONLY 1 MNCAD WILL BE RUN.
1	0	INHIBIT SAMPLING ADDITIONAL CHANNELS FOR STABLE INPUT.
1	1	ENABLE SAMPLING CHANNEL ZERO THROUGH CHANNEL SPECIFIED BY CLSTCH FOR STABLE INPUT (+-) TOLERANCE SPECIFIED BY OFFALL
2	0	USE CHANNEL ZERO ONLY FOR NOISE TESTING.
2	1	USE CHANNEL ZERO THROUGH THE CHANNEL SPECIFIED IN NLSTCH FOR NOISE TESTING.

2. THE FOLLOWING ARE LOCATIONS WITHIN THIS MODULE THAT ENABLE THE USER TO CHANGE LIMITS AND SPECIFY CHANNELS.

LOCATION	FUNCTION
ARMLIM	SPECIFIES MAXIMUM LIMIT FOR RMS NOISE. MAY BE CHANGED TO ZERO TO FORCE TYPEOUT OF RMS NOISE ON A CHANNEL RUNNING IN A SYSTEM ENVIRONMENT.
APKLIM	SPECIFIES MAXIMUM LIMIT FOR PEAK NOISE. MAY BE CHANGED TO ZERO TO FORCE TYPEOUT OF PEAK NOISE ON A CHANNEL RUNNING IN A SYSTEM ENVIRONMENT.
CLSTCH	IF SR1 BIT1=1, USED TO SPECIFY END CHANNEL FOR SAMPLING STABLE INPUT.
OFFALL	IF SR1 BIT1=1, USED TO SPECIFY TOLERANCE OF STABLE CHANNEL. PRESET BY MODULE TO "000002".
NLSTCH	IF SR1 BIT2=1, USED TO SPECIFY LAST CHANNEL FOR NOISE TESTING.

9.0 NON-STANDARD PRINTOUTS

1. IF A CHANNEL HAS EXCESSIVE RMS NOISE, IT REPORTS IT IN AN ERROR CALL AND A MSGN CALL:

   (EXAMPLE)

   ```
   MNCAD (A/D) UNIT #0  ON CHAN 00
   A/D RMS NOISE = 0.52 LSB (LIMIT = .20 LSB)
   ```

2. IF A CHANNEL HAS EXCESSIVE PEAK NOISE, IT REPORTS IT IN AN ERROR CALL AND A MSGN CALL:

   (EXAMPLE)

   ```
   MNCAD (A/D) UNIT #0   ON CHAN 00
   A/D PEAK NOISE = 2.57 LSB (LIMIT = 2.00 LSB)
   ```

3. IF THERE IS AN EXCESSIVE AMOUNT OF NOISE SO THAT THE DAC CANNOT BE ADJUSTED, IT REPORTS IT IN AN ERROR CALL AND A MSGN CALL:

   (EXAMPLE)

   ```
   MNCAD (A/D) UNIT #0
   A/D PEAK WRAPAROUND ERROR ON CHAN  00
   ```

4. IF A CHANNEL IS FOUND TO BE UNSTABLE IN STABLE INPUT SAMPLEING, IT REPORTS IT IN A MSGN CALL:

   (EXAMPLE)

