```
.REM    ]
```

IDENTIFICATION

PRODUCT CODE: AC-E875B-MC

PRODUCT NAME: CXARAB0 AR-11 MODULE

PRODUCT DATE: SEPTEMBER 1978

MAINTAINER: DEC/X11 SUPPORT GROUP

1. ABSTRACT

ARA IS AN IOMOD THAT EXERCISES THE AR11. IT WILL RUN IN ONE OF TWO COMPLETELY INDEPENDENT MODES.

A.) WITHOUT A G5036 WRAPAROUND TEST MODULE CONNECTED TO THE AR11.

B.) WITH A G5036 WRAPAROUND TEST MODULE CONNECTED TO THE AR11.

=NOTE=
IF A BC08R CABLE IS USED THE BERG AT THE AR11 END SHOULD BE CONNECTED A TO A; AND THE BERG AT THE G5036 END SHOULD BE CONNECTED A TO VV.

THE PROGRAM AUTOMATICALLY ENTERS THE APPROPRIATE MODE BY TESTING FOR THE PRESENCE OF A WRAPAROUND MODULE AT RUN TIME.

2. REQUIREMENTS

HARDWARE: AR11

STORAGE:: ARA REQUIRES:
1. DECIMAL WORDS: 1125
2. OCTAL WORDS: 02145
3. OCTAL BYTES: 4312

3. PASS DEFINITION

IN WRAPAROUND MODE, ONE PASS OF THE ARA MODULE CONSISTS OF SEQUENCING THROUGH 8 COMBINATIONS OF RMS AND PEAK NOISE CALCULATIONS ONE TIME FOR A TOTAL OF 81,920. CONVERSIONS.

IN NON-WRAPAROUND MODE, ONE PASS CONSISTS OF DISPLAYING EACH OF THE 16 A/D CHANNELS SEQUENTIALLY FOR A TOTAL OF 102,400 CONVERSIONS.

4. EXECUTION TIME

ONE PASS OF ARA RUNNING ALONE TAKES APPROXIMATELY 1 MINUTE.

5. CONFIGURATION REQUIREMENTS

DEFAULT PARAMETERS:

DVA: 1, VCT: 1, BR1: 6, BR2: 4

REQUIRED PARAMETERS:

ADDRESS AND VECTOR MUST BE SPECIFIED AT CONFIGURATION OR RUN TIME.

6. DEVICE/OPTION SETUP

TO RUN IN NON-WRAPAROUND MODE, DO NOT CONNECT THE G5036 WRAPAROUND MODULE SO THAT THE EXERCISER WILL NOT SENSE ITS PRESENCE AND WILL ENTER THE NON-WRAPAROUND MODE CORRECTLY.

TO RUN IN THE WRAPAROUND MODE, INSURE THAT THE G5036 WRAPAROUND MODULE IS CONNECTED PROPERLY TO THE AR11 AND IS OPERATING CORRECTLY BY RUNNING THE STAND ALONE WRAPAROUND DIAGNOSTIC. THIS WILL INSURE THAT THE EXERCISER WILL BE ABLE TO SENSE THE PRESENCE OF THE G5036 AND AUTOMATICALLY ENTER THE WRAPAROUND MODE.

7. MODULE OPERATION

AT THE BEGINNING OF THE DEC-X RUN, ARA WILL PRINTOUT ONE DOUBLE LINE MESSAGE TO INDICATE WHETHER IT IS RUNNING IN WRAPAROUND MODE OR NON-WRAPAROUND MODE. THIS TYPEOUT IS NOT AN ERROR.

IN NON-WRAPAROUND MODE, ARA DOES NOT MAKE ANY ERROR CHECKS ITSELF. PROPER OPERATION IS ASCERTAINED BY VISUAL EXAMINTAION OF THE DISPLAY AND THROUGH THE MONITOR'S ERROR REPORTING CAPABILITY. DATA IS ACQUIRED AND DISPLAYED FROM EACH OF THE 16 A/D CHANNELS SEQUENTIALLY WITH THE CHANNEL NUMBER DISPLAYED IN THE CENTER BOTTOM.

IN WRAPAROUND MODE, NOISE CALCULATIONS ARE MADE FROM STATISTICAL ANALYSIS OF LARGE NUMBERS OF SAMPLES AND COMPARED TO LIMITS IN THE FOLLOWING SEQUENCE:

A. A/D RMS NOISE USING FINE X WRAPAROUND DAC.
B. A/D RMS NOISE USING FINE Y WRAPAROUND DAC.
C. Y DAC RMS NOISE.
D. X DAC RMS NOISE.
E. A/D PEAK NOISE USING FINE X WRAPAROUND DAC.
F. A/D PEAK NOISE USING FINE Y WRAPAROUND DAC.
G. Y DAC PEAK NOISE.
H. X DAC PEAK NOISE.
I. END PASS.

8. OPERATION OPTIONS

LOCATIONS ARMLIM, DRMLIM, APKLIM AND DPKLIM CAN BE MODIFIED TO CHANGE THE MAXIMUM ALLOWABLE NOISE LIMITS. IF THE ACTUAL RMS AND PEAK NOISE FIGURES FOR THE AR11 RUNNING IN A SYSTEM ENVIRONMENT ARE DESIRED, THESE 4 LOCATIONS MAY BE CHANGED TO 000000 IN WHICH CASE ALL NOISE CALCULATIONS WILL BE TYPED OUT.

9. NON-STANDARD PRINTOUTS

A. A NON-ERROR MESSAGE AT THE BEGINNING OF THE DEC-X RUN WILL BE PRINTED:

AR11 RUNNING IN WRAPAROUND/NON-WRAPAROUND MODE

B. IF ARA FINDS EXCESSIVE ANALOG NOISE, IT REPORTS IT IN A MSGN CALL:

(EXAMPLE)
A/D PEAK NOISE = 2.57 LSB (LIMIT = 2.00 LSB)

FOLLOWED BY AN ERRORN CALL WHICH PRINTS OUT 6 LOCATIONS CONTAINING POWER SUPPLY VOLTAGE INFORMATION:

AVP14V SPP14V AVN14V SPN14V AVP5HQ SPP5HQ

DEFINED AS FOLLOWS:

AVP14V (AVERAGE VOLTAGE LEVEL ON +14V H.Q. SUPPLY)

THE AVERAGE OF 512 CONVERSIONS TAKEN ON CH.#3 (+2.5V) SCALED DOWN FROM THE +14 VOLT SUPPLY.

IF THE OCTAL NUMBER IN THIS COLUMN IS LESS THAN 1704, THEN THE +14 VOLT SUPPLY IS TOO LOW FOR NORMAL OPERATION.

VOLTAGE	OCTAL PRINTOUT	
>+13.9	001777	
13.75	001772	
13.5	001761	
13.25	001747	
13.0	001737	
12.75	001725	
12.5	001714	
12.25	001704	(LOWER LIMIT FOR NORMAL OPERATION)

SPP14V (COUNT SPREAD ON +14 VOLT H.Q. SUPPLY)

THE DIFFERENCE BETWEEN THE HIGHEST AND LOWEST CONVERSION OF 512 TAKEN ON CH.#3 (+2.5V) SCALED DOWN +14V.

IF THE OCTAL NUMBER PRINTED IN THIS COLUMN IS GREATER THAN 000010, THEN THE +14 VOLT SUPPLY IS CONSIDERED TOO NOISY FOR ACCURATE ANALOG OPERATION.

OCTAL PRINTOUT	PEAK TO PEAK NOISE	
000000-000004	<100 MILLIVOLTS NOISE	(TYPICAL LEVEL)
>000010	>200 MILLIVOLTS NOISE	(EXCESSIVELY NOISY)

AVN14V (AVERAGE VOLTAGE ON THE -14 VOLT H.Q. SUPPLY)

THE AVERAGE OF 512 CONVERSIONS TAKEN ON CH.#4 (-2.5V) SCALED DOWN FROM THE -14V. SUPPLY

AN OCTAL PRINTOUT IN THIS COLUMN OF GREATER THAN 000053 INDICATES THE -14 VOLT SUPPLY IS LOW.

VOLTAGE	OCTAL PRINTOUT	
<-13.9	000000	
13.75	000006	
13.5	000017	
13.25	000031	
13.0	000041	
12.75	000053	(LOWER LIMIT)
12.5	000064	

SPN14V (COUNT SPREAD ON -14 VOLT SUPPLY)

THE DIFFERENCE BETWEEN THE HIGHEST AND LOWEST CONVERSION OF 512 TAKEN ON CH.#4 (-2.5V) SCALED DOWN -14V.

