.REM $

IDENTIFICATION

PRODUCT CODE: AC-E866L-MC

PRODUCT NAME: CXDHAL0 DH11 16-LNE PROG

PRODUCT DATE: FEB 1979

MAINTAINER: DEC/X11 SUPPORT GROUP

1. ABSTRACT:

DHA IS AN IOMOD THAT EXERCISES UP TO FOUR (CONSECUTIVELY ADDRESSED) DH11 ASYNCHRONOUS INTERFACES. IT USES MAINTENANCE MODE TO TRANSMIT AND RECEIVE A BINARY COUNT PATTERN OUTPUTTED AND RECEIVED IN 64. CHARACTER BURSTS. THE MAJOR PORTION OF THE ERROR CHECKING IS DEFERRED TO LEVEL 0. ALL LINES SELECTED FOR TEST ARE ACTIVATED AND RUN CONCURRENTLY.

2. REQUIREMENTS:

HARDWARE: AT LEAST ONE DH11 INTERFACE

STORAGE:: DHA REQUIRES:
1. DECIMAL WORDS: 1066
2. OCTAL WORDS: 02052
3. OCTAL BYTES: 4124

3. PASS DEFINITION:

ONE PASS OF THE DHA MODULE CONSISTS OF TRANSMITTING AND RECEIVING 425984. CHARACTERS FOR EACH DH11 SELECTED

4. EXECUTION TIME:

VARIES WITH BAUD RATE BUT SHOULD TAKE AN AVERAGE OF ONE MINUTE TO COMPLETE ONE PASS WHEN RUNNING ALONE.

5. CONFIGURATION PARAMETERS:

DEFAULT PARAMETERS:

DVA: 1, VCT: 1, BR1: 5, BR2: 5, DVC: 1, SR1: 0

REQUIRED PARAMETERS:

AT CONFIGURATION TIME THE USER MUST SPECIFY:

DVA: ADDRESS OF FIRST DH11 CSR REG.
VCT: VECTOR ADDRESS OF FIRST DH11
DVC: NO OF DH11'S IF GREATER THAN 1
SR1: BIT15 MUST BE SET TO A "1" FOR
8. WORDS BETWEEN ADJACENT VECTORS
(2040 CONFIGURATION)

6. DEVICE OPTION SETUP:

NONE REQUIRED

7. MODULE OPERATION:

START: DETERMINE IF ANY ERRORS ARE SELECTED. DO NOT RUN THE MODULE IF NO DEVICES ARE SELECTED. IF THERE ARE SELECTED DEVICES, INITIALIZE THE BINARY COUNT PATTERN AT 0. CONTINUE PROCESSING.

RESTRT: INITIALIZE THE ITERATION COUNTER TO 6656. DETERMINE IF ANY DH11'S ARE SELECTED. LOAD THE INTERRUPT VECTORS TO POINT TO THE JSR LINKING TABLE.

SETUP2: INITIALIZE THE QUEUE POINTERS. CLEAR ALL THE BUFFERS AND QUEUES. CLEAR THE BUFFER ACCESS FLAG (LCKOUT) IN CASE IT WAS STILL SET BY A CONTROL C INTERRUPT OF THE PROGRAM.

ACTVATE: THIS BLOCK OF CODE GETS THE PHYSICAL ADDRESS OF THE TRANSMITTER TEXT BUFFER. IT INITIALIZES EACH DH11 SELECTED. IT SAVES THE EXTENDED ADDRESS INFORMATION IN THE APPROPRIATE SYSTEM CONTROL REGISTER BITS. EACH OF THE SIXTEEN LINES IS LOADED WITH THE PHYSICAL ADDRESS AND LENGTH OF THE TRANSMITTER BUFFER. ALSO, EIGHT BITS PER CHARACTER IS SELECTED. IF A BAUDRATE IS SELECTED, IT IS CALCULATED AND ASSIGNED. OTHERWISE THE DEFAULT RATE OF 9600 BAUD IS ASSUMED.

INITIAL: THE DATA PATTERN IS LOADED INTO THE TRANSMITTER BUFFER. IT IS A BINARY COUNT PATTERN WHICH, ON SUCCESSIVE ITERATIONS, BEGINS 0,4,10,14,20,...,177774. THE SILO ALARM LEVEL IS PLACED IN THE SILO STATUS REGISTER TRANSMITTER AND ERROR INTERRUPTS ARE ENABLED. ALL THE TRANSMITTERS FOR EACH SELECTED DH11 ARE ENABLED.

TMRSET: TMRCNT IS USED AS A MULTIPLYING FACTOR TO DETERMINE THE WAITING LENGTH FOR THE WATCHDOG TIMER. IT IS PRESENTLY SET AT 5 TO ALLOW SEVENTY-FIVE SECONDS TO ELAPSE BEFORE TAKING FURTHER ACTION.

TIMER: THIS IS THE WATCHDOG TIMER LOOP. IT IS CONTROLLED BY R4 AND TMRCNT. IF ALL DH11'S SELECTED GENERATED BOTH TRANSMIT AND RECEIVE INTERRUPTS, THE APPROPRIATE BIT IN DONFLG FOR THAT DH11 WILL BE CLEARED. IF THIS DOES NOT OCCUR IN THE GIVEN TIME, THE DEVICE NUMBER OF THE OFFENDING DH11 WILL BE CALCULATED, AND THIS WILL BE REPORTED IN A MODULE MESSAGE. THE

OFFENDING DEVICE IS DROPPED FROM THE EXERCISE. IF NO MORE DH11'S ARE SELECTED, THE MODULE ITSELF IS DROPPED FROM THE RUN. IF MORE REMAIN TO BE EXERCISED, HOWEVER, CONTROL IS TRANSFERRED TO "FINISH."

FINISH: CONTROL COMES HERE IF ALL SELECTED DH11'S WERE SUCCESSFULLY EXERCISED OR IF MORE DH11'S REMAIN AFTER ONE WAS HUNG. THE ITERATION COUNT IS DECREASED. IF THE COUNT DOES NOT REACH ZERO, CONTROL IS PASSED TO SETUP2 AND THE MODULE IS RUN AGAIN. WHEN THE COUNT REACHES ZERO, AN END OF PASS IS SIGNALLED.

XMTINT: THIS SEGMENT SAVES A POINTER TO THE CSR ADDRESS OF THE INTERRUPTING DH11 IN A FIRST-IN, FIRST-OUT, WRAPAROUND BUFFER. THE TRANSMITTER QUEUE. THE ENTRY POINTER IS THEN UPDATED TO POINT TO THE NEXT ENTRY IN THE QUEUE.

XMTSRV: THIS BLOCK FETCHES A POINTER TO A CSR ADDRESS FROM THE TRANSMITTER QUEUE, AND THE QUEUE IS UPDATED TO THE NEXT ENTRY. THE CSR IS TESTED TO DETERMINE WHAT KIND OF INTERRUPT OCCURRED. FALSE INTERRUPT AND NON-EXISTENT MEMORY INTERRUPT ARE REPORTED. IF EVERYTHING IS CORRECT, THE RECEIVER FOR THIS DH11 IS ENABLED.

RCVINT: THIS SEGMENT SAVES A POINTER TO THE CSR ADDRESS OF THE INTERRUPTING DH11 IN THE RECEIVER QUEUE. IT UPDATES THE QUEUE ENTRY AND RESTORES THE VALUE OF R5 WHICH WAS SAVED BY THE JSR INSTRUCTION FROM THE LINKAGE TABLE.

RCVSRV: THE FIRST TASK IS TO PREVENT VOLATILE REGISTER INFORMATION FROM BEING DESTROYED. THIS IS DONE BY TESTING A SEMAPHORE, "LCKOUT." IF IT IS SET, CONTROL IS RETURNED TO THE MONITOR TO WAIT FOR A WHILE. IF IT IS CLEAR, ACCESS IS PERMITTED. THE FLAG IS SET TO DENY OTHER ACCESSES TO THIS DEFERRED ROUTINE. A CSR ADDRESS IS OBTAINED FROM THE QUEUE, AND THE QUEUE ENTRY IS UPDATED. THE RECEIVER INTERRUPT AND INTERRUPT ENABLE ARE CLEARED. THE REGISTERS ARE SET UP TO RETRIEVE AS QUICKLY AS POSSIBLE THE DATA FROM THE DH11 SILO. EACH FETCH IS CHECKED TO SEE IF THE INFORMATION IS VALID. IF IT IS NOT, THE REGISTERS ARE SAVED AND A BREAK LOOP IS USED TO ALLOW MORE TIME FOR VALID INFORMATION TO BECOME AVAILABLE. IF AFTER THE ALLOTED TIME ALL THE CHARACTERS ARE STILL NOT RECEIVED, AN ERROR MESSAGE IS REPORTED. IN THE MESSAGE, THE NUMBER FOLLOWING STATC IS THE OCTAL NUMBER OF CHARACTERS MISSING.

CKDATA: THIS SEGMENT INITIALIZES THE LINE CHECK BUFFER (LNCKBF) TO THE FIRST DATUM THAT WAS TRANSMITTED. THE DEVICE NUMBER IS SAVED FOR LATER USAGE. THE RECEIVED INFORMATION IS CHECKED FOR VALIDITY AND TRANSMISSION ERRORS. ERRORS ARE HANDLED BY THE

'STATERR' (STATUS ERROR) AND 'DERROR' (DATA ERROR) ROUTINES.