   ```
   A/D ERROR ON CHAN 14; OLD VALUE = 4066  NEW VALUE = 4000
   !
   ```

```
                        ,TITLE  MNAA DEC/X11 SYSTEM EXERCISER MODULE
                ;       DDXCOM  VERSION  6       23-MAY-78
                                                .LIST   BIN
                                        ;*********************************************************************
      000000'                           BEGIN:
      000000' 047115  040501      040   MODNAM: .ASCII  /MNAA / ;MODULE NAME.
      000005'    000                    XFLAG:  .BYTE   OPEN            ;USED TO KEEP TRACK OF WBUFF USAGE
      000006' 171000                    ADDR:   171000+0                ;1ST DEVICE ADDR.
      000010' 000400                    VECTOR: 400+0                   ;1ST DEVICE VECTOR.
      000012'    300                    BR1:    .BYTE   PRTY6+0         ;1ST BR LEVEL.
      000013'    000                    BR2:    .BYTE   PRTY0+0         ;2ND BR LEVEL.
      000014' 000001                    DVID1:  0+1                     ;DEVICE INDICATOR 1.
      000016' 000000                    SR1:    OPEN                    ;SWITCH REGISTER 1
      000020' 000000                    SR2:    OPEN                    ;SWITCH REGISTER 2
      000022' 000000                    SR3:    OPEN                    ;SWITCH REGISTER 3
      000024' 000000                    SR4:    OPEN                    ;SWITCH REGISTER 4
                                        ;*********************************************************************
      000026' 140000                    STAT:   140000                  ;STATUS WORD.
      000030' 000274'                   INIT:   START                   ;MODULE START ADDR.
      000032' 000224'                   SPOINT: MODSP                   ;MODULE STACK POINTER.
      000034' 000000                    PASCNT: 0                       ;PASS COUNTER.
      000036' 000005                    ICONT:  5                       ;# OF ITERATIONS PER PASS=5
      000040' 000000                    ICOUNT: 0                       ;LOC TO COUNT ITERATIONS
      000042' 000000                    SOFCNT: 0                       ;LOC TO SAVE TOTAL SOFT ERRORS
      000044' 000000                    HRDCNT: 0                       ;LOC TO SAVE TOTAL HARD ERRORS
      000046' 000000                    SOFPAS: 0                       ;LOC TO SAVE SOFT ERRORS PER PASS
      000050' 000000                    HRDPAS: 0                       ;LOC TO SAVE HARD ERRORS PER PASS
      000052' 000000                    SYSCNT: 0                       ;# OF SYS ERRORS ACCUMULATED
      000054' 000000                    RANNUM: 0                       ;HOLDS RANDOM # WHEN RAND MACRO IS CALLED
      000056'                           CONFIG:                         ;RESERVED FOR MONITOR USE
      000056' 000000                    RES1:   0                       ;RESERVED FOR MONITOR USE
      000060' 000000                    RES2:   0                       ;RESERVED FOR MONITOR USE
      000062' 000000                    SVR0:   OPEN                    ;LOC TO SAVE R0.
      000064' 000000                    SVR1:   OPEN                    ;LOC TO SAVE R1.
      000066' 000000                    SVR2:   OPEN                    ;LOC TO SAVE R2.
      000070' 000000                    SVR3:   OPEN                    ;LOC TO SAVE R3.
      000072' 000000                    SVR4:   OPEN                    ;LOC TO SAVE R4.
      000074' 000000                    SVR5:   OPEN                    ;LOC TO SAVE R5.
      000076' 000000                    SVR6:   OPEN                    ;LOC TO SAVE R6.
      000100' 000000                    CSRA:   OPEN                    ;ADDR OF CURRENT CSR.
      000102'                           SBADR:                          ;ADDR OF GOOD DATA, OR
      000102' 000000                    ACSR:   OPEN                    ;CONTENTS OF CSR.
      000104'                           WASADR:                         ;ADDR OF BAD DATA, OR
      000104' 000000                    ASTAT:  OPEN                    ;STATUS REG CONTENTS.
      000106'                           ERRTYP:                         ;TYPE OF ERROR
      000106' 000000                    ASB:    OPEN                    ;EXPECTED DATA.
      000110' 000000                    AWAS:   OPEN                    ;ACTUAL DATA.
      000112' 000274'                   RSTRT:  RESTRT                  ;RESTART ADDRESS AFTER END OF PASS
      000114' 000000                    WDTO:   OPEN                    ;WORDS TO MEMORY PER ITERATION
      000116' 000000                    WDFR:   OPEN                    ;WORDS FROM MEMORY PER ITERATION
      000120' 000000                    INTR:   OPEN                    ;# OF INTERRUPTS PER ITERATION
      000122' 000000                    IDNUM:  0                       ;MODULE IDENTIFICATION NUMBER=0
      000224'                           MODSP:
                                        ;*********************************************************************
  333
```



```
  334                                   ;*OPTIONAL USER SUPPLIED VALUES
  335  000224' 000000             CLSTCH: .WORD   0               ;USER SUPPLIED LAST SAMPLED CH
  336  000226' 000002             OFFALL: .WORD   2               ;USER SUPPLIED TOLERANCE FOR SAMPLE
  337  000230' 000000             NLSTCH: .WORD   0               ;USER SUPPLIED LAST NOISE CH.
  338  000232' 000062             ARMLIM: 50.                     ;RMS NOISE LIMIT
  339  000234' 000310             APKLIM: 200.                    ;A/D PEAK NOISE LIMIT.
  340                                   ;*
  341                                   ;*REGISTER AND VECTOR ADDRESS
  342                                   ;*
  343  000236' 171000             ADSR:   .WORD   171000          ;A/D CSR ADDRESS
  344  000240'                    DAC:
  345  000240' 171002             ADBR:   .WORD   171002          ;DAC (WRITE ONLY) AND BUFFER REG (READ ONLY)
  346
  347                                   ;THE FOLLOWING ARE CLOCK ADDRESSES IF A MNCKW EXISTS.
  348
  349  000242' 171020             ASR:    .WORD   171020          ;CLOCK A CSR.
  350  000244' 171022             ABR:    .WORD   171022          ;CLOCK A PRESET BUFFER.
  351
  352                                   ;*
  353                                   ;*FLAGS, COUNTERS AND OTHER REGISTERS
  354                                   ;*
  355
  356  000246' 000000             WHO:    .WORD   0               ;0=RMS NOISE, 1=PEAK NOISE
  357  000250' 000000             INTFLG: .WORD   0               ;USED IN LOGIC TEST TO INDICATE AN A/D INTR.
  358  000252' 000000             FRED:   .WORD   0               ;USED TO HOLD CURRENT DAC VALUE.
  359  000254' 000000             EDGE:   .WORD   0               ;HOLD CURRENT EDGE VALUE.
  360  000256' 000000             TMP:    .WORD   0               ;TEMP WORKING AREA.
  361  000260' 000000             ARMX:   .WORD   0               ;USED TO HOLD RMS NOISE VALUE.
  362  000262' 000000             APKX:   .WORD   0               ;USED TO HOLD A/D PEAK NOISE VALUE.
  363  000264' 000000             NCCH:   .WORD   0               ;CURRENT NOISE CHAN.
  364  000266' 000000             CCH:    .WORD   0               ;CURRENT SAMPLE CHAN.
  365  000270' 000001             TEMP:   BIT0                    ;CURRENT UNIT BIT
  366  000272' 000400             VECTAD: 400                     ;A/D VECTOR
```

```
                                        ;*
                                        ;*START-UP AND INITILIZE SECTION
                                        ;*

000274'                         RESTRT:
000274'                         START:

000274' 012767  000001  177766  LOOP1:  MOV     #1,TEMP         ;LOAD UNIT BIT
000302' 016700  177500          MOV     ADDR,R0         ;GET BASE ADDR OF A/D.
000306' 010067  177724          MOV     R0,ADSR         ;FIX A/D CSR ADDR.
000312' 062700  000002          ADD     #2,R0           ;BUFFER REG=CSR+2.
000316' 010067  177716          MOV     R0,ADBR         ;FIX BUFFER REG ADDR.
000322' 016767  177462  177742  MOV     VECTOR,VECTAD   ;LOAD VECTOR VALUE
000330' 005067  177730          CLR     NCCH            ;CLEAR 1ST CHAN. FOR NOISE.

