```
1
2
3
```

```
.REM !
```

IDENTIFICATION

PRODUCT CODE: AC-E923B-MC

PRODUCT NAME: CXADCB0 ADV11 MODULE

PRODUCT DATE: SEPTEMBER 1978

MAINTAINER: DEC/X11 SUPPORT GROUP

1.0 ABSTRACT

ADC IS AN IOMOD THAT EXERCISES THE ADV11 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 KWV11 (REAL TIME CLOCK) DEVICE. THIS OPTION ALLOWS EXERCISING THE ADV11 ASYNCHRONOUSLY WITH THE LSI-11 CPU, THAT IS, ADV11 CONVERSIONS WILL BE STARTED AT RANDOM TIMES TO ALLOW FOR MAXIMUM BUS NOISE DURING THE CONVERSION. IF THIS OPTION IS SELECTED, YOU MUST DESELECT KWE 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 ADV11. 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.

2.0 REQUIREMENTS

HARDWARE: ONE ADV11
ONE BERG CONNECTOR (OPTIONAL)
ONE KWV11 (OPTIONAL)

STORAGE:: ADC REQUIRES:
1. DECIMAL WORDS: 933
2. OCTAL WORDS: 1645
3. OCTAL BYTES: 3512

3.0 PASS DEFINITION

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

4.0 EXECUTION TIME

ONE PASS OF THE ADC MODULE RUNNING ALONE TAKES APPROXIMATELY ONE MINUTE.

5.0 CONFIGURATION REQUIREMENTS

DEFAULT PARAMETERS:

DEVADR: 170400, VECTOR: 400, 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 PUT ON CH. 0.
SR1 BIT0=1	THE KWV11 OPTION MUST BE CONNECTED TO THE ADV11 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

VALUE OF 98% AREA OF NOISE (PEAK). IT IS THEN COMPARED AGAINST THE ALLOWED LIMIT TO SEE IF EXCESSIVE NOISE IS ON THE
PAGE 4

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 KWV11 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 CLOCK OPTION.
0	1	ENABLE USE OF CLOCK OPTION. NOTE: IF ENABLED, YOU MUST DESELECT KWEA FROM DEC/X11 RUN.
1	0	INHIBIT SAMPLING OTHER 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	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  ADCB DEC/X11 SYSTEM EXERCISER MODULE
                ;       DDXCOM  VERSION  6       23-MAY-78
                                .LIST   BIN
                        ;*********************************************************************
000000'                 BEGIN:
000000' 042101  041103      040 MODNAM: .ASCII  /ADCB / ;MODULE NAME.
000005'     000         XFLAG:  .BYTE   OPEN            ;USED TO KEEP TRACK OF WBUFF USAGE
000006' 170400          ADDR:   170400+0                ;1ST DEVICE ADDR.
000010' 000400          VECTOR: 400+0                   ;1ST DEVICE VECTOR.
000012'     300         BR1:    .BYTE   PRTY6+0         ;1ST RX 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' 000274'         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:
000056' 000000          RES1:   0                       ;RESERVED FOR MONITOR USE
000060' 000000          RES2:   0                       ;RESERVED FOR MONITOR USE
000062' 000000          SVR0:   OPEN                    ;RESERVED FOR MONITOR USE
000064' 000000          SVR1:   OPEN                    ;LOC TO SAVE R0.
000066' 000000          SVR2:   OPEN                    ;LOC TO SAVE R1.
000070' 000000          SVR3:   OPEN                    ;LOC TO SAVE R2.
000072' 000000          SVR4:   OPEN                    ;LOC TO SAVE R3.
000074' 000000          SVR5:   OPEN                    ;LOC TO SAVE R4.
000076' 000000          SVR6:   OPEN                    ;LOC TO SAVE R5.
000100' 000000          CSRA:   OPEN                    ;LOC TO SAVE R6.
000102'                 SBADR:                          ;ADDR OF CURRENT CSR.
000102' 000000          ACSR:   OPEN                    ;ADDR OF GOOD DATA, OR
000104'                 WASADR:                         ;CONTENTS OF CSR.
000104' 000000          ASTAT:  OPEN                    ;ADDR OF BAD DATA, OR
000106'                 ERRTYP:                         ;STATUS REG CONTENTS.
000106' 000000          ASB:    OPEN                    ;TYPE OF ERROR
000110' 000000          AWAS:   OPEN                    ;EXPECTED DATA.
000112' 001272'         RSTRT:  RESTRT                  ;ACTUAL DATA.
000114' 000000          WDTO:   OPEN                    ;RESTART ADDRESS AFTER END OF PASS
000116' 000000          WDFR:   OPEN                    ;WORDS TO MEMORY PER ITERATION
000120' 000000          INTR:   OPEN                    ;WORDS FROM MEMORY PER ITERATION
                                                        ;# OF INTERRUPTS PER ITERATION
```



```
000122' 000120          IDNUM:  120                     ;MODULE IDENTIFICATION NUMBER=120
000224'                 MODSP:
                        ;*********************************************************************
  332
  333
  334 000224' 000000              ;*OPTIONAL USER SUPPLIED IN
  335 000226' 000000      CLSTCH: .WORD   0               ;USER SUPPLIED LAST SAMPLED CH
  336 000230' 000002      NLSTCH: .WORD   0               ;USER SUPPLIED LAST NOISE CH.
  337                     OFFALL: .WORD   2               ;USER SUPPLIED TOLERANCE FOR SAMPLE
  338
  339                             ;*
  340                             ;*REGISTER AND VECTOR ADDRESS
  341 000232' 170400              ;*
  342 000234'             ADSR:   .WORD   170400          ;A/D CSR ADDRESS
  343 000234' 170402      DAC:
  344                     ADBR:   .WORD   170402          ;DAC (WRITE ONLY) AND BUFFER REG (READ ONLY)
  345
  346                             ;THE FOLLOWING ARE KW11K ADDRESSES IF A KW11K EXISTS.
  347 000236' 170420
  348 000240' 170422      ASR:    .WORD   170420          ;CLOCK A CSR.
  349                     ABR:    .WORD   170422          ;CLOCK A PRESET BUFFER.
  350
  351                             ;*
  352                             ;*FLAGS, COUNTERS AND OTHER REGISTERS
  353                             ;*
  354 000242' 000000      WHO:    .WORD   0               ;0=RMS NOISE, 1=PEAK NOISE
  355 000244' 000000      INTFLG: .WORD   0               ;USED IN LOGIC TEST TO INDICATE AN A/D INTR.
  356 000246' 000000      FRED:   .WORD   0               ;USED TO HOLD CURRENT DAC VALUE.
  357 000250' 000000      EDGE:   .WORD   0               ;HOLD CURRENT EDGE VALUE.
  358 000252' 000000      TMP:    .WORD   0               ;TEMP WORKING AREA.
  359 000254' 000000      ARMX:   .WORD   0               ;USED TO HOLD RMS NOISE VALUE.
  360 000256' 000062      ARMLIM: .WORD   50.             ;RMS NOISE LIMIT.
  361 000260' 000000      APKX:   .WORD   0               ;USED TO HOLD A/D PEAK NOISE VALUE.
  362 000262' 000310      APKLIM: .WORD   200.            ;A/D PEAK NOISE LIMIT.
  363 000264' 000000      NCCH:   .WORD   0               ;CURRENT NOISE CHAN.
  364 000266' 000000      CCH:    .WORD   0               ;CURRENT SAMPLE CHAN.
  365 000270' 000000      PASSCNT:.WORD   0               ;PASS CNT.
  366 000272' 000000      TEMP1:  .WORD   0               ;TEMPORARY STORAGE FOR ASCII CONVERSION
  367
  368                             ;*
  369
  370 000274' 012767  020064  177616  START:  MOV     #8244.,INTR     ;8244 INTERRUPTS/ITERATION
  371 000302' 012767  013000  177604          MOV     #5632.,WDTO     ;5632 WORDS TO MEM/ITERATION
  372 000310' 012767  013000  177600          MOV     #5632.,WDFR     ;5632 WORDS FROM MEM/ITEARTION
  373 000316' 016700  177464                  MOV     ADDR,R0         ;GET BASE ADDR OF A/D.
  374 000322' 010067  177704                  MOV     R0,ADSR         ;FIX A/D CSR ADDR.
  375 000326' 062700  000002                  ADD     #2,R0           ;BUFFER REG=CSR+2.
  376 000332' 010067  177676                  MOV     R0,ADBR         ;FIX BUFFER REG ADDR.
  377 000336' 005067  177722                  CLR     NCCH            ;CLEAR 1ST CHAN. FOR NOISE.
  378
  379                             ;*************************************************************
  380
  381                                                             ;CONVERT ARMLIM TO ASCII AND
  382 000342' 104421  000000' 000256'         BTOD$,BEGIN,ARMLIM,DECIM ;STORE AT DECIM
  383 000350' 002740'
  384                             ;*************************************************************
```

SEQ 0009