RCVDONE: THIS BLOCK CLEARS THE ACCESS SEMAPHORE TO ALLOW OTHER DEVICES TO USE THE LINE CHECK BUFFER. IT THEN BUILDS A ONE BIT MASK USING R0 AND THE CARRY BIT TO DELETE THE APPROPRIATE BIT IN THE WATCHDOG TIMER FLAG (DONFLG). WHEN THIS IS DONE, PROCESSING CONTROL IS RETURNED TO THE MONITOR.

SUBROUTINES

VCTLOAD: THIS ROUTINE IS CALLED IN 'SETUP1'. IT IS USED TO LOAD THE ADDRESS OF THE LINKING INSTRUCTION FOR INTERRUPT SERVICING INTO THE CORRESPONDING VECTOR SPACE. IT ALSO LOADS THE PRIORITY LEVEL AND THE DEVICE ADDRESS. THE LATTER IS LOADED INTO THE APPROPRIATE JSR TABLE ENTRY.

SAVREG: THIS ROUTINE SAVES THE FIVE VOLATILE INFORMATION REGISTERS IN A FIRST-IN, FIRST-OUT WRAPAROUND BUFFER, THE ERROR QUEUE.

GOTREG: THIS ROUTINE RETRIEVES THOSE SAME REGISTERS.

BAUDRTE: THIS ROUTINE CALCULATES THE BAUD RATE, ASSIGNS IT, AND SELECTS 8 BITS/CHARACTER COMMUNICATION MODE. IF SR1=0, THE DEFAULT RATE OF 9600. BAUD IS ASSIGNED. THE BAUD RATE SELECTED IS DETERMINED BY THE LEAST SIGNIFICANT (RIGHTMOST) SET BIT IN SR1.

STATERR: THIS ROUTINE DETERMINES WHETHER AN ERROR INDICATED IN THE RECEIVED CHARACTER INFORMATION WAS AN OVERRUN ERROR, A FRAMING ERROR, OR A PARITY ERROR. THE DEVICE NUMBER OF THE ERRING DEVICE IS REPORTED AS STATC. CSRA WILL BE CLEAR.

DERROR: THIS ROUTINE REPORTS A DATA ERROR.

8. OPERATOR OPTIONS

MODULE LOCATION DVID1 MAY BE MODIFIED (MOD. CMMD) TO EXERCISE ANY COMBINATION OF DH11'S.

MODULE LOCATION SR1 MAY BE MODIFIED TO SELECT A DIFFERENT BAUD RATE. THE FOLLOWING TABLE SHOULD BE USED:

FOR THE BAUD RATE	SR1=
9600	0
EXT. B	1
EXT. A	2

9600		4
4800		10
2400		20
1800		40
1200		100
600		200
300		400
200		1000
150		2000
134.5		4000
110		10000
75		20000
50		40000
2040	CONFIGURATION	100000 (8. WORDS BETWEEN VECTORS)

BIT15 MAY BE SET IN CONJUNCTION WITH ONE OF BITS
00 THRU 14 TO INDICATE SPECIFIC BAUD RATE OTHER THAN
DEFAULT OF 9600 BAUD. (SR<14:00> = 0)

THE DEFAULT RATE IS 9600 BAUD(SR1=0).

9. NON-STANDARD PRINTOUTS:

WHEN A STATUS ERROR IS DETECTED, DHA USES THE ERROR
CALL TO REPORT IT. THE LINE NUMBER IS
REPORTED IN STATC.

ALL OTHER PRINTOUT IS STANDARD.

$

```
                                        .LIST   SEQ,LOC,BIN
                                        IOMOD   <DHAL >1,1,5,5,,17000,25
                                        MODULE  140000,DHAL ,1,1,5,5,,17000,25
                                        .TITLE  DHAL DEC/X11 SYSTEM EXERCISER MODULE
                                ;       DDXCOM  VERSION  6       23-MAY-78
                                        .LIST   BIN
                                ;*****************************************************************
000000'                         BEGIN:
000000' 044104  046101      040 MODNAM: .ASCII  /DHAL / ;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'     240                 BR1:    .BYTE   PRTY5+0         ;1ST BR LEVEL.
000013'     240                 BR2:    .BYTE   PRTY5+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' 017000                  ICONT:  17000                   ;# OF ITERATIONS PER PASS=17000
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' 000244'                 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' 000025                  IDNUM:  25                      ;MODULE IDENTIFICATION NUMBER=25
                                        .REPT   SPSIZ           ;MODULE STACK STARTS HERE.
000040
```



