.REM_

IDENTIFICATION

PRODUCT CODE: AC-F422A-MC

PRODUCT NAME: CXMNCA0 MNCKW MODULE

PRODUCT DATE: FEBRUARY 1979

MAINTAINER: RAY SHOOP

1.0 ABSTRACT

THE MNC IS AN IOMOD THAT EXERCISES THE MNCKW REAL TIME CLOCK. ON START IT EXERCISES THE CSRS AND PRESET BUFFERS OF THE CLOCK. THEN AND FOR ALL RESTARTS, IT EXERCISES THE CLOCK AT EACH ONE OF ITS BASIC RATES. UP TO 8 MNCKW WILL BE EXERCISED WITH THIS MODULE. THE "MNA" (A/D) MODULE CAN BE ENABLED TO USE THE CLOCK TO START THE A/D. IF YOU HAVE ENABLED "MNA" TO USE THE CLOCK, YOU SHOULD DESELECT THIS EXERCISER MODULE.

2.0 REQUIREMENTS

HARDWARE: ONE MNCKW (CLOCK).

STORAGE:: MNC REQUIRES:
DECIMAL WORDS: 508
OCTAL WORDS: 775
OCTAL BYTES: 1772

3.0 PASS DEFINITION

ONE PASS OF THE MNC MODULE CONSISTS OF GENERATING INTERRUPTS FOR ONE SECOND AT EACH CLOCK RATE, UNTIL 60 SECONDS HAVE ELAPSED.

4.0 EXECUTION TIME

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

5.0 CONFIGURATION REQUIREMENTS

DEFAULT PARAMETERS:

DEVADR: 171020, VECTOR 440, BR1: 4

DEVCNT: 1, SR1: 0

REQUIRED PARAMETERS:

NONE.

6.0 DEVICE/OUTPUT SET-UP

THE FRONT PANEL SWITCHES MUST SELECT THE SCHMITT TRIGGER INPUT (PULL OUT "ST1" AND "ST2" SWITCHES AND ROTATE TO THE END).

7.0 MODULE OPERATION

TEST SEQUENCE:

1. (START) BIT EXERCISE CSR, PRESET REGISTER OF CLOCK.

2. (RESTRT) COUNT TESTS USING INTERRUPTS COUNT INTERRUPTS WILL OCCUR IN ONE SECOND AND ADVANCE THE TEST TO THE NEXT RATE.

   AFTER A RATE HAS BEEN SELECTED, A CHECK IS MADE TO SEE IF THE OPERATOR HAS INHIBITED THAT RATE FROM TEST. IF NOT, CONTROL IS TRANSFERRED TO THE PARTICULAR RATE ROUTINE (LISTED BELOW). EACH RATE ROUTINE MUST PRELOAD THE BUFFER REGISTER OF THE CLOCK TO THE COUNT THAT WILL CAUSE IT TO INTERRUPT IN ONE SECOND. AFTER THE BUFFER IS LOADED, THE CSR IS LOADED WITH THE PROPER BITS THAT SELECT THE RATE.

   A. COUNT TEST CLOCK RATE: 1MHZ.

   B. COUNT TEST CLOCK RATE: 100KHZ.

   C. COUNT TEST CLOCK RATE: 10KHZ.

   D. COUNT TEST CLOCK RATE: 1KHZ.

   E. COUNT TEST CLOCK RATE: 100HZ.

   F. COUNT TEST CLOCK RATE: PSEUDO RANDOM (1 OF 3 RATES).

8.0 OPERATION OPTIONS

VALID SR1 VALUES

SR1 BIT	ENABLE/DISABLE	FUNCTION
0	0	ENABLE TESTING 1MHZ
	1	DISABLE TESTING 1MHZ
1	0	ENABLE TESTING 100KHZ
	1	DISABLE TESTING 100KHZ
2	0	ENABLE TESTING 10KHZ
	1	DISABLE TESTING 10KHZ
3	0	ENABLE TESTING 1KHZ
	1	DISABLE TESTING 1KHZ
4	0	ENABLE TESTING 100HZ
	1	DISABLE TESTING 100HZ
5	0	*ENABLE TESTING RANDOM
	1	DISABLE TESTING RANDOM

*NOTE: IF RANDOM RATE SELECTED, THEN AN SR1 BIT DISABLING A PARTICULAR RATE WILL BE IGNORED.

