```
1
2
3
```

```
.REM !
```

IDENTIFICATION

PRODUCT CODE: AC-E926B-MC

PRODUCT NAME: CXADBB0 AD-11K MODULE

PRODUCT DATE: SEPTEMBER 1978

MAINTAINER: DEC/X11 SUPPORT GROUP

1.0 ABSTRACT

ADB IS AN IOMOD THAT EXERCISES THE AD11K ANOLOG MODULE. THIS MODULE REQUIRES ONLY AN ANOLOG 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 KW11K (DUAL REAL TIME CLOCK) DEVICE. THIS OPTION ALLOWS EXERCISING THE AD11K ASSYNCRONUS WITH THE PDP-11 CPU, THAT IS, AD11K CONVERSIONS WILL BE STARTED AT RANDOM TIMES TO ALLOW FOR MAXIMUM BUS NOISE DURING THE CONVERSION. IF THIS OPTION IS SELECTED, YOU MUST DESELECT THE MODULE KWD FROM A DEC/X11 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 AD11K. A CHECK IS MADE TO SEE THAT THE INPUT VOLTAGE REMAINS STABLE WITHIN AN ALLOWED TOLERANCE. LOCATIONS WITHING THIS MODULE TO CHANGE ANY LIMIT, OR TO FORCE TYPEOUT OF ANY VALUE.

2.0 REQUIREMENTS

HARDWARE: ONE AD11K
ONE WRAPAROUND MODULE (OPTIONAL)
ONE KW11K (OPTIONAL)

STORAGE:: ADB REQUIRES:
1. DECIMAL WORDS: 881
2. OCTAL WORDS: 1561
3. OCTAL BYTES: 3342

3.0 PASS DEFINITION

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

4.0 EXECUTION TIME

ONE PASS OF THE ADB MODULE RUNNING ALONG TAKES APPROXIMATELY ONE MINUTE.

5.0 CONFIGURATION REQUIREMENTS

DEFAULT PARAMETERS:

DEVADR: 170400, VECTOR: 340, BR1: 6

DEVCNT: 1, SR1: 0

REQUIRED PARAMETERS:

NONE IF SR1=1

IF SR1 BITS 1 2=1 THEN SEE OPERATION OPTIONS

6.0 DEVICE/OPTION SETUP

SR1=000	AN ANOLOG GROUND MUST BE PAT ON CH. 0.
SR1 BIT0=1	THE KW11K OPTION MUST BE CONNECTED TO THE AD11K OPTION.
SR1 BIT1=1	ALL CHANNELS SPECIFIED MUST HAVE AN ANOLOG GROUND.

7.0 MODULE OPERATION

1. (START) BIT EXERCISE CSR
2. SET TEST CHANNEL TO ZERO
3. (RESTRT) PREFORM RMS NOISE CHECK ON SPECIFIED CHANNEL.
   (ADRMS1)
   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.
   9. (ADPK1) THIS IS A PEAK NOISE TEST.
   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

SEQ 0004

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.

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 KW11K CLOCK OPTION IS SELECTED WE GET THE NUMBER THAT WE PUT INTO THE CLOCK PRESET REGISTER FROM THIS ROUTINE.

8.0 OPERATIONS OPTIONS

1. VALID SR1 VALUES

SR1 BIT	ENABLE/DISABLE	FUNCTION
0	0	INHIBIT USE OF CLOCK OPTION.
0	1	ENABLE USE OF CLOCK OPTION. NOTE: IF ENABLED, YOU MUST DESELECT KWDA FROM DEC/X11 RUN.
1	0	INHIBIT SAMPLING OTHER CHANNELS FOR STABLE INPUT.
1	1	ENABLE SAMPLING CHANNEL ZERO THROUGH CHANNEL SPECIFIED BY CLSTCH (164) FOR STABLE INPUT (+-) TOLERANCE SPECIFIED BY OFFALL (170)
2	0	USE CHANNEL ZERO ONLY FOR NOISE TESTING.
2	1	USE CHANNEL ZERO THROUGH THE CHANNEL SPECIFIED IN NLSTCH (166) FOR NOISE TESTING.

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

LOCATION	APC	FUNCTION
ARMLIM	216	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	220	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	164	IF SR1 BIT1=1, USED TO SPECIFY END CHANNEL FOR SAMPLING STABLE INPUT.
OFFALL	170	IF SR1 BIT1=1, USED TO SPECIFY TOLERANCE OF STABLE CHANNEL. PRESET BY MODULE TO "000002".
NLSTCH	166	IF SR1 BIT2=1, USED TO SPECIFY END 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)

   ON CH. 00 A/D RMS NOISE=0.52 LSB (LIMIT=.25LSB)

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

   (EXAMPLE)

   ON CH. 00 A/D PEAK NOISE=2.57LSB(LIMIT=2.00LSB)

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)

   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)