```
                                                                ;*****************************************************************
                                                                ;CONVERT APKLIM TO ASCII AND
000374' 104421  000000' 000262'                                 BTOD$,BEGIN,APKLIM,DECIM
000402' 002740'                                                 ;STORE AT DECIM
                                                                ;*****************************************************************
000404' 116767  002332  002551                                  MOVB    DECIM+2,P8      ;NOW WE WILL PUT IT
000412' 116767  002325  002545                                  MOVB    DECIM+3,P8+2    ;INTO THE ASCII MESSAGE
000420' 116767  002320  002540                                  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
                                                                ;*
000426' 005777  177600          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.
000432'                         LOG2:
000432' 104407  000000'                                         BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
000436' 104407  000000'                                         BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000442' 032767  000001  177346                                  BIT     #BIT0,SR1       ;IS CLOCK OPTION SELECTED?
000450' 001402                                                  BEQ     LOG3            ;BR IF NOT TO NEXT TEST.

000452' 005777  177560                                          TST     @ASR            ;CLOCK WAS SELECTED, WILL IT RESPOND?

                                                                ;*
                                                                ;*LOGIC TEST #3
                                                                ;*IN THIS TEST WE WILL SEE IF THE A/D WILL INTERRUPT.
                                                                ;*
000456' 005067  177562          LOG3:                           CLR     INTFLG          ;CLEAR A/D DID INTR. FLAG.
000462' 012777  000554' 177320                                  MOV     #1$,@VECTOR     ;SET UP VECTOR ADDR.
000470' 012777  000101  177534                                  MOV     #101,@ADSR      ;START A CONVERSION, SHOULD INTERRUPT.
                                                                ; BEFORE BREAK TIME IS OVER.

000476' 104407  000000'                                         BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
000502' 104407  000000'                                         BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.

000506' 005767  177532                                          TST     INTFLG          ;DID THE A/D INTERRUPT?
000512' 001027                                                  BNE     LOG4            ;IF SO, NEXT TEST
000514' 016767  177512  177356                                  MOV     ADSR,CSRA       ;SET ADDR. OF AD CSR FOR ERROR TYPEOUT.
000522' 017767  177504  177352                                  MOV     @ADSR,ACSR      ;RECORD CONTENTS OF CSR.
000530' 005077  177476                                          CLR     @ADSR           ;STOP A/D
```


SEQ 0010

```
000534' 012767  000023  177344                                  MOV     #23,ERRTYP      ;DEV FAILED TO INTERRUPT
                                                                ;*****************************************************************
000542' 104405  000000' 000000                                  ERDER$,BEGIN,NULL       ;A/D FAILED TO INTERRUPT
                                                                ;*****************************************************************
000550' 104410  000000'                                         END$,BEGIN              ;

                                                                ;*A/D SHOULD INTR. TO HERE.
000554' 005077  177452          1$:                             CLR     @ADSR           ;STOP A/D.
000560' 005777  177450                                          TST     @ADBR           ;CLEAR CSR BIT07.
000564' 005267  177454                                          INC     INTFLG          ;INDICATE THAT IT DID INTR.
000570' 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
                                                                ;*
000572' 032767  000001  177216  LOG4:                           BIT     #BIT0,SR1       ;IS THE CLOCK OPTION SELECTED?
000600' 001451                                                  BEQ     LOG5            ;IF NOT, GOTO NEXT TEST.

000602' 012777  177777  177430                                  MOV     #177777,@ABR    ;PRESET CLOCK FOR ALL ONES.
000610' 012777  000040  177414                                  MOV     #BIT5,@ADSR     ;SET OVERFLOW ENABLE IN A/D.
000616' 012777  000011  177412                                  MOV     #11,@ASR        ;START CLOCK, 1MHZ, GO.
000624' 005000                                                  CLR     R0              ;SET FOR DELAY LOOP.
000626'                         1S:
000626' 104407  000000'                                         BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
000632' 104407  000000'                                         BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000636' 105777  177374                                          TSTB    @ASR            ;IS THE CLOCK OVERFLOW SET?
000642' 100402                                                  BMI     2S              ;YES-EXIT LOOP.
000644' 105200                                                  INCB    R0              ;NO,IS DELAY EXCEEDED?
000646' 001367                                                  BNE     1S              ;NO-CONTINUE DELAY.
000650'                         2S:
000650' 017767  177362  177226                                  MOV     @ASR,ASTAT      ;RECORD CONTENTS OF CLOCK CSR.
000656' 005077  177354                                          CLR     @ASR            ;CLEAR THE CLOCK.