9.0 NON-STANDARD PRINTOUTS

ALL PRINTOUTS HAVE THE STANDARD FORMATS DESCRIBED IN THE DEC/X11 DOCUMENT.

```
                                        IOMOD   <MNCA >,171020,440,4,0,0,60.,0
                                        MODULE  140000,MNCA ,171020,440,4,0,0,60.,0
                                        .TITLE  MNCA DEC/X11 SYSTEM EXERCISER MODULE
                                ;       DDXCOM  VERSION  6       23-MAY-78
                                        .LIST   BIN
                                ;*****************************************************************
000000'                         BEGIN:
000000' 047115  040503      040 MODNAM: .ASCII  /MNCA / ;MODULE NAME.
000005'     000                 XFLAG:  .BYTE   OPEN                ;USED TO KEEP TRACK OF WBUFF USAGE
000006' 171020                  ADDR:   171020+0                    ;1ST DEVICE ADDR.
000010' 000440                  VECTOR: 440+0                       ;1ST DEVICE VECTOR.
000012'     200                 BR1:    .BYTE   PRTY4+0             ;1ST BR LEVEL.
000013'     000                 BR2:    .BYTE   PRTY0+0             ;2ND BR LEVEL.
000014' 000001                  DVID1:  0+1                         ;DEVICE INDICATOR 1.
000016' 000000                  SR1:    OPEN                        ;SWITCH REGISTER 1
000020' 000000                  SR2:    OPEN                        ;SWITCH REGISTER 2
000022' 000000                  SR3:    OPEN                        ;SWITCH REGISTER 3
000024' 000000                  SR4:    OPEN                        ;SWITCH REGISTER 4
                                ;*****************************************************************
000026' 140000                  STAT:   140000                      ;STATUS WORD.
000030' 000254'                 INIT:   START                       ;MODULE START ADDR.
000032' 000224'                 SPOINT: MODSP                       ;MODULE STACK POINTER.
000034' 000000                  PASCNT: 0                           ;PASS COUNTER.
000036' 000074                  ICONT:  60.                         ;# OF ITERATIONS PER PASS=60.
000040' 000000                  ICOUNT: 0                           ;LOC TO COUNT ITERATIONS
000042' 000000                  SOFCNT: 0                           ;LOC TO SAVE TOTAL SOFT ERROPS
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'                         SRADR:                              ;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' 000254'                 RSTRT:  RESTPT                      ;RESTART ADDRESS AFTER END OF PASS
000114' 000000                  WDTO:   OPEN                        ;WORDS TO MEMORY PER ITERATION
000116' 000000                  WDFR:   OPEN                        ;WORDS FROM MEMORY PER ITERATION
000120' 000000                  INTR:   OPEN                        ;# OF INTERRUPTS PER ITERATION
000122' 000000                  IDNUM:  0                           ;MODULE IDENTIFICATION NUMBER=0
        000040                          .REPT   SPSIZ               ;MODULE STACK STARTS HERE.
                                        .NLIST
```



```
                                        .WORD   0
                                        .LIST
                                        .ENDR
000224'                         MODSP:
                                ;*****************************************************************
000224' 000001                  TEMP:   1                           ;MASK OF CURRENT UNIT

                                ;MODULE REQUIRED REGISTERS - SET UP BY THIS MODULE.

000226' 171020                  ASR:    .WORD   171020              ;CLOCK A STATUS REG.
000230' 171022                  ABR:    .WORD   171022              ;CLOCK A BUFFER REG.

000232' 000440                  AVECT:  .WORD   440                 ;CLOCK A INTERRUPT VECTOR.
000234' 000442                  AVECT2: .WORD   442

000236' 000444                  BVECT:  .WORD   444                 ;CLOCK INTERRUPT VECTOR.
000240' 000446                  BVECT2: .WORD   446

000242' 000001                  RATEP:  .WORD   1                   ;POINTS TO CURRENT RATE
000244' 000000                  OFF:    .WORD   0                   ;OFFSET TO TAKE US TO RATE ROUTINE
000246' 000000                  RANA:   .WORD   0                   ;RANDOM NUMBER.
000250' 000000                  RANB:   .WORD   0                   ;RANDOM NUMBER.
000252' 000000                  AIFLG:  .WORD   0                   ;FLAG TO SHOW THAT CLOCK A HAS INTERRUPTED.

000254'                         RESTPT:
000254' 004767  000002          START:  JSR     PC,START0           ;PRIME THE ADDRESSES
000260' 000442                          BR      LOG1                ;RUN LOGIC TEST

000262' 012767  000001  177734  START0: MOV     #BIT0,TEMP          ;LOAD CURRENT UNIT MASK
000270' 016767  177512  177730  START1: MOV     ADDR,ASR            ;GET BASE ADDR.