   ON CHAN 14 OLD AVERAGE=4066 NEW AVERAGE=4000

```
                        !
                        .TITLE  ADBB DEC/X11 SYSTEM EXERCISER MODULE
                ;       DDXCOM  VERSION  6      23-MAY-78
                                .LIST   BIN
                        ;*********************************************************************
000000'                 BEGIN:
000000' 042101  041102      040 MODNAM: .ASCII  /ADBB / ;MODULE NAME.
000005'     000         XFLAG:  .BYTE   OPEN            ;USED TO KEEP TRACK OF WBUFF USAGE
000006' 170400          ADDR:   170400+0                ;1ST DEVICE ADDR.
000010' 000340          VECTOR: 340+0                   ;1ST DEVICE VECTOR.
000012'     300         BR1:    .BYTE   PRTY6+0         ;1ST BR LEVEL.
000013'     200         BR2:    .BYTE   PRTY4+0         ;2ND BR LEVEL.
000014' 000001          DVID1:  +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' 000300'         INIT:   START                   ;MODULE START ADDR.
000032' 000224'         SPOINT: MODSP                   ;MODULE STACK POINTER.
000034' 000000          PASCNT: 0                       ;PASS COUNTER.
000036' 000001          ICONT:  1                       ;# OF ITERATIONS PER PASS=1
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' 001130'         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' 000103          IDNUM:  103                     ;MODULE IDENTIFICATION NUMBER=103
000224'                 MODSP:
                        ;*********************************************************************
  331
  332
  333 000224' 000000    CLSTCH: .WORD   0               ;USER SUPPLIED LAST SAMPLED CH
  334 000226' 000000    NLSTCH: .WORD   0               ;USER SUPPLIED LAST NOISE CH.
  335 000230' 000002    OFFALL: .WORD   2               ;USER SUPPLIED TOLERANCE FOR SAMPLE
  336
  337                           ;*
  338                           ;*REGISTER AND VECTOR ADDRESS
  339                           ;*
  340 000232' 170400    ADSR:   .WORD   170400          ;A/D CSR ADDRESS
  341 000234'           DAC:
  342 000234' 170402    ADBR:   .WORD   170402          ;DAC (WRITE ONLY) AND BUFFER REG (READ ONLY)
  343
  344                           ;THE FOLLOWING ARE KW11K ADDRESSES IF A KW11K EXISTS.
  345
  346 000236' 170404    ASR:    .WORD   170404          ;CLOCK A CSR.
  347 000240' 170406    ABR:    .WORD   170406          ;CLOCK A PRESET BUFFER.
  348
  349                           ;*
  350                           ;*FLAGS, COUNTERS AND OTHER REGISTERS
  351                           ;*
  352
  353 000242' 000000    WHO:    .WORD   0               ;0=RMS NOISE, 1=PEAK NOISE
  354 000244' 000000    INTFLG: .WORD   0               ;USED IN LOGIC TEST TO INDICATE AN A/D INTR.
  355 000246' 000000    FRED:   .WORD   0               ;USED TO HOLD CURRENT DAC VALUE.
  356 000250' 000000    EDGR:   .WORD   0               ;HOLD CURRENT EDGE VALUE.
  357 000252' 000000    TMP:    .WORD   0               ;TEMP WORKING AREA.
  358 000254' 000000    ARMX:   .WORD   0               ;USED TO HOLD RMS NOISE VALUE.
  359 000256' 000062    ARMLIM: .WORD   50.             ;RMS NOISE LIMIT.
  360 000260' 000000    APKX:   .WORD   0               ;USED TO HOLD A/D PEAK NOISE VALUE.
  361 000262' 000310    APKLIM: .WORD   200.            ;A/D PEAK NOISE LIMIT.
  362 000264' 000000    NCCH:   .WORD   0               ;CURRENT NOISE CHAN.
  363 000266' 000000    CCH:    .WORD   0               ;CURRENT SAMPLE CHAN.
  364 000270' 000000    PASSCNT:.WORD   0               ;PASS CNT.
  365 000272' 000000    NUMBA1: .WORD   0
  366 000274' 000000    NUMBA2: .WORD   0
  367 000276' 000000    NUMBA3: .WORD   0
  368
  369                           ;*
  370
  371 000300' 012767  020064  177612  START:  MOV     #8244.,INTR     ;8244 INTERRUPTS/ITERATION
  372 000306' 012767  000100  177600          MOV     #64.,WDTO       ;64 WORDS TO MEM/ITERATION
  373 000314' 012767  000100  177574          MOV     #64.,WDFR       ;64 WORDS FROM /ITERATION
  374 000322' 016700  177460                  MOV     ADDR,R0         ;GET BASE ADDR OF A/D.
  375 000326' 010067  177700                  MOV     R0,ADSR         ;FIX A/D CSR ADDR.
  376 000332' 062700  000002                  ADD     #2,R0           ;BUFFER REG=CSR+2.
  377 000336' 010067  177672                  MOV     R0,ADBR         ;FIX BUFFER REG ADDR.
  378 000342' 062700  000002                  ADD     #2,R0           ;CLOCK CSR=BUFFER REG+2.
  379 000346' 010067  177664                  MOV     R0,ASR          ;FIX CLOCK CSR ADDR.
  380 000352' 062700  000002                  ADD     #2,R0           ;PRESET ADDR=CSR+2.
  381 000356' 010067  177656                  MOV     R0,ABR          ;FIX CLOCK PRESET REG ADDR.
  382 000362' 005067  177676                  CLR     NCCH            ;CLEAR 1ST CHAN. FOR NOISE.
  383
```

```
                                        ;*************************************************************
                                                                ;CONVERT ARMLIM TO ASCII AND
                                                                ;STORE AT DECIM
000366' 104421  000000' 000256'         BTOD$,BEGIN,ARMLIM,DECIM
000374' 002570'                         ;*************************************************************
                                        ;*************************************************************
000376' 116767  002170  002375          MOVB    DECIM+2,P7      ;NOW WE WILL PUT IT
000404' 116767  002163  002371          MOVB    DECIM+3,P7+2    ;INTO THE ASCII
000412' 116767  002156  002364          MOVB    DECIM+4,P7+3    ;MESSAGE THAT WE TYPE OUT.
                                        ;*************************************************************
                                                                ;CONVERT APKLIM TO ASCII AND
                                                                ;STORE AT DECIM
000420' 104421  000000' 000262'         BTOD$,BEGIN,APKLIM,DECIM
000426' 002570'                         ;*************************************************************
000430' 116767  002136  002355          MOVB    DECIM+2,P8      ;NOW WE WILL PUT IT
000436' 116767  002131  002351          MOVB    DECIM+3,P8+2    ;INTO THE ASCII MESSAGE
000444' 116767  002124  002344          MOVB    DECIM+4,P8+3    ;THAT WE TYPE OUT.


                                        ;*
                                        ;*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
                                        ;*

000452' 005777  177554          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.

000456'                         LOG2:
000456' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
000462' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000466' 032767  000001  177322          BIT     #BIT0,SR1       ;IS CLOCK OPTION SELECTED?