                                        ;*************************************************************
                                                        ;CONVERT ARMLIM TO ASCII AND
                                                        ;STORE AT DECIM
000334' 104421  000000' 000232'         BTOD$,BEGIN,ARMLIM,DECIM
000342' 002600'
                                        ;*************************************************************

000344' 116767  002232  002451          MOVB    DECIM+2,P7      ;NOW WE WILL PUT IT
000352' 116767  002225  002445          MOVB    DECIM+3,P7+2    ;INTO THE ASCII
000360' 116767  002220  002440          MOVB    DECIM+4,P7+3    ;MESSAGE THAT WE TYPE OUT.

                                        ;*************************************************************
                                                        ;CONVERT APKLIM TO ASCII AND
                                                        ;STORE AT DECIM
000366' 104421  000000' 000234'         BTOD$,BEGIN,APKLIM,DECIM
000374' 002600'
                                        ;*************************************************************

000376' 116767  002200  002431          MOVB    DECIM+2,P8      ;NOW WE WILL PUT IT
000404' 116767  002173  002425          MOVB    DECIM+3,P8+2    ;INTO THE ASCII MESSAGE
000412' 116767  002166  002420          MOVB    DECIM+4,P8+3    ;THAT WE TYPE OUT.

                                ;LOOP TO HERE IF MULTIPLE MNCAD'S
                                ;       FIX THE UNIT # FOR THE ERROR TYPEOUT
000420' 016700  177644          LOOP:   MOV     TEMP,R0         ;GET CURRENT UNIT #
000424' 005001                          CLR     R1              ;INIT VALUE
000426' 006200                  1$:     ASR     R0              ;MOVE IT
000430' 001402                          BEQ     2$              ;BR IF LAST
000432' 005201                          INC     R1              ;UPDATE COUNT
000434' 000774                          BR      1$
000436' 062701  000060          2$:     ADD     #60,R1          ;MAKE ASCII VALUE
000442' 110167  002442                  MOVB    R1,P10U         ;SAVE IN MESSAGE
```



```
                                        ;*
                                        ;*LOGIC TEST #1
                                        ;*IN THIS TEST WE WILL SEE IF THE A/D RESPONDS TO ITS
                                        ;*ADDR. IF NOT, A DEC/X11 "SYS ERROR..." WILL OCCUR
                                        ;*

000446' 005777  177564          LOG1:   TST     @ADSR           ;ADDRESS THE A/D

                                        ;*
                                        ;*LOGIC TEST #2
                                        ;*IN THIS TEST WE WILL SEE IF THE CLOCK RESPONDS TO ITS ADDR, ONLY
                                        ;*IF THE CLOCK OPTION IS SELECTED BY SR1.

000452'                         LOG2:
000452' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
000456' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000462' 032767  000001  177326          BIT     #BIT0,SR1       ;IS CLOCK OPTION SELECTED?
000470' 001402                          BEQ     LOG3            ;BR IF NOT TO NEXT TEST.

000472' 005777  177544                  TST     @ASR            ;CLOCK WAS SELECTED, WILL IT RESPOND?

                                        ;*
                                        ;*LOGIC TEST #3
                                        ;*IN THIS TEST WE WILL SEE IF THE A/D WILL INTERRUPT.
                                        ;*
000476' 005067  177546          LOG3:   CLR     INTFLG          ;CLEAR A/D DID INTR. FLAG.
000502' 012777  000566' 177562          MOV     #1$,@VECTAD     ;SET UP VECTOR ADDR.
000510' 012777  000101  177520          MOV     #101,@ADSR      ;START A CONVERSION, SHOULD INTERRUPT,
                                                                ; BEFORE BREAK TIME IS OVER.

000516' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
000522' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.

000526' 005767  177516                  TST     INTFLG          ;DID THE A/D INTERRUPT?
000532' 001024                          BNE     LOG4            ;IF SO, NEXT TEST
000534' 016767  177476  177336          MOV     ADSR,CSRA       ;SET ADDR. OF AD CSR FOR ERROR TYPEOUT.
000542' 017767  177470  177332          MOV     @ADSR,ACSR      ;RECORD CONTENTS OF CSR.
000550' 005077  177462                  CLR     @ADSR           ;STOP A/D
                                        ;*************************************************************
000554' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;A/D FAILED TO INTERRUPT
                                        ;*************************************************************
000562' 104410  000000'                 END$,BEGIN              ;DROP MODULE ON THIS FAILURE

                                        ;*A/D SHOULD INTR. TO HERE.
000566' 005077  177444          1$:     CLR     @ADSR           ;STOP A/D.
000572' 005777  177442                  TST     @ADBR           ;CLEAR CSR BIT07.
000576' 005267  177446                  INC     INTFLG          ;INDICATE THAT IT DID INTR.
000602' 000002                          RTI                     ;EXIT INTR.
```

```
                                        ;*LOGIC TEST #4
                                        ;*THIS TEST WILL ONLY BE DONE IF THE CLOCK OPTION IS SELECTED
                                        ;*IN THIS TEST WE WILL SEE IF THE OVERFLOW OF CLOCK 1 WILL
                                        ;*TRIGGER A CONVERSION IN THE A/D
                                        ;*
000604' 032767  000001  177204  LOG4:   BIT     #BIT0,SR1       ;IS THE CLOCK OPTION SELECTED?
000612' 001446                          BEQ     LOG5            ;IF NOT, GOTO NEXT TEST.

000614' 012777  177777  177422          MOV     #177777,@ABR    ;PRESET CLOCK FOR ALL ONES.
000622' 012777  000040  177406          MOV     #BIT5,@ADSR     ;SET OVERFLOW ENABLE IN A/D.
000630' 012777  000011  177404          MOV     #11,@ASR        ;START CLOCK, 1MHZ, GO.
000636' 005000                          CLR     R0              ;SET FOR DELAY LOOP.

000640'                         1$:
000640' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
000644' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000650' 105777  177366                  TSTB    @ASR            ;IS THE CLOCK OVERFLOW SET?
000654' 100402                          BMI     2$              ;YES=EXIT LOOP.
000656' 105200                          INCB    R0              ;NO,IS DELAY EXCEEDED?
000660' 001367                          BNE     1$              ;NO=CONTINUE DELAY.
000662'                         2$:
000662' 017767  177354  177214          MOV     @ASR,ASTAT      ;RECORD CONTENTS OF CLOCK CSR.
000670' 005077  177346                  CLR     @ASR            ;CLEAR THE CLOCK.

000674' 105777  177336                  TSTB    @ADSR           ;IS DONE FLAG SET?
000700' 100413                          BMI     LOG5            ;IF YES NEXT TEST.
                                                                ;IF NOT, ERROR
000702' 016767  177330  177170          MOV     ADSR,CSRA       ;RECORD A/D CSR ADDR.
000710' 017767  177322  177164          MOV     @ADSR,ACSR      ;RECORD CONTENTS OF A/D CSR.
                                        ;****************************************************************
000716' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;CLOCK OVERFLOW FAILED TO TRIGGER A/D CONVERSION
                                        ;****************************************************************

                                                                ;FOR THIS ERROR YOU MIGHT CHECK
                                                                ;TO SEE IF A OVERFLOW IS WIRED TO
                                                                ;A/D INPUT.
                                                                ;DROP THIS MODULE = FATAL ERROR.
000724' 104410  000000'                 END$,BEGIN              ;CAN'T CONTINUE IF OVERFLOW DOESN'T TRIGGER THE CONVERSI
```



```
                                        ;*
                                        ;*LOGIC TEST #5
                                        ;*IN THIS TEST WE CHECK THAT WHEN CHANNEL 0 IS CONVERTED