000276' 016767  177506  177726          MOV     VECTOR,AVECT        ;GET BASE VECTOR ADDR.
000304' 016767  177716  177716  START2: MOV     ASR,ABR             ;NOW WE'RE GONNA FIX
000312' 062767  000002  177710          ADD     #2,ABR              ;ALL CLOCK ADDRESSES BASED ON ASR.
000320' 016700  177706                  MOV     AVECT,R0            ;NOW FIX VECTOR ADDRESSES
000324' 062700  000004                  ADD     #4,R0
000330' 010067  177702                  MOV     R0,BVECT
000334' 016767  177672  177672          MOV     AVECT,AVECT2
000342' 062767  000002  177664          ADD     #2,AVECT2
000350' 016767  177662  177662          MOV     BVECT,BVECT2
000356' 062767  000002  177654          ADD     #2,BVECT2
000364' 000207                          RTS     PC                  ;EXIT
```

```
                                        ;*
                                        ;*LOGIC TEST #1 BE SURE A CLOCK EXISTS AT THE
                                        ;*SPECIFIED ADDR, IF NO CLOCK, THEN A
                                        ;*DEC/X11 SYS ERROR WILL OCCUR.
                                        ;*

000366' 005777  177634          LOG1:   TST     @ASR            ;ADDRESS THE CLOCK, IF SYS ERROR
                                        ;OCCURS, THEN CLOCK DID NOT
                                        ;RETURN SLAVE-SYN WHEN
                                        ;ADDRESSED.

                                        ;*
                                        ;*LOGIC TEST #2, MAKE SURE CLOCK CSR BITS
                                        ;*14,11,6,5,2 AND 0 CAN BE SET + CLEARED.
                                        ;*

000372' 012767  044125  177504  LOG2:   MOV     #044125,ASTAT   ;GENERATE + RECORD PATTERN TO BE USED.
000400' 016777  177500  177620          MOV     ASTAT,@ASR      ;SET THEM IN CSR OF CLOCK A.
000406' 017767  177614  177466          MOV     @ASR,ACSR       ;READ THEM BACK
000414' 026767  177464  177460          CMP     ASTAT,ACSR      ;DID THEY ALL SET?

000422' 001412                          BEQ     2$              ;YES - GO TO NEXT TEST.
000424' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
000430' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000434' 016767  177566  177436  1$:     MOV     ASR,CSRA        ;RECORD CSR'S ADDR
                                        ;****************************************************************
000442' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;;PATTERN 044125 FAILED
                                        ;****************************************************************
000450' 005077  177552          2$:     CLR     @ASR            ;TRY CLEARING THE BITS
000454' 017767  177546  177420          MOV     @ASR,ACSR       ;READ IT BACK.
000462' 001414                          BEQ     LOG3            ;IF ZERO CSR GOOD.
000464' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
000470' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000474' 005067  177404          3$:     CLR     ASTAT           ;EXPECT ZERO CSR.
000500' 016767  177522  177372          MOV     ASR,CSRA        ;RECORD CSR'S ADDR.
                                        ;****************************************************************
000506' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;;CSR FAILED TO CLEAR
                                        ;****************************************************************
```



```
                                        ;*
                                        ;*LOGIC TEST #3, MAKE SURE CLOCK CSR BITS
                                        ;*13,5,3 AND 1 CAN BE SET + CLEARED
                                        ;*

000514' 012767  020052  177362  LOG3:   MOV     #020052,ASTAT   ;GENERATE + RECORD PATTERN TO BE USED.
000522' 016777  177356  177476          MOV     ASTAT,@ASR      ;SET THEM IN CSR OF CLOCK A.
000530' 017767  177472  177344          MOV     @ASR,ACSR       ;READ THEM BACK
000536' 026767  177342  177336          CMP     ASTAT,ACSR      ;DID THEY ALL SET?
000544' 001412                          BEQ     2$              ;YES - GO TO NEXT TEST.
000546' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
000552' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000556' 016767  177444  177314  1$:     MOV     ASR,CSRA        ;RECORD CSR'S ADDR.
                                        ;****************************************************************
000564' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;;CSR PATTERN 020052 FAILED
                                        ;****************************************************************
000572' 005077  177430          2$:     CLR     @ASR            ;TRY CLEARING THE BITS
000576' 017767  177424  177276          MOV     @ASR,ACSR       ;READ IT BACK.
000604' 001414                          BEQ     LOG4            ;IF ZERO CSR GOOD.
000606' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