000662' 105777  177344                                          TSTB    @ADSR           ;IS DONE FLAG SET?
000666' 100416                                                  BMI     LOG5            ;IF YES NEXT TEST.
                                                                ;IF NOT, ERROR
000670' 016767  177336  177202                                  MOV     ADSR,CSRA       ;RECORD A/D CSR ADDR.
000676' 017767  177330  177176                                  MOV     @ADSR,ACSR      ;RECORD CONTENTS OF A/D CSR.
000704' 012767  000046  177174                                  MOV     #46,ERRTYP      ;CLK FAILED TO TRIGGER A/D CONVERSION
                                                                ;*****************************************************************
000712' 104405  000000' 000000                                  ERDER$,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.
000720' 104410  000000'                                         END$,BEGIN              ;DROP THIS MODULE - FATAL ERROR.
                                                                                        ;CAN'T CONTINUE IF OVERFLOW DOESN'T TRIGGER THE CONVERSI


                                                                ;*
```

SEQ 0011

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

000724'                         LOG5:
000724' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
000730' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000734' 005777  177274                  TST     @ADBR           ;FALSE READ OF A/D BUFFER
000740' 012777  001016' 177042          MOV     #2$,@VECTOR     ;SET UP INTERRUPT VECTOR
000746' 012777  000105  177256          MOV     #105,@ADSR      ;START A/D WITH MAINTENANCE SET
000754' 104400  000000'                 EXIT$,BEGIN             ;EXIT TO MONITOR. MODULE WAIT FOR INTERRUPT.
000760' 005777  177250          1$:     TST     @ADBR           ;TEST FOR ALL 0'S RESULT
                                                                ;NOTE: INTERRUPT SERVICE WILL
                                                                ;BEGIN USING HERE AFTER A PIRQ.
000764' 001417                          BEQ     LOG6            ;IF RESULT IS ALL 0'S, GOTO NEXT TEST
000766' 016767  177240  177104          MOV     ADSR,CSRA       ;RECORD A/D CSR ADDRESS
000774' 017767  177232  177100          MOV     @ADSR,ACSR      ;RECORD CONTENTS OF A/D CSR
001002' 012767  000026  177076          MOV     #26,ERRTYP      ;A/D CONV. OUT OF SPEC
                                        ;*****************************************************
001010' 104405  000000' 000000          ERROR$,BEGIN,NULL       ;FAILED TO GET ALL 0'S RESULT FROM CONVERSION
                                        ;*****************************************************

001016'                         2$:     ;*A/D INTERRUPTS TO HERE
                                        ;-------------------------------------------------------------
001016' 000004  000000' 000760'         PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1S 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

001024' 012777  001104' 176756  LOG6:   MOV     #2$,@VECTOR     ;SET UP INTERRUPT VECTOR
001032' 012777  000505  177172          MOV     #505,@ADSR      ;START A/D WITH MAINTENANCE SET
001040' 104400  000000'                 EXIT$,BEGIN             ;EXIT TO MONITOR. MODULE WAIT FOR INTERRUPT.
001044' 022777  007777  177162  1$:     CMP     #7777,@ADBR     ;TEST FOR ALL 1'S RESULT
001052' 001417                          BEQ     LOG7            ;IF RESULT IS ALL 1'S, GOTO NEXT TEST
001054' 016767  177152  177016          MOV     ADSR,CSRA       ;RECORD A/D CSR ADDRESS
001062' 017767  177144  177012          MOV     @ADSR,ACSR      ;RECORD CONTENTS OF A/D CSR
001070' 012767  000026  177010          MOV     #26,ERRTYP      ;A/D CONV OUT OF SPEC
                                        ;*****************************************************
001076' 104405  000000' 000000          ERROR$,BEGIN,NULL       ;FAILED TO GET ALL 1'S RESULT FROM CONVERSION
                                        ;*****************************************************
001104'                         2$:     ;*A/D INTERRUPTS TO HERE
                                        ;-------------------------------------------------------------
001104' 000004  000000' 001044'         PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1S AND RTI
                                        ;-------------------------------------------------------------


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


SEQ 0012

```
001112'                         LOG7:
001112' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
001116' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
001122' 005777  177106                  TST     @ADBR           ;FALSE READ OF A/D BUFFER.
001126' 005077  177100                  CLR     @ADSR           ;CLEAR A/D'S CSR
001132' 005067  177130                  CLR     CCH             ;START ON CH. 0.
001136' 012777  001264' 176644          MOV     #4$,@VECTOR     ;SET UP INTR. VECTOR.

001144' 012700  177770          1$:     MOV     #-8.,R0         ;SET TO DO 8 CONVERSIONS.
001150' 005001                          CLR     R1              ;R1 WILL CONTAIN SUM OF CONVERSIONS.
001152' 016767  177110  177072          MOV     CCH,    TMP     ;GET CH. NUMBER.
001160' 000367  177066                  SWAB    TMP             ;PUT IN CORRECT CSR POSITION.
001164' 052767  000101  177060          BIS     #BIT6!BIT0,TMP  ;ADD INTR. ENABLE AND GO.
001172' 016777  177054  177032  2$:     MOV     TMP,@ADSR       ;START A/D.
001200' 104400  000000'                 EXIT$,BEGIN             ;EXIT TO MONITOR. MODULE WAIT FOR INTERRUPT.
001204'                         3$:
001204' 067701  177024                  ADD     @ADBR,R1        ;SUM THIS RESULT.
                                                                ;NOTE INTR SERVICE WILL
001210' 005200                          INC     R0              ;BEING USE HERE AFTER A PIRQ
001212' 100767                          BMI     2$              ;DONE 8 CONVERSIONS?
                                                                ;NO - DO ANOTHER ONE.
001214' 006201                          ASR     R1              ;NOW WE MUST
001216' 006201                          ASR     R1              ;CALCULATE THE AVERAGE
001220' 006201                          ASR     R1              ;OF THE SAMPLES THAT
001222' 005501                          ADC     R1              ;WE JUST TOOK.
001224' 016702  177036                  MOV     CCH,R2          ;PICK UP CH. NUMBER.
001230' 006302                          ASL     R2              ;USE IT AS AN OFFSET.
001232' 010162  003262'                 MOV     R1,RECSAM(2)    ;STORE THIS AVERAGE.

001236' 005267  177024                  INC     CCH             ;UPDATE CH. NUMBER.
001242' 026767  177020  176756          CMP     CCH,NLSTCH      ;DONE ALL NOISE CHANNELS?
001250' 003735                          BLE     1$              ;NO-DO NEXT ONE.
001252' 026767  177010  176744          CMP     CCH,    CLSTCH  ;DONE ALL STABLE CHS?
001260' 003731                          BLE     1$              ;NO -DO NEXT ONE.
001262' 000403                          BR      RESTRT          ;YES-GOTO NOISE TESTS.