IF THE OCTAL PRINTOUT IN THIS COLUMN IS GREATER THAN 000010, THEN THE -14 VOLTS IS TOO NOISY FOR ACCURATE ANALOG OPERATION.

OCTAL PRINTOUT	PEAK TO PEAK NOISE	
000000-000004	<100 MILLIVOLTS NOISE	(TYPICAL)
>000010	>200 MILLIVOLTS NOISE	(EXCESSIVE NOISE)

AVP5HQ (AVERAGE VOLTAGE LEVEL ON +5V H.Q.)

THE AVERAGE OF 512 CONVERSIONS TAKEN ON CH.#17 (+4V) SCALED DOWN FROM THE +5 H.Q. VOLTAGE.

AN OCTAL PRINTOUT OF LESS THAN 001321 OR GREATER THAN 001524 IN THIS COLUMN INDICATES THE +5HQ IS OUTSIDE OF ITS NORMAL OPERATING RANGE.

VOLTAGE	OCTAL PRINTOUT	
5.2	001524	(UPPER LIMIT)
5.1	001504	
5.0	001464	
4.9	001443	
4.8	001423	
4.7	001402	
4.6	001362	
4.5	001341	
4.4	001321	(LOWER LIMIT)

SPP5HQ (COUNT SPREAD ON +5V H.Q.)

THE DIFFERENCE BETWEEN THE HIGHEST AND LOWST CONVERSION OF 512 TAKEN ON CH.#17 (+4V) SCALED DOWN +5HQ.

AN OCTAL PRINTOUT IN THIS COLUMN OF GREATER THAN 000030 IS REGARDED AS NOISY, AND 000040 (200 MILLIVOLTS NOISE) IS HIGHLY QUESTIONABLE.