000612' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000616' 005067  177262          3$:     CLR     ASTAT           ;EXPECT ZERO CSR.
000622' 016767  177400  177250          MOV     ASR,CSRA        ;RECORD CSR'S ADDR.
                                        ;****************************************************************
000630' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;;CSR FAILED TO CLEAR
                                        ;****************************************************************
                                        ;*
                                        ;*LOGIC TEST #4, MAKE SURE CLOCK BUFFER REG
                                        ;*PATTERN 125252 CAN BE SET + CLEARED.
                                        ;*

000636' 012767  125252  177240  LOG4:   MOV     #125252,ASTAT   ;GENERATE + RECORD PATTERN TO BE USED.
000644' 016777  177234  177356          MOV     ASTAT,@ABR      ;SET THEM IN BUFFER REG OF CLOCK.
000652' 017767  177352  177222          MOV     @ABR,ACSR       ;READ THEM BACK
000660' 026767  177220  177214          CMP     ASTAT,ACSR      ;DID THEY ALL SET?
000666' 001412                          BEQ     2$              ;YES - GO TO NEXT TEST.
000670' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
000674' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000700' 016767  177324  177172  1$:     MOV     ABR,CSRA
                                        ;****************************************************************
000706' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;;BUFFER REG PATTERN 125252 FAILED
                                        ;****************************************************************
000714' 005077  177310          2$:     CLR     @ABR            ;TRY CLEARING THE BITS
000720' 017767  177304  177154          MOV     @ABR,ACSR       ;READ IT BACK.
000726' 001414                          BEQ     LOG5            ;IF ZERO BUFFER GOOD.
000730' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
000734' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000740' 005067  177140          3$:     CLR     ASTAT           ;EXPECT ZERO BUFFER.
000744' 016767  177260  177126          MOV     ABR,CSRA        ;RECORD ADDR. OF BUFFER REG.
                                        ;****************************************************************
000752' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;;BUFFER REG FAILED TO CLEAR
                                        ;****************************************************************
                                        ;*LOGIC TEST #5, MAKE SURE CLOCK BUFFER REG
                                        ;*PATTERN 052525 CAN BE SET + CLEARED

000760' 012767  052525  177116  LOG5:   MOV     #052525,ASTAT   ;GENERATE + RECORD PATTERN TO BE USED.
000766' 016777  177112  177234          MOV     ASTAT,@ABR      ;SET THEM IN BUFFER OF CLOCK A.
```

```
001774' 017767  177230  177100          MOV     @ABR,ACSR       ;READ THEM BACK
001002' 026767  177076  177072          CMP     ASTAT,ACSR      ;DID THEY ALL SET?
001010' 001412                          BEQ     2$              ;YES - GO TO NEXT TEST.
001012' 104407  000003'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
001016' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
001022' 016767  177202  177050  1$:     MOV     ABR,CSRA        ;RECORD BUFFER REG ADDR.
                                        ;*************************************************************
001030' 104405  000000' 000000          HRDER$,BEGIN,NULI       ;;BUFF REG PATTERN 052525 FAILED
                                        ;*************************************************************
001036' 005077  177166          2$:     CLR     @ABP            ;TRY CLEARING THE BITS
001042' 017767  177162  177032          MOV     @ABR,ACSR       ;READ IT BACK.
001050' 001414                          BEQ     RFST0           ;IF ZERO BUFFER GOOD,
001052' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
001056' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
001062' 005067  177016          3$:     CLR     ASTAT           ;EXPECT ZERO BUFFER,
001066' 016767  177136  177004          MOV     ABR,CSRA        ;RECORD BUFFER REG A ADDR,
                                        ;*************************************************************
001074' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;;BUFFER REG FAILED TO CLEAR
                                        ;*************************************************************
001102' 006367  177116          REST0:  ASL     TEMP            ;CHECK FOR NEXT UNIT
001106' 022767  000400  177110          CMP     #BIT8,TEMP      ;TEST IF VALID UNIT
001114' 001420                          BEQ     REST1           ;BR IF NOT
001116' 036767  177102  176670          BIT     TEMP,DVID1      ;TEST IF THIS UNIT IS SELECTED
001124' 001414                          BEQ     REST1           ;BR IF NOT