000474' 001402                          BEQ     LOG3            ;BR IF NOT TO NEXT TEST.

000476' 005777  177534                  TST     @ASR            ;CLOCK WAS SELECTED, WILL IT RESPOND?

                                        ;*
                                        ;*LOGIC TEST #3
                                        ;*IN THIS TEST WE WILL SEE IF THE A/D WILL INTERRUPT.
                                        ;*
000502' 005067  177536          LOG3:   CLR     INTFLG          ;CLEAR A/D DID INTR. FLAG.
000506' 012777  000600' 177274          MOV     #1$,@VECTOR     ;SET UP VECTOR ADDR.
000514' 012777  000101  177510          MOV     #101,@ADSR      ;START A CONVERSION, SHOULD INTERRUPT.
                                                                ; BEFORE BREAK TIME IS OVER.

000522' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
000526' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
```



```
000532' 005767  177506                  TST     INTFLG          ;DID THE A/D INTERRUPT?
000536' 001027                          BNE     LOG4            ;IF SO, NEXT TEST
000540' 016767  177466  177332          MOV     ADSR,CSRA       ;SET ADDR. OF AD CSR FOR ERROR TYPEOUT.
000546' 017767  177460  177326          MOV     @ADSR,ACSR      ;RECORD CONTENTS OF CSR.
000554' 005077  177452                  CLR     @ADSR           ;STOP A/D
000560' 012767  000011  177320          MOV     #11,ERRTYP      ;MISSING INTERRUPT
                                        ;*************************************************************
000566' 104405  000000' 000000          HRDERR$,BEGIN,NULL      ;A/D FAILED TO INTERRUPT
                                        ;*************************************************************
000574' 104410  000000'                 END$,BEGIN              ;

                                        ;*A/D SHOULD INTR. TO HERE.
000600' 005077  177426          1$:     CLR     @ADSR           ;STOP A/D.
000604' 005777  177424                  TST     @ADBR           ;CLEAR CSR BIT07.
000610' 005267  177430                  INC     INTFLG          ;INDICATE THAT IT DID INTR.
000614' 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
                                        ;*

000616' 032767  000001  177172  LOG4:   BIT     #BIT0,SR1       ;IS THE CLOCK OPTION SELECTED?
000624' 001451                          BEQ     LOG5            ;IF NOT, GOTO NEXT TEST.

000626' 012777  177777  177404          MOV     #177777,@ABR    ;PRESET CLOCK FOR ALL ONES.
000634' 012777  000040  177370          MOV     #BIT5,@ADSR     ;SET OVERFLOW ENABLE IN A/D.
000642' 012777  000011  177366          MOV     #11,@ASR        ;START CLOCK, 1MHZ, GO.
000650' 005000                          CLR     R0              ;SET FOR DELAY LOOP.

000652'                         1$:
000652' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
000656' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000662' 105777  177350                  TSTB    @ASR            ;IS THE CLOCK OVERFLOW SET?
000666' 100402                          BMI     2$              ;YES-EXIT LOOP.
000670' 105200                          INCB    R0              ;NO,IS DELAY EXCEEDED?
000672' 001367                          BNE     1$              ;NO-CONTINUE DELAY.
000674'                         2$:
000674' 017767  177336  177202          MOV     @ASR,ASTAT      ;RECORD CONTENTS OF CLOCK CSR.
000702' 005077  177330                  CLR     @ASR            ;CLEAR THE CLOCK.

000706' 105777  177320                  TSTB    @ADSR           ;IS DONE FLAG SET?
000712' 100416                          BMI     LOG5            ;IF YES NEXT TEST.
                                                                ;IF NOT, ERROR
000714' 016767  177312  177156          MOV     ADSR,CSRA       ;RECORD A/D CSR ADDR.
000722' 017767  177304  177152          MOV     @ADSR,ACSR      ;RECORD CONTENTS OF A/D CSR.
000730' 012767  000046  177150          MOV     #46,ERRTYP      ;CLOCK FAILED TO TRIGGER A/D
                                        ;*************************************************************
000736' 104405  000000' 000000          HRDERR$,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.
000744' 104410  000000'         ENDS,BEGIN              ;CAN'T CONTINUE IF OVERFLOW DOESN'T TRIGGER THE CONVERSI
                                ;*
                                ;*LOGIC TEST #5
                                ;*IN THIS TEST WE WILL SAMPLE ALL CHANNELS SELECTED
                                ;*FOR TEST AND STORE AWAY THEIR RESULTS.
000750'                 LOG5:
000750' 104407  000000'         BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
000754' 104407  000000'         BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000760' 005777  177250          TST     @ADBR           ;FALSE READ OF A/D BUFFER.
000764' 005077  177242          CLR     @ADSR           ;CLEAR A/D'S CSR
000770' 005067  177272          CLR     CCH             ;START ON CH. 0.
000774' 012777  001122' 177006  MOV     #4$,@VECTOR     ;SET UP INTR. VECTOR.
001002' 012700  177770  1$:     MOV     #-8.,R0         ;SET TO DO 8 CONVERSIONS.
001006' 005001          CLR     R1              ;R1 WILL CONTAIN SUM OF CONVERSIONS.
001010' 016767  177252  177234  MOV     CCH,    TMP     ;GET CH. NUMBER.
001016' 000367  177230          SWAB    TMP             ;PUT IN CORRECT CSR POSITION.
001022' 052767  000101  177222  BIS     #BIT6!BIT0,TMP  ;ADD INTR. ENABLE AND GO.
001030' 016777  177216  177174  2$:     MOV     TMP,@ADSR       ;START A/D.
001036' 104400  000000'         EXIT$,BEGIN                     ;EXIT TO MONITOR. MODULE WAIT FOR INTERRUPT.
001042'                 3$:
001042' 067701  177166          ADD     @ADBR,R1        ;SUM THIS RESULT.
                                ;NOTE INTR SERVICE WILL
                                ;BEING USE HERE AFTER A PIRQ
001046' 005200          INC     R0              ;DONE 8 CONVERSIONS?
001050' 100767          BMI     2$              ;NO - DO ANOTHER ONE.
001052' 006201          ASR     R1              ;NOW WE MUST
001054' 006201          ASR     R1              ;CALCULATE THE AVERAGE
001056' 006201          ASR     R1              ;OF THE SAMPLES THAT
001060' 005501          ADC     R1              ;WE JUST TOOK.
001062' 016702  177200  MOV     CCH,R2          ;PICK UP CH. NUMBER.
001066' 006302          ASL     R2              ;USE IT AS AN OFFSET.
001070' 010162  003112' MOV     R1,RECSAM(2)    ;STORE THIS AVERAGE.
001074' 005267  177166          INC     CCH             ;UPDATE CH. NUMBER.
001100' 026767  177162  177120  CMP     CCH,NLSTCH      ;DONE ALL NOISE CHANNELS?
001106' 003735          BLE     1$              ;NO-DO NEXT ONE.
001110' 026767  177152  177106  CMP     CCH,    CLSTCH  ;DONE ALL STABLE CHS?
001116' 003731          BLE     1$              ;NO -DO NEXT ONE.
001120' 000403          BR      RESTRT          ;YES-GOTO NOISE TESTS.