C. A MSG STATING "WRAPAROUND ERROR" FOLLOWED BY A DROPPED MESSAGE INDICATES THAT ARA DID SENSE A WRAPAROUND MODULE BUT THAT THROUGH A MALFUNCTION OR MALADJUSTMENT, ALL "ZEROES" OR ALL "ONES" WAS RETURNED BY THE SUCCESSIVE APPROXIMATION ROUTINE. RUN THE STAND ALONE WRAPAROUND DIAGNOSTIC TO DETERMINE WHY THE A/D VALUE WAS OUT OF THE RANGE OF THE DAC.

```
                                ]
                                        IONOD   <ARAB >,1,1,6,4,,1,133
                                        MODULE  140000,ARAB ,1,6,4,,1,133
                                        .TITLE  ARAB DEC/X11 SYSTEM EXERCISER MODULE
                                ;       DDXCOM  VERSION  6       23-MAY-78
                                        .LIST   BIN
                                ;*****************************************************************************
000000'                         BEGIN:
000000' 051101  041101      040 MODNAM: .ASCII  /ARAB / ;MODULE NAME.
000005'     000                 XFLAG:  .BYTE   OPEN        ;USED TO KEEP TRACK OF WBUFF USAGE
000006' 000001                  ADDR:   1+0                 ;1ST DEVICE ADDR.
000010' 000001                  VECTOR: 1+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' 000224'                 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' 000224'                 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' 000133                  IDNUM:  133                 ;MODULE IDENTIFICATION NUMBER=133
        000040                          .REPT   SPSIZ       ;MODULE STACK STARTS HERE.
```



```
                                        .NLIST
                                        .WORD   0
                                        .LIST
                                        .ENDR
000224'                         MODSP:
                                ;*****************************************************************************
```

```
                        START:
000224'                 RESTRT: MOV     ADDR,R0         ;SETUP REGISTER ADDRESSES
000224' 016700  177556          MOV     R0,ADSR
000230' 010067  003236          TST     (R0)+
000234' 005720                  MOV     R0,ADBR
000236' 010067  003232          TST     (R0)+
000242' 005720                  MOV     R0,CLKSR
000244' 010067  003226          TST     (R0)+
000250' 005720                  MOV     R0,CLKBR
000252' 010067  003222          TST     (R0)+
000256' 005720                  MOV     R0,DPSR
000260' 010067  003216          TST     (R0)+
000264' 005720                  MOV     R0,XDAC
000266' 010067  003212          TST     (R0)+
000272' 005720                  MOV     R0,YDAC
000274' 010067  003206          TST     (R0)+
000300' 005720                  MOV     R0,CLKCNT
000302' 010067  003202
                        LIMIN:
000306'                         ;*****************************************************************
                                ;CONVERT ARMLIM TO ASCII AND
                                ;STORE AT DECIM
000306' 104421  000000' 004302' BTOD$,BEGIN,ARMLIM,DECIM
000314' 003514'                 ;*****************************************************************
000316' 116767  003174  002550  MOVB    DECIM+2,P7
000324' 116767  003167  002544  MOVB    DECIM+3,P7+2
000332' 116767  003162  002537  MOVB    DECIM+4,P7+3
                                ;*****************************************************************
                                ;CONVERT APKLIM TO ASCII AND
                                ;STORE AT DECIM
000340' 104421  000000' 004306' BTOD$,BEGIN,APKLIM,DECIM
000346' 003514'                 ;*****************************************************************
000350' 116767  003142  002530  MOVB    DECIM+2,P8
000356' 116767  003135  002524  MOVB    DECIM+3,P8+2
000364' 116767  003130  002517  MOVB    DECIM+4,P8+3
                                ;*****************************************************************
                                ;CONVERT DRMLIM TO ASCII AND
                                ;STORE AT DECIM
000372' 104421  000000' 004304' BTOD$,BEGIN,DRMLIM,DECIM
000400' 003514'                 ;*****************************************************************
000402' 116767  003110  002510  MOVB    DECIM+2,P9
000410' 116767  003103  002504  MOVB    DECIM+3,P9+2
000416' 116767  003076  002477  MOVB    DECIM+4,P9+3
                                ;*****************************************************************
                                ;CONVERT DPKLIM TO ASCII AND
                                ;STORE AT DECIM
000424' 104421  000000' 004310' BTOD$,BEGIN,DPKLIM,DECIM
000432' 003514'                 ;*****************************************************************
000434' 116767  003056  002470  MOVB    DECIM+2,P10
000442' 116767  003051  002464  MOVB    DECIM+3,P10+2
000450' 116767  003044  002457  MOVB    DECIM+4,P10+3
```



```
000456' 012767  177326          SENSE:  MOV     VECTOR,R0       ;SET VECTOR AND PSW FOR DSP. INTERRUPT
000462' 012720  000550'                 MOV     #SAINT,(R0)+
000466' 116710  177320                  MOVB    BR1,@R0
000472' 005067  003014                  CLR     WRPFLG          ;CLEAR WRAPAROUND INDICATOR
000476' 012777  000120  002766          MOV     #120,@ADSR      ;EXTERNAL START AND INTERRUPT ENABLE
000504' 012777  000001  002770          MOV     #1,@DPSR        ;INTENSIFY
000512' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
000516' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000522' 005767  002764                  TST     WRPFLG          ;SEE IF A/D INTERUPTED
000526' 001413                          BEQ     NOWRAP
000530' 012767  000020  177356          MOV     #16.,WDTO       ;16. WORDS TO MEM/ITERATION
000536' 012767  000020  177352          MOV     #16.,WDFR       ;16. WORDS FROM MEM/ITERATION
000544' 000167  000416                  JMP     WRAP
000550' 005267  002736          SAINT:  INC     WRPFLG          ;FLAG THAT INTERRUPT HAS OCCURRED
000554' 000002                          RTI
000556' 012767  000200  177334  NOWRAP: MOV     #128.,INTR      ;128. INTERRUPTS/ITERATION
000564' 012767  003340  177324          MOV     #1760,WDFR      ;1760  WORDS FROM MEM
000572' 104403  000000' 003410'         MSG$S,BEGIN,MSG9        ;ASCII MESSAGE CALL WITH COMMON HEADER
000600' 012777  000100  002664          MOV     #100,@ADSR      ;SET INTERRUPT ENABLE
000606' 005077  002672                  CLR     @XDAC           ;INIT TO LEFT SIDE
000612' 012703  003523'                 MOV     #CARGEN,R3      ;INIT POINTER TO CHARACTER GENERATOR
000616' 012767  000676' 177266          MOV     #SWEEP,RSTRT    ;MOVE RESTART FOR ENDPASS
000624' 012777  000110  002650          MOV     #110,@DPSR      ;INTENSIFY ON LOAD Y AND INTERRUPT ENABLE
000632' 012777  000102  002636          MOV     #102,@CLKSR     ;INTERRUPT ENABLE AND 10USEC. RATE
000640' 016700  177144                  MOV     VECTOR,R0       ;USE R0 TO SET UP VECTORS
000644' 012720  000726'                 MOV     #NAINT,(R0)+    ;A/D VECTOR
000650' 005720                          TST     (R0)+
000652' 012720  000720'                 MOV     #NCINT,(R0)+    ;CLOCK VECTOR
000656' 116720  177130                  MOVB    BR1,(R0)+       ;STATUS
000662' 105720                          TSTB    (R0)+
000664' 012720  000736'                 MOV     #NDINT,(R0)+    ;DISPLAY VECTOR
000670' 116710  177117                  MOVB    BR2,@R0         ;STATUS
000674' 005740                          TST     -(R0)           ;BACK TO DISPLAY VECTOR
000676' 016701  000262          SWEEP:  MOV     FRMCNT,R1       ;NUMBER OF FRAMES PER CHANNEL
000702' 016777  000254  002570  PRIME:  MOV     CLCNT,@CLKBR    ;SET INTERVAL
000710' 005277  002562                  INC     @CLKSR          ;START CLOCK RUNNING
000714' 104400  000000'                 EXIT$,BEGIN             ;EXIT TO MONITOR. MODULE WAIT FOR INTERRUPT.
000720' 005277  002546          NCINT:  INC     @ADSR           ;START AN A/D CONVERSION
000724' 000002                          RTI
000726' 017777  002542  002552  NAINT:  MOV     @ADBR,@YDAC     ;LOAD Y
000734' 000002                          RTI
000736'                         NDINT:
                                ;-----------------------------------------------------------------
000736' 000004  000000' 000744' PIRQ$,BFGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
                                ;-----------------------------------------------------------------
000744' 062777  000020  002532  1$:     ADD     #16.,@XDAC      ;MOVE RIGHT 16
000752' 005777  002526                  TST     @XDAC           ;TEST FOR END OF SCREEN
000756' 001351                          BNE     PRIME           ;DO NEXT POINT IF NOT AT END OF SCREEN
000760' 012710  001006'                 MOV     #NDINT2,@R0     ;CHANGE VECTOR FOR DISPLAYING CH.#
000764' 012705  000012                  MOV     #10.,R5         ;LOOP COUNTER
000770' 010304                          MOV     R3,R4           ;WORKING POINTER
```

```
441  000772' 012477  002506          CHNUM:  MOV     (R4)+,@XDAC     ;LOAD X
442  000776' 012477  002504                  MOV     (R4)+,@YDAC     ;LOAD Y AND INTENSIFY
443  001002' 104400  000000'                  EXITS,BEGIN             ;EXIT TO MONITOR. MODULE WAIT FOR INTERRUPT.
444  001006'                          NDINT2:
445                                           ;-------------------------------------------------------------------