                                        ;*WITH THE MAINTENANCE BIT SET THE RESULT IS 0

000730'                         LOG5:
000730' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
000734' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000740' 005777  177274                  TST     @ADBR           ;FALSE READ OF A/D BUFFER
000744' 012777  001014' 177320          MOV     #2$,@VECTAD     ;SET UP INTERRUPT VECTOR
000752' 012777  000105  177256          MOV     #105,@ADSR      ;START A/D WITH MAINTENANCE SET
000760' 104400  000000'                 EXIT$,BEGIN                     ;EXIT TO MONITOR, MODULE WAIT FOR INTERRUPT.
000764' 005777  177250          1$:     TST     @ADBR           ;TEST FOR ALL 0'S RESULT
                                                                ;NOTE: INTERRUPT SERVICE WILL
                                                                ;BEGIN USING HERE AFTER A PIRQ.
000770' 001414                          BEQ     LOG6            ;IF RESULT IS ALL 0'S, GOTO NEXT TEST
000772' 016767  177240  177100          MOV     ADSR,CSRA       ;RECORD A/D CSR ADDRESS
001000' 017767  177232  177074          MOV     @ADSR,ACSR      ;RECORD CONTENTS OF A/D CSR
                                        ;****************************************************************
001006' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;FAILED TO GET ALL 0'S RESULT FROM CONVERSION
                                        ;****************************************************************

                                        ;*A/D INTERRUPTS TO HERE
001014'                         2$:
                                        ;----------------------------------------------------------------
001014' 000004  000000' 000764'         PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
                                        ;----------------------------------------------------------------

                                        ;*
                                        ;*LOGIC TEST #6
                                        ;*IN THIS TEST WE CHECK THAT WHEN CHANNEL 1 IS CONVERTED
                                        ;*WITH THE MAINTENANCE BIT SET THE RESULT IS 7777

001022' 012777  001074' 177242  LOG6:   MOV     #2$,@VECTAD     ;SET UP INTERRUPT VECTOR
001030' 012777  000505  177200          MOV     #505,@ADSR      ;START A/D WITH MAINTENANCE SET
001036' 104400  000000'                 EXIT$,BEGIN                     ;EXIT TO MONITOR, MODULE WAIT FOR INTERRUPT.
001042' 022777  007777  177170  1$:     CMP     #7777,@ADBR     ;TEST FOR ALL 1'S RESULT
001050' 001414                          BEQ     LOG7            ;IF RESULT IS ALL 1'S, GOTO NEXT TEST
001052' 016767  177160  177020          MOV     ADSR,CSRA       ;RECORD A/D CSR ADDRESS
001060' 017767  177152  177014          MOV     @ADSR,ACSR      ;RECORD CONTENTS OF A/D CSR
                                        ;****************************************************************
001066' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;FAILED TO GET ALL 1'S RESULT FROM CONVERSION
                                        ;****************************************************************
                                        ;*A/D INTERRUPTS TO HERE
001074'                         2$:
                                        ;----------------------------------------------------------------
001074' 000004  000000' 001042'         PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
                                        ;----------------------------------------------------------------
```

```
                                        ;*
                                        ;*LOGIC TEST #7
                                        ;*IN THIS TEST WE WILL SAMPLE ALL CHANNELS SELECTED
                                        ;*FOR TEST AND STORE AWAY THEIR RESULTS.

001102'                         LOG7:
001102' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
001106' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
001112' 012777  000200  177120          MOV     #200,@ADBR      ;LOAD CORRECTION DAC
001120' 005777  177114                  TST     @ADBR           ;FALSE READ OF A/D BUFFER.
001124' 005077  177106                  CLR     @ADSR           ;CLEAR A/D'S CSR
001130' 005067  177132                  CLR     CCH             ;START ON CH. 0.
001134' 012777  001262' 177130          MOV     #4$,@VECTAD     ;SET UP INTR. VECTOR.

001142' 012700  177770          1$:     MOV     #-8.,R0         ;SET TO DO 8 CONVERSIONS.
001146' 005001                          CLR     R1              ;R1 WILL CONTAIN SUM OF CONVERSIONS.
001150' 016767  177112  177100          MOV     CCH,TMP         ;GET CH. NUMBER.
001156' 000367  177074                  SWAB    TMP             ;PUT IN CORRECT CSR POSITION.
001162' 052767  000101  177066          BIS     #BIT6!BIT0,TMP  ;ADD INTR. ENABLE AND GO.
001170' 016777  177062  177040  2$:     MOV     TMP,@ADSR       ;START A/D.
001176' 104400  000000'                 EXIT$,BEGIN                     ;EXIT TO MONITOR, MODULE WAIT FOR INTERRUPT.
001202'                         3$:
001202' 067701  177032                  ADD     @ADBR,R1        ;SUM THIS RESULT.
                                                                ;NOTE INTR SERVICE WILL
                                                                ;BEING USE HERE AFTER A PIRQ
001206' 005200                          INC     R0              ;DONE 8 CONVERSIONS?
001210' 100767                          BMI     2$              ;NO = DO ANOTHER ONE.

001212' 006201                          ASR     R1              ;NOW WE MUST
001214' 006201                          ASR     R1              ;CALCULATE THE AVERAGE
001216' 006201                          ASR     R1              ;OF THE SAMPLES THAT
001220' 005501                          ADC     R1              ;WE JUST TOOK.
001222' 016702  177040                  MOV     CCH,R2          ;PICK UP CH. NUMBER.
001226' 006302                          ASL     R2              ;USE IT AS AN OFFSET.
001230' 010162  003162'                 MOV     R1,RECSAM(2)    ;STORE THIS AVERAGE.

001234' 005267  177026                  INC     CCH             ;UPDATE CH. NUMBER.
001240' 026767  177022  176762          CMP     CCH,NLSTCH      ;DONE ALL NOISE CHANNELS?
001246' 003735                          BLE     1$              ;NO=DO NEXT ONE.
001250' 026767  177012  176746          CMP     CCH,CLSTCH      ;DONE ALL STABLE CHS?
001256' 003731                          BLE     1$              ;NO =DO NEXT ONE.
001260' 000403                          BR      REST0           ;YES=GOTO NOISE TESTS.


                                        ;*A/D INTERRUPTS TO HERE

001262'                         4$:
                                        ;---------------------------------------------------------------
001262' 000004  000000' 001202'         PIRQ$,BEGIN,3$          ; QUEUE UP TO CONTINUE AT 3$ AND RTI
                                        ;---------------------------------------------------------------
```



```
001270' 005077  176742          REST0:  CLR     @ADSR           ;CLEAR A/D'S CSR.
001274' 016700  176772                  MOV     VECTAD,R0       ;SET VECTOR AND PSW
001300' 012720  002262'                 MOV     #WAINT,(R0)+    ;LOAD INTR. ADDR. INTO VECTOR ADDR.
001304' 116710  176502                  MOVB    BR1,(R0)        ;LOAD PRIORITY
001310' 005067  176750                  CLR     NCCH            ;INIT THE CHANNEL #
                                ;CALCULATE A/D RMS NOISE USING VERNIER DAC.

001314' 012700  000657          ADRMS1: MOV     #431.,R0        ;R0 = 84.13% OF 512 CONVERSIONS
001320' 016767  176740  176730          MOV     NCCH,TMP        ;GET THE CHAN #
001326' 000367  176724                  SWAB    TMP             ;PUT IN PROPER CSR POSITION.
001332' 052767  000100  176716          BIS     #100,TMP        ;ADD INTR. ENABLE.
001340' 016701  176720                  MOV     NCCH,R1         ;PICK UP CH. NUMBER.
001344' 006301                          ASL     R1              ;FORM AN OFFSET.
001346' 016167  003162' 176700          MOV     RECSAM(R1),EDGE ;GET EDGE VALUE FOR THIS CH.
001354' 005267  176666                  INC     WHO             ;INDICATE RMS NOISE TEST.