001126' 005077  177074                  CLR     @ASR            ;ENSURE CLOCK IS STOPPED
001132' 062767  000004  177066          ADD     #4,ASR          ;UPDATE STATUS ADDRESS
001140' 062767  000010  177064          ADD     #10,AVECT       ;UPDATE VECTOR ADDRESS
001146' 004767  177132                  JSR     PC,START2       ;FIX OTHER VALUES
001152' 000167  177210                  JMP     LOG1            ;CONTINUE AT LOG1
001156' 012767  000100  177056  REST1:  MOV     #BIT6,RATEP     ;FIRST PASS THRU LOOP, OFFSET=0, RATES WILL=1.
001164' 012777  001574' 177040  REST2:  MOV     #INSERV,@AVECT  ;SET UP CLOCK A'S INTER. VECTOR.
001172' 116777  176614  177034  REST3:  MOVB    BR1,@AVECT2     ;SET PRIORITY ON CLOCK A'S INTR.
001200' 106167  177036  LOOP:   ROLB    RATEP           ;GET NEXT RATE.
001204' 100005                          BPL     1$              ;IF NOT END THEN CONTINUE.
001206' 005067  177032                  CLR     OFF             ;CLEAR THE OFFSET
001212' 012767  000001  177022          MOV     #1,RATEP        ;LOOK AT FIRST RATE
001220' 062767  000002  177016  1$:     ADD     #2,OFF
001226' 022767  000016  177010          CMP     #16,OFF         ;TEST IF =16
001234' 001761                          BEQ     LOOP            ;BR
001236' 036767  177000  176552          BIT     RATEP,SR1       ;IS THIS RATE INHIBITED?
001244' 001355                          BNE     LOOP
001246' 005067  177000                  CLR     AIFLG           ;CLR FLAG INDICATING CLOCK A HAS INTERRUPTED.
001252' 016701  176766                  MOV     OFF,R1          ;PICK UP OFFSET
001256' 000171  001262'                 JMP     @LISTP(R1)      ;GO SET THE RATE + START THE CLOCK.
```



```
                                        ;THE FOLLOWING (LISTP) ARE POINTERS TO VARIOUS RATE
                                        ;ROUTINES. THEY ARE USED BY "LOOP". "LOOP" GENERATES
                                        ;AN OFFSET OF A RATE WE WISH TO EXERCISE. THE OFFSET
                                        ;IS STORED IN R1. WE INDEX "LISTP" BY R1 (JMP @LISTP(R1))
                                        ;TO GET THE ADDRESS OF THE RATE ROUTINE TO EXERCISE.
                                        ;

001262' 000001          LISTP:  .WORD   1               ;
001264' 001340'                 .WORD   RATE0           ;POINTER TO 1MHZ ROUTINE
001266' 001362'                 .WORD   RATE1           ;POINTER TO 100KHZ ROUTINE
001270' 001404'                 .WORD   RATE2           ;POINTER TO 10KHZ ROUTINE
001272' 001426'                 .WORD   RATE3           ;POINTER TO 1KHZ ROUTINE
001274' 001450'                 .WORD   RATE4           ;POINTER TO 100HZ ROUTINE
001276' 001472'                 .WORD   RATE5           ;POINTER TO RANDOM ROUTINE
001300' 000001                  .WORD   1

                                        ;
                                        ;THE FOLLOWING (RATEAL) ARE THE PRESET VALUES THAT THE
                                        ;VARIOUS RATE ROUTINES NEED. THEY ARE LOADED INTO
                                        ;CLOCK A'S PRESET BUFFER. "RATEAL" IS INDEXED BY
                                        ;AN OFFSET IN R1 BY THE RATE ROUTINES TO GET THE
                                        ;PRESET VALUE
                                        ;

001302' 000001          RATEAL: .WORD   1               ;OFFSET ZERO, NO RATE,
001304' 036260                  .WORD   -50000.         ;VALUE FOR 1MHZ PRESET.
001306' 036260                  .WORD   -50000.         ;PRESET VALUE FOR 100 KHZ
001310' 154360                  .WORD   -10000.         ;PRESET VALUE FOR 10 KHZ
001312' 176030                  .WORD   -1000.          ;PRESET VALUE FOR 1 KHZ
001314' 177634                  .WORD   -100.           ;PRESET VALUE FOR 100 HZ
001316' 000000                  .WORD   0               ;PRESET VALUE FOR RANDOM
001320' 000001                  .WORD   1

                                        ;
                                        ;THE FOLLOWING (RSAL) IS USED BY THE RANDOM
                                        ;RATE ROUTINE (RATE5). THEY ARE THE VALUES NEEDED
                                        ;TO BE PUT INTO THE CLOCK'S CSR FOR A PARTICULAR RATE.
                                        ;

001322' 000000          RSAL:   .WORD   0               ;OFFSET ZERO, NO RATE.
001324' 000113                  .WORD   113             ;1 MHZ, GO., MODE 1
001326' 000123                  .WORD   123             ;100 KHZ, GO., MODE 1
001330' 000131                  .WORD   131             ;10 KHZ, GO.
001332' 000141                  .WORD   141             ;1 KHZ, GO.
001334' 000151                  .WORD   151             ;100 HZ, GO.
001336' 000000                  .WORD   0
```

```
                                        ;*THIS ROUTINE PRESETS CLOCK A FOR
                                        ;*1 MHZ RATE    CLOCK A INTRS IN 1/20 SEC., 25 TIMES.