446  001006' 000004  000000' 001014'         PIRQS,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
447                                           ;-------------------------------------------------------------------
448  001014' 005305                   1$:     DEC     R5              ;COUNT THROUGH LOOP
449  001016' 001365                           BNE     CHNUM           ;BRANCH IF NOT DONE
450  001020' 032777  004000  002444           BIT     #BIT11,@ADSR    ;CH.# 10 OR GREATER?
451  001026' 001421                           BEQ     CHEXIT          ;BRANCH IF NOT
452  001030' 012704  004222'                  MOV     #D10,R4         ;SET UP TO DISPLAY TEN
453  001034' 012705  000012                   MOV     #10.,R5         ;LOOP COUNTER
454  001040' 012710  001060'                  MOV     #NDINT3,@R0     ;CHANGE VECTOR AGAIN
455  001044' 012477  002434           CHTEN:  MOV     (R4)+,@XDAC     ;LOAD X
456  001050' 012477  002432                   MOV     (R4)+,@YDAC     ;LOAD Y AND INTENSIFY
457  001054' 104400  000000'                  EXITS,BEGIN             ;EXIT TO MONITOR. MODULE WAIT FOR INTERRUPT.
458  001060'                          NDINT3:
459                                           ;-------------------------------------------------------------------
460  001060' 000004  000000' 001066'         PIRQS,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
461                                           ;-------------------------------------------------------------------
462  001066' 005305                   1$:     DEC     R5              ;COUNT THROUGH LOOP
463  001070' 001365                           BNE     CHTEN           ;BRANCH IF NOT DONE
464  001072' 012710  000736'          CHEXIT: MOV     #NDINT,@R0      ;RESTORE DISPLAY VECTOR
465  001076' 005077  002402                   CLR     @XDAC           ;BACK TO LEFT SIDE OF SCREEN
466  001102' 005301                           DEC     R1              ;COUNT FRAMES
467  001104' 001276                           BNE     PRIME           ;DO ANOTHER FRAME IF NOT DONE WITH CHANNEL
468  001106' 062777  000400  002356           ADD     #400,@ADSR      ;SELECT NEXT CHANNEL
469  001114' 017702  002352                   MOV     @ADSR,R2        ;FETCH STATUS
470  001120' 042702  174377                   BIC     #174377,R2      ;ISOLATE LOWER 3 BITS OF CH.#
471  001124' 006202                           ASR     R2
472  001126' 006202                           ASR     R2
473  001130' 006202                           ASR     R2              ;R2 = CH.# X 32
474  001132' 010203                           MOV     R2,R3           ;R3 = CH.# X 32
475  001134' 006202                           ASR     R2
476  001136' 006202                           ASR     R2              ;R2 = CH.# X 8
477  001140' 060203                           ADD     R2,R3           ;R3 = CH.# X 40
478  001142' 062703  003522'                  ADD     #CARGEN,R3      ;R3 = POINTER TO CORRESPONDING CHARACTER
479  001146' 027727  002320  000100           CMP     @ADSR,#100      ;DONE ALL 16 CHANNELS?
480  001154' 001250                           BNE     SWEEP           ;DO NEXT CHANNEL IF NOT DONE
481  001156' 104413  000000'                  ENDITS,BEGIN            ;SIGNAL END OF ITERATION.
482                                                                   ;MONITOR SHALL TEST END OF PASS
483  001162' 000200                   CLCNT:  200                     ;CLOCK INTERVAL
484  001164' 000156                   FRMCNT: 110.                    ;FRAMES PER CHANNEL
```



```
485  001166'                          WRAP:
486  001166' 104403  000000' 003422'          MSGNS,BEGIN,MSG10       ;ASCII MESSAGE CALL WITH COMMON HEADER
487  001174' 012777  000002  002274           MOV     #2,@CLKSR       ;SET CLOCK FOR 10S. RATE, SINGLE INTERVAL
488  001202' 016700  176602                   MOV     VECTOR,R0       ;SET VECTOR AND PSW
489  001206' 012720  002300'                  MOV     #WAINT,(R0)+
490  001212' 116710  176574                   MOVB    BR1,@R0
491  001216' 012767  001224' 176666           MOV     #ADRMS1,RSTRT   ;SET RESTART ADDRESS FOR WRAPAROUND MODE
492
493                                   ;CALCULATE A/D RMS NOISE USING FINE X WRAPAROUND DAC.
494
495  001224' 012700  000657           ADRMS1: MOV     #431.,R0        ;R0 = 84.13% OF 512 CONVERSIONS
496  001230' 012777  004140  002234           MOV     #4140,@ADSR     ;CH.#10, CLK. ST. AND INTERRUPT ENABLE
497  001236' 016701  002242                   MOV     XDAC,R1         ;R1 = ADDRESS OF SAR DAC
498  001242' 004767  000772                   JSR     PC,SAR          ;GET DAC VALUE THAT PRODUCES 16/84 SPLIT
499  001246' 011105                           MOV     @R1,R5          ;SAVE LEFT BOUNDARY
500  001250' 012700  000121                   MOV     #81.,R0         ;R0 = 15.87% OF 512 CONVERSIONS
501  001254' 004767  000760                   JSR     PC,SAR          ;GET DAC VALUE THAT PRODUCES 84/16 SPLIT
502  001260' 161105                           SUB     @R1,R5          ;R5 = BREADTH OF NOISE @ 68% AREA
503  001262' 010567  003004                   MOV     R5,ARMX         ;SAVE FOR DAC NOISE CALCULATIONS
504  001266' 020567  003010                   CMP     R5,ARMLIM       ;< OR = RMS LIMIT?
505  001272' 003413                           BLE     ADRMS2          ;IF WITHIN LIMIT THEN CONTINUE AT ADRMS2
506  001274' 004767  001100                   JSR     PC,ERCOM        ;GET ERROR PARAMETERS
507
508  001300' 104403  000000' 003250'          MSGNS,BEGIN,MSG1        ;ASCII MESSAGE CALL WITH COMMON HEADER
509
510  001306' 012767  000031  176572           MOV     #31,ERRTYP      ;A/D NOISE ERROR
511                                           ;*******************************************************************
512  001314' 104405  000000' 003440'          ERDERS,BEGIN,VOLTS      ;A/D RMS NOISE EXCEEDED LIMIT
513                                           ;*******************************************************************
514
515
516                                   ;CALCULATE A/D RMS NOISE USING FINE Y WRAPAROUND DAC.
517
518  001322' 012700  000657           ADRMS2: MOV     #431.,R0        ;R0 = 84.13% OF 512 CONVERSIONS
519  001326' 012777  003540  002136           MOV     #3540,@ADSR     ;CH.#7, CLK. ST. AND INTERRUPT ENABLE
520  001334' 016701  002146                   MOV     YDAC,R1         ;R1 = SAR DAC
521  001340' 004767  000674                   JSR     PC,SAR          ;GET DAC VALUE THAT PRODUCES 16/84 SPLIT
522  001344' 011105                           MOV     @R1,R5          ;SAVE DAC VALUE
523  001346' 012700  000121                   MOV     #81.,R0         ;CHANGE TO 15.87% OF 512 CONVERSIONS
524  001352' 004767  000662                   JSR     PC,SAR          ;GET DAC VALUE THAT PRODUCES 84/16 SPLIT
525  001356' 161105                           SUB     @R1,R5          ;R5 = RMS NOISE X 2
526  001360' 010567  002710                   MOV     R5,ARMY         ;SAVE FOR DAC NOISE CALCULATIONS
527  001364' 020567  002712                   CMP     R5,ARMLIM       ;< OR = RMS LIMIT?
528  001370' 003413                           BLE     YRMS            ;BRANCH IF NO ERROR
529  001372' 004767  001002                   JSR     PC,ERCOM        ;GET ERROR PARAMETERS
530
531  001376' 104403  000000' 003264'          MSGNS,BEGIN,MSG2        ;ASCII MESSAGE CALL WITH COMMON HEADER
532
533  001404' 012767  000031  176474           MOV     #31,ERRTYP      ;A/D NOISE ERROR
534                                           ;*******************************************************************
535  001412' 104405  000000' 003440'          ERDERS,BEGIN,VOLTS      ;A/D RMS NOISE EXCEEDED LIMIT
536                                           ;*******************************************************************
537
```

```
                                ;CALCULATE Y DAC RMS NOISE USING COARSE Y, FINE X WRAPAROUND DACS.
001420' 012700  000657          YRMS:   MOV     #431.,R0        ;R0 = 84.13% OF 512 CONVERSIONS
001424' 012777  003140  002040          MOV     #3140,@ADSR     ;CH.#6, CLK. ST. AND INTERRUPT ENABLE
001432' 012777  001000  002046          MOV     #1000,@YDAC     ;PUT YDAC AT MID-RANGE
001440' 016701  002040                  MOV     XDAC,R1         ;R1 = SAR DAC
001444' 004767  000570                  JSR     PC,SAR          ;GET DAC VALUE THAT PRODUCES 16/84 SPLIT
001450' 011105                          MOV     @R1,R5          ;SAVE LEFT BOUNDARY
001452' 012700  000121                  MOV     #81.,R0         ;R0 = 15.87% OF 512 CONVERSIONS
001456' 004767  000556                  JSR     PC,SAR          ;GET DAC VALUE THAT PRODUCES 84/16 SPLIT
001462' 161105                          SUB     @R1,R5          ;R5 = COMBINED A/D AND Y DAC NOISE X 2
001464' 166705  002602                  SUB     ARMX,R5         ;TAKE AWAY A/D NOISE
001470' 020567  002610                  CMP     R5,DRMLIM       ;< OR = DAC RMS NOISE LIMIT?
001474' 003413                          BLE     XRMS            ;BRANCH IF NO ERROR
001476' 004767  000676                  JSR     PC,ERCOM        ;FETCH ERROR PARAMETERS

001502' 104403  000000' 003300'         MSGN$,BEGIN,MSG3        ;ASCII MESSAGE CALL WITH COMMON HEADER

001510' 012767  000032  176370          MOV     #32,ERRTYP      ;D/A ERROR
                                ;*****************************************************************
001516' 104405  000000' 003440'         HRDERS,BEGIN,VOLTS      ;Y-D/A RMS NOISE EXCEEDED LIMIT