001360' 016777  176672  176650          MOV     TMP,@ADSR       ;CH,#0, AND INTERRUPT ENABLE
001366' 016701  176646                  MOV     DAC,R1          ;R1 = ADDRESS OF SAR DAC
001372' 004767  000552                  JSR     PC,SAR          ;GET DAC VALUE THAT PRODUCES 16/84 SPLIT
001376' 016705  176650                  MOV     FRED,R5         ;SAVE LEFT BOUNDARY
001402' 012700  000121                  MOV     #81.,R0         ;R0 = 15.87% OF 512 CONVERSIONS
001406' 004767  000536                  JSR     PC,SAR          ;GET DAC VALUE THAT PRODUCES 84/16 SPLIT
001412' 166705  176634                  SUB     FRED,R5         ;R5 = BREADTH OF NOISE @ 68% AREA
001416' 010567  176636                  MOV     R5,ARMX         ;SAVE FOR DAC NOISE CALCULATIONS
001422' 020567  176604                  CMP     R5,ARMLIM       ;< OR = RMS LIMIT?
001426' 003410                          BLE     ADPK1           ;IF WITHIN LIMIT THEN CONTINUE AT ADRMS2
001430' 004767  001052                  JSR     PC,ERCOM        ;GET ERROR PARAMETERS
                                                                ;ERROR PARAMETERS LOADED BY "ERCOM"
                                        ;***************************************************************
001434' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;;RMS NOISE ERROR=SEE NEXT TYPEOUT
                                        ;***************************************************************
001442' 104403  000000' 002606'         MSGN$,BEGIN,MSG1        ;ASCII MESSAGE CALL WITH COMMON HEADER
                                ;CALCULATE A/D PEAK NOISE USING VERNIER DAC.

001450' 012700  000775          ADPK1:  MOV     #509.,R0        ;FOR EAK OF 2 1/2 SIGMA
001454' 016767  176604  176574          MOV     NCCH,TMP        ;GET CHAN. NUMBER.
001462' 000367  176570                  SWAB    TMP             ;PUT IN CORRECT CSR POSITION.
001466' 052767  000100  176562          BIS     #100,TMP        ;ADD INTR. ENABLE
001474' 005067  176546                  CLR     WHO             ;INDICATE PEAK NOISE TEST.
001500' 016777  176552  176530          MOV     TMP,@ADSR       ;CH.AND INTERRUPT ENABLE, AND CH
001506' 016701  176526                  MOV     DAC,R1          ;R1 = ADDRESS OF SAR DAC
001512' 004767  000432                  JSR     PC,SAR          ;GET DAC VALUE THAT GIVES .6% LOW COUNT
001516' 016705  176530                  MOV     FRED,R5         ;R5 = LEFT BOUNDARY
001522' 012700  000003                  MOV     #3,R0           ;CHANGE SPLIT FOR RIGHT BOUNDARY
001526' 004767  000416                  JSR     PC,SAR          ;GET DAC VALUE THAT GIVES .6% HIGH COUNT
001532' 166705  176514                  SUB     FRED,R5         ;R5 = A/D PEAK TO PEAK NOISE
001536' 010567  176520                  MOV     R5,APKX         ;SAVE FOR DAC NOISE CALCULATIONS
001542' 020567  176466                  CMP     R5,APKLIM       ;< OR = A/D PEAK NOISE LIMIT?
001546' 003410                          BLE     ENDP            ;BRANCH IF NO ERROR
001550' 004767  000732                  JSR     PC,ERCOM        ;GET ERROR PARAMETERS
                                                                ;ERROR PARAMETERS LOADED BY "ERCOM"
                                        ;***************************************************************
001554' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;;PEAK NOISE ERROR=SEE NEXT TYPE OUT
                                        ;***************************************************************
001562' 104403  000000' 002630'         MSGN$,BEGIN,MSG5        ;ASCII MESSAGE CALL WITH COMMON HEADER
```

```
001570' 032767  000004  176220  ENDP:  BIT     #BIT2,SR1       ;DOING MULTIPLE NOISE CHANNELS?
001576' 001411                         BEQ     3$              ;NO - BYPASS
001600' 000402                         BR      2$
001602' 000167  177506          1$:    JMP     ADRMS1          ;NO DO  AGAIN.
001606' 005267  176452          2$:    INC     NCCH            ;POINT TO NEXT CH.
001612' 026767  176446  176410         CMP     NCCH,NLSTCH     ;EXCEED LAST NOISE CH?
001620' 101770                         BLOS    1$              ;NO-TEST THIS CH.
                               ;BY NOW THE PROGRAM HAS RUN THE LOGIC AND NOISE TESTS ON ALL SELECTED CHANNELS ON THIS U
001622' 032767  000002  176166  3$:    BIT     #BIT1,SR1       ;ARE WE DOING VOLTAGE SAMPLING?
001630' 001510                         BEQ     EENDP           ;NO - END PASS!
001632' 005067  176430                 CLR     CCH             ;YES - START WITH CH 0.
                               ;VOLTAGE SAMPLING HAS BEEN ENABLED -- NOW TEST IF BEFORE REPORTING EOP
001636' 026767  176424  176360  4$:    CMP     CCH,CLSTCH      ;DONE ALL CHANNELS?
001644' 003102                         BGT     EENDP           ;YES - REPORT END PASS.
001646' 005003                         CLR     R3              ;R3 WILL HOLD SAMPLE.
001650' 012700  000010                 MOV     #8.,R0          ;SET TO TAKE 8 SAMPLES
001654' 012777  001704' 176410         MOV     #6$,@VECTAD     ;SET VECTOR FOR INTERRUPT
001662' 016701  176400                 MOV     CCH,R1          ;GET CH. NUMBER
001666' 000301                         SWAB    R1              ;FIX RIGHT POSITION IN CSR.
001670' 052701  000101                 BIS     #101,R1         ;ADD INTR, ENABLE AND GO.
001674' 010177  176336          5$:    MOV     R1,@ADSR        ;START A/D.
001700' 104400  000000'                EXIT$,BEGIN                     ;EXIT TO MONITOR, MODULE WAIT FOR INTERRUPT.
001704'                        6$:
                               ;---------------------------------------------------------------
001704' 000004  000000' 001712'        PIRQ$,BEGIN,7$          ; QUEUE UP TO CONTINUE AT 7$ AND RTI
                               ;---------------------------------------------------------------

001712' 067703  176322          7$:    ADD     @ADBR,R3        ;ADD THIS SAMPLE TO TOTAL
001716' 005300                         DEC     R0              ;ALL DONE ALL SAMPLES?
001720' 001365                         BNE     5$              ;NO - DO NEXT ONE.

001722' 006203                         ASR     R3              ;YES NOW WE
001724' 006203                         ASR     R3              ;MUST DIVIDE BY
001726' 006203                         ASR     R3              ;8 TO GET THE
001730' 005503                         ADC     R3              ;SAMPLE AVERAGE.
001732' 016700  176330                 MOV     CCH,R0          ;NOW GET CH. NUMBER
001736' 006300                         ASL     R0              ;FORM AN OFFSET
001740' 010301                         MOV     R3,R1           ;SAVE NEW SAMPLE VALUE.
001742' 166003  003162'                SUB     RECSAM(R0),R3   ;GET THE DIFFERENCE BETWEEN
                                                               ;AND THE NEW ONE WE JUST GOT
001746' 100001                         BPL     8$              ;IF POSITIVE WE'RE OK.
001750' 005403                         NEG     R3              ;OTHERWISE MAKE IT POSTIVE.
001752' 020367  176250          8$:    CMP     R3,OFFALL       ;IS IT WITHIN TOLERANCE?
001756' 003431                         BLE     10$             ;YES DO NEXT CH

                                       ;***************************************************************
                                                               ;CONVERT CCH TO ASCII AND
                                                               ;STORE AT CHANN
001760' 104420  000000' 000266'        OTOA$,BEGIN,CCH,CHANN
001766' 003402'
                                       ;***************************************************************

001770' 016067  003162' 001364         MOV     RECSAM(R0),OLDS ;GET OLD VALUE
                                       ;***************************************************************
                                                               ;CONVERT OLDS TO ASCII AND
                                                               ;STORE AT OLDS
```



```
001776' 104420  000000' 003362'        OTOA$,BEGIN,OLDS,OLDS
002004' 003362'
                                       ;***************************************************************

002006' 010167  001360                 MOV     R1,NEWS         ;GET NEW VALUE