                                        ;*A/D INTERRUPTS TO HERE
001264'                         4$:
                                        ;-------------------------------------------------------------
001264' 000004  000000' 001204'         PIRQ$,BEGIN,3$          ; QUEUE UP TO CONTINUE AT 3S AND RTI
                                        ;-------------------------------------------------------------

001272' 005077  176734          RESTRT: CLR     @ADSR           ;CLEAR A/D'S CSR.
001276' 005067  176766                  CLR     PASSCNT


001302'                         1$:
001302' 016700  176502                  MOV     VECTOR,R0       ;SET VECTOR AND PSW
001306' 012720  002230'                 MOV     #VAINT,(R0)+    ;LOAD INTR. ADDR. INTO VECTOR ADDR.
001312' 116710  176474                  MOVB    BR1,(R0)        ;LOAD PRIORITY


                                        ;CALCULATE A/D RMS NOISE USING VERNIER DAC.
```

```
001316' 012700  000657          ADRMS1: MOV     #431.,R0        ;R0 = 84.13% OF 512 CONVERSIONS
001322' 016767  176736  176722          MOV     NCCH,TMP        ;GET THE CHAN #
001330' 000367  176716                  SWAB    TMP             ;PUT IN PROPER CSR POSITION.
001334' 052767  000100  176710          BIS     #100,TMP        ;ADD INTR. ENABLE.
001342' 016701  176716                  MOV     NCCH,R1         ;PICK UP CH. NUMBER.
001346' 006301                          ASL     R1              ;FORM AN OFFSET.
001350' 016167  003262' 176672          MOV     RECSAM(1),EDGE  ;GET EDGE VALUE FOR THIS CH.
001356' 005267  176660                  INC     WHO             ;INDICATE RMS NOISE TEST.
001362' 016777  176664  176642          MOV     TMP,@ADSR       ;CH.#0, AND INTERRUPT ENABLE
001370' 016701  176640                  MOV     DAC,R1          ;R1 = ADDRESS OF SAR DAC
001374' 004767  000516                  JSR     PC,SAR          ;GET DAC VALUE THAT PRODUCES 16/84 SPLIT
001400' 016705  176642                  MOV     FRED,R5         ;SAVE LEFT BOUNDARY
001404' 012700  000121                  MOV     #81.,R0         ;R0 = 15.87% OF 512 CONVERSIONS
001410' 004767  000502                  JSR     PC,SAR          ;GET DAC VALUE THAT PRODUCES 84/16 SPLIT
001414' 166705  176626                  SUB     FRED,R5         ;R5 = BREADTH OF NOISE @ 68% AREA
001420' 010567  176630                  MOV     R5,ARMX         ;SAVE FOR DAC NOISE CALCULATIONS
001424' 020567  176626                  CMP     R5,ARMLIM       ;< OR = RMS LIMIT?
001430' 003413                          BLE     ADPK1           ;IF WITHIN LIMIT THEN CONTINUE AT ADRMS2
001432' 004767  001210                  JSR     PC,ERCOM        ;GET ERROR PARAMETERS

001436' 012767  000031  176442          MOV     #31,ERRTYP      ;ERROR PARAMETERS LOADED BY ".ERCOM"
                                        ;NOISE EXCEEDED
                                        ;*****************************************************************
001444' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;;RMS NOISE ERROR-SEE NEXT TYPEOUT
                                        ;*****************************************************************
001452' 104403  000000' 002746'         MSGN$,BEGIN,MSG1        ;ASCII MESSAGE CALL WITH COMMON HEADER

                        ;CALCULATE A/D PEAK NOISE USING VERNIER DAC.

001460' 012700  000775          ADPK1:  MOV     #509.,R0        ;FOR EAK OF 2 1/2 SIGMA
001464' 016767  176574  176560          MOV     NCCH,TMP        ;GET CHAN. NUMBER.
001472' 000367  176554                  SWAB    TMP             ;PUT IN CORRECT CSR POSITION.
001476' 052767  000100  176546          BIS     #100,TMP        ;ADD INTR. ENABLE
001504' 005067  176532                  CLR     WHO             ;INDICATE PEAK NOISE TEST.
001510' 016777  176536  176514          MOV     TMP,@ADSR       ;CH.AND INTERRUPT ENABLE, AND CH
001516' 016701  176512                  MOV     DAC,R1          ;R1 = ADDRESS OF SAR DAC
001522' 004767  000370                  JSR     PC,SAR          ;GET DAC VALUE THAT GIVES .6% LOW COUNT
001526' 016705  176514                  MOV     FRED,R5         ;R5 = LEFT BOUNDARY
001532' 012700  000003                  MOV     #3,R0           ;CHANGE SPLIT FOR RIGHT BOUNDARY
001536' 004767  000354                  JSR     PC,SAR          ;GET DAC VALUE THAT GIVES .6% HIGH COUNT
001542' 166705  176500                  SUB     FRED,R5         ;R5 = A/D PEAK TO PEAK NOISE
001546' 010567  176506                  MOV     R5,APKX         ;SAVE FOR DAC NOISE CALCULATIONS
001552' 020567  176504                  CMP     R5,APKLIM       ;< OR = A/D PEAK NOISE LIMIT?
001556' 003413                          BLE     ENDP            ;BRANCH IF NO ERROR
001560' 004767  001062                  JSR     PC,ERCOM        ;GET ERROR PARAMETERS

001564' 012767  000031  176314          MOV     #31,ERRTYP      ;A/D NOISE LIMIT EXCEEDED
                                        ;*****************************************************************
001572' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;;PEAK NOISE ERROR-SEE NEXT TYPE OUT
                                        ;*****************************************************************
001600' 104403  000000' 002766'         MSGN$,BEGIN,MSG5        ;ASCII MESSAGE CALL WITH COMMON HEADER

001606'                         ENDP:
001606' 005267  176456                  INC     PASSCNT         ;UPDATE PASS COUNT.
001612' 032767  000004  176176          BIT     #BIT2,SR1       ;DOING MULTIPLE NOISE CHANNELS?
001620' 001012                          BNE     2$              ;YES - GET THE NEXT ONE
001622' 026727  176442  000013  1$:     CMP     PASSCNT,#13     ;DONE ENOUGH PASSES?
```



```
001630' 001417                          BEQ     3$              ;YES-DO ENDPASS.
001632' 032767  000001  176156          BIT     #BIT0,SR1       ;ARE WE CONNECTED TO THE KW11K OPTION?
001640' 001013                          BNE     3$              ;IF SO THE 1 MIN IS UP ON ONE PASS.

001642' 000167  177450                  JMP     ADRMS1          ;NO DO  AGAIN.

001646' 005267  176412          2$:     INC     NCCH            ;POINT TO NEXT CH.
001652' 026787  176406  176346          CMP     NCCH,NLSTCH     ;EXCEED LAST NOISE CH?
001660' 101760                          BLOS    1$              ;NO-TEST THIS CH.
001662' 005067  176376                  CLR     NCCH            ;YES - START AGAIN ON CH. 0.
001666' 000755                          BR      1$              ;GO TEST CH 0.

001670' 032767  000002  176120  3$:     BIT     #BIT1,SR1       ;ARE WE DOING VOLTAGE SAMPLING?
001676' 001505                          BEQ     EENDP           ;NO - END PASS!
001700' 005067  176362                  CLR     CCH             ;YES - START WITH CH 0.