                                ;*****************************************************************


                                ;CALCULATE X DAC RMS NOISE USING COARSE X, FINE Y WRAPAROUND DACS.

001524' 012700  000657          XRMS:   MOV     #431.,R0        ;R0 = 84.13% OF 512 CONVERSIONS
001530' 012777  002540  001734          MOV     #2540,@ADSR     ;CH.#5, CLK. ST. AND INTERRUPT ENABLE
001536' 012777  001000  001740          MOV     #1000,@XDAC     ;SET X DAC AT MID-RANGE
001544' 016701  001736                  MOV     YDAC,R1         ;R1 = SAR DAC
001550' 004767  000464                  JSR     PC,SAR          ;GET DAC VALUE THAT PRODUCES 16/84 SPLIT
001554' 011105                          MOV     @R1,R5          ;SAVE LEFT BOUNDARY
001556' 012700  000121                  MOV     #81.,R0         ;R0 = 15.87% OF 512 CONVERSIONS
001562' 004767  000452                  JSR     PC,SAR          ;GET DAC VALUE THAT PRODUCES 84/16 SPLIT
001566' 161105                          SUB     @R1,R5          ;R5 = (A/D PLUS X DAC NOISE) X 2
001570' 166705  002500                  SUB     ARMY,R5         ;SUBTRACT OUT THE A/D NOISE
001574' 020567  002504                  CMP     R5,DRMLIM       ;< OR = DAC RMS NOISE LIMIT?
001600' 003413                          BLE     ADPK1           ;BRANCH IF NO ERROR
001602' 004767  000572                  JSR     PC,ERCOM        ;GET ERROR DATA

001606' 104403  000000' 003314'         MSGN$,BEGIN,MSG4        ;ASCII MESSAGE CALL WITH COMMON HEADER

001614' 012767  000032  176264          MOV     #32,ERRTYP      ;D/A NOISE ERROR
                                ;*****************************************************************
001622' 104405  000000' 003440'         HRDERS,BEGIN,VOLTS      ;X-D/A RMS NOISE EXCEEDED LIMIT
                                ;*****************************************************************
```



```
                                ;CALCULATE A/D PEAK NOISE USING FINE X WRAPAROUND DAC.

001630' 012700  000775          ADPK1:  MOV     #509.,R0        ;FOR PEAK OF 2 1/2 SIGMA
001634' 012777  004140  001630          MOV     #4140,@ADSR     ;CH.#10, CLK. ST. AND INTERRUPT ENABLE
001642' 016701  001636                  MOV     XDAC,R1         ;R1 = ADDRESS OF SAR DAC
001646' 004767  000366                  JSR     PC,SAR          ;GET DAC VALUE THAT GIVES .6% LOW COUNT
001652' 011105                          MOV     @R1,R5          ;R5 = LEFT BOUNDARY
001654' 012700  000003                  MOV     #3,R0           ;CHANGE SPLIT FOR RIGHT BOUNDARY
001660' 004767  000354                  JSR     PC,SAR          ;GET DAC VALUE THAT GIVES .6% HIGH COUNT
001664' 161105                          SUB     @R1,R5          ;R5 = A/D PEAK TO PEAK NOISE
001666' 010567  002404                  MOV     R5,APKX         ;SAVE FOR DAC NOISE CALCULATIONS
001672' 020567  002410                  CMP     R5,APKLIM       ;< OR = A/D PEAK NOISE LIMIT?
001676' 003413                          BLE     ADPK2           ;BRANCH IF NO ERROR
001700' 004767  000474                  JSR     PC,ERCOM        ;GET ERROR PARAMETERS

001704' 104403  000000' 003330'         MSGN$,BEGIN,MSG5        ;ASCII MESSAGE CALL WITH COMMON HEADER

001712' 012767  000031  176166          MOV     #31,ERRTYP      ;A/D NOISE ERROR
                                ;*****************************************************************
001720' 104405  000000' 003440'         HRDERS,BEGIN,VOLTS      ;A/D PEAK NOISE EXCEEDED LIMIT
                                ;*****************************************************************


                                ;CALCULATE A/D PEAK NOISE USING FINE Y WRAPAROUND DAC.

001726' 012700  000775          ADPK2:  MOV     #509.,R0        ;DEFINE SPLIT
001732' 012777  003540  001532          MOV     #3540,@ADSR     ;CH.#7, CLK. ST. AND INTERRUPT ENABLE
001740' 016701  001542                  MOV     YDAC,R1         ;R1 = ADDRESS OF SAR DAC
001744' 004767  000270                  JSR     PC,SAR          ;GET DAC VALUE THAT PRODUCES SPLIT
001750' 011105                          MOV     @R1,R5          ;R5 = LEFT BOUNDARY
001752' 012700  000003                  MOV     #3,R0           ;CHANGE SPLIT TO RIGHT BOUNDARY
001756' 004767  000256                  JSR     PC,SAR          ;AND GET DAC VALUE
001762' 161105                          SUB     @R1,R5          ;R5 = A/D PEAK TO PEAK NOISE
001764' 010567  002310                  MOV     R5,APKY         ;SAVE FOR DAC NOISE CALCULATIONS
001770' 020567  002312                  CMP     R5,APKLIM       ;< OR = A/D PEAK NOISE LIMIT?
001774' 003413                          BLE     YPEAK           ;BRANCH IF NO ERROR
001776' 004767  000376                  JSR     PC,ERCOM        ;GET ERROR DATA

002002' 104403  000000' 003344'         MSGN$,BEGIN,MSG6        ;ASCII MESSAGE CALL WITH COMMON HEADER

002010' 012767  000031  176070          MOV     #31,ERRTYP      ;A/D NOISE ERROR
                                ;*****************************************************************
002016' 104405  000000' 003440'         HRDERS,BEGIN,VOLTS      ;A/D PEAK NOISE EXCEEDED LIMIT
                                ;*****************************************************************
```

```
                                ;CALCULATE Y DAC PEAK NOISE USING COARSE Y, FINE X WRAPAROUND DACS.

002024' 012700  000775          YPEAK:  MOV     #509.,R0        ;.6% SPLIT
002030' 012777  003140  001434          MOV     #3140,@ADSR     ;CH.#6, CLK. ST. AND INTERRUPT ENABLE
002036' 012777  001000  001442          MOV     #1000,@YDAC     ;SET Y DAC FOR MID-RANGE
002044' 016701  001434                  MOV     XDAC,R1         ;R1 = ADDRESS OF SAR DAC
002050' 004767  000164                  JSR     PC,SAR          ;GET DAC VALUE THAT PRODUCES SPLIT
002054' 011105                          MOV     @R1,R5          ;R5 = LEFT BOUNDARY
002056' 012700  000003                  MOV     #3,R0           ;CHANGE SPLIT FOR RIGHT BOUNDARY
002062' 004767  000152                  JSR     PC,SAR          ;GET DAC VALUE THAT PRODUCES SPLIT
002066' 161105                          SUB     @R1,R5          ;A/D PLUS Y DAC NOISE
002070' 166705  002202                  SUB     APKX,R5         ;SUBTRACT OUT A/D NOISE
002074' 020567  002206                  CMP     R5,APKLIM       ;< OR = DAC PEAK NOISE LIMIT?
002100' 003413                          BLE     XPEAK           ;BRANCH IF NO ERROR
002102' 004767  000272                  JSR     PC,ERCOM        ;GET ERROR INFO

002106' 104403  000000' 003360'         MSGN$,BEGIN,MSG7        ;ASCII MESSAGE CALL WITH COMMON HEADER

002114' 012767  000032  175764          MOV     #32,ERRTYP      ;D/A ERROR
                                        ;***************************************************************
002122' 104405  000000' 003440'         HRDERS,BEGIN,VOLTS      ;Y-D/A PEAK NOISE EXCEEDED LIMIT
                                        ;***************************************************************


                                ;CALCULATE X DAC PEAK NOISE USING COARSE X, FINE Y WRAPAROUND DACS.

002130' 012700  000775          XPEAK:  MOV     #509.,R0        ;.6% SPLIT
002134' 012777  002540  001330          MOV     #2540,@ADSR     ;CH.#5, CLK. ST. AND INTERRUPT ENABLE
002142' 012777  001000  001334          MOV     #1000,@XDAC     ;SET X DAC TO MID-RANGE
002150' 016701  001332                  MOV     YDAC,R1         ;R1 = SAR DAC
002154' 004767  000060                  JSR     PC,SAR          ;GET DAC VALUE THAT PRODUCES SPLIT
002160' 011105                          MOV     @R1,R5          ;R5 = LEFT BOUNDARY
002162' 012700  000003                  MOV     #3,R0           ;CHANGE SPLIT FOR RIGHT BOUNDARY
002166' 004767  000046                  JSR     PC,SAR          ;GET DAC VALUE THAT PRODUCES SPLIT
002172' 161105                          SUB     @R1,R5          ;A/D PLUS X DAC PEAK TO PEAK NOISE
002174' 166705  002100                  SUB     APKY,R5         ;TAKE AWAY A/D NOISE
002200' 020567  002104                  CMP     R5,DPKLIM       ;< OR = DAC PEAK NOISE LIMIT?
002204' 003413                          BLE     DONE            ;BRANCH IF WITHIN LIMIT
002206' 004767  000166                  JSR     PC,ERCOM        ;GET ERROR PARAMETERS

002212' 104403  000000' 003374'         MSGN$,BEGIN,MSG8        ;ASCII MESSAGE CALL WITH COMMON HEADER

002220' 012767  000032  175660          MOV     #32,ERRTYP      ;D/A ERROR
                                        ;***************************************************************
002226' 104405  000000' 003440'         HRDERS,BEGIN,VOLTS      ;X-D/A PEAK NOISE EXCEEDED LIMIT
                                        ;***************************************************************


002234'                         DONE:
002234' 104413  000000'                 ENDIT$,BEGIN            ;SIGNAL END OF ITERATION.
                                                                ;MONITOR SHALL TEST END OF PASS
```



```
                                ;FIND DAC VALUE WHICH PRODUCES THE NUMBER OF HIGH COUNTS DEFINED IN R0
                                ;USING SUCCESSIVE APPROXIMATION AND WRAPAROUND DAC DEFINED IN R1.

002240' 012702  001000          SAR:    MOV     #1000,R2        ;R2 = MSB OF DAC
002244' 005011                  CLR     @R1             ;GET RID OF "ONES"
002246' 060211                  BIT:    ADD     R2,@R1          ;TRY THIS BIT
002250' 005003                          CLR     R3              ;INIT HIGH COUNT