                                        ;*
                                        ;*

001340'                         RATE0:
001340' 005077  176662                  CLR     @ASR            ;CLEAR CLOCK A.
001344' 016177  001302' 176656          MOV     RATEAL(R1),@ABR ;PRESET COUNT IN CLOCK A.
001352' 016177  001322' 176646          MOV     RSAL(R1),@ASR   ;START CLOCK A.
                                                                ;NOW WAIT FOR INTERRUPT
001360' 104400                          EXIT$                   ;EXIT TO MONITOR. MODULE WAIT FOR INTERRRUPT.

                                        ;*
                                        ;*THIS ROUTINE PRESETS CLOCK A FOR
                                        ;*100 KHZ RATE CLOCK A INTRS IN .5 SEC., TWICE.
                                        ;*
                                        ;*

001362'                         RATE1:
001362' 005077  176640                  CLR     @ASR            ;CLEAR CLOCK A.
001366' 016177  001302' 176634          MOV     RATEAL(R1),@ABR ;PRESET COUNT IN CLOCK A.
001374' 016177  001322' 176624          MOV     RSAL(R1),@ASR   ;START CLOCK A.
                                                                ;NOW WAIT FOR INTERRUPT.
001402' 104400                          EXIT$                   ;EXIT TO MONITOR. MODULE WAIT FOR INTERRUPT.

                                        ;*
                                        ;*THIS ROUTINE PRESETS CLOCK A FOR
                                        ;*10 KHZ RATE   CLOCK A INTRS IN 1.0 SEC.
                                        ;*
                                        ;*

001404'                         RATE2:
001404' 005077  176616                  CLR     @ASR            ;CLEAR CLOCK A.
001410' 016177  001302' 176612          MOV     RATEAL(R1),@ABR ;PRESET COUNT IN CLOCK A.
001416' 016177  001322' 176602          MOV     RSAL(R1),@ASR   ;START CLOCK A.
                                                                ;NOW WAIT FOR INTERRUPT.
001424' 104400                          EXIT$                   ;EXIT TO MONITOR. MODULE WAST FOR INTERRUPT.

                                        ;*
                                        ;*THIS ROUTINE PRESETS CLOCK FOR
                                        ;*1 KHZ RATE    CLOCK A INTRS IN 1.0 SEC.
                                        ;*
                                        ;*

001426'                         RATE3:
001426' 005077  176574                  CLR     @ASR            ;CLEAR CLOCK A.
001432' 016177  001302' 176570          MOV     RATEAL(R1),@ABR ;PRESET COUNT IN CLOCK A.
001440' 016177  001322' 176560          MOV     RSAL(R1),@ASR   ;START CLOCK A.
                                                                ;NOW WAIT FOR INTERRUPT.
001446' 104400                          EXIT$                   ;EXIT TO MONITOR. MODULE WAIT FOR INTERRUPT.
```



```
                                        ;*THIS ROUTINE PRESETS CLOCK FOR
                                        ;*100 HZ RATE   CLOCK A INTRS IN 1.0 SEC.
                                        ;*
                                        ;*

001450'                         RATE4:
001450' 005077  176552                  CLR     @ASR            ;CLEAR CLOCK A.
001454' 016177  001302' 176546          MOV     RATEAL(R1),@ABR ;PRESET COUNT IN CLOCK A.
001462' 016177  001322' 176536          MOV     RSAL(R1),@ASR   ;START CLOCK A.
                                                                ;NOW WAIT FOR INTERRUPT.
001470' 104400                          EXIT$                   ;EXIT TO MONITOR. MODULE WAIT FOR INTERRUPT.

                                        ;*THIS ROUTINE PRESETS CLOCK FOR
                                        ;*RANDOM RATES

001472' 004767  000246          RATE5:  JSR     PC,RANDOM       ;GET 2 RANDOM NUMBERS.

001476' 042767  177771  176542          BIC     #177771,RANA    ;MAKE NUMBER < 10.
001504' 042767  177771  176536          BIC     #177771,RANB    ;MAKE 2ND NUMBER < 10.

                                        ;NUMBERS MUST BE 2, 4, OR 6

001512' 005767  176530          3$:     TST     RANA            ;IS NUMBER ZERO?
001516' 001003                          BNE     4$              ;NO - GO AHEAD.
001520' 062767  000002  176520          ADD     #2,RANA         ;MAKE IT NON-ZERO.
001526' 005767  176516          4$:     TST     RANB            ;IS NUMBER ZERO?
001532' 001003                          BNE     5$              ;NO GO AHEAD.
001534' 062767  000002  176506          ADD     #2,RANB         ;MAKE IT NON-ZERO.
001542'                         5$:
001542' 005077  176460                  CLR     @ASR            ;CLEAR CLOCK A
001546' 016701  176474                  MOV     RANA,R1
001552' 010167  176466                  MOV     R1,OFF          ;RECORD THE OFFSET
001556' 016177  001302' 176444          MOV     RATEAL(R1),@ABR ;PRESET CLOCK A.