001704' 026767  176356  176312  4$:     CMP     CCH,CLSTCH      ;DONE ALL CHANNELS?
001712' 003077                          BGT     EENDP           ;YES - REPORT END PASS.
001714' 005003                          CLR     R3              ;R3 WILL HOLD SAMPLE.
001716' 012700  000010                  MOV     #8.,R0          ;SET TO TAKE 8 SAMPLES
001722' 012777  001752' 176060          MOV     #6$,@VECTOR     ;SET VECTOR FOR INTERRUPT
001730' 016701  176332                  MOV     CCH,R1          ;GET CH. NUMBER
001734' 000301                          SWAB    R1              ;FIX RIGHT POSITION IN CSR.
001736' 052701  000101                  BIS     #101,R1         ;ADD INTR. ENABLE AND GO.
001742' 010177  176264          5$:     MOV     R1,@ADSR        ;START A/D.
001746' 104400  000000'                 EXITS,BEGIN             ;EXIT TO MONITOR. MODULE WAIT FOR INTERRUPT.
001752'                         6$:
                                        ;---------------------------------------------------------------
001752' 000004  000000' 001760'         PIRQ$,BEGIN,7$          ; QUEUE UP TO CONTINUE AT 7$ AND RTI
                                        ;---------------------------------------------------------------

001760' 067703  176250          7$:     ADD     @ADDR,R3        ;ADD THIS SAMPLE TO TOTAL
001764' 005300                          DEC     R0              ;ALL DONE ALL SAMPLES?
001766' 001365                          BNE     5$              ;NO - DO NEXT ONE.

001770' 006203                          ASR     R3              ;YES NOW WE
001772' 006203                          ASR     R3              ;MUST DIVIDE BY
001774' 006203                          ASR     R3              ;8 TO GET THE
001776' 005503                          ADC     R3              ;SAMPLE AVERAGE.
002000' 016700  176262                  MOV     CCH,R0          ;NOW GET CH. NUMBER
002004' 006300                          ASL     R0              ;FORM AN OFFSET
002006' 010301                          MOV     R3,R1           ;SAVE NEW SAMPLE VALUE.
002010' 166003  003262'                 SUB     RECSAM(0),R3    ;GET THE DIFFERENCE BETWEEN
                                                                ;AND THE NEW ONE WE JUST GOT
002014' 100001                          BPL     8$              ;IF POSITIVE WE'RE OK.
002016' 005403                          NEG     R3              ;OTHERWISE MAKE IT POSTIVE.
002020' 020367  176204          8$:     CMP     R3,OFFALL       ;IS IT WITHIN TOLERANCE?
002024' 003426                          BLE     10$             ;YES DO NEXT CH

                                        ;*****************************************************************
                                                                ;CONVERT CCH TO ASCII AND
002026' 104420  000000' 000266'         OTOA$,BEGIN,CCH,CHANN   ;STORE AT CHANN
002034' 003502'
                                        ;*****************************************************************
```

```
                                MOV     RECSAM(0),TEMP1 ;GET OLD VALUE
                                ;***************************************************************
                                                        ;CONVERT TEMP1 TO ASCII AND
002044' 104420  000000' 000272' OTOA$,BEGIN,TEMP1,OLDS  ;STORE AT OLDS
002052' 003462'
                                ;***************************************************************

002054' 010167  176212          MOV     R1,TEMP1        ;GET NEW VALUE
                                ;***************************************************************
                                                        ;CONVERT TEMP1 TO ASCII AND
002060' 104420  000000' 000272' OTOA$,BEGIN,TEMP1,NEWS  ;STORE AT NEWS
002066' 003472'
                                ;***************************************************************

002070' 010160  003262'         MOV     R1,RECSAM(R0)   ;PUT NEW INTO OLD.

002074' 104403  000000' 003024' MSGN$,BEGIN,MSG12        ;ASCII MESSAGE CALL WITH COMMON HEADER

002102' 005267  176160          INC     CCH             ;LOOK AT NEXT CHAN.
002106' 000167  177572          JMP     4$              ;GO TEST IT

002112'
002112' 104413  000000'         ENDIT$,BEGIN            ;SIGNAL END OF ITERATION.
                                                        ;MONITOR SHALL TEST END OF PASS

                                ;USING SUCCESSIVE APPROXIMATION AND VERNIER DAC DEFINED IN R1.

002116' 012702  000200          MOV     #200,R2         ;R2 = MSB OF DAC
002122' 005011                  CLR     (R1)            ;GET RID OF 'ONES'
002124' 005067  176116          CLR     FRED            ;START WITH ZERO DAC
002130' 060267  176112          ADD     R2,FRED         ;TRY THIS BIT
002134' 016711  176106          MOV     FRED,(R1)       ;LOAD DAC.
002140' 005003                  CLR     R3              ;INIT HIGH COUNT
002142' 012704  001000          MOV     #512.,R4        ;R4 = # OF SAMPLES IN A BURST
002146'

002146' 032767  000001  175642  BIT     #BIT0,SR1       ;IS CLOCK SELECTED?
002154' 001004                  BNE     1$              ;YES GOTO HANDLER.

002156' 052777  000001  176046  BIS     #BIT0,@ADSR     ;NO - SET GO BIT IN A/D.
002164' 000420                  BR      2$              ;GOTO 2$

002166' 004767  000364          JSR     PC,RANDV        ;GET A RANDOM NUMBER.
002172' 005077  176040          CLR     @ASR            ;MAKE SURE THE CLOCK'S CSR IS CLEAR.
002176' 052767  177770  000434  BIS     #177770,RNA     ;MAKE SURE OF HIGH NUMBER.
002204' 016777  000430  176026  MOV     RNA,@ABR        ;SET CLOCK PRESET REG.
002212' 052777  000040  176012  BIS     #BIT5,@ADSR     ;SET OVFRFLOW ENABLE.
002220' 012777  000011  176010  MOV     #11,@ASR        ;START CLOCK
002226'
002226' 104400          2$:     EXIT$                   ;RETURN TO MONITOR.
002230'                 WAINT:
                        ;-------------------------------------------------------------------
002230' 000004  000000' 002236' PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1S AND RTI
                        ;-------------------------------------------------------------------
```



```
002236' 027767  175772  176004 1$:     CMP     @ADBR,EDGE      ;> OR = EDGE?
002244' 103401                 BLO     2$              ;BRANCH IF <EDGE
002246' 005203                 INC     R3              ;COUNT IF > OR =EDGE
002250' 005304                 2$:     DEC     R4              ;COUNT THROUGH BURST
002252' 001335                 BNE     CONV            ;BRANCH IF NOT DONE
002254' 020300                 CMP     R3,R0           ;HIGH COUNT > % OF 512 CONVERSIONS?
002256' 003402                 BLE     3$              ;BRANCH TO LAVE BIT IN
002260' 160267  175762         SUB     R2,FRED         ;TAKE BIT OUT
002264' 006202                 3$:     ASR     R2              ;NEXT BIT
002266' 001320                 BNE     BIT             ;AND GO RESOLVE IT IF NOT DONE
002270' 005767  175752         TST     FRED            ;CHECK FOR ALL 'ZEROES'
002274' 001405                 BEQ     WRPERR          ;BRANCH IF INVALID RESULT
002276' 026727  175744  000377 CMP     FRED,#377       ;CHECK FOR ALL 'ONES'
002304' 001401                 BEQ     WRPERR          ;BRANCH IF INVALID RESULT
002306' 000207                 RTS     PC              ;RETURN TO ADRMS1 OR ADPK1.
                        ;VOLTAGE NEEDED TO PRODUCE # OF HIGH COUNTS SPECIFIED IS OUT OF THE
                        ;RANGE OF THE WRAPAROUND DAC.
                        ;CLEAR INTERRUPTS, PRINT MESSAGE AND DROP MODULE.