002252' 012704  001000                  MOV     #512.,R4        ;R4 = # OF SAMPLES IN A BURST
002256' 004767  000436          CONV:   JSR     PC,RANDY        ;GET A RANDOM NUMBER
002262' 016777  000524  001210          MOV     RNA,@CLKBR      ;PUT IT IN CLOCK PRESET REGISTER
002270' 005277  001202                  INC     @CLKSR          ;START CLOCK RUNNING
002274' 104400  000000'                 EXIT$,BEGIN                     ;EXIT TO MONITOR. MODULE WAIT FOR INTERRUPT.
002300'                         WAINT:
                                        ;-------------------------------------------------------------------
002300' 000004  000000' 002306'         PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
                                        ;-------------------------------------------------------------------
002306' 027727  001162  001000  1$:     CMP     @ADBR,#1000     ;> OR = 1000?
002314' 100401                          BMI     .+4             ;BRANCH IF < 1000
002316' 005203                          INC     R3              ;COUNT IF > OR = 1000
002320' 005304                          DEC     R4              ;COUNT THROUGH BURST
002322' 001355                          BNE     CONV            ;BRANCH IF NOT DONE
002324' 020300                          CMP     R3,R0           ;HIGH COUNT > % OF 512 CONVERSIONS?
002326' 003401                          BLE     .+4             ;BRANCH TO LEAVE BIT IN
002330' 160211                          SUB     R2,@R1          ;TAKE BIT OUT
002332' 006202                          ASR     R2              ;NEXT BIT
002334' 001344                          BNE     BIT             ;AND GO RESOLVE IT IF NOT DONE
002336' 005711                          TST     @R1             ;CHECK FOR ALL "ZEROS"
002340' 001404                          BEQ     WRPERR          ;BRANCH IF INVALID RESULT
002342' 021127  001777                  CMP     @R1,#1777       ;CHECK FOR ALL "ONES"
002346' 001401                          BEQ     WRPERR          ;BRANCH IF INVALID RESULT
002350' 000207                          RTS     PC

                                ;VOLTAGE NEEDED TO PRODUCE # OF HIGH COUNTS SPECIFIED IS OUT OF THE
                                ;RANGE OF THE WRAPAROUND DAC.
                                ;CLEAR INTERUPTS, PRINT MESSAGE AND DROP MODULE.

002352' 005077  001114          WRPERR: CLR     @ADSR
002356' 005077  001120                  CLR     @DPSR
002362' 005077  001110                  CLR     @CLKSR
002366' 104403  000000' 003432'         MSGN$,BEGIN,MSG11       ;ASCII MESSAGE CALL WITH COMMON HEADER
002374' 104410  000000'                 ENDS,BEGIN              ;OUT OF THE DAC'S RANGE
```

```
                                        ;CONVERT NOISE RESULTS TO DECIMAL AND
                                        ;MEASURE AR11'S POWER SUPPLY VOLTAGES AND CALCULATE AVERAGE AND SPREAD

002400' 016767  001066  175472   ERCOM:  MOV     ADSR,CSRA       ;LOAD HEADER FOR ERRORN CALL
002406' 017767  001060  175466          MOV     @ADSR,ACSR
002414' 016767  175406  175462          MOV     STAT,ASTAT
002422' 010546                          MOV     R5,-(SP)        ;STACK BINARY NOISE VALUE
                                        ;*****************************************************************
                                                                ;CONVERT DECIM TO ASCII AND
                                                                ;STORE AT
002424' 104421  000000' 003514'         BTOD$,BEGIN,DECIM,
002432' 000000
                                        ;*****************************************************************
002434' 116767  001056  000502          MOVB    DECIM+2,VALUE   ;MOVE CONVERTED VALUES TO ASCII BUFFER
002442' 116767  001051  000476          MOVB    DECIM+3,VALUE+2
002450' 116767  001044  000471          MOVB    DECIM+4,VALUE+3
002456' 012777  002636' 175324          MOV     #ERINT,@VECTOR  ;SETUP VECTOR FOR AVERAGING ROUTINE
002464' 012777  001540  001000          MOV     #1540,@ADSR     ;CH.#3, CLK. ST. AND INTERRUPT ENABLE
002472' 012703  001740                  MOV     #1740,R3        ;R3 = EXPECTED VALUE
002476' 004767  000074                  JSR     PC,AVER         ;GET AVERAGE VALUE AND COUNT SPREAD
002502' 010067  000750                  MOV     R0,AVP14V       ;AVERAGE +14 VOLTS
002506' 010167  000746                  MOV     R1,SPP14V       ;COUNT SPREAD ON +14V
002512' 012777  002140  000752          MOV     #2140,@ADSR     ;CH.#4, CLK. ST. AND INTERRUPT ENABLE
002520' 012703  000040                  MOV     #40,R3          ;R3 = EXPECTED VALUE
002524' 004767  000046                  JSR     PC,AVER         ;GET AVERAGE AND SPREAD
002530' 010067  000726                  MOV     R0,AVN14V       ;AVERAGE -14V
002534' 010167  000724                  MOV     R1,SPN14V       ;COUNT SPREAD ON -14V
002540' 012777  027540  000724          MOV     #27540,@ADSR    ;CH.#17, CLK. ST., UNIPOLAR AND I. E.
002546' 012703  001400                  MOV     #1400,R3        ;R3 = EXPECTED VALUE
002552' 004767  000020                  JSR     PC,AVER         ;GET AVERAGE AND SPREAD
002556' 010067  000704                  MOV     R0,AVP5HQ       ;AVERAGE +5HQ
002562' 010167  000702                  MOV     R1,SPP5HQ       ;COUNT SPREAD ON +5HQ
002566' 012777  002300' 175214          MOV     #@INT,@VECTOR   ;SET VECTOR BACK FOR SAR
002574' 000207                          RTS     PC
```



```
                                        ;GET THE AVERAGE OF 512 SAMPLES AND THEIR COUNT SPREAD

002576' 005000          AVER:   CLR     R0              ;INIT R0 FOR RUNNING SUM
002600' 012704  001000          MOV     #512.,R4        ;BURST OF 512 CONVERSIONS
002604' 012701  100000          MOV     #BIT15,R1       ;INIT HIGH COUNT TO NEGATIVE #
002610' 012702  040000          MOV     #BIT14,R2       ;INIT LOW COUNT TO POSITIVE #
002614' 004767  000100  ECON:   JSR     PC,RANDY        ;GET A RANDOM NUMBER
002620' 016777  000166  000652  MOV     RNA,@CLKBR      ;PUT IT IN CLOCK PRESET REGISTER
002626' 005277  000644          INC     @CLKSR          ;START CLOCK RUNNING
002632' 104400  000000'         EXIT$,BEGIN                     ;EXIT TO MONITOR. MODULE WAIT FOR INTERRUPT.
002636'                 ERINT:
                                        ;---------------------------------------------------------------
002636' 000004  000000' 002644'         PIRQ$,BEGIN,1S          ; QUEUE UP TO CONTINUE AT 1S AND RTI
                                        ;---------------------------------------------------------------
002644' 067700  000624          1S:     ADD     @ADBR,R0        ;R0 = RUNNING SUM OF DIFFERENCES
002650' 160300                  SUB     R3,R0           ;OFFSET BY EXPECTED VALUE
002652' 027701  000616          CMP     @ADBR,R1        ;HIGHER THAN PREVIOUS HIGH?
002656' 003402                  BLE     2S              ;BRANCH IF NOT A NEW HIGH
002660' 017701  000610          MOV     @ADBR,R1        ;NEW HIGH RESULT
002664' 027702  000604  2S:     CMP     @ADBR,R2        ;LOWER THAN PREVIOUS LOW?
002670' 002002                  BGE     3S              ;BRANCH IF NO
002672' 017702  000576          MOV     @ADBR,R2        ;NEW LOW RESULT
002676' 005304          3S:     DEC     R4              ;COUNT THROUGH BURST
002700' 001345                  BNE     ECON            ;DO ANOTHER SAMPLE IF NOT DONE
002702' 000300                  SWAB    R0              ;DIVIDE SUM OF DIFFERENCES BY 256
002704' 110000                  MOVB    R0,R0           ;SIGN EXTEND
002706' 006200                  ASR     R0              ;DIVIDE BY 2 MORE
002710' 005500                  ADC     R0              ;ROUND UP FOR > 1/2
002712' 060300                  ADD     R3,R0           ;OFFSET AVERAGE BY EXPECTED RESULT
002714' 160201                  SUB     R2,R1           ;R1 = COUNT SPREAD
002716' 000207                  RTS     PC

                                        ;GENERATE A RANDOM NUMBER

002720' 066767  000070  000064  RANDY:  ADD     RNB,RNA
002726' 066767  000064  000056          ADD     RNC,RNA
002734' 005567  000052                  ADC     RNA
002740' 066767  000046  000046          ADD     RNA,RNB
002746' 066767  000044  000040          ADD     RNC,RNB
002754' 005567  000034                  ADC     RNB
002760' 066767  000026  000030          ADD     RNA,RNC
002766' 066767  000022  000022          ADD     RNB,RNC
002774' 005567  000016                  ADC     RNC
003000' 126727  000006  000300          CMPB    RNA,#300        ;CHECK FOR SMALL NEGATIVE NUMBERS
003006' 101344                  BHI     RANDY           ;GET ANOTHER RANDOM NUMBER
003010' 000207                  RTS     PC
003012' 063241          RNA:    063241
003014' 142315          RNB:    142315
003016' 127623          RNC:    127623
```

```
                                ;PHRASES AND MSG POINTERS
003020' 040445  042057  000040  P1:     .ASCIZ          "%A/D "
003026' 054045  042055  040457  P2:     .ASCIZ          "%X-D/A "
003034' 000040
003036' 054445  042055  040457  P3:     .ASCIZ          "%Y-D/A "
003044' 000040
003046' 046522  020123     000  P4:     .ASCIZ          "RMS "
003053'    000                          .BYTE
003054' 042520  045501  000040  P5:     .ASCIZ          "PEAK "
003062' 047516  051511  020105  P6:     .ASCIZ          "NOISE = "
003070' 020075     000
003073'    000                          .BYTE
003074' 027060  032462  046040  P7:     .ASCIZ          "0.25 LSB)"