                                       ;***************************************************************
                                                               ;CONVERT NEWS TO ASCII AND
                                                               ;STORE AT NEWS
002012' 104420  000000' 003372'        OTOA$,BEGIN,NEWS,NEWS
002020' 003372'
                                       ;***************************************************************

002022' 010160  003162'                MOV     R1,RECSAM(R0)   ;PUT NEW INTO OLD.

                                       ;***************************************************************
002026' 104405  000000' 000000         HRDER$,BEGIN,NULL       ;REPORT NEW VALUE EXCEEDS THE ALLOWED TOLERANCE
                                       ;***************************************************************
002034' 104403  000000' 002672'        MSGN$,BEGIN,MSG12       ;ASCII MESSAGE CALL WITH COMMON HEADER

002042' 005267  176220          10$:   INC     CCH             ;LOOK AT NEXT CHAN.
002046' 000167  177564                 JMP     4$              ;GO TEST IT

002052' 032767  000001  175736  EENDP: BIT     #BIT0,SR1       ;TEST IF MNCKW ENABLED
002060' 001025                         BNE     1$              ;BR IF YES <DO NOT DO MULTIPLE UNITS>
002062' 006367  176202                 ASL     TEMP            ;MOVE OVER
002066' 022767  000020  176174         CMP     #BIT4,TEMP      ;TEST IF MAX. UNIT
002074' 001417                         BEQ     1$              ;BR IF MAX
002076' 036767  176166  175710         BIT     TEMP,DVID1      ;TEST IF SELECTED
002104' 001413                         BEQ     1$              ;BR IF NOT
002106' 062767  000004  176122         ADD     #4,ADSR         ;UPDATE ADDRESS
002114' 062767  000004  176116         ADD     #4,ADBR
002122' 062767  000010  176142         ADD     #10,VECTAD      ;UPDATE VECTOR
002130' 000167  176264                 JMP     LOOP            ;LOOP AND TEST NEXT UNIT
002134' 005067  176124          1$:    CLR     NCCH            ;INIT THE CHANNEL #
002140' 104413  000000'                ENDIT$,BEGIN            ;SIGNAL END OF ITERATION.
                                                               ;MONITOR SHALL TEST END OF PASS
002144' 000167  176124                 JMP     RESTRT

                               ;USING SUCCESSIVE APPROXIMATION AND VERNIER DAC DEFINED IN R1.

002150' 012702  000200          SAR:   MOV     #200,R2         ;R2 = MSB OF DAC
002154' 005011                         CLR     (R1)            ;GET RID OF 'ONES'
002156' 005067  176070                 CLR     FRED            ;START WITH ZERO DAC
002162' 060267  176064          BIT:   ADD     R2,FRED         ;TRY THIS BIT
002166' 016711  176060                 MOV     FRED,(R1)       ;LOAD DAC.
002172' 005003                         CLR     R3              ;INIT HIGH COUNT
002174' 012704  001000                 MOV     #512.,R4        ;R4 = # OF SAMPLES IN A BURST
002200'                        CONV:

002200' 032767  000001  175610         BIT     #BIT0,SR1       ;IS CLOCK SELECTED?
002206' 001004                         BNE     1$              ;YES GOTO HANDLER.

002210' 052777  000001  176020         BIS     #BIT0,@ADSR     ;NO - SET GO BIT IN A/D.
002216' 000420                         BR      2$              ;GOTO 2$

002220' 004767  000172          1$:    JSR     PC,RANDY        ;GET A RANDOM NUMBER.
```

```
768  002224' 005077  176012          CLR     @ASR            ;MAKE SURE THE CLOCK'S CSR IS CLEAR.
769  002230' 052767  177770  000242  BIS     #177770,RNA     ;MAKE SURE OF HIGH NUMBER.
770  002236' 016777  000236  176000  MOV     RNA,@ABR        ;SET CLOCK PRESET REG.
771  002244' 052777  000040  175764  BIS     #BIT5,@ADSR     ;SET OVERFLOW ENABLE.
772  002252' 012777  000011  175762  MOV     #11,@ASR        ;START CLOCK
773  002260'                          2$:
774  002260' 104400                   EXIT$                   ;RETURN TO MONITOR.
775  002262'                          WAINT:
776                                   ;-------------------------------------------------------------
777  002262' 000004  000000' 002270'  PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
778                                   ;-------------------------------------------------------------
779  002270' 027767  175744  175756  1$:     CMP     @ADBR,EDGE      ;> OR = EDGE?
780  002276' 103401                   BLO     2$              ;BRANCH IF <EDGE
781  002300' 005203                   INC     R3              ;COUNT IF > OR =EDGE
782  002302' 005304                   2$:     DEC     R4              ;COUNT THROUGH BURST
783  002304' 001335                   BNE     CONV            ;BRANCH IF NOT DONE
784  002306' 020300                   CMP     R3,R0           ;HIGH COUNT > % OF 512 CONVERSIONS?
785  002310' 003402                   BLE     3$              ;BRANCH TO LAVE BIT IN
786  002312' 160267  175734           SUB     R2,FRED         ;TAKE BIT OUT
787  002316' 006202                   3$:     ASR     R2              ;NEXT BIT
788  002320' 001320                   BNE     BIT             ;AND GO RESOLVE IT IF NOT DONE
789  002322' 005767  175724           TST     FRED            ;CHECK FOR ALL 'ZEROES'
790  002326' 001405                   BEQ     WRPERR          ;BRANCH IF INVALID RESULT
791  002330' 026727  175716  000377   CMP     FRED,#377       ;CHECK FOR ALL 'ONES'
792  002336' 001401                   BEQ     WRPERR          ;BRANCH IF INVALID RESULT
793  002340' 000207                   RTS     PC              ;RETURN TO ADRMS1 OR ADPK1.
794                                   ;VOLTAGE NEEDED TO PRODUCE # OF HIGH COUNTS SPECIFIED IS OUT OF THE
795                                   ;RANGE OF THE WRAPAROUND DAC.
796                                   ;CLEAR INTERRUPTS, PRINT MESSAGE AND DROP MODULE.
797
798  002342' 005077  175670           WRPERR: CLR     @ADSR           ;STOP A/D
799  002346' 004767  000134           JSR     PC,ERCOM        ;GET ERROR PARAMETERS
800                                                           ;ERROR PARAMETERS LOADED BY "ERCOM"
801                                   ;*************************************************************
802  002352' 104405  000000' 000000   HRDER$,BEGIN,NULL       ;;NOISE TEST WRAPAROUND ERROR
803                                   ;*************************************************************
804  002360' 005767  175662           TST     WHO             ;SEE WHICH TEST WE WERE DOING.
805  002364' 001004                   BNE     1$              ;WHO=1 THEN RMS NOISE TEST.
806  002366' 104403  000000' 002652'  MSGN$,BEGIN,MSG11       ;ASCII MESSAGE CALL WITH COMMON HEADER
807  002374' 000403                   BR      2$              ;GO AHEAD.
808  002376'                          1$:
809  002376' 104403  000000' 002716'  MSGN$,BEGIN,MSG13       ;ASCII MESSAGE CALL WITH COMMON HEADER
810  002404' 005267  175440           2$:     INC     HRDPAS          ;UPDATE THE ERROR COUNT.
811  002410' 005726                   TST     (SP)+           ;RESTORE STACK FROM THE JSR PC THAT
812                                                           ;GOT US TO "SAR".
813  002412' 000167  177152           JMP     ENDP            ;GET NEXT DIRECTIVE.
```



```
814                                   ;GENERATE A RANDOM NUMBER
815
816  002416' 066767  000060  000054  RANDY:  ADD     RNB,RNA         ;THESE FOLLOW SEQUENCE OF
817  002424' 066767  000054  000046           ADD     RNC,RNA         ; INSTRUCTIONS GENERATE
818  002432' 005567  000042           ADC     RNA             ; A RANDOM NUMBER
819  002436' 066767  000036  000036           ADD     RNA,RNB         ; TO BE USED
820  002444' 066767  000034  000030           ADD     RNC,RNB         ; TO BE LOADED
821  002452' 005567  000024           ADC     RNB             ; INTO THE CLOCK'S PRESET
822  002456' 066767  000016  000020           ADD     RNA,RNC         ; BUFFER, IF SELECTED FOR TEST.
823  002464' 066767  000012  000012           ADD     RNB,RNC         ; ONLY RANA WILL BE USED.
824  002472' 005567  000006           ADC     RNC
825  002476' 000207                   RTS     PC              ;RETURN TO "CONV"
826  002500' 063241                   RNA:    063241          ;RANDOM TO NUMBER A.