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

```
                                ;START ROUTINE. DETERMINE IF ANY DH'S ARE SELECTED
                                ;IF SO, BEGIN MODULE PROCESSING... IF NOT, DROP THE MODULE FROM THE RUN

000224' 105067  003616          START:  CLRB    DATA            ;INITIALIZE THE DATA PATTERN WORD
000230' 016767  177560  003566          MOV     DVID1,SELECT    ;COPY THE DEVICE SELECTION PARAMETER. ARE
                                                                ;ANY DEVICES SELECTED?
000236' 001002                          BNE     RESTRT          ;IF SO, BEGIN PROCESSING. IF NOT, DROP THE MODULE
000240'                         DROP:
000240' 104410  000000'                 END$,REGIN              ;

                                ;INITIALIZE THE NUMBER OF PASSES TO BE MADE. SET UP THE DH11 INTERRUPT
                                ;VECTORS TO INTERRUPT IN THE SUBROUTINE LINKING TABLE. CALL SUBROUTINE
                                ;VCTLOAD FOR THIS PURPOSE.

000244' 005000                  RESTRT: CLR     R0              ;INIT POINTER
000246' 016701  177544                  MOV     SR1,R1          ;GET BAUD RATE
000252' 042701  100000                  BIC     #100000,R1      ;THROW 2040 CNF BIT AWAY
000256' 001405                          BEQ     2$              ;LEAVE IF SR1 IS 0
000260' 062700  000002          1$:     ADD     #2,R0           ;ELSE BUMP POINTER
000264' 006001                          ROR     R1              ;LOOK FOR THE SR1 BIT
000266' 103401                          BCS     2$              ;LEAVE  IF WE FOUND IT
000270' 000773                          BR      1$              ;ELSE DO IT AGAIN
000272' 016067  002602' 177536  2$:     MOV     CNTTBL(R0),ICONT        ;SET ITERATION COUNT
000300' 042767  177760  003516          BIC     #177760,SELECT  ;REMOVE UNWANTED BITS FROM THE DEVICE SELECTION
                                                                ;PARAMETER
000306' 016701  003512          SETUP1: MOV     SELECT,R1       ;COPY THE DEVICE SELECTION PARAMETER INTO R1
000312' 001752                          BEQ     DROP            ;IF NO DEVICES SELECTED (ALL FLAGS CLEAR) DROP THE
                                                                ;MODULE
000314' 016700  177470                  MOV     VECTOR,R0       ;LOAD THE VECTOR ADDRESS IN R0
000320' 016702  177462                  MOV     ADDR,R2         ;LOAD R2 WITH ADDRESS OF FIRST DH11
000324' 012703  002502'                 MOV     #LNKTAB,R3      ;POINT R3 TO THE BEGINNING OF JSR LINKING TABLE
000330' 006201                  1$:     ASR     R1              ;ISOLATE A SELECTION FLAG IN THE CARRY BIT
000332' 103415                          BCS     2$              ;IF THE FLAG IS SET, GO SET UP THE VECTORS
000334' 001430                          BEQ     SETUP2          ;IF NO FLAGS ARE LEFT, GO SET UP THE BUFFERS
000336' 062700  000010                  ADD     #10,R0          ;IF MORE FLAGS ARE SET, ADJUST POINTERS. THE
                                                                ;VECTOR POINTER...
000342' 005767  177450                  TST     SR1             ;EIGHT WORDS BETWEEN VECTORS ?
000346' 100002                          BPL     4$              ;BR IF NOT
000350' 062700  000010                  ADD     #10,R0          ;FIX VECTOR POINTER
000354' 062703  000020          4$:     ADD     #20,R3          ;THE ADDRESS POINTER...
000360' 062702  000020          3$:     ADD     #20,R2          ;AND THE LINKAGE TABLE POINTER
000364' 000761                          BR      1$              ;GO SET UP THE NEXT DH11 ADDRESSSES
000366' 004567  001426          2$:     JSR     R5,VCTLOAD      ;CALL THE VECTOR SETUP ROUTINE FOR RECEIVER
000372' 000013'                         BR2                     ;VECTOR, PASSING THE RECEIVER BR LEVEL AS THE
                                                                ;ARGUMENT
000374' 004567  001420                  JSR     R5,VCTLOAD      ;SET UP THE TRANSMITTER VECTOR PASSING THE
000400' 000012'                         BR1                     ;TRANSMITTER BR LEVEL AS THE ARGUMENT
000402' 005767  177410                  TST     SR1             ;EIGHT WORDS BETWEEN VECTORS ?
000406' 100002                          BPL     5$              ;BR IF NOT
000410' 062700  000010                  ADD     #10,R0          ;FIX VCTR POINTER
000414' 000761                  5$:     BR      3$              ;GO SETUP THE NEXT DH11

                                ;THIS BLOCK RESETS ALL THE QUEUE POINTERS, CLEARS THE TRANSMITTER TEXT BUFFER, THE
                                ;RECEIVER BUFFERS, AND THE QUEUES. THIS IS THE BEGINNING OF THE ITERATIVE PART OF THE
                                ;PROGRAM.
```



```
000416' 012767  002642' 003364  SETUP2: MOV     #XMTQUE,XMTQPI  ;POINT THE TRANSMIT QUEUE ENTRY (IN) POINTER
                                                                ;TO THE BEGINNING OF THE QUEUE
000424' 012767  002642' 003360          MOV     #XMTQUE,XMTQPO  ;POINT THE TRANSMIT QUEUE RETRIEVAL (OUT)
                                                                ;POINTER TO THE BEGINNING OF THE QUEUE
000432' 012767  002662' 003354          MOV     #RCVQUE,RCVQPI  ;SET UP THE RECEIVER QUEUE POINTERS
000440' 012767  002662' 003350          MOV     #RCVQUE,RCVQPO  ;
000446' 012767  002702' 003344          MOV     #ERRQUE,ERRQPI  ;SETUP THE ERROR QUEUE POINTERS
000454' 012767  002702' 003340          MOV     #ERRQUE,ERRQPO  ;
000462' 012703  002642'                 MOV     #XMTQUE,R3      ;POINT R3 TO THE BEGINNING OF THE QUEUE AREA
000466' 012704  000446                  MOV     #294.,R4        ;USING R4 AS A COUNTER CLEAR -
000472' 005023                  1$:     CLR     (R3)+           ;TRANSMITTER, RECEIVERS, AND ERROR QUEUES....
000474' 005304                          DEC     R4              ;TRANSMITTER TEXT BUFFER...
000476' 001375                          BNE     1$              ;RECEIVER SILO BUFFERS
000500' 105067  003345                  CLRB    LCKOUT          ;CLEAR THE BUFFER ACCESS FLAG

                                ;THIS BLOCK SETS THE WATCHDOG TIMER FLAG, INITIALIZES THE DH11 CONDITIONS, SETS THE CHAR
                                ;AND BAUD RATE, AND SETS THE EXTENDED ADDRESSING BITS FOR THE ACTIVE DH11S

000504' 016700  003314  ACTVATE:MOV     SELECT,R0       ;COPY THE DEVICE SELECTION PARAMETER
000510' 110067  003333          MOVB    R0,DONFLG       ;SET THE WATCHDOG TIMER DEVICE COMPLETION FLAG
000514' 016701  177266          MOV     ADDR,R1         ;LOAD THE ADDRESS OF THE FIRST DH11
000520' 012767  002764' 003304          MOV     #XMTBUF,VA      ;GET THE PHYSICAL ADDRESS OF THE TRANSMITTER
                                                                ;BUFFER
000526' 104415  000000' 004032'         GETPAS,BEGIN, VA        ;GET PHYSICAL ADDRESS FROM 16-BIT VA
000534' 006200                  1$:     ASR     R0              ;ISOLATE A SELECTION FLAG IN THE CARRY BIT
000536' 103404                          BCS     3$              ;IF SELECTED, GO SET UP THE DEVICE REGISTERS
000540' 001431                          BEQ     INITIAL         ;IF NO MORE SELECTED, GO SET UP THE DATA
000542' 062701  000020          2$:     ADD     #20,R1          ;POINT R1 TO THE NEXT DH11
000546' 000772                          BR      1$              ;PROCESS NEXT DH11
000550' 012711  004000          3$:     MOV     #BIT11,(R1)     ;MASTER CLEAR THIS DH11 AND SELECT LINE 0

000554' 052711  001000                  BIS     #BIT9,(R1)      ;ENABLE MAINTENANCE MODE OPERATION
000560' 042711  000060                  BIC     #60,(R1)        ;ERASE INVALID INFORMATION FROM THE EA BITS IN THE CSR
000564' 056711  003246                  BIS     EA,(R1)         ;SET THE EXTENDED ADDRESS BITS OF THE TEXT
000570' 012702  000020                  MOV     #20,R2          ;USE R2 TO COUNT 16 LINES BEING SET UP
000574' 016761  003234  000006  4$:     MOV     PA,6(R1)        ;LOAD THE PHYSICAL ADDRESS OF THE TRANSMITTER
                                                                ;BUFFER INTO THE CURRENT ADDRESS REGISTER
000602' 012761  177774  000010          MOV     #-4,10(R1)      ;LOAD THE BYTE COUNT REGISTER WITH THE NUMBER
                                                                ;OF CHARACTERS TO BE TRANSMITTED
000610' 004767' 001316                  JSR     PC,BAUDRTE      ;GO CALCULATE THE BAUD RATE TO BE USED
000614' 005302                          DEC     R2              ;REDUCE THE COUNT. ARE ALL 16 LINES SET?
000616' 001751                          BEQ     2$              ;IF YES, GO PROCESS NEXT DH11
000620' 005211                          INC     (R1)            ;IF NO, INCREMENT THE LINE SELECTION PARAMETER
000622' 000764                          BR      4$              ;GO SET UP THE NEXT LINE

                                ;THIS BLOCK SETS UP THE TRANSMITTER TEXT WITH THE TEST DATA . IT INSURES THAT THE