002310' 005077  175716 WRPERR: CLR     @ADSR           ;STOP A/D
002314' 004767  000326         JSR     PC,ERCOM        ;GET ERROR PARAMETERS
                                                        ;ERROR PARAMETERS LOADED BY "ERCOM"
002320' 012767  000031  175560 MOV     #31,ERRTYP      ;A/D NOISE LIMIT EXCEEDED
                                ;***************************************************************
002326' 104405  000000' 000000 HRDER$,BEGIN,NULL       ;;NOISE TEST WRAPAROUND ERROR
                                ;***************************************************************
002334' 005767  175702         TST     WHO             ;SEE WHICH TEST WE WERE DOING.
002340' 001004                 BNE     1$              ;WHO=1 THEN RMS NOISE TEST.
002342' 104403  000000' 003006' MSGN$,BEGIN,MSG11       ;ASCII MESSAGE CALL WITH COMMON HEADER
002350' 000403                 BR      2$              ;GO AHEAD.
002352'                 1$:
002352' 104403  000000' 003046' MSGN$,BEGIN,MSG13       ;ASCII MESSAGE CALL WITH COMMON HEADER
002360' 005267  175460 2$:     INC     HRDCNT          ;UPDATE THE ERROR COUNT.
002364' 005726                 TST     (6)+            ;RESTORE STACK FROM THE JSR PC THAT

002366' 000167  177214         JMP     ENDP            ;GOT US TO "SAR".
                                                        ;GET NEXT DIRECTIVE.

                        ;GET THE AVERAGE OF 512 SAMPLES AND THEIR COUNT SPREAD

002372'                 AVER:
002372' 005000                 CLR     R0              ;INIT R0 FOR RUNNING SUM
002374' 012704  001000         MOV     #512.,R4        ;BURST OF 512 CONVERSIONS
002400' 012701  100000         MOV     #BIT15,R1       ;INIT HIGH COUNT TO NEGATIVE #
002404' 012702  040000         MOV     #BIT14,R2       ;INIT LOW COUNT TO POSITIVE #
002410'                 ECON:

002410' 032767  000001  175400 BIT     #BIT0,SR1       ;IS CLOCK SELECTED?
002416' 001004                 BNE     1$              ;YES GOTO HANDLER.

002420' 052777  000001  175604 BIS     #BIT0,@ADSR     ;NO - SET GO BIT IN A/D.
002426' 000420                 BR      2$              ;GOTO 2$

002430' 004767  000122         1$:     JSR     PC,RANDV        ;GET A RANDOM NUMBER.
002434' 005077  175576         CLR     @ASR            ;MAKE SURE THE CLOCK'S CSR IS CLEAR.
002440' 052767  177770  000172 BIS     #177770,RNA     ;MAKE SURE OF HIGH NUMBER.
002446' 016777  000166  175564 MOV     RNA,@ABR        ;SET CLOCK PRESET REG.
```

```
002454' 052777  000040  175550          BIS     #BIT5,@ADSR     ;SET OVERFLOW ENABLE.
002462' 012777  000011  175546          MOV     #11,@ASR        ;START CLOCK
002470'                         2$:
002470' 104400  000000'         ERINT:  EXIT$,BEGIN                     ;EXIT TO MONITOR. MODULE WAIT FOR INTERRUPT.
002474'                         ;-------------------------------------------------------------------
002474' 000004  000000' 002502' PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
                                ;-------------------------------------------------------------------
002502' 067700  175526          1$:     ADD     @ADBR,R0        ;R0 = RUNNING SUM OF DIFFERENCES
002506' 160300                  SUB     R3,R0           ;OFFSET BY EXPECTED VALUE
002510' 027701  175520          CMP     @ADBR,R1        ;HIGHER THAN PREVIOUS HIGH?
002514' 003402                  BLE     2$              ;BRANCH IF NOT A NEW HIGH
002516' 017701  175512          MOV     @ADBR,R1        ;NEW HIGH RESULT
002522' 027702  175506          2$:     CMP     @ADBR,R2        ;LOWER THAN PREVIOUS LOW?
002526' 002002                  BGE     3$              ;BRANCH IF NO
002530' 017702  175500          MOV     @ADBR,R2        ;NEW LOW RESULT
002534' 005304                  3$:     DEC     R4              ;COUNT THROUGH BURST
002536' 001324                  BNE     ECON            ;DO ANOTHER SAMPLE IF NOT DONE
002540' 000300                  SWAB    R0              ;DIVIDE SUM OF DIFFERENCES BY 256
002542' 110000                  MOVB    R0,R0           ;SIGN EXTEND
002544' 006200                  ASR     R0              ;DIVIDE BY 2 MORE
002546' 005500                  ADC     R0              ;ROUND UP FOR > 1/2
002550' 060300                  ADD     R3,R0           ;OFFSET AVERAGE BY EXPTECTED RESULT
002552' 160201                  SUB     R2,R1           ;R1 = COUNT SPREAD
002554' 000207                  RTS     PC

                                ;GENERATE A RANDOM NUMBER

002556' 066767  000060  000054  RANDY:  ADD     RNB,RNA         ;THESE FOLLOW SEQUENCE OF
002564' 066767  000054  000046          ADD     RNC,RNA         ; INSTRUCTIONS GENERATE
002572' 005567  000042                  ADC     RNA             ; A RANDOM NUMBER
002576' 066767  000036  000036          ADD     RNA,RNB         ; TO BE USED
002604' 066767  000034  000030          ADD     RNC,RNB         ; TO BE LOADED
002612' 005567  000024                  ADC     RNB             ; INTO THE CLOCK'S PRESET
002616' 066767  000016  000020          ADD     RNA,RNC         ; BUFFER, IF SELECTED FOR TEST.
002624' 066767  000012  000012          ADD     RNB,RNC         ; ONLY RNA WILL BE USED.
002632' 005567  000006                  ADC     RNC
002636' 000207                  RTS     PC
002640' 063241                  RNA:    063241                  ;RETURN TO "CONV"
002642' 142315                  RNB:    142315                  ;RANDOM TO NUMBER A.
002644' 127623                  RNC:    127623                  ;RANDOM NUMBER B.
                                                                ;RANDOM NUMBER C.