003102' 041123  000051
003106' 027062  030060  046040  P8:     .ASCIZ          "2.00 LSB)"
003114' 041123  000051
003120' 027060  032462  046040  P9:     .ASCIZ          "0.25 LSB)"
003126' 041123  000051
003132' 027062  030060  046040  P10:    .ASCIZ          "2.00 LSB)"
003140' 041123  000051
003144' 027130  054130  046040  VALUE:  .ASCIZ          "X.XX LSB (LIMIT = "
003152' 041123  024040  044514
003160' 044515  020124  020075
003166'    000
003167'    000                          .BYTE
003170' 040445  030522  020061  P11:    .ASCIZ          "%AR11 RUNNING IN "
003176' 052522  047116  047111
003204' 020107  047111  000040
003212' 047516  026516     000  P12:    .ASCIZ          "NON-"
003217'    000                          .BYTE
003220' 051127  050101  051101  P13:    .ASCIZ          "WRAPAROUND "
003226' 052517  042116  000040
003234' 047515  042504     000  P14:    .ASCIZ          "MODE"
003241'    000                          .BYTE
003242' 051105  047522  000122  P15:    .ASCIZ          "ERROR"
003250' 003020'                 MSG1:   P1
003252' 003046'                         P4
003254' 003062'                         P6
003256' 003144'                         VALUE
003260' 003074'                         P7
003262' 177777                          177777
003264' 003020'                 MSG2:   P1
003266' 003046'                         P4
003270' 003062'                         P6
003272' 003144'                         VALUE
003274' 003074'                         P7
003276' 177777                          177777
```



```
003300' 003036'                 MSG3:   P3
003302' 003046'                         P4
003304' 003062'                         P6
003306' 003144'                         VALUE
003310' 003120'                         P9
003312' 177777                          177777
003314' 003026'                 MSG4:   P2
003316' 003046'                         P4
003320' 003062'                         P6
003322' 003144'                         VALUE
003324' 003120'                         P9
003326' 177777                          177777
003330' 003020'                 MSG5:   P1
003332' 003054'                         P5
003334' 003062'                         P6
003336' 003144'                         VALUE
003340' 003106'                         P8
003342' 177777                          177777
003344' 003020'                 MSG6:   P1
003346' 003054'                         P5
003350' 003062'                         P6
003352' 003144'                         VALUE
003354' 003106'                         P8
003356' 177777                          177777
003360' 003036'                 MSG7:   P3
003362' 003054'                         P5
003364' 003062'                         P6
003366' 003144'                         VALUE
003370' 003132'                         P10
003372' 177777                          177777
003374' 003026'                 MSG8:   P2
003376' 003054'                         P5
003400' 003062'                         P6
003402' 003144'                         VALUE
003404' 003132'                         P10
003406' 177777                          177777
003410' 003170'                 MSG9:   P11
003412' 003212'                         P12
003414' 003220'                         P13
003416' 003234'                         P14
003420' 177777                          177777
003422' 003170'                 MSG10:  P11
003424' 003220'                         P13
003426' 003234'                         P14
003430' 177777                          177777
003432' 003220'                 MSG11:  P13
003434' 003242'                         P15
003436' 177777                          177777
```

```
  901  003440' 003456'                    VOLTS:  AVP14V
  902  003442' 003460'                            SPP14V
  903  003444' 003462'                            AVN14V
  904  003446' 003464'                            SPN14V
  905  003450' 003466'                            AVP5HQ
  906  003452' 003470'                            SPP5HQ
  907  003454' 177777                             177777
  908  003456' 000000                  AVP14V: OPEN
  909  003460' 000000                  SPP14V: OPEN
  910  003462' 000000                  AVN14V: OPEN
  911  003464' 000000                  SPN14V: OPEN
  912  003466' 000000                  AVP5HQ: OPEN
  913  003470' 000000                  SPP5HQ: OPEN
  914  003472' 000000                  ADSR:   OPEN
  915  003474' 000000                  ADBR:   OPEN
  916  003476' 000000                  CLKSR:  OPEN
  917  003500' 000000                  CLKBR:  OPEN
  918  003502' 000000                  DPSR:   OPEN
  919  003504' 000000                  XDAC:   OPEN
  920  003506' 000000                  YDAC:   OPEN
  921  003510' 000000                  CLKCNT: OPEN
  922  003512' 000000                  WRPFLG: OPEN
  923  003514' 000003                  DECIM:  .BLKW   3
  924  003522' 001002  000004  001002  CARGEN: .WORD   1002,4,1002,10,1002,14,1006,0,1006,20,1012,0,1012,20,1016,4,1016,10,1016
  925  003530' 000010  001002  000014
  926  003536' 001006  000000  001006
  927  003544' 000020  001012  000000
  928  003552' 001012  000020  001016
  929  003560' 000004  001016  000010
  930  003566' 001016  000014
  931  003572' 001004  000000  001005          .WORD   1004,0,1005,15,1006,0,1010,0,1010,4,1010,10,1010,14,1010,20,1012,0,1014,
  932  003600' 000015  001006  000000
  933  003606' 001010  000000  001010
  934  003614' 000004  001010  000010
  935  003622' 001010  000014  001010
  936  003630' 000020  001012  000000
  937  003636' 001014  000000
  938  003642' 001002  000000  001002          .WORD   1002,0,1002,4,1006,0,1006,10,1006,20,1012,0,1012,10,1012,20,1016,0,1016,
  939  003650' 000004  001006  000000
  940  003656' 001006  000010  001006
  941  003664' 000020  001012  000000
  942  003672' 001012  000010  001012
  943  003700' 000020  001016  000000
  944  003706' 001016  000014
  945  003712' 001002  000000  001002          .WORD   1002,0,1002,20,1006,0,1006,10,1006,20,1012,0,1012,10,1012,20,1016,4,1016
  946  003720' 000020  001006  000000
  947  003726' 001006  000010  001006
  948  003734' 000020  001012  000000
  949  003742' 001012  000010  001012
  950  003750' 000020  001016  000004
  951  003756' 001016  000014
```



```
  952  003762' 001002  000010  001002          .WORD   1002,10,1002,14,1002,20,1006,10,1012,0,1012,4,1012,10,1012,14,1012,20,10
  953  003770' 000014  001002  000020
  954  003776' 001006  000010  001012
  955  004004' 000000  001012  000004
  956  004012' 001012  000010  001012
  957  004020' 000014  001012  000020
  958  004026' 001016  000010
  959  004032' 001002  000002  001002          .WORD   1002,2,1002,12,1002,20,1007,0,1007,12,1010,20,1014,2,1014,11,1016,6,1016
  960  004040' 000012  001002  000020
  961  004046' 001007  000000  001007
  962  004054' 000012  001010  000020
  963  004062' 001014  000002  001014
  964  004070' 000011  001016  000006
  965  004076' 001016  000020
  966  004102' 001002  000004  001002          .WORD   1002,4,1002,10,1002,14,1006,0,1006,10,1006,20,1012,0,1012,10,1012,20,101
  967  004110' 000010  001002  000014
  968  004116' 001006  000000  001006
  969  004124' 000010  001006  000020
  970  004132' 001012  000000  001012
  971  004140' 000010  001012  000020
  972  004146' 001016  000004
  973  004152' 001002  000020  001005          .WORD   1002,20,1005,20,1010,20,1011,0,1012,4,1013,7,1013,20,1014,12,1015,15,101
  974  004160' 000020  001010  000020
  975  004166' 001011  000000  001012
  976  004174' 000004  001013  000007
  977  004202' 001013  000020  001014
  978  004210' 000012  001015  000015
  979  004216' 001016  000020
  980  004222' 000761  000000  000762  D10:    .WORD   761,0,762,15,763,0,765,0,765,4,765,10,765,14,765,20,767,0,771,0
  981  004230' 000015  000763  000000
  982  004236' 000765  000000  000765
  983  004244' 000004  000765  000010
  984  004252' 000765  000014  000765
  985  004260' 000020  000767  000000
  986  004266' 000771  000000
  987
  988
  989
  990  004272' 000000                  ARMX:   OPEN
  991  004274' 000000                  ARMY:   OPEN
  992  004276' 000000                  APKX:   OPEN
  993  004300' 000000                  APKY:   OPEN
  994
  995
  996                                  ;NOISE LIMITS IN 1/100THS LSB
  997  004302' 000031                  ARMLIM: 25.                     ;A/D RMS NOISE LIMIT
  998  004304' 000031                  DRMLIM: 25.                     ;D/A RMS NOISE LIMIT
  999  004306' 000310                  APKLIM: 200.                    ;A/D PEAK NOISE LIMIT
 1000  004310' 000310                  DPKLIM: 200.                    ;D/A PEAK NOISE LIMIT
 1001          000001                          .END
```

```
ACSR    000102R          316#    725*    696     769     771     773     774     776     915#
ADBR    003474R          338*    429
ADDR    000006R          282#    335
ADDR22= 001000           334#
ADPK1   001630R          575     587#
ADPK2   001726R          597     610#
ADRMS1  001224R          491     495#
ADRMS2  001322R          505     518#
ADSR    003472R          336*    393#    408*    427*    450     468*    469     479     496*    519*    541*    565*    588*
                         611*    633*    657*    716*    724     725     738*    743*    748*    914#
APKLIM  004306R          365     596     619     642     999#
APKX    004276R          595*    641     992#