                                ;EXTENDED ADDRESS BITS ARE SET, THEN STARTS EACH RECEIVER AND EACH TRANSMITTER SELECTED

000624' 012701  002764'         INITIAL:MOV     #XMTBUF,R1      ;POINT R1 TO THE START OF THE BUFFER AREA
000630' 116767  003212  003204          MOVB    DATA,BGNDATA    ;COPY THE FIRST DATUM BEING TRANSMITTED IN THIS
000636' 116767  003204  003177          MOVB    DATA,BGNDATA+1  ;GROUP .BGNDATA WILL BE USED TO SET
                                                                ;UP LNCKBF(THE LINE CHECK BUFFER ) IN THE CKDATA ROUTINE
000644' 012702  000004                  MOV     #4,R2           ;USE R2 TO COUNT THE LOOP ITERATIONS
000650' 116721  003172          1$:     MOVB    DATA,(R1)+      ;PUT A CHARACTER IN THE BUFFER, BUILDING THEM
000654' 105267  003166                  INCB    DATA            ;IN DATA, WHICH WILL CREATE A BINARY INCREMENT PATTERN.
000660' 005302                          DEC     R2              ;REDUCE COUNT. HAVE ALL CHARACTERS BEEN MADE?
```

```
000662' 001372                  BNE     1$              ;IF NO, GO LOAD THE NEXT CHARACTER
000664' 016700  003134          MOV     SELECT,R0       ;COPY THE DEVICE SELECTION PARAMETER
000670' 016701  177112          MOV     ADDR,R1         ;LOAD THE ADDRESS OF THE FIRST DH11
000674' 006200          2$:     ASR     R0              ;ISOLATE A SELECTION FLAG IN THE CARRY BIT
000676' 103404                  BCS     4$              ;IF SELECTED, GO START THE DEVICE
000700' 001414                  BEQ     TMRSET          ;IF NO MORE SELECTED, GO START WATCHDOG TIMER
000702' 062701  000020  3$:     ADD     #20,R1          ;POINT R1 TO THE NEXT DH11
000706' 000772                  BR      2$              ;GO PROCESS THE NEXT DH11
000710' 012761  000040  000015  4$:     MOV     #32.,16(R1)     ;LOAD THE SILO ALARM LEVEL INTO THE SILO STATUS
                                                        ;REGISTER
000716' 052711  020000          BIS     #BIT13,(R1)     ;ENABLE TRANSMITTER AND NON-EXISTENT MEMORY
                                                        ;INTERRUPTS
000722' 012761  177777  000012  MOV     #-1,12(R1)      ;ENABLE THE TRANSMITTERS FOR ALL 16. LINES
000730' 000764                  BR      3$              ;PROCESS THE NEXT DH11

                        ;THIS IS THE WATCHDOG TIMER. WHEN A DEVICE HAS SUCCESSFULLY TRANSMITTED AND
                        ;RECEIVED ALL DATA. IT CLEARS ITS CORRESPONDING BIT IN DONFLG. IF THIS
                        ;DOES NOT OCCUR, AN ERROR MESSAGE IS REPORTED. STATC IN THE
                        ;ERROR MESSAGE TELLS WHICH DEVICE WAS SLOW OR HUNG. CSRA WILL BE ZERO.
                        ;A MODULE MESSAGE IS REPORTED. IF MORE DEVICES ARE TO BE PROCESSED, THIS
                        ;ITERATION IS CONSIDERED COMPLETE. IF NO DEVICES REMAIN TO BE PROCESSED,
                        ;THE MODULE IS DROPPED.

000732' 012767  000005  003070  TMRSET: MOV     #5,TMRCNT       ;SET THE TIMER COUNT FOR THE BREAK LOOP
000740' 005004          TIMER:  CLR     R4              ;USING R4, RETURN TO MONITOR 65535 TIMES
000742'                 1$:
000742' 104407  000000'         BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
000746' 104407  000000'         BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000752' 105767  003071          TSTB    DONFLG          ;IF DONFLG IS CLEAR, EACH SELECTED DEVICE WAS
                                                        ;SUCCESSFUL
000756' 001454                  BEQ     FINISH          ;IF SO, PERFORM ENDPASS PROCESSING
000760' 005304                  DEC     R4              ;IF NOT, REDUCE COUNT AND BREAK AGAIN
000762' 001367                  BNE     1$              ;BREAK IF COUNT NOT EXCEEDED
000764' 005367  003040          DEC     TMRCNT          ;REDUCE THE OVERALL TIME.IS IT EXCEEDED?
000770' 001363                  BNE     TIMER           ;IF NO,START ANOTHER BREAK LOOP

000772' 116703  003051          MOVB    DONFLG,R3       ;IF TIMEOUT OCCURRED, SAVE THE REMAINING FLAG
                                                        ;IN R3.
000776' 040367  003022          BIC     R3,SELECT       ;CLEAR THE SELECTION FLAG FOR THIS DEVICE
001002' 005067  177072          CLR     CSRA            ;/SETUP TO REPORT CSR
001006' 006003          2$:     ROR     R3              ;DETERMINE WHICH DEVICE WAS READ FOR REPORTING
                                                        ;PURPOSES
001010' 103405                  BCS     3$              ;IF THIS IS THE DEVICE, R4 CONTAINS THE CORRECT
                                                        ;LINE NUMBER... GO REPORT IT
001012' 005204                  INC     R4              ;IF NOT, INCREMENT R4, WHICH WAS INITIALLY 0
                                                        ;FROM THE PREVIOUS LOOP
001014' 062767  000020  177056  ADD     #20,CSRA        ;/ADD DEVICE OFFSET FOR CORRECT DH11
001022' 000771                  BR      2$              ;GO SEE IF THIS IS THE DEVICE
001024' 010467  003022  3$:     MOV     R4,NUMBA1       ;THE DEVICE NUMBER WILL BE REPORTED AS STATC
                                ;*************************************************************
                                                        ;CONVERT NUMBA1 TO ASCII AND
                                                        ;STORE AT M2
001030' 104420  000000' 004052'         OTOA$,BEGIN,NUMBA1,M2
001036' 004067'                 ;*************************************************************
001040' 004767  000772          JSR     PC,SAVREG       ;SAVE THE REGISTERS BEFORE LEAVING THE MODULE
```



```
001044' 066767  176736  177026  ADD     ADDR,CSRA       ;/ADD BASE ADDRESS TO OFFSET FOR ERROR CALL
001052' 005067  177030          CLR     ERRTYP          ;UNKNOWN
                                ;*************************************************************
001056' 104405  000000' 000000  ERDER$,BEGIN,NULL       ;
                                ;*************************************************************
001064' 104403  000000' 004054' MSGN$,BEGIN,HUNG         ;ASCII MESSAGE CALL WITH COMMON HEADER
001072' 004767  000776          JSR     PC,GETREG       ;RESTORE THE PREVIOUS REGISTER VALUES
001076' 005767  002722          TST     SELECT          ;ARE THERE ANY DEVICES STILL ACTIVE?
001102' 001002                  BNE     FINISH          ;IF YES, REDUCE COUNT AND CONTINUE
                                                        ;IF NO, DROP THE MODULE
001104' 104410  000000'         END$,BEGIN              ;

                        ;THIS BLOCK REDUCES THE ITERATION COUNT AND GOES TO SETUP2 IF THE COUNT IS NOT EXCEEDED

001110'                 FINISH:
001110' 104413  000000'         ENDIT$,BEGIN            ;SIGNAL END OF ITERATION.
                                                        ;MONITOR SHALL TEST END OF PASS
001114' 000167  177276          JMP     SETUP2

                        ;TRANSMITTER INTERRUPT SERVICE ROUTINE
                        ;THIS ROUTINE STORES THE ADDRESS OF THE CSR POINTER (OFFSET IN THE JSR LINKAGE
                        ;TABLE) IN THE TRANSMITTER QUEUE. IT DETERMINES THIS A NON-EXISTENT MEMORY ERROR
                        ;OR A DATA INTERRUPT. IF IT IS THE FORMER, IT IS SETUP FOR REPORTING. IF IT
                        ;IS THE LATTER, FURTHER SERVICE IS DEFERRED TO PRIORITY LEVEL 0 BY A PROGRAM
                        ;INTERRUPT REQUEST. THE ROUTINE THEN ENABLES INTERRUPTS FOR THE CORRESPONDING RECEIVER
                        ;AND RETURNS CONTROL TO THE MONITOR.

001120' 010577  002664  XMTINT: MOV     R5,@XMTQPI      ;LOAD THE OFFSET TO THE CSR INTO TRANSMITTER QUEUE
001124' 062767  000002  002656  ADD     #2,XMTQPI       ;UPDATE THE QUEUE ENTRY POINTER
001132' 022767  002660' 002650  CMP     #XMTQUE+16,XMTQPI ;HAS THE QUEUE BOUNDARY BEEN EXCEEDED?
001140' 103003                  BHIS    1$              ;IF NO, CONTINUE PROCESSING
001142' 012767  002642' 002640  MOV     #XMTQUE,XMTQPI  ;IF YES, RESET THE POINTER TO QUEUE BEGINNING
001150' 012605          1$:     MOV     (SP)+,R5        ;RESTORE THE PREVIOUS R5 VALUE
001152' 010046                  MOV     R0,-(SP)
001154' 010146                  MOV     R1,-(SP)
001156' 010246                  MOV     R2,-(SP)
001160' 010346                  MOV     R3,-(SP)
001162' 010446                  MOV     R4,-(SP)
001164' 017701  002622  XMTSRV: MOV     @XMTQPO,R1      ;FETCH THE OFFSET FROM THE QUEUE
001170' 062767  000002  002614  ADD     #2,XMTQPO       ;UPDATE THE QUEUE RETRIEVAL POINTER
001176' 022767  002660' 002605  CMP     #XMTQUE+16,XMTQPO ;HAS THE QUEUE BOUNDARY BEEN EXCEEDED?
001204' 103003                  BHIS    1$              ;IF NO, CONTINUE PROCESSING
001206' 012767  002642' 002576  MOV     #XMTQUE,XMTQPO  ;IF YES, RESET POINTER TO BEGINNING OF THE QUEUE
001214' 011100          1$:     MOV     (R1),R0         ;LOAD CSR ADDRESS INTO R0