                                ;CONVERT NOISE RESULT TO DECIMAL

002646' 016767  175360  175224  ERCOM:  MOV     ADSR,CSRA       ;LOAD HEADER FOR ERRORN CALL
002654' 017767  175352  175220          MOV     @ADSR,ACSR
002662' 016767  175140  175214          MOV     STAT,ASTAT
002670' 010567  175376                  MOV     R5,TEMP1        ;STACK BINARY NOISE VALUE
                                ;*************************************************************
                                                                ;CONVERT TEMP1 TO ASCII AND
002674' 104421  000000' 000272'         BTOD$,BEGIN,TEMP1,DECIM ;STORE AT DECIM
002702' 002740'
                                ;*************************************************************
002704' 116767  000032  000323          MOVB    DECIM+2,VALUE   ;MOVE CONVERTED VALUES TO ASCII BUFFER
002712' 116767  000025  000317          MOVB    DECIM+3,VALUE+2 ;FOR TYPEOUT OF ERROR
```



```
002720' 116767  000020  000312          MOVB    DECIM+4,VALUE+3 ; ON RETURN
                                ;*************************************************************
                                                                ;CONVERT NCCH TO ASCII AND
002726' 104420  000000' 000264'         OTOA$,BEGIN,NCCH,CHANN  ;STORE AT CHANN
002734' 003502'
                                ;*************************************************************
002736' 000207                  RTS     PC                      ; EXIT TO CALLER.
002740' 000003                  DECIM:  .BLKW   3

                                ;ASCII MESSAGES AND POINTERS

002746' 003072'                 MSG1:   P2                      ;ON CHAN
003750' 003506'                         CHANN+4                 ; (CHAN NUMBER 2 DIGITS)
002752' 003064'                         P1                      ; % A/D
002754' 003123'                         P4                      ; RMS
002756' 003137'                         P6                      ; NOISE =
002760' 003235'                         VALUE                   ; (CONVERTED BELOW) X.XX LSB (LIMIT =
002762' 003151'                         P7                      ; 0.50 LSB) "THIS VALUE WILL CHANGE IF OPERATOR CHANGES
002764' 177777                          -1                      ; MESSAGE TERMINATOR.

002766' 003072'                 MSG5:   P2                      ;ON CHAN
002770' 003506'                         CHANN+4                 ; (CHAN NUMBER 2 DIGITS)
002772' 003064'                         P1                      ; % A/D
002774' 003131'                         P5                      ; PEAK
002776' 003137'                         P6                      ; NOISE =
003000' 003235'                         VALUE                   ; (CONVERTED VALUE) X.XX LSB (LIMIT =
003002' 003163'                         P8                      ; 2.00 LSB) "THIS VALUE WILL CHANGE IF OPERATOR CHANGES
003004' 177777                          -1                      ; MESSAGE TERMINATOR

003006' 003064'                 MSG11:  P1                      ; % A/D
003010' 003131'                         P5                      ; PEAK
003012' 003213'                         P13                     ; WRAPAROUND
003014' 003227'                         P15                     ; ERROR
003016' 003072'                         P2                      ; ON CHAN
003020' 003506'                         CHANN+4                 ; (CHAN NUMBER 2 DIGITS)
003022' 177777                          -1                      ; MESSAGE TERMINATOR

003024' 003064'                 MSG12:  P1                      ; % A/D
003026' 003227'                         P15                     ; ERROR
003030' 003072'                         P2                      ; ON CHAN
003032' 003506'                         CHANN+4                 ; (CHAN NUMBER 2 DIGITS)
003034' 003104'                         P3                      ; OLD VALUE =
003036' 003464'                         OLDS+2                  ; "A/D READING 4 DIGITS
003040' 003175'                         P9                      ; NEW VALUE =
003042' 003474'                         NEWS+2                  ; "NEW A/D READING 4 DIGITS
003044' 177777                          -1                      ; MESSAGE TERMINATOR.

003046' 003064'                 MSG13:  P1                      ; % A/D
003050' 003123'                         P4                      ; RMS
003052' 003213'                         P13                     ; WRAPAROUND
003054' 003227'                         P15                     ; ERROR
003056' 003072'                         P2                      ; ON CHAN
003060' 003506'                         CHANN+4                 ; (CHAN NUMBER 2 DIGITS)
003062' 177777                          -1                      ; MESSAGE TERMINATOR.
```

```
945
946   003064' 040445  042057  000040  P1:     .ASCIZ  "%A/D "
947   003072' 047440  020116  044103  P2:     .ASCIZ  " ON CHAN "
948   003100' 047101  000040
949   003104' 020073  046117  020104  P3:     .ASCIZ  "; OLD VALUE = "
950   003112' 040526  052514  020105
951   003120' 020075     000
952   003123'    122  051515  000040  P4:     .ASCIZ  "RMS "
953   003130'    000                          .BYTE
954   003131'    120  040505  020113  P5:     .ASCIZ  "PEAK "
955   003136'    000
956   003137'    116  044517  042523  P6:     .ASCIZ  "NOISE = "
957   003144' 036440  000040
958   003150'    000                          .BYTE
959   003151'    060  032456  020060  P7:     .ASCIZ  "0.50 LSB)"
960   003156' 051514  024502     000
961   003163'    062  030056  020060  P8:     .ASCIZ  "2.00 LSB)"
962   003170' 051514  024502     000
963
964   003175'    040  042516  020127  P9:     .ASCIZ  " NEW VALUE = "
965   003202' 040526  052514  020105
966   003210' 020075     000
967   003213'    127  040522  040520  P13:    .ASCIZ  "WRAPAROUND "
968   003220' 047522  047125  020104
969   003226'    000
970   003227'    105  051122  051117  P15:    .ASCIZ"ERROR"
971   003234'    000
972
973   003235'    130  054056  020130  VALUE:  .ASCIZ  "X.XX LSB (LIMIT = "
974   003242' 051514  020102  046050
975   003250' 046511  052111  036440
976   003256' 000040
977   003260'    000                          .BYTE
978
979           003262'                         .EVEN
980   003262' 000100                  RECSAM: .BLKW   64.          ;USED TO STORE A/D RESULTS ON UP TO 64. CHANS.
981
982   003462' 000003                  OLDS:   .BLKW   3            ;USED FOR STORE OF ASCII OF OLD SAMPLE VALUE.
983   003470' 000000                          .WORD   0
984   003472' 000003                  NEWS:   .BLKW   3            ;USED FOR STORE OF ASCII NEW SAMPLE VALUE.
985   003500' 000000                          .WORD   0
986   003502' 000003                  CHANN:  .BLKW   3            ;USED FOR STORAGE OF ASCII OF CHAN. NUMBER.
987   003510' 000000                          .WORD   0
988           000001                          .END
```