                                ;*A/D INTERRUPTS TO HERE
001122'                 4$:
                                ;-------------------------------------------------------------------
001122' 000004  000000' 001042' PIRQ$,BEGIN,3$          ; QUEUE UP TO CONTINUE AT 3$ AND RTI
                                ;-------------------------------------------------------------------
001130' 005077  177076  RESTRT: CLR     @ADSR           ;CLEAR A/D'S CSR.
001134' 005067  177130          CLR     PASSCNT
```



```
001140'                 1$:
001140' 016700  176644          MOV     VECTOR,R0       ;SET VECTOR AND PSW
001144' 012720  002066'         MOV     #WAINT,(R0)+    ;LOAD INTR. ADDR. INTO VECTOR ADDR.
001150' 116710  176636          MOVB    BR1,(R0)        ;LOAD PRIORITY
                                ;CALCULATE A/D RMS NOISE USING WRAPAROUND DAC.
001154' 012700  000657  ADRMS1: MOV     #431.,R0        ;R0 = 84.13% OF 512 CONVERSIONS
001160' 016767  177100  177064  MOV     NCCH,TMP        ;GET THE CHAN #
001166' 000367  177060          SWAB    TMP             ;PUT IN PROPER CSR POSITION.
001172' 052767  000100  177052  BIS     #100,TMP        ;ADD INTR. ENABLE
001200' 016701  177060          MOV     NCCH,R1         ;PICK UP CH. NUMBER.
001204' 006301          ASL     R1              ;FORM AN OFFSET.
001206' 016167  003112' 177034  MOV     RECSAM(1),EDGE  ;GET EDGE VALUE FOR THIS CH.
001214' 005267  177022          INC     WHO             ;INDICATE RMS NOISE TEST.
001220' 016777  177026  177004  MOV     TMP,@ADSR       ;CH.#0, AND INTERRUPT ENABLE
001226' 016701  177002          MOV     DAC,R1          ;R1 = ADDRESS OF SAR DAC
001232' 004767  000516          JSR     PC,SAR          ;GET DAC VALUE THAT PRODUCES 16/84 SPLIT
001236' 016705  177004          MOV     FRED,R5         ;SAVE LEFT BOUNDARY
001242' 012700  000121          MOV     #81.,R0         ;R0 = 15.87% OF 512 CONVERSIONS
001246' 004767  000502          JSR     PC,SAR          ;GET DAC VALUE THAT PRODUCES 84/16 SPLIT
001252' 166705  176770          SUB     FRED,R5         ;R5 = BREADTH OF NOISE @ 68% AREA
001256' 010567  176772          MOV     R5,ARMX         ;SAVE FOR DAC NOISE CALCULATIONS
001262' 020567  176770          CMP     R5,ARMLIM       ;< OR = RMS LIMIT?
001266' 003413          BLE     ADPK1           ;IF WITHIN LIMIT THEN CONTINUE AT ADRMS2
001270' 004767  001202          JSR     PC,ERCOM        ;GET ERROR PARAMETERS
                                ;ERROR PARAMETERS LOADED BY ".ERCOM"
001274' 012767  000031  176604  MOV     #31,ERRTYP      ;A/D NOISE ERROR
                                ;*****************************************************************
001302' 104405  000000' 000000  HRDER$,BEGIN,NULL       ;;RMS NOISE ERROR-SEE NEXT TYPEOUT
                                ;*****************************************************************
001310' 104403  000000' 002576' MSGN$,BEGIN,MSG1        ;ASCII MESSAGE CALL WITH COMMON HEADER
                                ;CALCULATE A/D PEAK NOISE USING WRAPAROUND DAC.
001316' 012700  000775  ADPK1:  MOV     #509.,R0        ;FOR EAK OF 2 1/2 SIGMA
001322' 016767  176736  176722  MOV     NCCH,TMP        ;GET CHAN. NUMBER.
001330' 000367  176716          SWAB    TMP             ;PUT IN CORRECT CSR POSITION.
001334' 052767  000100  176710  BIS     #100,TMP        ;ADD INTR.  ENABLE
001342' 005067  176674          CLR     WHO             ;INDICATE PEAK NOISE TEST.
001346' 016777  176700  176656  MOV     TMP,@ADSR       ;CH.AND INTERRUPT ENABLE, AND CH
001354' 016701  176654          MOV     DAC,R1          ;R1 = ADDRESS OF SAR DAC
001360' 004767  000370          JSR     PC,SAR          ;GET DAC VALUE THAT GIVES .6% LOW COUNT
001364' 016705  176656          MOV     FRED,R5         ;R5 = LEFT BOUNDARY
001370' 012700  000003          MOV     #3,R0           ;CHANGE SPLIT FOR RIGHT BOUNDARY
001374' 004767  000354          JSR     PC,SAR          ;GET DAC VALUE THAT GIVES .6% HIGH COUNT
001400' 166705  176642          SUB     FRED,R5         ;R5 = A/D PEAK TO PEAK NOISE
001404' 010567  176650          MOV     R5,APKX         ;SAVE FOR DAC NOISE CALCULATIONS
001410' 020567  176646          CMP     R5,APKLIM       ;< OR = A/D PEAK NOISE LIMIT?
001414' 003413          BLE     ENDP            ;BRANCH IF NO ERROR
001416' 004767  001054          JSR     PC,ERCOM        ;GET ERROR PARAMETERS
                                ;ERROR PARAMETERS LOADED BY "ERCOM"
001422' 012767  000031  176456  MOV     #31,ERRTYP      ;A/D NOISE ERROR
                                ;*****************************************************************
001430' 104405  000000' 000000  HRDER$,BEGIN,NULL       ;;PEAK NOISE ERROR-SEE NEXT TYPE OUT
```

```
                                ;*****************************************************************
001436' 104403  000000' 002616' MSGN$,BEGIN,MSG5        ;ASCII MESSAGE CALL WITH COMMON HEADER

001444'                 ENDP:
001444' 005267  176620          INC     PASSCNT         ;UPDATE PASS COUNT.
001450' 032767  000004  176340  BIT     #BIT2,SR1       ;DOING MULTIPLE NOISE CHANNELS?
001456' 001012          BNE     2$                      ;YES - GET THE NEXT ONE
001460' 026727  176604  000001 1$:     CMP     PASSCNT,#1      ;DONE ENOUGH PASSES?
001466' 001417          BEQ     3$                      ;YES-DO  ENDPASS.
001470' 032767  000001  176320  BIT     #BIT0,SR1       ;ARE WE CONNECTED TO THE KW11K OPTION?
001476' 001013          BNE     3$                      ;IF SO THE 1 MIN IS UP ON ONE PASS.

001500' 000167  177450          JMP     ADRMS1          ;NO DO  AGAIN.

001504' 005267  176554  2$:     INC     NCCH            ;POINT TO NEXT CH.
001510' 026767  176550  176510  CMP     NCCH,NLSTCH     ;EXCEED LAST NOISE CH?
001516' 101760          BLOS    1$                      ;NO-TEST THIS CH.
001520' 005067  176540          CLR     NCCH            ;YES - START AGAIN ON CH. 0.
001524' 000755          BR      1$                      ;GO TEST CH 0.

001526' 032767  000002  176262 3$:     BIT     #BIT1,SR1       ;ARE WE DOING VOLTAGE SAMPLING?
001534' 001503          BEQ     ENDP                    ;NO - END PASS!
001536' 005067  176524          CLR     CCH             ;YES - START WITH CH 0.

001542' 026767  176520  176454 4$:     CMP     CCH,CLSTCH      ;DONE ALL CHANNELS?
001550' 003075          BGT     ENDP                    ;YES - REPORT END PASS.
001552' 005003          CLR     R3                      ;R3  WILL  HOLD  SAMPLE.
001554' 012700  000010          MOV     #8.,R0          ;SET TO TAKE 8 SAMPLES
001560' 012777  001610' 176222  MOV     #6$,@VECTOR     ;SET VECTOR FOR INTERRUPT
001566' 016701  176474          MOV     CCH,R1          ;GET CH. NUMBER
001572' 000301          SWAB    R1                      ;FIX RIGHT POSITION IN CSR.
001574' 052701  000101          BIS     #101,R1         ;ADD INTR, ENABLE AND GO.
001600' 010177  176426  5$:     MOV     R1,@ADSR        ;START A/D.