001564' 016177  001322' 176434          MOV     RSAL(R1),@ASR   ;START CLOCK A.

001572' 104400                          EXIT$                   ;EXIT TO MONITOR. MODULE WAIT FOR INTERRUPT
```

```
                                        ;*INTERRUPT SERVICE ROUTINE FOR CLOCK.
001574' 005267  176452          INSERV: INC     AIFLG           ;INDICATE CLOCK A HAS INTERRUPTED
001600' 042777  000200  176420          BIC     #200,@ASR       ;CLEAR OVERFLOW FLAG.
001606' 026727  176432  000002    2$:   CMP     OFF,#2          ;ARE WE RUNNING 1MHZ RATE?
001614' 001005                          BNE     3$              ;IF NOT, 3$
001616' 026727  176430  000024          CMP     AIFLG,#20.      ;1 MHZ, 1 SEC. UP?
001624' 001412                          BEQ     4$              ;YES - STOP
001626' 000002                          RTI                     ;NO, ALLOW ANOTHER COUNT.
001630' 026727  176410  000004    3$:   CMP     OFF,#4          ;100 KHZ RATE?
001636' 001005                          BNE     4$              ;NO - 4$
001640' 026727  176406  000002          CMP     AIFLG,#2        ;YES, COUNTED TWICE?
001646' 001401                          BEQ     4$              ;YES - 4$,
001650' 000002                          RTI                     ;NO - COUNT ONE MORE TIME,
001652' 005077  176350          4$:   CLR     @ASR            ;STOP CLOCK A.
                                        ;---------------------------------------------------------------
001656' 000004  000000' 001664'         PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
                                        ;---------------------------------------------------------------
001664' 006367  176334          1$:   ASL     TEMP            ;CHECK NEXT UNIT
001670' 022767  000400  176326          CMP     #BIT8,TEMP      ;TEST IF VALID UNIT
001676' 001004                          BNE     11$             ;BR IF OK
001700'                           10$:
001700' 104413  000000'                 ENDIT$,BEGIN            ;SIGNAL END OF ITERATION,
                                                                ;MONITOR SHALL TEST END OF PASS
001704' 000167  177254                  JMP     REST2
001710' 036767  176310  176076  11$:   BIT     TEMP,DVID1      ;TEST IF UNIT SELECTED TO RUN ?
001716' 001770                          BEQ     10$             ;BR IF NOT
001720' 062767  000004  176300          ADD     #4,ASR          ;UPDATE ADDRESS
001726' 062767  000010  176276          ADD     #10,AVECT       ;UPDATE VECTOR
001734' 004767  176344                  JSR     PC,START2       ;FIX ADDRESSES
001740' 000167  177226                  JMP     REST3
                                        ;*THIS SUBROUTINE GENERATES TWO RANDOM NUMBERS
                                        ;*CALL = JSR PC, RANDOM
                                        ;*RETURNS WITH NUMBERS IN RANDA AND RANDB
001744' 066767  176276  176276  RANDOM: ADD     RANA,RANB
001752' 005567  176270                  ADC     RANA
001756' 066767  176266  176262          ADD     RANB,RANA
001764' 005567  176260                  ADC     RANB
001770' 000207                          RTS     PC
        000001                          .END
```



```
ABR      000230R          246#   267*   268*   347*   348    353    357*   358    363    370*   371    376    380*
                          381    386    468*   481*   494*   507*   518*   543*
ACSR     000102R          223#   295*   296    306*   322*   323    332*   348*   349    358*   371*   372    381*
ADDR     000006R          189#   265
ADDR22= 001000            241#
AIFLG    000252R          258#   412*   548*   552    557
ASB      000106R          227#
ASR      000226R          245#   265*   267    283    294*   295    301    305*   306    311    321*   322    327
                          331*   332    337    395*   396*   467*   469*   480*   482*   493*   495*   506*   508*
                          517*   519*   540*   544*   549*   560*   573*
ASTAT    000104R          225#   293*   294    296    310*   320*   321    323    336*   346*   347    349    362*
                          369*   370    372    385*
AVECT    000232R          248#   266*   269    272    397*   401*   574*
AVECT2   000234R          249#   272*   273*   402*
AWAS     000110R          228#
BEGIN    000000R          186#   299    300    303    308    309    313    325    326    329    334    335    339
                          351    352    355    360    361    365    374    375    378    383    384    388    562
                          568
BIT0  = 000001            241#   264
BIT1  = 000002            241#
BIT10 = 002000            241#
BIT11 = 004000            241#
BIT12 = 010000            241#
BIT13 = 020000            241#
BIT14 = 040000            241#
BIT15 = 100000            241#
BIT2  = 000004            241#
BIT3  = 000010            241#
BIT4  = 000020            241#
BIT5  = 000040            241#
BIT6  = 000100            241#   400