```
ABR       000240R    348#   463*   768*   832*
ACSR      000102R    319#   439*   483*   513*   536*   878*
ADBR      000234R    343#   376*   449    504    508    533    556    569    696    777    841    843    845
                     846    848
ADDR      000006R    285#   373
ADDR22=   001000     332#
ADPK1     001460R    626    637#
ADRMS1    001316R    609#   669
ADSR      000232R    341#   374*   409    430*   438    439    440*   448*   464*   479    482    483    506*
                     512    513    531*   535    536    557*   566*   598*   617*   642*   689*   762*   769*
                     796*   826*   833*   877    878
APKLIM    000262R    362#   394    650
APKX      000260R    361#   649*
ARMLIM    000256R    360#   382    625
ARMX      000254R    359#   624*
ASB       000106R    323#
ASR       000236R    347#   422    465*   471    476    477*   766*   770*   830*   834*
ASTAT     000104R    321#   476*   879*
AVER      002372R    816#
AWAS      000110R    324#
BEGIN     000000R    282#   382    394    417    418    433    434    443    445    469    470    486    493
                     502    503    507    516    522    532    539    544    554    555    567    595    631
                     633    656    658    690    693    717    725    733    739    745    775    801    805
                     808    836    839    884    893
BIT       002130R    753#   786
BIT0   =  000001     332#   419    460    565    666    759    762    823    826
BIT1   =  000002     332#   677
BIT10  =  002000     332#
BIT11  =  004000     332#
BIT12  =  010000     332#
BIT13  =  020000     332#
BIT14  =  040000     332#   820
BIT15  =  100000     332#   819
BIT2   =  000004     332#   662
BIT3   =  000010     332#
BIT4   =  000020     332#
BIT5   =  000040     332#   464    769    833
BIT6   =  000100     332#   565
BIT7   =  000200     332#
BIT8   =  000400     332#
BIT9   =  001000     332#
BREAK$=   104407     332#   417    418    433    434    469    470    502    503    554    555
BR1       000012R    287#   605
BR2       000013R    288#
BTOD$  =  104421     332#   382    394    884
CCH       000266R    364#   558*   563    579    583*   584    586    679*   681    686    704    717    741*
CDATA$=   104412     332#
CHANN     003502R    717    893    903    912    925    931    943    986#
CLSTCH    000224R    334#   586    681
CONFIG    000056R    307#
CONV      002146R    757#   781
CSRA      000100R    317#   438*   482*   512*   535*   877*
DAC       000234R    342#   618    643
DATCK$=   104411     332#
DATER$=   104404     332#
DECTM     002740R    382    387    388    389    394    398    399    400    884    887    888    889    898#
```

```
DVID1   000014R   289#
ECON    002410R   821#   850
EDGE    000250R   357#   615*   777
EENDP   002112R   678    682    744#
ENDITS= 104413    332#   745
ENDP    001606R   651    660#   812
ENDS    = 104410  332#   445    493
ERCOM   002646R   627    652    797    877#
ERINT   002474R   837#
ERRTYP  000106R   322#   441*   484*   514*   537*   629*   654*   799*
EXITS   = 104400  332#   507    532    567    690    772    836
FRED    000246R   356#   620    623    645    648    752*   753*   754    784*   787    789
GETPAS= 104415    332#
GWBUFS= 104414    332#
HRDCNT  000044R   302#   809*
HRDERS= 104405    332#   443    486    516    539    631    656    801
HRDPAS  000050R   304#
ICONT   000036R   299#
ICOUNT  000040R   300#
IDNUM   000122R   329#
INIT    000030R   296#
INTFLG  000244R   355#   428*   436    450*
INTR    000120R   328#   370*
LOG1    000426R   409#
LOG2    000432R   416#
LOG3    000456R   420    428#
LOG4    000572R   437    460#
LOG5    000724R   461    480    501#
LOG6    001024R   511    530#
LOG7    001112R   534    553#
MAP22$= 104416    332#
MODNAM  000000R   283#
MODSP   000224R   297    330#
MSGN$   = 104403  332#   633    658    739    805    808
MSGS$   = 104402  332#
MSG$    = 104401  332#
MSG1    002746R   633    902#
MSG11   003006R   805    920#
MSG12   003024R   739    928#
MSG13   003046R   808    938#
MSG5    002766R   658    911#
NCCH    000264R   363#   377*   610    613    638    671*   672    674*   893
NEWS    003472R   733    935    984#
NLSTCH  000226R   335#   584    672
NULL    = 000000  332#   443    486    516    539    631    656    801
OFFALL  000230R   336#   711
OLDS    003462R   725    933    982#
OPEN    = 000000  264    290    291    292    293    310    311    312    313    314    315    316    317
                  319    321    323    324    326    327    328    332#
OTOA$   = 104420  332#   717    725    733    893
PASCNT  000034R   298#
PASSCN  000270R   365#   599*   661*   664
PIRQ$   = 000004  332#   522    544    595    693    775    839
POPSP   = 005726  332#
POPSP2= 022626    332#
PRTY    = 000000  332#
```



```
PRTY0   = 000000  332#
PRTY1   = 000040  332#
PRTY2   = 000100  332#
PRTY3   = 000140  332#
PRTY4   = 000200  288    332#
PRTY5   = 000240  332#
PRTY6   = 000300  287    332#
PRTY7   = 000340  332#
PS      = 177776  332#
PSW     = 177776  332#
PUSH    = 005746  332#
PUSH2   = 024646  332#
P1      003064R   904    913    920    928    938    946#
P13     003213R   922    940    967#
P15     003227R   923    929    941    970#
P2      003072R   902    911    924    930    942    947#
P3      003104R   932    949#
P4      003123R   905    939    952#
P5      003131R   914    921    954#
P6      003137R   906    915    956#
P7      003151R   387*   388*   389*   908    959#
P8      003163R   398*   399*   400*   917    961#
P9      003175R   934    964#
RANDY   002556R   765    829    861#
RAND$   = 104417  332#
RANNUM  000054R   306#
RECSAM  003267R   581*   615    707    721    737*   980#
RESTRT  001272R   325    588    598#
RES1    000056R   308#
RES2    000060R   309#
RNA     002640R   767*   768    831*   832    861*   862*   863*   864    867    871#
RNB     002642R   861    864*   865*   866*   868    872#
RNC     002644R   862    865    867*   868*   869*   873#
RSTRT   000112R   325#
SAR     002116R   619    622    644    647    750#
SBADR   000102R   318#
SOFCNT  000042R   301#
SOFEN$= 104406    332#
SOFPAS  000046R   303#
SPOINT  000032R   297#
SPSIZ   = 000040  1#     330
SR1     000016R   290#   419    460    662    666    677    759    823
SR2     000020R   291#
SR3     000022R   292#
SR4     000024R   293#
START   000274R   296    370#
STAT    000026R   295#   879
SVR0    000062R   310#
SVR1    000064R   311#
SVR2    000066R   312#
SVR3    000070R   313#
SVR4    000072R   314#
SVR5    000074R   315#
SVR6    000076R   316#
SYSCNT  000052R   305#
TEMP1   000272R   366#   721*   725    729*   733    880*   884
```

```
TMP     000252R          358#    563*    564*    565*    566     610*    611*    612*    617     638*    639*    640*    642
TRPDFD= 000022           332#
VALUE   003235R          887*    888*    889*    907     916     973#
VECTOR  000010R          286#    429*    505*    530*    559*    603     685*
WAINT   002230R          604     773#
WASADR  000104R          320#
WDFR    000116R          327#    372*
WDTO    000114R          326#    371*
WHO     000242R          354#    616*    641*    803
WRPERR  002310R          788     790     796#
XPLAG   000005R          284#
.       = 003512R        898#    979#    980#    982#    984#    986#

. ABS.  000000      000
        003512      001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 XADCB0,XADCB0/SOL/CRF:SYM=DDXCOM,XADCB0
 RUN-TIME:  2  2  .3 SECONDS
 RUN-TIME RATIO: 2975=5.5
 CORE USED:  7K  (13 PAGES)
```