001604' 104400  000000'         EXIT$,BEGIN                     ;EXIT TO MONITOR. MODULE WAIT FOR INTERRUPT.
001610'                 6$:
                                ;---------------------------------------------------------------
001610' 000004  000000' 001616' PTRQ$,BEGIN,7$          ; QUEUE UP TO CONTINUE AT 7$ AND RTI
                                ;---------------------------------------------------------------

001616' 067703  176412  7$:     ADD     @ADBR,R3        ;ADD THIS SAMPLE TO TOTAL
001622' 005300          DEC     R0                      ;ALL DONE ALL SAMPLES?
001624' 001365          BNE     5$                      ;NO - DO NEXT ONE.

001626' 006203          ASR     R3              ;YES NOW WE
001630' 006203          ASR     R3              ;MUST DIVIDE BY
001632' 006203          ASR     R3              ;8 TO GET THE
001634' 005503          ADC     R3              ;SAMPLE AVERAGE.
001636' 016700  176424  MOV     CCH,R0          ;NOW GET CH. NUMBER
001642' 006300          ASL     R0              ;FORM AN OFFSET
001644' 010301          MOV     R3,R1           ;SAVE NEW SAMPLE VALUE.
001646' 166003  003112' SUB     RECSAM(0),R3    ;GET THE DIFFERENCE BETWEEN
                                                ;AND THE NEW ONE WE JUST GOT
001652' 100001          BPL     8$              ;IF POSITIVE WE'RE OK.
001654' 005403          NEG     R3              ;OTHERWISE MAKE IT POSTIVE.
001656' 020367  176346  8$:     CMP     R3,OFFALL       ;IS IT WITHIN TOLERANCE?
001662' 003424          BLE     10$             ;YES DO NEXT CH
```



```
                                ;*****************************************************************
                                                        ;CONVERT CCH TO ASCII AND
                                                        ;STORE AT CHANN
001664' 104420  000000' 000266' OTOA$,BEGIN,CCH,CHANN
001672' 003332'
                                ;*****************************************************************

001674' 016067  003112' 176372  MOV     RECSAM(0),NUMBA2
                                ;*****************************************************************
                                                        ;CONVERT NUMBA2 TO ASCII AND
                                                        ;STORE AT OLDS
001702' 104420  000000' 000274' OTOA$,BEGIN,NUMBA2,OLDS
001710' 003312'
                                ;*****************************************************************


                                ;*****************************************************************
                                                        ;CONVERT NUMBA3 TO ASCII AND
                                                        ;STORE AT NEWS
001712' 104420  000000' 000276' OTOA$,BEGIN,NUMBA3,NEWS
001720' 003322'
                                ;*****************************************************************
001722' 010160  003112'         MOV     R1,RECSAM(R0)   ;PUT NEW INTO OLD.

001726' 104403  000000' 002654' MSGN$,BEGIN,MSG12        ;ASCII MESSAGE CALL WITH COMMON HEADER

001734' 005267  176326  10$:    INC     CCH             ;LOOK AT NEXT CHAN.
001740' 000167  177576          JMP     4$              ;GO TEST IT

001744'                 ENDP:
001744' 104413  000000'         ENDIT$,BEGIN            ;SIGNAL END OF ITERATION.
                                                        ;MONITOR SHALL TEST END OF PASS
001750' 000167  177154          JMP     RESTRT

                                ;USING SUCCESSIVE APPROXIMATION AND WRAPAROUND DAC DEFINED IN R1.

001754' 012702  000200  SAR:    MOV     #200,R2         ;R2 = MSB OF DAC
001760' 005011          CLR     (R1)                    ;GET RID OF "ONES"
001762' 005067  176260          CLR     FRED            ;START WITH ZERO DAC
001766' 060267  176254  BIT:    ADD     R2,FRED         ;TRY THIS BIT
001772' 016711  176250          MOV     FRED,(R1)       ;LOAD DAC.
001776' 005003          CLR     R3                      ;INIT HIGH COUNT
002000' 012704  001000          MOV     #512.,R4        ;R4 = # OF SAMPLES IN A BURST
002004'                 CONV:

002004' 032767  000001  176004  BIT     #BIT0,SR1       ;IS CLOCK SELECTED?
002012' 001004          BNE     1$                      ;YES GOTO HANDLER.

002014' 052777  000001  176210  BIS     #BIT0,@ADSR     ;NO - SET GO BIT IN A/D.
002022' 000420          BR      2$                      ;GOTO 2$

002024' 004767  000356  1$:     JSR     PC,RANDY        ;GET A RANDOM NUMBER.
002030' 005077  176202          CLR     @ASR            ;MAKE SURE THE CLOCK'S CSR IS CLEAR.
002034' 052767  177770  000426  BIS     #177770,RNA     ;MAKE SURE OF HIGH NUMBER.
002042' 016777  000422  176170  MOV     RNA,@ABR        ;SET CLOCK PRESET REG.
002050' 052777  000040  176154  BIS     #BIT5,@ADSR     ;SET OVERFLOW ENABLE.
```

```
  720  002056' 012777  000011  176152          MOV     #11,@ASR        ;START CLOCK
  721  002064'                                 2$:
  722  002064' 104400                          EXIT$                   ;RETURN TO MONITOR.
  723  002066'                                 WAINT:
  724                                          ;-------------------------------------------------------------
  725  002066' 000004  000000' 002074'        PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
  726                                          ;-------------------------------------------------------------
  727  002074' 027767  176134  176146  1$:    CMP     @ADBR,EDGE      ;> OR = EDGE?
  728  002102' 103401                          BLO     2$              ;BRANCH IF <EDGE
  729  002104' 005203                          INC     R3              ;COUNT IF > OR =EDGE
  730  002106' 005304                  2$:     DEC     R4              ;COUNT THROUGH BURST
  731  002110' 001335                          BNE     CONV            ;BRANCH IF NOT DONE
  732  002112' 020300                          CMP     R3,R0           ;HIGH COUNT > % OF 512 CONVERSIONS?
  733  002114' 003402                          BLE     3$              ;BRANCH TO LAVE BIT IN
  734  002116' 160267  176124                  SUB     R2,FRED         ;TAKE BIT OUT
  735  002122' 006202                  3$:     ASR     R2              ;NEXT BIT
  736  002124' 001320                          BNE     BIT             ;AND GO RESOLVE IT IF NOT DONE
  737  002126' 005767  176114                  TST     FRED            ;CHECK FOR ALL "ZEROES"
  738  002132' 001405                          BEQ     WRPERR          ;BRANCH IF INVALID RESULT
  739  002134' 026727  176106  000377          CMP     FRED,#377       ;CHECK FOR ALL "ONES"
  740  002142' 001401                          BEQ     WRPERR          ;BRANCH IF INVALID RESULT
  741  002144' 000207                          RTS     PC              ;RETURN TO ADRMS1 OR ADPK1.
  742                                  ;VOLTAGE NEEDED TO PRODUCE # OF HIGH COUNTS SPECIFIED IS OUT OF THE