001216' 005710                  TST     (R0)            ;IS THIS A VALID INTERRUPT?
001220' 100467                  BMI     3$              ;IF YES, GO ENABLE RECEIVER INTERRUPT
001222' 032710  002000          BIT     #BIT10,(R0)     ;IF NO, DETERMINE THE TYPE OF ERROR
001226' 001032                  BNE     2$              ;IF NON-EXISTENT MEMORY (BIT 10 SET) GO PROCESS
                                                        ;THAT TYPE OF ERROR. OTHERWISE IT IS A FALSE INTERRUPT
001230' 010067  176644          MOV     R0,CSRA         ;LOAD THE DEVICE ADDRESS
001234' 011067  176642          MOV     (R0),ACSR       ;SAVE THE CSR CONTENTS
001240' 004767  000572          JSR     PC,SAVREG       ;SAVE THE REGISTERS BEFORE ERROR MESSAGE
001244' 012604                  MOV     (SP)+,R4
001246' 012603                  MOV     (SP)+,R3
001250' 012602                  MOV     (SP)+,R2
001252' 012601                  MOV     (SP)+,R1
```

```
001254' 012600                          MOV     (SP)+,R0
                                        ;-------------------------------------------------------------
001256' 000004  000000' 001264'         PIRQ$,BEGIN,10$         ; QUEUE UP TO CONTINUE AT 10$ AND RTI
                                        ;-------------------------------------------------------------
001264' 012767  000011  176614  10$:    MOV     #11,ERRTYP      ;ILLEGAL INTERRUPT
                                        ;*************************************************************
001272' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;FALSE INTERRUPT REQUEST FROM TRANSMITTER
                                        ;*************************************************************
001300' 004767  000570                  JSR     PC,GETREG       ;RESTORE THE REGISTERS
001304' 042710  120000                  BIC     #120000,(R0)    ;DELETE THE FALSE INTERRUPT AND THE INTERRUPT
                                                                ;ENABLE
001310' 104400  000000'                 EXIT$,BEGIN             ;EXIT TO MONITOR. MODULE WAIT FOR INTERRUPT.
001314' 010067  176560          2$:     MOV     R0,CSRA         ;LOAD THE DEVICE ADDRESS
001320' 011067  176556                  MOV     (R0),ACSR       ;SAVE THE CSR CONTENTS
001324' 004767  000506                  JSR     PC,SAVREG       ;SAVE THE REGISTER VALUE BEFORE ERROR MESS.
001330' 012604                          MOV     (SP)+,R4
001332' 012603                          MOV     (SP)+,R3
001334' 012602                          MOV     (SP)+,R2
001336' 012601                          MOV     (SP)+,R1
001340' 012600                          MOV     (SP)+,R0
                                        ;-------------------------------------------------------------
001342' 000004  000000' 001350'         PIRQ$,BEGIN,11$         ; QUEUE UP TO CONTINUE AT 11$ AND RTI
                                        ;-------------------------------------------------------------
001350' 012767  000010  176530  11$:    MOV     #10,ERRTYP      ;BAD ADDRESS
                                        ;*************************************************************
001356' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;NON-EXISTENT MEMORY ADDRESSING FAILURE
                                        ;*************************************************************
001364' 004767  000504                  JSR     PC,GETREG       ;RESTORE THE REGISTERS
001370' 042710  122000                  BIC     #122000,(R0)    ;CLEAR INTERRUPT ENABLE AND INTERRUPT BITS
001374' 104400  000000'                 EXIT$,BEGIN             ;EXIT TO MONITOR. MODULE WAIT FOR INTERRUPT.
001400' 052710  000100          3$:     BIS     #BIT6,(R0)      ;ENABLE RECEIVER INTERRUPTS
001404' 012604                          MOV     (SP)+,R4
001406' 012603                          MOV     (SP)+,R3
001410' 012602                          MOV     (SP)+,R2
001412' 012601                          MOV     (SP)+,R1
001414' 012600                          MOV     (SP)+,R0
001416' 000002                          RTI

                                        ;RECEIVER INTERRUPT SERVICE ROUTINE
                                ;THIS ROUTINE STORES A POINTER TO THE CSR ADDRESS OF THE INTERRUPTING DEVICE IN THE
                                ;RECEIVER QUEUE. SERVICE IS DEFERRED TO PRIORITY LEVEL 0. AT DEFERRED SERVICE (RCVSRV),
                                ;THE OFFSET IS FETCHED FROM THE QUEUE. THE INTERRUPT AND INTERRUPT ENABLE BITS ARE
                                ;CLEARED AND THE SILO IS EMPTIED INTO THE SOFTWARE BUFFER. IF ALL CHARACTERS
                                ;WERE NOT RECEIVED, THIS IS REPORTED. CHECK THE BUFFER AND REPORT DATA ERRORS. IF
                                ;ANOTHER RECEIVER IS ALREADY USING THE LINE CHECK BUFFER (LNCKBF), SUBSEQUENT
                                ;RECEIVERS WILL NOT BE PERMITTED ACCESS UNTIL THE FIRST IS COMPLETE. IF THERE
                                ;ARE ANY CHARACTER ERRORS REPORT THEM. WHEN ALL DATA ARE CHECKED, IT RELEASES
                                ;THE LINE CHECK BUFFER AND CLEAR THE CORRESPONDING BIT IN DONFLG.

001420' 010577  002370          RCVINT: MOV     R5,@RCVQPI      ;STORE THE OFFSET IN THE RETRIEVAL QUEUE
001424' 062767  000002  002362          ADD     #2,RCVQPI       ;UPDATE THE QUEUE ENTRY POINTER
001432' 022767  002700' 002354          CMP     #RCVQUE+16,RCVQPI ;HAS THE QUEUE BOUNDARY BEEN EXCEEDED?
001440' 103003                          BHIS    1$              ;IF NOT, CONTINUE PROCESSING
001442' 012767  002662' 002344          MOV     #RCVQUE,RCVQPI  ;RESET THE POINTER TO QUEUE BEGINNING
001450' 012605                  1$:     MOV     (SP)+,R5        ;RESTORE THE PREVIOUS VALUE OF R5
                                        ;-------------------------------------------------------------
```



```
001452' 000004  000000' 001460'         PIRQ$,BEGIN,RCVSRV              ; QUEUE UP TO CONTINUE AT RCVSRV AND RTI
                                        ;-------------------------------------------------------------
001460' 105767  002365          RCVSRV: TSTB    LCKOUT          ;TEST THE BUFFER LOCK FLAG. IS ACCESS PERMITTED?
001464' 001405                          BEQ     11$             ;IF YES, GO SET THE LOCK AND CHECK THE DATA
001466' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
001472' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
001476' 000770                          BR      RCVSRV          ;TRY TO ACCESS AGAIN
001500' 105367  002345          11$:    DECB    LCKOUT          ;DENY OTHER ACCESSES TO THE BUFFER
001504' 017700  002306                  MOV     @RCVQPO,R0      ;FETCH THE OFFSET FROM THE QUEUE
001510' 062767  000002  002300          ADD     #2,RCVQPO       ;UPDATE THE QUEUE RETRIEVAL POINTER
001516' 022767  002700' 002272          CMP     #RCVQUE+16,RCVQPO ;HAS THE QUEUE BOUNDARY BEEN EXCEEDED?
001524' 103003                          BHIS    1$              ;IF NO, CONTINUE PROCESSING
001526' 012767  002662' 002262          MOV     #RCVQUE,RCVQPO  ;IF YES, RESET THE POINTER TO THE QUEUE BEGINNING
001534' 012001                  1$:     MOV     (R0)+,R1        ;LOAD THE CSR ADDRESS INTO R1
001536' 042721  000300                  BIC     #300,(R1)+      ;DISABLE RECEIVER INTERRUPT AND INTERRUPT ENABLE
                                                                ;AND POINT R1 TO THE NEXT RECEIVED CHAR. REGISTER
001542' 011002                          MOV     (R0),R2         ;LOAD THE SOFTWARE SILO BUFFER ADDRESS
001544' 012767  077000  002272          MOV     #77000,RCVTMR   ;SET THE RECEIVER BREAK LOOP TIMER
001552' 012703  000100                  MOV     #64.,R3         ;SET THE COUNT FOR STORING THE DATA
001556' 011122                  2$:     MOV     (R1),(R2)+      ;STORE A WORD IN THE SOFTWARE SILO
001560' 002003                          BGE     3$              ;IF THE DATA WASN'T VALID, GO ALLOW A LITTLE TIME FOR IT
001562' 005303                          DEC     R3              ;IF IT WAS VALID, REDUCE THE COUNT.
001564' 001374                          BNE     2$              ;IF NOT ZERO, GO STORE THE NEXT WORD
001566' 000434                          BR      CKDATA          ;IF ALL CHARACTERS RECEIVED, GO CHECK THEM
001570' 005742                  3$:     TST     -(R2)           ;RESET R2 TO THE PROPER BUFFER LOCATION
001572' 004767  000240                  JSR     PC,SAVREG       ;SAVE THE REGISTER VALUES
001576' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
001602' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
001606' 004767  000262                  JSR     PC,GETREG       ;RESTORE THE REGISTER VALUES
001612' 005367  002226                  DEC     RCVTMR          ;HAS ENOUGH TIME BEEN PERMITTED FOR A CHARACTER?
001616' 100357                          BPL     2$              ;TRY TO GET ONE.
001620' 010367  176260                  MOV     R3,ASTAT        ;STATC WILL SHOW HOW MANY CHARACTERS