827  002502' 142315                   RNB:    142315          ;RANDOM NUMBER B.
828  002504' 127623                   RNC:    127623          ;RANDOM NUMBER C.
829
830                                   ;CONVERT NOISE RESULT TO DECIMAL
831
832  002506' 016767  175524  175364  ERCOM:  MOV     ADSR,CSRA       ;LOAD HEADER FOR ERRORN CALL
833  002514' 017767  175516  175360           MOV     @ADSR,ACSR
834  002522' 016767  175300  175354           MOV     STAT,ASTAT
835  002530' 010567  000044           MOV     R5,DECIM        ;STACK BINARY NOISE VALUE
836                                   ;*************************************************************
837                                                           ;CONVERT DECIM TO ASCII AND
838                                                           ;STORE AT DECIM
839  002534' 104421  000000' 002600'  BTOD$,BEGIN,DECIM,DECIM
840  002542' 002600'
841                                   ;*************************************************************
842  002544' 116767  000032  000363   MOVB    DECIM+2,VALUE   ;MOVE CONVERTED VALUES TO ASCII BUFFER
843  002552' 116767  000025  000357   MOVB    DECIM+3,VALUE+2 ;FOR TYPEOUT OF ERROR
844  002560' 116767  000020  000352   MOVB    DECIM+4,VALUE+3 ; ON RETURN
845                                   ;*************************************************************
846                                                           ;CONVERT NCCH TO ASCII AND
847                                                           ;STORE AT CHANN
848  002566' 104420  000000' 002264'  OTOA$,BEGIN,NCCH,CHANN
849  002574' 003402'
850                                   ;*************************************************************
851  002576' 000207                   RTS     PC              ; EXIT TO CALLER.
852
853  002600' 000003                   DECIM:  .BLKW   3
```

```
                                        ;ASCII MESSAGES AND POINTERS
002606' 003065'                 MSG1:   P10                     ;UNIT #
002610' 002744'                         P2                      ;ON CHAN
002612' 003406'                         CHANN+4                 ; (CHAN NUMBER 2 DIGITS)
002614' 002736'                         P1                      ; % A/D
002616' 002775'                         P4                      ; RMS
002620' 003011'                         P6                      ; NOISE =
002622' 003135'                         VALUE                   ; (CONVERTED BELOW) X,XX LSB (LIMIT =
002624' 003023'                         P7                      ; 0,50 LSB) "THIS VALUE WILL CHANGE IF OPERATOR CHANGES
002626' 177777                          -1                      ; MESSAGE TERMINATOR.

002630' 003065'                 MSG5:   P10                     ;UNIT #
002632' 002744'                         P2                      ;ON CHAN
002634' 003406'                         CHANN+4                 ; (CHAN NUMBER 2 DIGITS)
002636' 002736'                         P1                      ; % A/D
002640' 003003'                         P5                      ; PEAK
002642' 003011'                         P6                      ; NOISE =
002644' 003135'                         VALUE                   ; (CONVERTED VALUE) X,XX LSB (LIMIT =
002646' 003035'                         P8                      ; 2,00 LSB) "THIS VALUE WILL CHANGE IF OPERATOR CHANGES
002650' 177777                          -1                      ; MESSAGE TERMINATOR

002652' 003065'                 MSG11:  P10                     ;UNIT #
002654' 002736'                         P1                      ; % A/D
002656' 003003'                         P5                      ; PEAK
002660' 003113'                         P13                     ; WRAPAROUND
002662' 003127'                         P15                     ; ERROR
002664' 002744'                         P2                      ; ON CHAN
002666' 003406'                         CHANN+4                 ; (CHAN NUMBER 2 DIGITS)
002670' 177777                          -1                      ; MESSAGE TERMINATOR

002672' 003065'                 MSG12:  P10                     ;UNIT #
002674' 002736'                         P1                      ; % A/D
002676' 003127'                         P15                     ; ERROR
002700' 002744'                         P2                      ; ON CHAN
002702' 003406'                         CHANN+4                 ; (CHAN NUMBER 2 DIGITS)
002704' 002756'                         P3                      ; ;OLD VALUE =
002706' 003364'                         OLDS+2                  ; "A/D READING 4 DIGITS
002710' 003047'                         P9                      ; NEW VALUE =
002712' 003374'                         NEWS+2                  ; "NEW A/D READING 4 DIGITS
002714' 177777                          -1                      ; MESSAGE TERMINATOR.

002716' 003065'                 MSG13:  P10                     ;UNIT #
002720' 002736'                         P1                      ; % A/D
002722' 002775'                         P4                      ; RMS
002724' 003113'                         P13                     ; WRAPAROUND
002726' 003127'                         P15                     ; ERROR
002730' 002744'                         P2                      ; ON CHAN
002732' 003406'                         CHANN+4                 ; (CHAN NUMBER 2 DIGITS)
002734' 177777                          -1                      ; MESSAGE TERMINATOR.
```



```
002736' 040445  042057  000040  P1:     .ASCIZ  '%A/D '
002744' 047440  020116  044103  P2:     .ASCIZ  ' ON CHAN '
002752' 047101  000040
002756' 020073  046117  020104  P3:     .ASCIZ  '; OLD VALUE = '
002764' 040526  052514  020105
002772' 020075     000
002775'    122  051515  000040  P4:     .ASCIZ  'RMS '
003002'    000                         .BYTE
003003'    120  040505  020113  P5:     .ASCIZ  'PEAK '
003010'    000
003011'    116  044517  042523  P6:     .ASCIZ  'NOISE = '
003016' 036440  000040
003022'    000                         .BYTE
003023'    060  032456  020060  P7:     .ASCIZ  '0,50 LSB)'
003030' 051514  024502     000
003035'    062  030056  020060  P8:     .ASCIZ  '2,00 LSB)'
003042' 051514  024502     000

003047'    040  042516  020127  P9:     .ASCIZ  ' NEW VALUE = '
003054' 040526  052514  020105
003062' 020075     000
003065'    045  047115  040503  P10:    .ASCII  '%MNCAD (A/D) UNIT #'
003072' 020104  040450  042057
003100' 020051  047125  052111
003106' 021440
003110'    060     040     000  P10U:   .BYTE   60,40,0
003113'    127  040522  040520  P13:    .ASCIZ  'WRAPAROUND '
003120' 047522  047125  020104
003126'    000
003127'    105  051122  051117  P15:    .ASCIZ  'ERROR'
003134'    000

003135'    130  054056  020130  VALUE:  .ASCIZ  'X,XX LSB (LIMIT = '
003142' 051514  020102  046050
003150' 046511  052111  036440
003156' 000040
003160'    000                         .BYTE

         003162'                        .EVEN
003162' 000100                  RECSAM: .BLKW   64.              ;USED TO STORE A/D RESULTS ON UP TO 64. CHANS.

003362' 000003                  OLDS:   .BLKW   3               ;USED FOR STORE OF ASCII OF OLD SAMPLE VALUE.
003370' 000000                          .WORD   0
003372' 000003                  NEWS:   .BLKW   3               ;USED FOR STORE OF ASCII NEW SAMPLE VALUE.
003400' 000000                          .WORD   0
003402' 000003                  CHANN:  .BLKW   3               ;USED FOR STORAGE OF ASCII OF CHAN. NUMBER.
003410' 000000                          .WORD   0
         000001                         .END
```

```
ABR     000244R          350#    473*    770#
ACSR    000102R          320#    452*    493*    520*    542*    833*
ADBR    000240R          345#    379*    461     511     515     539     559*    560     573     683     742*    779
ADDR    000006R          286#    376
ADDR22= 001000           333#
ADPK1   001450R          626     635#
ADRMS1  001314R          609#    659
ADSR    000236R          343#    377*    422     443*    451     452     453*    460*    474*    489     492     493     513*
                         519     520     537*    541     542     561*    570*    602*    617*    640*    676*    741*    764*
                         771*    798*    832     833