  743                                  ;RANGE OF THE WRAPAROUND DAC.
  744                                  ;CLEAR INTERRUPTS, PRINT MESSAGE AND DROP MODULE.
  745
  746  002146' 005077  176060          WRPERR: CLR     @ADSR           ;STOP A/D
  747  002152' 004767  000320                  JSR     PC,ERCOM        ;GET ERROR PARAMETERS
  748                                          ;ERROR PARAMETERS LOADED BY "ERCOM"
  749                                          ;*************************************************************
  750  002156' 104405  000000' 000000          MRDER$,BEGIN,NULL       ;;NOISE TEST WRAPAROUND ERROR
  751                                          ;*************************************************************
  752  002164' 005767  176052                  TST     WHO             ;SEE WHICH TEST WE WERE DOING.
  753  002170' 001004                          BNE     1$              ;WHO=1 THEN RMS NOISE TEST.
  754  002172' 104403  000000' 002636'         MSGN$,BEGIN,MSG11       ;ASCII MESSAGE CALL WITH COMMON HEADER
  755  002200' 000403                          BR      2$              ;GO AHEAD.
  756  002202'                                 1$:
  757  002202' 104403  000000' 002676'         MSGN$,BEGIN,MSG13       ;ASCII MESSAGE CALL WITH COMMON HEADER
  758  002210' 005267  175630          2$:     INC     HRDCNT          ;UPDATE THE ERROR COUNT.
  759  002214' 005726                          TST     (6)+            ;RESTORE STACK FROM THE JSR PC THAT
  760                                          ;GOT US TO "SAR".
  761  002216' 000167  177222                  JMP     ENDP            ;GET NEXT DIRECTIVE.
  762
  763                                  ;GET THE AVERAGE OF 512 SAMPLES AND THEIR COUNT SPREAD
  764
  765  002222'                         AVER:
  766  002222' 005000                          CLR     R0              ;INIT R0 FOR RUNNING SUM
  767  002224' 012704  001000                  MOV     #512.,R4        ;BURST OF 512 CONVERSIONS
  768  002230' 012701  100000                  MOV     #BIT15,R1       ;INIT HIGH COUNT TO NEGATIVE #
  769  002234' 012702  040000                  MOV     #BIT14,R2       ;INIT LOW COUNT TO POSITIVE #
  770  002240'                         ECON:
  771
  772  002240' 032767  000001  175550          BIT     #BIT0,SR1       ;IS CLOCK SELECTED?
  773  002246' 001004                          BNE     1$              ;YES GOTO HANDLER.
  774
  775  002250' 052777  000001  175754          BIS     #BIT0,@ADSR     ;NO - SET GO BIT IN A/D.
```



```
  776  002256' 000420                          BR      2$              ;GOTO 2$
  777
  778  002260' 004767  000122          1$:     JSR     PC,RANDY        ;GET A RANDOM NUMBER.
  779  002264' 005077  175746                  CLR     @ASR            ;MAKE SURE THE CLOCK'S CSR IS CLEAR.
  780  002270' 052767  177770  000172          BIS     #177770,RNA     ;MAKE SURE OF HIGH NUMBER.
  781  002276' 016777  000166  175734          MOV     RNA,@ABR        ;SET CLOCK PRESET REG.
  782  002304' 052777  000040  175720          BIS     #BIT5,@ADSR     ;SET OVERFLOW ENABLE.
  783  002312' 012777  000011  175716          MOV     #11,@ASR        ;START CLOCK
  784  002320'                                 2$:
  785  002320' 104400  000000'                 EXIT$,BEGIN                     ;EXIT TO MONITOR. MODULE WAIT FOR INTERRUPT.
  786  002324'                                 ERINT:
  787                                          ;-------------------------------------------------------------
  788  002324' 000004  000000' 002332'        PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
  789                                          ;-------------------------------------------------------------
  790  002332' 067700  175676          1$:     ADD     @ADBR,R0        ;R0 = RUNNING SUM OF DIFFERENCES
  791  002336' 160300                          SUB     R3,R0           ;OFFSET BY EXPECTED VALUE
  792  002340' 027701  175670                  CMP     @ADBR,R1        ;HIGHER THAN PRVIOUS HIGH?
  793  002344' 003402                          BLE     2$              ;BRANCH IF NOT A NEW HIGH
  794  002346' 017701  175662                  MOV     @ADBR,R1        ;NEW HIGH RESULT
  795  002352' 027702  175656          2$:     CMP     @ADBR,R2        ;LOWER THAN PREVIOUS LOW?
  796  002356' 002002                          BGE     3$              ;BRANCH IF NO
  797  002360' 017702  175650                  MOV     @ADBR,R2        ;NEW LOW RESULT
  798  002364' 005304                  3$:     DEC     R4              ;COUNT THROUGH BURST
  799  002366' 001324                          BNE     ECON            ;DO ANOTHER SAMPLE IF NOT DONE
  800  002370' 000300                          SWAB    R0              ;DIVIDE SUM OF DIFFERENCES BY 256
  801  002372' 110000                          MOVB    R0,R0           ;SIGN EXTEND
  802  002374' 006200                          ASR     R0              ;DIVIDE BY 2 MORE
  803  002376' 005500                          ADC     R0              ;ROUND UP FOR > 1/2
  804  002400' 060300                          ADD     R3,R0           ;OFFSET AVERAGE BY EXPTECTED RESULT
  805  002402' 160201                          SUB     R2,R1           ;R1 = COUNT SPREAD
  806  002404' 000207                          RTS     PC
  807
  808                                  ;GENERATE A RANDOM NUMBER
  809
  810  002406' 066767  000060  000054  RANDY:  ADD     RNB,RNA         ;THESE FOLLOW SEQUENCE OF
  811  002414' 066767  000054  000046          ADD     RNC,RNA         ; INSTRUCTIONS GENERATE
  812  002422' 005567  000042                  ADC     RNA             ; A RANDOM NUMBER
  813  002426' 066767  000036  000036          ADD     RNA,RNB         ; TO BE USED
  814  002434' 066767  000034  000030          ADD     RNC,RNB         ; TO BE LOADED
  815  002442' 005567  000024                  ADC     RNB             ; INTO THE CLOCK'S PRESET
  816  002446' 066767  000016  000020          ADD     RNA,RNC         ; BUFFER, IF SELECTED FOR TEST.
  817  002454' 066767  000012  000012          ADD     RNB,RNC         ; ONLY RANA WILL BE USED.
  818  002462' 005567  000006                  ADC     RNC
  819  002466' 000207                          RTS     PC              ;RETURN TO "CONV"
  820  002470' 063241                  RNA:    063241                  ;RANDOM TO NUMBER A.
  821  002472' 142315                  RNB:    142315                  ;RANDOM NUMBER B.
  822  002474' 127623                  RNC:    127623                  ;RANDOM NUMBER C.
  823
  824
  825                                  ;CONVERT NOISE RESULT TO DECIMAL
  826  002476' 016767  175530  175374  ERCOM:  MOV     ADSR,CSRA       ;LOAD HEADER FOR ERRORN CALL
  827  002504' 017767  175522  175370          MOV     @ADSR,ACSR
  828  002512' 016767  175310  175364          MOV     STAT,ASTAT