BIT7  = 000200            241#
BIT8  = 000400            241#   391    565
BIT9  = 001000            241#
BREAK$= 104407            241#   299    300    308    309    325    326    334    335    351    352    360    361
                          374    375    383    384
BR1      000012R          191#   402
BR2      000013R          192#
BTOD$ = 104421            241#
BVECT    000236R          251#   271*   274
BVECT2   000240R          252#   274*   275*
CDATA$= 104412            241#
CONFIG   000056R          211#
CSRA     000100R          221#   301*   311*   327*   337*   353*   363*   376*   386*
DATCK$= 104411            241#
DATER$= 104401            241#
DVID1    000014R          193#   393    571
ENDIT$= 104413            241#   568
END$  = 104410            241#
ERRTYP   000106R          226#
EXIT$ = 104400            241#   471    484    497    510    521    546
GETPA$= 104415            241#
GWBUF$= 104414            241#
HRDCNT   000044R          206#
HRDER$= 104405            241#   303    313    329    339    355    365    378    388
HRDPAS   000050R          208#
```

```
ICONT   000036R          203#
ICOUNT  000040R          204#
IDNUM   000122R          233#
INIT    000030R          200#
INSERV  001574R          401     548#
INTR    000120R          232#
LISTP   001262R          414     422#
LOG1    000366R          262     283#    399
LOG2    000372R          293#
LOG3    000514R          307     320#
LOG4    000636R          333     346#
LOG5    000760R          359     369#
LOOP    001200R          403#    409     411
MAP22$= 104416           241#
MODNAM  000000R          187#
MODSP   000224R          291     239#
MSGN$ = 104403           241#
MSGS$ = 104402           241#
MSG$  = 104401           241#
NULL  = 000000           241#    303     313     329     339     355     365     378     388
OFF     000244R          255#    405#    407#    408     413     542#    550     555
OPEN  = 000000           188     194     195     196     197     214     215     216     217     218     219     220     221
                         223     225     227     228     230     231     232     241#
OTOA$ = 104420           241#
PASCNT  000034R          202#
PIRQ$ = 000004           241#    562
POPSP = 005726           241#
POPSP2= 022626           241#
PRTY  = 000000           241#
PRTY0 = 000000           192     241#
PRTY1 = 000040           241#
PRTY2 = 000100           241#
PRTY3 = 000140           241#
PRTY4 = 000200           191     241#
PRTY5 = 000240           241#
PRTY6 = 000300           241#
PRTY7 = 000340           241#
PS    = 177776           241#
PSW   = 177776           241#
PUSH  = 005746           241#
PUSH2 = 024646           241#
RANA    000246R          256#    528#    533     535#    541     580     581#    582#
RANB    000250R          257#    529#    536     538#    580#    582     583#
RANDOM  001744R          526     580#
RAND$ = 104417           241#
RANNUM  000054R          210#
RATEAL  001302R          439#    468     481     494     507     518     543
RATEP   000242R          254#    400#    403#    406#    410
RATE0   001340R          423     466#
RATE1   001362R          424     479#
RATE2   001404R          425     492#
RATE3   001426R          426     505#
RATE4   001450R          427     516#
RATE5   001472R          428     526#
RESTRT  000254R          229     260#
REST0   001102R          392     390#
```



```
REST1   001156R          392     394     400#
REST2   001164R          401#    570
REST3   001172R          402#    576
RES1    000056R          212#
RES2    000060R          213#
RSAL    001322R          454#    469     482     495     508     519     544
RSTRT   000112R          229#
SBADR   000102R          222#
SOFCNT  000042R          205#
SOFER$= 104406           241#
SOFPAS  000046R          207#
SPOINT  000032R          201#
SPSIZ = 000040             1#    234
SR1     000016R          194#    410
SR2     000020R          195#
SR3     000022R          196#
SR4     000024R          197#
START   000254R          200     261#
START0  000262R          261     264#
START1  000270R          265#
START2  000304R          267#    398     575
STAT    000026R          199#
SVR0    000062R          214#
SVR1    000064R          215#
SVR2    000066R          216#
SVR3    000070R          217#
SVR4    000072R          218#
SVR5    000074R          219#
SVR6    000076R          220#
SYSCNT  000052R          209#
TEMP    000224R          241#    264#    390#    391     393     564#    565     571
TRPDFD= 000022           241#
VECTOR  000010R          190#    266
WASADR  000104R          224#
WDFR    000116R          231#
WDTO    000114R          230#
XFLAG   000005R          188#


. ABS.  000000     000
        001772     001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 PACK:XMNCA0,PACK:XMNCA0/SOL/CRF:SYM=DDXCOM,PACK:XMNCA0
 RUN-TIME:  3  5  .7 SECONDS
 RUN-TIME RATIO: 43/10=4.1
 CORE USED:  7K  (13 PAGES)
```