APKLIM  000234R          339#    397     648
APKX    000262R          362#    647*
ARMLIM  000232R          338#    386     625
ARMX    000260R          361#    624*
ASB     000106R          324#
ASR     000242R          349#    435     475*    481     486     487*    768*    772*
ASTAT   000104R          322#    486*    834*
AWAS    000110R          325#
BEGIN   000000R          283#    386     397     430     431     446     447     455     457     479     480     495     502
                         509     510     514     522     528     538     544     549     557     558     571     599     630
                         632     653     655     677     680     704     712     720     727     729     746     777     802
                         806     809     839     848
BIT     002162R          755#    788
BIT0  = 000001           333#    365     432     470     569     734     761     764
BIT1  = 000002           333#    664
BIT10 = 002000           333#
BIT11 = 004000           333#
BIT12 = 010000           333#
BIT13 = 020000           333#
BIT14 = 040000           333#
BIT15 = 100000           333#
BIT2  = 000004           333#    656
BIT3  = 000010           333#
BIT4  = 000020           333#    737
BIT5  = 000040           333#    474     771
BIT6  = 000100           333#    569
BIT7  = 000200           333#
BIT8  = 000400           333#
BIT9  = 001000           333#
BREAK$= 104407           333#    430     431     446     447     479     480     509     510     557     558
BR1     000012R          288#    605
BR2     000013R          289#
BTOD$ = 104421           333#    386     397     839
CCH     000266R          364#    562*    567     583     587*    588     590     666*    668     673     691     704     731*
CDATA$= 104412           333#
CHANN   003402R          704     848     858     868     882     889     902     949#
CLSTCH  000224R          335#    590     668
CONFIG  000056R          308#
CONV    002200R          759#    783
CSRA    000100R          318#    451*    492*    519*    541*    832*
DAC     000240R          344#    618     641
DATCK$= 104411           333#
DATER$= 104404           333#
DECIM   002600R          386     390     391     392     397     401     402     403     835*    839     842     843     844
                         853#
DVID1   000014R          290#    739
```



```
EDGE    000254R          359#    615*    779
EENDP   002052R          665     669     734#
ENDIT$= 104413           333#    746
ENDP    001570R          649     656#    813
END$  = 104410           333#    457     502
ERCOM   002506R          627     650     799     832#
ERRTYP  000106R          323#
EXIT$ = 104400           333#    514     538     571     677     774
FRED    000252R          358#    620     623     643     646     754*    755*    756     786*    789     791
GETPA$= 104415           333#
GWBUF$= 104414           333#
HRDCNT  000044R          303#
HRDER$= 104405           333#    455     495     522     544     630     653     727     802
HRDPAS  000050R          305#    810*
ICONT   000036R          300#
ICOUNT  000040R          301#
IDNUM   000122R          330#
INIT    000030R          297#
INTFLG  000250R          357#    441*    449     462*
INTR    000120R          329#
LOG1    000446R          422#
LOG2    000452R          429#
LOG3    000476R          433     441#
LOG4    000604R          450     470#
LOG5    000730R          471     490     508#
LOG6    001022R          518     536#
LOG7    001102R          540     556#
LOOP    000420R          407#    744
LOOP1   000274R          375#
MAP22$= 104416           333#
MODNAM  000000R          284#
MODSP   000224P          298     331#
MSGN$ = 104403           333#    632     655     729     806     809
MSGS$ = 104402           333#
MSG$  = 104401           333#
MSG1    002606R          632     856#
MSG11   002652R          806     876#
MSG12   002672R          729     885#
MSG13   002716R          809     896#
MSG5    002630R          655     866#
NCCH    000264R          363#    381*    606*    610     613     636     660*    661     745*    848
NEW$    003372R          716*    720     893     947#
NLSTCH  000230R          337#    588     661
NULL  = 000000           333#    455     495     522     544     630     653     727     802
OFFALL  000226R          336#    698
OLD$    003362R          708*    712     891     945#
OPEN  = 000000           285     291     292     293     294     311     312     313     314     315     316     317     318
                         320     322     324     325     327     328     329     333#
OTOA$ = 104420           333#    704     712     720     848
PASCNT  000034R          299#
PIRQ$ = 000004           333#    528     549     599     680     777
POPSP = 005726           333#
POPSP2= 022626           333#
PRTY  = 000000           333#
PRTY0 = 000000           289     333#
PRTY1 = 000040           333#
```

```
PRTY2 = 000100            333#
PRTY3 = 000140            333#
PRTY4 = 000200            333#
PRTY5 = 000240            333#
PRTY6 = 000300            288     333#
PRTY7 = 000340            333#
PS    = 177776            333#
PSW   = 177776            333#
PUSH  = 005746            333#
PUSH2 = 024646            333#
P1      002736R           859     869     877     886     897     904#
P10     003065R           856     866     876     885     896     925#
P10U    003110R           414*    929#
P13     003113R           879     899     930#
P15     003127R           880     887     900     933#
P2      002744R           857     867     881     888     901     905#
P3      002756R           890     907#
P4      002775R           860     898     910#
P5      003003R           870     878     912#
P6      003011R           861     871     914#
P7      003023R           390*    391*    392*    863     917#
P8      003035R           401*    402*    403*    873     919#
P9      003047R           892     922#
RANDY   002416R           767     816#
RAND$ = 104417            333#
RANNUM  000054R           307#
RECSAM  003162R           585*    615     694     708     724*    943#
RESTRT  000274R           326     372#    748
REST0   001270R           592     602#
RES1    000056R           309#
RES2    000060R           310#
RNA     002500R           769*    770     816*    817*    818*    819     822     826#
RNB     002502R           816     819*    820*    821*    823     827#
RNC     002504R           817     820     822*    823*    824*    828#
RSTRT   000112R           326#
SAR     002150R           619     622     642     645     752#
SBADR   000102R           319#
SOFCNT  000042R           302#
SOFER$= 104406            333#
SOFPAS  000046R           304#
SPOINT  000032R           298#
SPSIZ = 000040              1#    331
SR1     000016R           291#    432     470     656     664     734     761
SR2     000020R           292#
SR3     000022R           293#
SR4     000024R           294#
START   000274R           297     373#
STAT    000026R           296#    834
SVR0    000062R           311#
SVR1    000064R           312#
SVR2    000066R           313#
SVR3    000070R           314#
SVR4    000072R           315#
SVR5    000074R           316#
SVR6    000076R           317#
SYSCNT  000052R           306#
```



```
TEMP    000270R           365#    375*    407     736*    737     739
TMP     000256R           360#    567*    568*    569*    570     610*    611*    612*    617     636*    637*    638*    640
TRPDFD= 000022            333#
VALUE   003135R           842#    843*    844*    862     872     936#
VECTAD  000272R           366#    380*    442*    512*    536*    563*    603     672*    743*
VECTOR  000010R           287#    380
WAINT   002262R           604     775#
WASADR  000104R           321#
WDFR    000116R           328#
WDTO    000114R           327#
WHO     000246R           356#    616*    639*    804
WRPERR  002342R           790     792     798#
XFLAG   000005R           285#
.     = 003412R           853#    942#    943#    945#    947#    949#

. ABS.  000000     000
        003412     001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 XMNAA0,XMNAA0/SOL/CRF:SYM=DDXCOM,XMNAA0
 RUN-TIME:  2  2  .3 SECONDS
 RUN-TIME RATIO: 147/5=26.8
 CORE USED:  7K  (13 PAGES)
```