  829  002520' 010567  175546                  MOV     R5,NUMBA1       ;
  830                                          ;*************************************************
  831                                          ;CONVERT NUMBA1 TO ASCII AND
```

```
                                                        ;STORE AT DECIM
002524' 104421  000000' 000272'         BTOD$,BEGIN,NUMBA1,DECIM
002532' 002570'
                                        ;*************************************************************
002534' 116767  000032  000323          MOVB    DECIM+2,VALUE   ;MOVE CONVERTED VALUES TO ASCII BUFFER
002542' 116767  000025  000317          MOVB    DECIM+3,VALUE+2 ;FOR TYPEOUT OF ERROR
002550' 116767  000020  000312          MOVB    DECIM+4,VALUE+3 ;ON RETURN
                                        ;*************************************************************
                                                        ;CONVERT NCCH TO ASCII AND
                                                        ;STORE AT CHANN
002556' 104420  000000' 000264'         OTOA$,BEGIN,NCCH,CHANN
002564' 003332'
                                        ;*************************************************************
002566' 000207                          RTS     PC              ; EXIT TO CALLER.

002570' 000003                  DECIM:  .BLKW   3

                                        ;ASCII MESSAGES AND POINTERS

002576' 002722'                 MSG1:   P2                      ;ON CHAN
002600' 003336'                         CHANN+4                 ; (CHAN NUMBER 2 DIGITS)
002602' 002714'                         P1                      ; % A/D
002604' 002753'                         P4                      ; RMS
002606' 002767'                         P6                      ; NOISE =
002610' 003065'                         VALUE                   ; (CONVERTED BELOW) X.XX LSB (LIMIT =
002612' 003001'                         P7                      ; 0.50 LSB) "THIS VALUE WILL CHANGE IF OPERATOR CHANGES
002614' 177777                          -1                      ; MESSAGE TERMINATOR.

002616' 002722'                 MSG5:   P2                      ;ON CHAN
002620' 003336'                         CHANN+4                 ; (CHAN NUMBER 2 DIGITS)
002622' 002714'                         P1                      ; % A/D
002624' 002761'                         P5                      ; PEAK
002626' 002767'                         P6                      ; NOISE =
002630' 003065'                         VALUE                   ; (CONVERTED VALUE) X.XX LSB (LIMIT =
002632' 003013'                         P8                      ; 2.00 LSB) "THIS VALUE WILL CHANGE IF OPERATOR CHANGES
002634' 177777                          -1                      ; MESSAGE TERMINATOR

002636' 002714'                 MSG11:  P1                      ; % A/D
002640' 002761'                         P5                      ; PEAK
002642' 003043'                         P13                     ; WRAPAROUND
002644' 003057'                         P15                     ; ERROR
002646' 002722'                         P2                      ; ON CHAN
002650' 003336'                         CHANN+4                 ; (CHAN NUMBER 2 DIGITS)
002652' 177777                          -1                      ; MESSAGE TERMINATOR

002654' 002714'                 MSG12:  P1                      ; % A/D
002656' 003057'                         P15                     ; ERROR
002660' 002722'                         P2                      ; ON CHAN
002662' 003336'                         CHANN+4                 ; (CHAN NUMBER 2 DIGITS)
002664' 002734'                         P3                      ;OLD VALUE =
002666' 003314'                         OLDS+2                  ;4A/D READING 4 DIGITS
002670' 003025'                         P9                      ; NEW VALUE =
002672' 003324'                         NEWS+2                  ;4NEW A/D READING 4 DIGITS
002674' 177777                          -1                      ; MESSAGE TERMINATOR.

002676' 002714'                 MSG13:  P1                      ; % A/D
```



```
002700' 002753'                         P4                      ; RMS
002702' 003043'                         P13                     ; WRAPAROUND
002704' 003057'                         P15                     ; ERROR
002706' 002722'                         P2                      ; ON CHAN
002710' 003336'                         CHANN+4                 ; (CHAN NUMBER 2 DIGITS)
002712' 177777                          -1                      ; MESSAGE TERMINATOR.

002714' 040445  042057  000040  P1:     .ASCIZ  "%A/D "
002722' 047440  020116  044103  P2:     .ASCIZ  " ON CHAN "
002730' 047101  000040
002734' 020073  046117  020104  P3:     .ASCIZ  "; OLD VALUE = "
002742' 040526  052514  020105
002750' 020075      000
002753'     122  051515  000040  P4:     .ASCIZ  "RMS "
002760'     000                         .BYTE
002761'     120  040505  020113  P5:     .ASCIZ  "PEAK "
002766'     000
002767'     116  044517  042523  P6:     .ASCIZ  "NOISE = "
002774' 036440  000040
003000'     000                         .BYTE
003001'     060  032456  020060  P7:     .ASCIZ  "0.50 LSB)"
003006' 051514  024502      000
003013'     062  030056  020060  P8:     .ASCIZ  "2.00 LSB)"
003020' 051514  024502      000

003025'     040  042516  020127  P9:     .ASCIZ  " NEW VALUE = "
003032' 040526  052514  020105
003040' 020075      000
003043'     127  040522  040520  P13:    .ASCIZ  "WRAPAROUND "
003050' 047522  047125  020104
003056'     000
003057'     105  051122  051117  P15:    .ASCIZ"ERROR"
003064'     000

003065'     130  054056  020130  VALUE:  .ASCIZ  "X.XX LSB (LIMIT = "
003072' 051514  020102  046050
003100' 046511  052111  036440
003106' 000040
003110'     000                         .BYTE

                                        .EVEN
003112' 000100                  RECSAM: .BLKW   64.              ;USED TO STORE A/D RESULTS ON UP TO 64. CHANS.

003312' 000003                  OLDS:   .BLKW   3               ;USED FOR STORE OF ASCII OF OLD SAMPLE VALUE.
003320' 000000                          .WORD   0
003322' 000003                  NEWS:   .BLKW   3               ;USED FOR STORE OF ASCII NEW SAMPLE VALUE.
003330' 000000                          .WORD   0
003332' 000003                  CHANN:  .BLKW   3               ;USED FOR STORAGE OF ASCII OF CHAN. NUMBER.
003340' 000000                          .WORD   0
        000001                          .END
```

```
ABR     000240R  347#  381*  467*  718*  781*
ACSR    000102R  316#  443*  487*  827*
ADBR    000234R  342#  377*  453   507   520   647   727   790   792   794   795   797
ADDR    000006R  284#  374
ADDR22= 001000   331#
ADPK1   001316R  577   588#
ADRMS1  001154R  560#  620
ADSR    000232R  340#  375*  413   434*  442   443   444*  452*  468*  483   486   487   508*