                                                                ;WERE NOT RECEIVED
001624' 014067  176250                  MOV     -(R0),CSRA      ;LOAD THE DEVICE ADDRESS
001630' 013067  176246                  MOV     @(R0)+,ACSR     ;LOAD THE CSR CONTENTS
001634' 004767  000176                  JSR     PC,SAVREG       ;SAVE THE REGISTERS
001640' 012767  000017  176240          MOV     #17,ERRTYP      ;UNKNOWN RCVR ERROR
                                        ;*************************************************************
001646' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;DID NOT RECEIVE ALL 64. CHARACTERS-STATC=NO. MISSING
                                        ;*************************************************************
001654' 004767  000214                  JSR     PC,GETREG

                                ;THIS ROUTINE PERMITS ONLY ONE DEVICE TO HAVE ITS DATA CHECKED AT A TIME. PENDING
                                ;DEVICES RETURN TO THE MONITOR IN A WAIT LOOP. WHEN ACCESS IS PERMITTED, THE DATA IS CHE
                                ;FOR EACH LINE OF THE DEVICE. DATA ERRORS ARE REPORTED.

001660' 012703  000010          CKDATA: MOV     #8.,R3          ;SET UP A COUNT FOR CLEARING THE BUFFER
001664' 012705  003770'                 MOV     #LNCKBF,R5      ;POINT TO THE BEGINNING OF THE BUFFER
001670' 016725  002146          2$:     MOV     BGNDATA,(R5)+   ;SET UP TWO LINE CHECK BYTES
001674' 005303                          DEC     R3              ;REDUCE THE COUNT. HAVE 16. BYTES BEEN CLEARED?
001676' 001374                          BNE     2$              ;IF NO, GO CLEAR THE REMAINING
001700' 011002                          MOV     (R0),R2         ;LOAD THE SOFTWARE SILO ADDRESS IN R2
001702' 016067  000010  002130          MOV     10(R0),DVCNMBR  ;SAVE THE NUMBER OF THIS DEVICE
001710' 012703  000100                  MOV     #64.,R3         ;LOAD COUNT TO COMPARE 64. CHARACTERS
001714' 012200                  3$:     MOV     (R2)+,R0        ;FETCH A WORD FROM THE SILO INTO R0
001716' 002024                          BGE     RCVDONE         ;IF IT'S INVALID, GO CLEAN UP BUFFERS
```

```
001720' 032700  070000          BIT     #70000,R0       ;ARE THERE ANY TRANSMISSION ERRORS?
001724' 001402                  BEQ     4$              ;IF NO, GO DETERMINE THE LINE NUMBER
001726' 004767  000304          JSR     PC,STATERR      ;IF YES, GO DETERMINE THE ERROR TYPE
001732' 110067  002112  4$:     MOVB    R0,RCVDATA      ;SAVE THE RECEIVED CHARACTER
001736' 000300                  SWAB    R0              ;PUT THE LINE NUMBER IN RO'S LOWER BYTE
001740' 042700  177760          BIC     #177760,R0      ;ISOLATE THE LINE NUMBER IN R0
001744' 126067  003770' 002075  CMPB    LNCKBF(R0),RCVDATA ;IS THE DATA CORRECT?
001752' 001402                  BEQ     5$              ;IF YES, GO CHECK THE NEXT CHARACTER
001754' 004767  000422          JSR     PC,DERROR       ;IF NO, GO REPORT A DATA ERROR
001760' 105260  003770'  5$:     INCB    LNCKBF(R0)      ;SET UP THE NEXT CHARACTER FOR THIS LINE
001764' 005303                  DEC     R3              ;REDUCE THE COUNT, ARE ALL 64. CHARACTERS CHECKED?
001766' 001352                  BNE     3$              ;IF NO, GO CHECK THE NEXT CHARACTER.

                        ;THIS ROUTINE UNLOCKS THE LINE CHECK BUFFER TO PERMIT ACCESS BY OTHER DEVICES.
                        ;IT THEN COMPUTES THE LINE NUMBER AND CLEARS THE APPROPIATE FLAG FROM THE WATCHDOG
                        ;TIMER BYTE.

001770' 105067  002055  RCVDONE:CLRB    LCKOUT          ;RESET THE BUFFER ACCESS FLAG
001774' 005000                  CLR     R0              ;THE FLAG WILL PROPAGATE IN R0
001776' 000261                  SEC                     ;USE THE CARRY BIT TO BUILD THE DELETION FLAG
002000' 006100          1$:     ROL     R0              ;MOVE THE FLAG TO THE NEXT DEVICE
002002' 005367  002032          DEC     DVCNMBR         ;WHEN THIS NUMBER IS NEGATIVE, THE CORRECT
002006' 002374                  BGE     1$              ;DEVICE IS POINTED TO.
002010' 140067  002033          BICB    R0,DONFLG       ;CLEAR THIS FLAG
002014' 104400  000000'         EXITS,BEGIN             ;EXIT TO MONITOR. MODULE WAIT FOR INTERRUPT.

                        ;SUBROUTINES
                        ;THE FIRST IS THE VECTOR LOADING SUBROUTINE. THE VECTOR ADDRESS IS PASSED IN R0,
                        ;THE DEVICE ADDRESS IS PASSED IN R2. THE LINKING TABLE ADDRESS IS PASSED IN R3.
                        ;THE BUSS REQUEST PRIORITY LEVEL IS A PARAMETER TAKEN INDIRECTLY THROUGH R5.

002020' 010320          VCTLOAD:MOV     R3,(R0)+        ;LOAD THE ADDRESS OF THE LINKING INSTRUCTION
002022' 113520                  MOVB    @(R5)+,(R0)+    ;AND THE PRIORITY LEVEL INTO THE VECTOR
002024' 005200                  INC     R0              ;POINT R0 TO THE NEXT VECTOR ADDRESS
002026' 022323                  CMP     (R3)+,(R3)+     ;POINT R3 TO THE CSR INSERT LOCATION
002030' 010223                  MOV     R2,(R3)+        ;LOAD THE DEVICE ADDRESS IN THE LINK TABLE
002032' 005723                  TST     (R3)+           ;ALIGN R3 FOR THE NEXT DEVICE
002034' 000205                  RTS     R5              ;RETURN TO CALLING BLOCK

                        ;THIS ROUTINE SAVES THE REGISTER VALUES IN THE ERROR QUEUE, A FIRST-IN,
                        ;FIRST-OUT DISCIPLINE WRAPAROUND BUFFER.

002036' 016705  001756  SAVREG: MOV     ERRQPI,R5       ;USE R5 FOR INDEXING CAPABILITY
002042' 010025                  MOV     R0,(R5)+        ;SAVE R0...
002044' 010125                  MOV     R1,(R5)+        ;SAVE R1...
002046' 010225                  MOV     R2,(R5)+        ;SAVE R2...
002050' 010325                  MOV     R3,(R5)+        ;SAVE R3...
002052' 010425                  MOV     R4,(R5)+        ;AND R4
002054' 022705  002762'         CMP     #ERRQUE+60,R5   ;HAS THE QUEUE BOUNDARY BEEN EXCEEDED?
002060' 103002                  BHIS    1$              ;IF NO, GO RELOAD THE POINTER
002062' 012705  002702'         MOV     #ERRQUE,R5      ;RESET R5 TO THE QUE BEGINNING
002066' 010567  001726  1$:     MOV     R5,ERRQPI       ;SET THE ENTRY POINTER TO NEXT ENTRY POINT
002072' 000207                  RTS     PC              ;RETURN TO THE CALLING BLOCK

                        ;THIS ROUTINE RETRIEVES REGISTER VALUES FROM THE ERROR QUEUE
```



```
002074' 016705  001722  GETREG: MOV     ERRQPO,R5       ;USE R5 FOR INDEXING CAPABILITY
002100' 012500                  MOV     (R5)+,R0        ;RETRIEVE R0...
002102' 012501                  MOV     (R5)+,R1        ;        R1...
002104' 012502                  MOV     (R5)+,R2        ;        R2...
002106' 012503                  MOV     (R5)+,R3        ;        R3...
002110' 012504                  MOV     (R5)+,R4        ;AND R4
002112' 022705  002762'         CMP     #ERRQUE+60,R5   ;HAS THE QUEUE BOUNDARY BEEN EXCEEDED?
002116' 103002                  BHIS    1$              ;IF NO, GO RELOAD THE POINTER
002120' 012705  002702'         MOV     #ERRQUE,R5      ;RESET R5 TO THE QUEUE BEGINNING
002124' 010567  001672  1$:     MOV     R5,ERRQPO       ;SET THE RETRIEVAL POINTER TO THE NEXT FETCH POINT.
002130' 000207                  RTS     PC              ;RETURN TO THE CALLING BLOCK

                        ;THIS ROUTINE CALCULATES THE BAUD RATE FROM THE SR1 OPTION. IF SR1 IS 0,THE DEFAULT
                        ;RATES OF 9600. ARE ASSIGNED. THE LEAST SIGNIFICANT BIT IS THE ONLY ONE CONSIDERED.
                        ;THE BAUD RATES ARE THEN LOADED INTO THE PROPER LINE PARAMETER REGISTER.

002132' 005767  175660  BAUDRTE:TST     SR1             ;IS A SPECIFIC RATE SELECTED?
002136' 001433                  BEQ     3$              ;IF NO,GO ASSIGN THE DEFAULT RATE
002140' 016704  175652          MOV     SR1,R4          ;IF YES COPY SR1
002144' 100003                  BPL     4$              ;BR IF NOT 8. WORDS BETWEEN VECTORS
002146' 042704  100000          BIC     #BIT15,R4       ;CLEAR VECTOR SELECT BIT
002152' 001425                  BEQ     3$              ;BR IF DEFAULT - USE 9600 BAUD
002154' 012703  000017  4$:     MOV     #17,R3          ;SET UP THE BIT CONFIGURATION FOR A BAUD RATE SELECTION
002160' 006004          1$:     ROR     R4              ;ISOLATE THE NEXT BIT IN THE CARRY BIT
002162' 103402                  BCS     2$              ;IF THIS IS THE BIT, PUT TOGETHER THE BAUD RATE
002164' 005303                  DEC     R3              ;IF NOT,CALCULATE THE NEXT BIT CONFIGURATION
002166' 000774                  BR      1$              ;GO CHECK THE NEXT ALTERNATIVE
002170' 000257          2$:     CCC                     ;ELIMINATE EXTRANEOUS FLAGS BEFORE ROTATING
002172' 010305                  MOV     R3,R5           ;MAKE A DUPLICATE COPY OF THE BAUD RATE
002174' 006105                  ROL     R5              ;BUILD THE TRANSMITTER RATE IN R5
002176' 006105                  ROL     R5