APKY    004300R          618*    665     993#
ARMLIM  004302R          356     504     527     997#
ARMX    004272R          503*    549     990#
ARMY    004274R          526*    573     991#
ASB     000106R          320#
ASTAT   000104R          318#    726*
AVER    002576R          740     745     750     757#
AVN14V  003462R          746*    903     910#
AVP14V  003456R          741*    901     908#
AVP5HQ  003466R          751*    905     912#
AWAS    000110R          321#
BEGIN   000000R          279#    356     365     374     383     395     396     407     426     433     443     446     457
                         460     481     486     508     512     531     535     554     558     578     582     600     604
                         623     627     646     650     670     674     678     691     694     719     720     731     764
                         767
BIT     002246R          685#    705
BIT0    = 000001         334#
BIT1    = 000002         334#
BIT10   = 002000         334#
BIT11   = 004000         334#    450
BIT12   = 010000         334#
BIT13   = 020000         334#
BIT14   = 040000         334#    760
BIT15   = 100000         334#    759
BIT2    = 000004         334#
BIT3    = 000010         334#
BIT4    = 000020         334#
BIT5    = 000040         334#
BIT6    = 000100         334#
BIT7    = 000200         334#
BIT8    = 000400         334#
BIT9    = 001000         334#
BREAK$= 104407           334#    395     396
BRI     000012R          284#    391     418     490
BR2     000013R          285#    421
BTOD$ = 104421           334#    356     365     374     383     731
CARGEN  003522R          410     478     924#
CDATA$= 104412           334#
CHEXIT  001072R          451     464#
CHNUM   000772R          441#    449
CHTEN   001044R          455#    463
CLCNT   001162R          424     483#
CLKBR   003500R          342*    424*    689*    762*    917#
CLKCNT  003510R          350*    921#
```



```
CLKSR   003476R          340*    413*    425*    487*    690*    718*    763*    916#
CONFIG  000056R          304#
CONV    002256R          688#    700
CSRA    000100R          314#    724*
DATCK$= 104411           334#
DATER$= 104404           334#
DECIM   003514R          356     359     360     361     365     368     369     370     374     377     378     379     383
                         386     387     388     731     734     735     736     923#
DONE    002234R          667     677#
DPKLIM  004310R          383     666     1000#
DPSR    003502R          344*    394*    412*    717*    918#
DRMLIM  004304R          374     550     574     998#
DVID1   000014R          286#
DIO     004222R          452     980#
ECON    002614R          761#    778
END1T$= 104413           334#    481     678
END$  = 104410           334#    720
ERCOM   002400R          506     529     552     576     598     621     644     668     724#
ERINT   002636R          737     765#
ERRTYP  000106R          319#    510*    533*    556*    580*    602*    625*    648*    672*
EXIT$ = 104400           334#    426     443     457     691     764
FRMCNT  001164R          423     484#
GETPA$= 104415           334#
GWBUF$= 104414           334#
HRDCNT  000044R          299#
HRDER$= 104405           334#    512     535     558     582     604     627     650     674
HRDPAS  000050R          301#
ICONT   000036R          296#
ICOUNT  000040R          297#
IDNUM   000122R          326#
INIT    000030R          293#
INTR    000120R          325#    405*
LIMIN   000306R          352#
MAP22$= 104416           334#
MODNAM  000000R          280#
MODSP   000224R          294     332#
MSGN$ = 104403           334#    407     486     508     531     554     578     600     623     646     670     719
MSGS$ = 104402           334#
MSG$  = 104401           334#
MSG1    003250R          508     841#
MSG10   003422R          486     894#
MSG11   003432R          719     898#
MSG2    003264R          531     847#
MSG3    003300R          554     853#
MSG4    003314R          578     859#
MSG5    003330R          600     865#
MSG6    003344R          623     871#
MSG7    003360R          646     877#
MSG8    003374R          670     883#
MSG9    003410R          407     889#
NAINT   000726R          415     429#
NCINT   000720R          417     427#
NDINT   000736R          420     431#    464
NDINT2  001006R          438     444#
NDINT3  001060R          454     458#
NOWRAP  000556R          398     405#
```

```
NULL   = 000000            334#
OPEN   = 000000            281     287     288     289     290     307     308     309     310     311     312     313     314
                           316     319     320     321     323     324     325     334#    908     909     910     911     912
                           913     914     915     916     917     918     919     920     921     922     990     991     992
                           993
OTOAS  = 104420            334#
PASCNT  000034R            295#
PIRQS  = 000004            334#    433     446     460     694     767
POPSP  = 005726            334#
POPSP2= 022626             334#
PRIME   000702R            424#    437     467
PRTY   = 000000            334#
PRTY0  = 000000            334#
PRTY1  = 000040            334#
PRTY2  = 000100            334#
PRTY3  = 000140            334#
PRTY4  = 000200            285     334#
PRTY5  = 000240            334#
PRTY6  = 000300            284     334#
PRTY7  = 000340            334#
PS     = 177776            334#
PSW    = 177776            334#
PUSH   = 005746            334#
PUSH2  = 024646            334#
P1      003020R            807#    841     847     865     871
P10     003132R            386*    387*    388*    824#    881     887
P11     003170R            831#    889     894
P12     003212R            834#    890
P13     003220R            836#    891     895     898
P14     003234R            838#    892     896
P15     003242R            840#    899
P2      003026R            808#    859     883
P3      003036R            810#    853     877
P4      003046R            812#    842     848     854     860
P5      003054R            814#    866     872     878     884
P6      003062R            815#    843     849     855     861     867     873     879     885
P7      003074R            359*    360*    361*    818#    845     851
P8      003106R            368*    369*    370*    820#    869     875
P9      003120R            377*    378*    379*    822#    857     863
RANDY   002720R            688     761     790#    800
RAND$  = 104417            334#
RANNUM  000054R            303#
RESTRT  000224R            322     335#
RES1    000056R            305#
RES2    000060R            306#
RNA     003012R            689     762     790*    791*    792*    793     796     799     802#
RNB     003014R            790     793*    794*    795*    797     803#
RNC     003016R            791     794     796*    797*    798*    804#
RSTRT   000112R            322#    411*    491*
SAINT   000550R            390     402#
SAR     002240R            498     501     521     524     544     547     568     571     590     593     613     616     636
                           639     660     663     683#
SBADR   000102R            315#
SENSE   000456R            389#
SOFCNT  000042R            298#
SOFERS= 104406             334#
```



```
SOFPAS  000046R            300#
SPN14V  003464R            747*    904     911#
SPOINT  000032R            294#
SPP14V  003460R            742*    902     909#
SPPSHQ  003470R            752*    906     913#
SPSIZ  = 000040              1#    327
SR1     000016R            287#
SR2     000020R            288#
SR3     000022R            289#
SR4     000024R            290#
START   000224R            293     334#
STAT    000026R            292#    726
SVR0    000062R            307#
SVR1    000064R            308#
SVR2    000066R            309#
SVR3    000070R            310#
SVR4    000072R            311#
SVR5    000074R            312#
SVR6    000076R            313#
SWEEP   000676R            411     423#    480
SYSCNT  000052R            302#
TRPDFD= 000022             334#
VALUE   003144R            734*    735*    736*    826#    844     850     856     862     868     874     880     886
VECTOR  000010R            283#    389     414     488     737*    751*
VOLTS   003440R            512     535     558     582     604     627     650     674     901#
WAINT   002300R            489     692#    753
WASADR  000104R            317#
WDFR    000116R            324#    400*    406*
WDTO    000114R            323#    399*
WRAP    001166R            401     485#
WRPERR  002352R            707     709     716#
WRPFLG  003512R            392*    397     402*    922#
XDAC    003504R            346*    409*    435*    436     441*    455*    465*    497     543     566*    589     635     658*
                           919#
XFLAG   000005R            281#
XPEAK   002130R            643     656#
XRMS    001524R            551     564#
YDAC    003506R            348*    429*    442*    456*    520     542*    567     612     634*    659     920#
YPEAK   002024R            620     632#
YRMS    001420R            528     540#
.      = 004312R            697     702     923#

. ABS.  000000      000
        004312      001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 XARAB0,XARAB0/SOL/CRF:SYM=DDXCOM,XARAB0
 RUN-TIME:  1  2  .5 SECONDS
 RUN-TIME RATIO: 20/5=3.7
 CORE USED:  7K  (13 PAGES)
```