                 517*  549*  568*  593*  640*  712*  719*  746*  775*  782*  826   827
APKLIM  000262R  361#  399   601
APKX    000260R  360#  600*
ARMLIM  000256R  359#  387   576
ARMX    000254R  358#  575*
ASB     000106R  322#
ASR     000236R  346#  379*  426   469*  475   480   481*  716*  720*  779*  783*
ASTAT   000104R  320#  480*  828*
AVER    002222R  765#
AWAS    000110R  323#
BEGIN   000000R  281#  387   399   421   422   437   438   447   449   473   474   490   497
                 505   506   518   546   582   584   607   609   641   644   668   676   683
                 688   694   725   750   754   757   785   788   833   842
BIT     001766R  703#  736
BIT0  = 000001   331#  423   464   516   617   709   712   772   775
BIT1  = 000002   331#  628
BIT10 = 002000   331#
BIT11 = 004000   331#
BIT12 = 010000   331#
BIT13 = 020000   331#
BIT14 = 040000   331#  769
BIT15 = 100000   331#  768
BIT2  = 000004   331#  613
BIT3  = 000010   331#
BIT4  = 000020   331#
BIT5  = 000040   331#  468   719   782
BIT6  = 000100   331#  516
BIT7  = 000200   331#
BIT8  = 000400   331#
BIT9  = 001000   331#
BREAK$= 104407   331#  421   422   437   438   473   474   505   506
BR1     000012R  286#  556
BR2     000013R  287#
BTOD$ = 104421   331#  387   399   833
CCH     000266R  363#  509*  514   530   534*  535   537   630*  632   637   655   668   690*
CDATA$= 104412   331#
CHANN   003332R  668   842   852   861   874   880   892   935#
CLSTCH  000224R  333#  537   632
CONFIG  000056R  306#
CONV    002004R  707#  731
CSRA    000100R  316#  442*  486*  826*
DAC     000234R  341#  569   594
DATCK$= 104411   331#
DATER$= 104404   331#
DRCTW   002570R  387   392   393   394   399   402   403   404   833   836   837   838   847#
DVID1   000014R  288#
ECON    002240R  770#  799
EDGE    000250R  356#  566*  727
```



```
ENDP    001744R  629   633   693#
ENDTT$= 104413   331#  694
ENDP    001444R  602   611#  761
ENDS  = 104410   331#  449   497
ERCOM   002476R  578   603   747   826#
ERINT   002324R  786#
ERRTYP  000106R  321#  445*  488*  580*  605*
EXIT$ = 104400   331#  518   641   722   785
FRED    000246R  355#  571   574   596   599   702*  703*  704   734*  737   739
GETPA$= 104415   331#
GWBUF$= 104414   331#
HRDCNT  000044R  301#  758*
HRDER$= 104405   331#  447   490   582   607   750
HRDPAS  000050R  303#
ICUNT   000036R  298#
ICOUNT  000040R  299#
IDNUM   000122R  328#
INIT    000030R  295#
INTFLG  000244R  354#  432*  440   454*
INTR    000120R  327#  371*
LOG1    000452R  413#
LOG2    000456R  420#
LOG3    000502R  424   432#
LOG4    000616R  441   464#
LOG5    000750R  465   484   504#
MAP22$= 104416   331#
MODNAM  000000R  282#
MODSP   000224R  296   329#
MSGN$ = 104403   331#  584   609   688   754   757
MSGS$ = 104402   331#
MSG$  = 104401   331#
MSG1    002576R  584   851#
MSG11   002636R  754   869#
MSG12   002654R  688   877#
MSG13   002676R  757   887#
MSG5    002616R  609   860#
NCCH    000264R  362#  382*  561   564   589   622*  623   625*  842
NEWS    003322R  683   884   933#
NLSTCH  000226R  334#  535   623
NULL  = 000000   331#  447   490   582   607   750
NUMBA1  000272R  365#  829*  833
NUMBA2  000274R  369#  672*  676
NUMBA3  000276R  367#  683
OFFALL  000230R  335#  662
OLDS    003312R  676   882   931#
OPEN  = 000000   283   289   290   291   292   309   310   311   312   313   314   315   316
                 319   320   322   323   325   326   327   331#
OTOA$ = 104420   331#  668   676   683   842
PASCNT  000034R  297#
PASSCN  000270R  364#  550*  612*  615
PIRQ$ = 000004   331#  546   644   725   788
POPSP = 005726   331#
POPSP2= 022626   331#
PRTY  = 000000   331#
PRTY0 = 000000   331#
PRTY1 = 000040   331#
```

```
PRTY2 = 000100              331#
PRTY3 = 000140              331#
PRTY4 = 000200              287     331#
PRTY5 = 000240              331#
PRTY6 = 000300              286     331#
PRTY7 = 000340              331#
PS    = 177776              331#
PSW   = 177776              331#
PUSH  = 005746              331#
PUSH2 = 024646              331#
P1      002714R             853     862     869     877     887     895#
P13     003043R             871     889     916#
P15     003057R             872     878     890     919#
P2      002722R             851     860     873     879     891     896#
P3      002734R             881     898#
P4      002753R             854     888     901#
P5      002761R             863     870     903#
P6      002767R             855     864     905#
P7      003001R             392*    393*    394*    857     908#
P8      003013R             402*    403*    404*    866     910#
P9      003025R             883     913#
RANDV   002406R             715     778     810#
RANDS = 104417              331#
RANNUM  000054R             305#
RECSAM  003112R             532*    566     658     672     686*    929#
RFSTRT  001130R             324     539     549#    696
RES1    000056R             307#
RES2    000060R             308#
RNA     002470R             717*    718     780*    781     810*    811*    812*    813     816     820#
RNB     002472R             810     813*    814*    815*    817     821#
RNC     002474R             811     814     816*    817*    818*    822#
RSTRT   000112R             324#
SAR     001754R             570     573     595     598     700#
SBADR   000102R             317#
SOPCNT  000042R             300#
SOPERS= 104406              331#
SOPPAS  000046R             302#
SPOINT  000032R             296#
SPSIZ = 000040                1#    329
SR1     000016R             289#    423     464     613     617     628     709     772
SR2     000020R             290#
SR3     000022R             291#
SR4     000024R             292#
START   000300R             295     371#
STAT    000026R             294#    828
SVR0    000062R             309#
SVR1    000064R             310#
SVR2    000066R             311#
SVR3    000070R             312#
SVR4    000072R             313#
SVR5    000074R             314#
SVR6    000076R             315#
SYSCNT  000052R             304#
TMP     000252R             357#    514*    515*    516*    517     561*    562*    563*    568     589*    590*    591*    593
TRPDFD= 000022              331#
VALUE   003065R             836*    837*    838*    856     865     922#
```



```
VECTOR  000010R             285#    433*    510*    554     636*
WAINT   002066R             555     723#
WASADR  000104R             319#
WDFR    000116R             326#    373*
WDTO    000114R             325#    372*
WHO     000242R             353#    567*    592*    752
WRPERR  002146R             738     740     746#
XFLAG   000005R             263#
.     = 003342R             847#    928#    929#    931#    933#    935#

. ABS.  000000      000
        003342      001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 XADBB0,XADBB0/SOL/CRF:SYM=DDXCOM,XADBB0
 RUN-TIME:  2  2  .4 SECONDS
 RUN-TIME RATIO: 18/5=3.4
 CORE USED:  7K  (13 PAGES)
```