002200' 006105                  ROL     R5
002202' 006105                  ROL     R5
002204' 060305                  ADD     R3,R5           ;ADD THE RECEIVER RATE TO THE TRANSMITTER RATE
002206' 000305                  SWAB    R5              ;ALIGN THE RATE TO BIT POSITIONS 6-13
002210' 006005                  ROR     R5
002212' 006005                  ROR     R5
002214' 052705  000003          BIS     #3,R5
002220' 050561  000004          BIS     R5,4(R1)        ;PLACE THE BAUD RATES IN THE LINE PARAMETER REGISTER
002224' 000207                  RTS     PC              ;RETURN TO THE CALLING PROCEDURE
002226' 052761  033503  000004  3$:     BIS     #33503,4(R1)    ;SET THE DEFAULT RATE(9600. BAUD)
002234' 000207                  RTS     PC              ;RETURN TO CALLING PROCEDURE

                        ;THIS ROUTINE DETERMINES WHAT TYPE OF ERROR WAS INDICATED BY THE SILO AND REPORTS EACH

002236' 004767  177574  STATERR:JSR     PC,SAVREG       ;/SAVE THE REGISTERS
002242' 010167  175634          MOV     R1,ACSR         ;/REPORT THE CSR CONTENTS
002246' 005004                  CLR     R4              ;/SETUP TO CALCULATE
002250' 016703  001564          MOV     DVCNMBR,R3      ;/THE CSR
002254' 001404                  BEQ     2$              ;/DEVICE 0 JUST ADD ADDRESS
002256' 062704  000020  1$:     ADD     #20,R4          ;/POINT NEXT DH11
002262' 005303                  DEC     R3              ;/DECREMENT DH NUMBER
002264' 001374                  BNE     1$              ;/RIGHT DH11?
002266' 066704  175514  2$:     ADD     ADDR,R4         ;/ADD BASE ADDRESS TO OFFSET
002272' 010467  175602          MOV     R4,CSRA         ;/REPORT CSR ADDRESS
002276' 010001                  MOV     R0,R1           ;CALC LINE #
```

```
002300' 000301                  SWAB    R1              ;GET CORRECT HALF
002302' 042701  177770          BIC     #177770,R1      ;LEAVE ONLY LINE #
002306' 010167  175572          MOV     R1,ASTAT        ;PUT IT IN ASTAT
002312' 006100                  ROL     R0              ;PLACE THE OVERRUN BIT IN THE CARRY BIT
002314' 006100                  ROL     R0
002316' 103006                  BCC     3$              ;IF NO ERROR, GO CHECK THE NEXT BIT
002320' 012767  000021  175560  MOV     #21,ERRTYP      ;OVERRUN ERROR
                                ;*****************************************************************
002326' 104406  000000' 000000  SOFERS,BEGIN,NULL       ;OVERRUN ERROR-LINE NUMBER IN STATC
                                ;*****************************************************************
002334' 006100          3$:     ROL     R0              ;PLACE THE FRAME BIT IN THE CARRY BIT
002336' 103006                  BCC     4$              ;IF NO ERROR, GO CHECK THE NEXT BIT
002340' 012767  000022  175540  MOV     #22,ERRTYP      ;FRAMING ERROR
                                ;*****************************************************************
002346' 104406  000000' 000000  SOFERS,BEGIN,NULL       ;FRAMING ERROR-LINE NUMBER IN STATC
                                ;*****************************************************************
002354' 006100          4$:     ROL     R0              ;CHECK FOR A PARITY ERROR
002356' 103006                  BCC     5$              ;IF NO ERROR, GO RESTORE THE REGISTERS
002360' 012767  000023  175520  MOV     #23,ERRTYP      ;PARITY ERROR
                                ;*****************************************************************
002366' 104406  000000' 000000  SOFERS,BEGIN,NULL       ;PARITY ERROR-LINE NUMBER IN STATC
                                ;*****************************************************************
002374' 004767  177474  5$:     JSR     PC,GETREG       ;RESTORE THE REGISTERS
002400' 000207                  RTS     PC              ;RETURN TO CALLING PROCEDURE

                        ;THIS ROUTINE CALCULATES THE BUFFER ADDRESSES OF THE INCORRECT DATA AND REPORTS THE ERRO

002402' 004767  177430  DERROR: JSR     PC,SAVREG       ;SAVE THE PRESENT REGISTER VALUES
002406' 005004                  CLR     R4              ;/SETUP
002410' 016703  001424          MOV     DVCNMBR,R3      ;/TO CALCULATE
002414' 001404                  BEQ     2$              ;/THE CSR
002416' 062704  000020  1$:     ADD     #20,R4          ;/POINT NEXT DH11
002422' 005303                  DEC     R3              ;/DECREMENT DH NUMBER
002424' 001374                  BNE     1$              ;/RIGHT DH11 ?
002426' 066704  175354  2$:     ADD     ADDR,R4         ;/ADD BASE ADDRESS TO OFFSET
002432' 010467  175442          MOV     R4,CSRA         ;/REPORT CSR ADDRESS
002436' 116067  003770' 175442  MOVB    LNCKBF(R0),ASB  ;LOAD THE CORRECT DATA VALUE
002444' 116767  001400  175436  MOVB    RCVDATA,AWAS    ;LOAD THE ACTUAL DATA VALUE
002452' 005742                  TST     -(R2)           ;GET THE SILO ADDRESS OF THE DATA
002454' 010267  175424          MOV     R2,WASADR       ;LOAD IT FOR REPORTING
002460' 062700  003770'         ADD     #LNCKBF,R0      ;CALCULATE THE CORRECT VALUE ADDRESS
002464' 010067  175412          MOV     R0,SBADR        ;REPORT IT
                                ;*****************************************************************
002470' 104404  000000'         DATERS,BEGIN            ;DATA ERROR!!!
                                ;*****************************************************************
002474' 004767  177374          JSR     PC,GETREG       ;RESTORE THE REGISTERS
002500' 000207                  RTS     PC              ;RETURN TO THE CALLING PROCEDURE

                        ;LINKAGE TABLE
                        ;EACH ENTRY CONSISTS OF A JSR INSTRUCTION FOLLOWED BY A LOCATION INTO WHICH THE CSR IS
                        ;LOADED, FOLLOWED BY THE SILO BUFFER ADDRESS, FOLLOWED BY A JSR FOR THE TRANSMITTER
                        ;INTERRUPT, THE CSR ADDRESS WORD, AND THE NUMBER OF THE INTERRUPTING DEVICE

002502' 004567  176712  LNKTAB: JSR     R5,RCVINT       ;LINK FOR RECEIVER 0
002506' 000000                  0
002510' 002770'                 SILO0
```



```
002512' 004567  176402          JSR     R5,XMTINT       ;LINK FOR TRANSMITTER 0
002516' 000000                  0
002520' 000000                  0
002522' 004567  176672          JSR     R5,RCVINT       ;LINK FOR RECEIVER 1
002526' 000000                  0
002530' 003170'                 SILO1
002532' 004567  176362          JSR     R5,XMTINT       ;LINK FOR TRANSMITTER 1
002536' 000000                  0
002540' 000001                  1
002542' 004567  176652          JSR     R5,RCVINT       ;LINK FOR RECEIVER 2
002546' 000000                  0
002550' 003370'                 SILO2
002552' 004567  176342          JSR     R5,XMTINT       ;LINK FOR TRANSMITTER 2
002556' 000000                  0
002560' 000002                  2
002562' 004567  176632          JSR     R5,RCVINT       ;LINK FOR RECEIVER 3
002566' 000000                  0
002570' 003570'                 SILO3
002572' 004567  176322          JSR     R5,XMTINT       ;LINK FOR TRANSMITTER 3
002576' 000000                  0
002600' 000003                  3

002602' 017000  CNTTBL: 17000   ;DIFFERENT ITERATION COUNTS FOR
002604' 017000          17000   ;DIFFERENT BAUD RATES
002606' 017000          17000
002610' 017000          17000
002612' 011700          11700
002614' 006000          6000
002616' 005100          5100
002620' 003000          3000
002622' 001600          1600
002624' 000750          750
002626' 000500          500
002630' 000350          350
002632' 000300          300
002634' 000260          260
002636' 000165          165
002640' 000110          110

                        ;THESE ARE THE QUEUES.

002642' 000010  XMTQUE: .BLKW   8.              ;TRANSMITTER SERVICE QUEUE
002662' 000010  RCVQUE: .BLKW   8.              ;RECEIVER SERVICE QUEUE
002702' 000031  ERRQUE: .BLKW   25.             ;ERROR SERVICE QUEUE

                        ;THESE ARE THE BUFFERS

002764' 000004  XMTBUF: .BLKB   4.              ;FOUR CHARACTER TRANSMITTER BUFFER
002770' 000100  SILO0:  .BLKW   64.             ;RECEIVER BUFFER FOR DEVICE 0.
003170' 000100  SILO1:  .BLKW   64.             ;RECEIVER BUFFER FOR DEVICE 1
003370' 000100  SILO2:  .BLKW   64.             ;RECEIVER BUFFER FOR DEVICE 2
003570' 000100  SILO3:  .BLKW   64.             ;RECEIVER BUFFER FOR DEVICE 3
003770' 000020  LNCKBF: .BLKB   16.             ;BUFFER FOR CHECKING INDIVIDUAL LINE DATA

                        ;THESE ARE THE QUEUE POINTERS
```

```
GETREG  002074R          538     596     615     672     684     756#    834     857
GWBUF$= 104414           364#
HDRCNT  000044R          329#
HDRS$= 104405            364#    535     594     613     682
HAS     000050R          331#
I       004054R          537     951#
IT      000036R          326#    386*
ICOUNT  000040R          327#
IDNUM   000122R          356#
INIT    000030R          323#
INITIA  000624R          446     468#
INTR    000120R          355#
LCKOUT  004051R          433*    646     651*    717*    945#
LNCKBF  003770R          691     706     709*    848     852     920#
LNKTAB  002502R          394     865#
MAP22$= 104416           364#
MODNAM  000000R          310#
MODSP   000224R          324     362#
MSGN$ = 104403           364#    537
MSGS$ = 104402           364#
MSG$  = 104401           364#
M1      004060R          951     954#
M2      004067R          528     956#
M3      004075R          957#
NULL  = 000000           364#    535     594     613     682     820     826     832
NUMBA1  004052R          524*    528     946#
OPEN  = 000000           311     317     318     319     320     337     338     339     340     341     342     343     344
                         346     348     350     351     353     354     355     364#    924     925     926     927     928
                         929     933     934     935     936     937     938     939     940     942     943     944     945
                         946
OTOA$ = 104420           364#    528
PA      004034R          455     937#
PASCNT  000034R          325#
PIRQ$ = 000004           364#    590     609     644
POPSP = 005726           364#
POPSP2= 022626           364#
PRTY  = 000000           364#
PRTY0 = 000000           364#
PRTY1 = 000040           364#
PRTY2 = 000100           364#
PRTY3 = 000140           364#
PRTY4 = 000200           364#
PRTY5 = 000240           314     315     364#
PRTY6 = 000300           364#
PRTY7 = 000340           364#
PS    = 177776           364#
PSW   = 177776           364#
PUSH  = 005746           364#
PUSH2 = 024646           364#
RAND$ = 104417           364#
RANNUM  000054R          333#
RCVDAT  004050R          703*    706     849     944#
RCVDON  001770R          699     717#
RCVINT  001420R          637#    865     871     877     883
RCVQPI  004014R          424*    637*    638*    639     641*    926#
RCVQPO  004016R          425*    652     653*    654     656*    927#
```



```
RE      002662R          424     425     639     641     654     656     910#
RECV    001460R          644     646#    650
REPAR   004044R          661*    673*    941#
RESTRT  000244R          352     370     378#
RES1    000056R          335#
RES2    000060R          336#
RSTRT   000112R          352#
SAVREG  002036R          531     583     602     669     679     742#    801     839
SBADR   000102R          345#    853*
SELECT  004024R          368*    387*    389     438     477     514*    539     933#
SETUP1  000306R          389#
SETUP2  000416R          397     420#    549
SILO0   002770R          867     916#
SILO1   003170R          873     917#
SILO2   003370R          879     918#
SILO3   003570R          885     919#
SOFCNT  000042R          328#
SOFERS= 104406           364#    820     826     832
SOFPAS  000046R          330#
SPOINT  000032R          324#
SPSIZ = 000040             1#    357
SR1     000016R          317#    379     400     411     772     774
SR2     000020R          318#
        000022R          319#
        000024R          320#
        000024R          323     367#
        000026R          322#
STACER  002236R          702     801#
SVR0    000062R          337#
SVR1    000064R          338#
SVR2    000066R          339#
SVR3    000070R          340#
SVR4    000072R          341#
SVR5    000074R          342#
SVR6    000076R          343#
SYSCNT  000052R          332#
TIMER   000740R          500#    510
TMRCNT  004030R          499*    509*    935#
TMRSET  000732R          481     499#
TRPDFD= 000022           364#
VA      004032R          441*    443     936#
VCTLOA  002020R          406     409     731#
VECTOR  000010R          313#    392
WASADR  000104R          347#    851*
WDFR    000116R          354#
WDTO    000114R          353#
XFLAG   000005R          311#
XMTBUF  002764R          441     468     915#
XMTINT  001120R          559#    868     874     880     886
XMTQPI  004010R          420*    559*    560*    561     563*    924#
XMTQPO  004012R          422*    570     571*    572     574*    925#
XMTQUE  002642R          420     422     428     561     563     572     574     909#
XMTSRV  001164R          570#
.     = 004124R          909#    910#    911#    915#    916#    917#    918#    919#    920#    956#    961#
```

```
004010' 000000                  XMTQPI: OPEN                    ;TRANSMITTER QUEUE ENTRY (IN) POINTER
004012' 000000                  XMTQPO: OPEN                    ;TRANSMITTER QUEUE RETRIEVAL POINTER (OUT)
004014' 000000                  RCVQPI: OPEN                    ;RECEIVER QUEUE POINTERS
004016' 000000                  RCVQPO: OPEN
004020' 000000                  ERRQPI: OPEN                    ;ERROR QUE POINTERS
004022' 000000                  ERRQPO: OPEN

                                ;THESE ARE THE PROGRAM PARAMETERS

004024' 000000                  SELECT: OPEN                    ;ACTIVE DEVICE SELECTION PARAMETER
004026' 000000                  COUNT:  OPEN                    ;COUNTER FOR THE NUMBER OF ITERATIONS
004030' 000000                  TMRCNT: OPEN                    ;COUNTER FOR WATCHDOG TIMER
004032' 000000                  VA:     OPEN                    ;LOCATION FOR VIRTUAL ADDRESS(USED BY GETPA)
004034' 000000                  PA:     OPEN                    ;ARGUMENT RETURNED BY GETPA(PHYSICAL ADDRESS)
004036' 000000                  EA:     OPEN                    ;ARGUMENT RETURNED BY GETPA(EXTENDED ADDRESS BITS)
004040' 000000                  DVCNMBR:OPEN                    ;NUMBER OF DEVICE BEING PROCESSED
004042' 000000                  BGNDATA:OPEN                    ;TWO COPIES OF THE FIRST CHARACTER IN TRANSMIT PATTERN
004044' 000000                  RCVTMR: 0                               ;RECEIVER BREAK LOOP TIMER
004046'     000                 DATA:   .BYTE   OPEN            ;CHARACTER BUILDING BYTE
004047'     000                 DONFLG: .BYTE   OPEN            ;WATCHDOG FLAG FOR BUSY DEVICES
004050'     000                 RCVDATA:.BYTE   OPEN            ;BUFFER FOR CHARACTER CHECKING
004051'     000                 LCKOUT: .BYTE   OPEN            ;BUFFER ACCESS FLAG
004052' 000000                  NUMBA1: OPEN
                                .EVEN

                                ;THIS IS THE ASCII MESSAGE

004054' 004060'                 HUNG:   M1
004056' 177777                  -1

004060' 042504  044526  042503  M1:     .ASCII  'DEVICE '
004066'     040
004067' 000006                  M2:     .BLKB   6
004075'     040 052510  043515  M3:     .ASCIZ  ' HUNG%IT #AS DROPPED%'
004102' 044445  020124  040527
004110' 020123  051104  050117
004116' 042520  022504      000
004124'
000001                          .EVEN
                                .END
```



```
ACSR    000102R  346#  582*  601*  678*  802*
ACTVAT  000504R  438#
ADDR    000006R  312#  393   440   478   532   809   846
ADDR22= 001000   364#
ASB     000106R  350#  848*
ASTAT   000104R  348#  675*  814*
AWAS    000110R  351#  849*
BAUDRT  002132R  459   772#
BEGIN   000000R  309#  372   443   502   503   528   535   537   542   547   590   594   599
                 609   613   617   644   648   649   670   671   682   724   820   826   832
                 855
BGNDAT  004042R  469*  470*  692   940#
BIT0  = 000001   364#
BIT1  = 000002   364#
BIT10 = 002000   364#  578
BIT11 = 004000   364#  449
BIT12 = 010000   364#
BIT13 = 020000   364#  486
BIT14 = 040000   364#
BIT15 = 100000   364#  776
BIT2  = 000004   364#
BIT3  = 000010   364#
BIT4  = 000020   364#
BIT5  = 000040   364#
BIT6  = 000100   364#  618
BIT7  = 000200   364#
BIT8  = 000400   364#
BIT9  = 001000   364#  451
BREAK$= 104407   364#  502   503   648   649   670   671
BR1     000012R  314#  410
BR2     000013R  315#  407
RTOD$ = 104421   364#
CDATA$= 104412   364#
CKDATA  001660R  667   690#
CNTTBL  002602R  386   890#
CONFIG  000056R  334#
COUNT   004026R  934#
CSRA    000100R  344#  515*  522*  532*  581*  600*  677*  810*  847*
DATA    004046R  367*  469   470   473   474*  942#
DATCK$= 104411   364#
DATER$= 104404   364#  855
DERROR  002402R  708   839#
DONFLG  004047R  439*  504   512   723*  943#
DROP    000240R  371#  390
DVCNMB  004040R  696*  721*  804   841   939#
DVIDI   000014R  316#  368
EA      004036R  453   938#
ENDIT$= 104413   364#  547
END$  = 104410   364#  372   542
ERRQPI  004020R  426*  742   751*  928#
ERRQPO  004022R  427*  756   765*  929#
ERRQUF  002702R  426   427   748   750   762   764   911#
ERRTYP  000106R  349#  533*  592*  611*  680*  818*  824*  830*
EXIT$ = 104400   364#  599   617   724
FINISH  001110R  506   540   546#
GETPA$= 104415   364#  443
```

```
. ABS.  000000    000
        004124    001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 XDHAL0,XDHAL0/SOL/CRF:SYM=DDXCOM,XDHAL0
 RUN-TIME:  1  2  .3 SECONDS
 RUN-TIME RATIO:  24/5=4.8
 CORE USED:  7K  (13 PAGES)
```