

DAV11-A,
DAV11-B

DAV11 INTER EXER
CZDAVAO

AH-T257A-MC
FICHE 1 OF 1

MAY 1983
COPYRIGHT © 82-83
MADE IN USA



Microfiche grid containing multiple frames of data. The data is organized into columns and rows, with some frames containing headers and numerical values. The content is too faint to transcribe accurately.

.REM 8

I D E N T I F I C A T I O N

PRODUCT CODE: AC-T256A-MC
PRODUCT NAME: CZDAVAO DAV11 INTERPROC EXER
PRODUCT DATE: NOVEMBER 1982
MAINTAINER: CSS WEST DIAGNOSTIC ENGINEERING
 COSTA MESA, CALIFORNIA
AUTHOR: JOHN M. MARTIN

COPYRIGHT (C) 1982,1983
DIGITAL EQUIPMENT CORP, MAYNARD, MASSACHUSETTS 01754

THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON
A SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE
INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR
ANY OTHER COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE
MADE AVAILABLE TO ANY OTHER PERSON EXCEPT FOR USE ON SUCH
SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE TERMS. TITLE
TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES REMAIN
IN DEC.

THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT
NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL
EQUIPMENT CORPORATION.

DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF
ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.

T A B L E O F C O N T E N T S

1.0	ABSTRACT
2.0	REQUIREMENTS
2.1	EQUIPMENT
2.2	CONFIGURATION
2.3	STORAGE
3.0	LOADING PROCEDURE
4.0	STARTING PROCEDURE
5.0	SOFTWARE SWITCH REGISTER
5.1	OPTIONS
5.2	CONTROL
6.0	ERRORS
6.1	ERROR COMMENT
6.2	ERROR DATA
6.3	ERROR RECOVERY
7.0	MISCELLANEOUS
7.1	DEVICE BUS AND VECTOR ADDRESS
7.2	MODIFICATION
8.0	EXECUTION TIME
9.0	PROGRAM DESCRIPTION
9.1	GENERAL
9.2	PROGRAM SEGMENTS
10.0	MODIFICATION HISTORY

1.0 ABSTRACT

THE FUNCTION OF THE DAV11 INTERPROCESSOR EXERCISER IS TO VERIFY THE OPTION OPERATES ACCORDING TO SPECIFICATIONS. THE DIAGNOSTIC WILL ALSO VERIFY THAT THE DAV11 OPERATES IN ITS ENVIRONMENT SUCH AS THE SYSTEM IN WHICH IT IS INSTALLED.

THE DAV11 OPTION PROVIDES A LINK BETWEEN TWO Q-BUS PROCESSORS (DAV11-A) OR BETWEEN ONE Q-BUS AND ONE PDP-11 (UNIBUS) PROCESSOR (DAV11-B), USING THE FOLLOWING PROCESSOR/DAV11 CONFIGURATIONS:

Q-BUS PROCESSOR----- (DAV11-A) ----- UNIBUS PROCESSOR

Q-BUS PROCESSOR----- (DAV11-B) ----- Q-BUS PROCESSOR

CZDAV WILL TEST ALL COMPONENTS OF THE DAV11 OPTION, INCLUDING BOTH DR11B/DRV11B AND M7230/M5927 MODULES, AND THE INTERCONNECT CABLES (BC08R-25). THE DIAGNOSTIC WILL RUN IN BOTH PROCESSORS CONCURRENTLY.

2.0 REQUIREMENTS

2.1 EQUIPMENT

2.1.1 DAV11-A

ANY PDP-11 FAMILY CPU (WITH MINIMUM 16K MEMORY W/CONSOLE DEVICE)
PDP-11/03 OR ANY LSI-11 CPU (WITH MINIMUM 16K MEMORY W/CONSOLE)

- DR11-B UNIBUS DMA INTERFACE
- M7230 DUAL DIFFERENTIAL MODULE
- DRV11-B (M7950) DMA INTERFACE
- BC08R (2) DRV-11 INTERCONNECT CABLE
- BC08R-25 (3) INTERPROCESSOR INTERCONNECT CABLE

2.1.2 DAV11-B

TWO PDP-11/03 OR ANY LSI-11 CPUS (WITH MINIMUM 16K MEMORY W/CONSOLE)

- DRV11-B (2) (M7950) DMA INTERFACE
- M5927 (2) QUAD DIFFERENTIAL MODULE
- BC08R (4) DRV-11 INTERCONNECT CABLE
- BC08R-25 (3) INTERPROCESSOR INTERCONNECT CABLE

2.2 CONFIGURATION

THIS DIAGNOSTIC WILL SUPPORT BOTH THE DAV11-A AND DAV11-B OPTIONS
ON ANY PDP-11 FAMILY AND LSI-11 FAMILY PROCESSORS AND HAS BEEN
FULLY TESTED ON THE FOLLOWING PROCESSOR CONFIGURATIONS:

2.2.1 DAV11-A

- PDP-11/03 <-----> PDP-11/34
- LSI-11/23 <-----> PDP-11/34
- PDP-11/03 <-----> PDP-11/70
- LSI-11/23 <-----> PDP-11/70

2.2.2 DAV11-B

- PDP-11/03 <-----> LSI-11/23
- LSI-11/23 <-----> LSI-11/23

2.3 STORAGE

THE PROGRAM USES THE LOWER 4K WORDS OF SYSTEM MEMORY.

3.0 LOADING PROCEDURE

3.1 METHOD

THIS DIAGNOSTIC IS DISTRIBUTED ON BOOTABLE XXDP+ MEDIA. THE OPERATOR SHOULD FIRST BOOT THE DIAGNOSTIC SUPERVISOR FROM THE DISTRIBUTION KIT, AND RUN THE DIAGNOSTIC USING THE XXDP+ COMMANDS. REFER TO THE APPROPRIATE DAV11 OPTION DESCRIPTION FOR DIAGNOSTIC HIERARCHY AND INTSTALLATION INSTRUCTIONS PRIOR TO USING THIS DIAGNOSTIC.

4.0 STARTING PROCEDURE

BEFORE STARTING THE DIAGNOSTIC, COMPARE THE DEVICE REGISTER BASE ADDRESS AND INTERRUPT VECTOR ADDRESS WITH THE VALUES IN LOCATIONS DAVADR AND DRVECT, AS LISTED BELOW. IF THESE VALUES ARE INCORRECT, LOAD THE DIAGNOSTIC, HALT THE PROCESSOR, AND MODIFY THE LOCATIONS TO CONTAIN THE CORRECT VALUES, USING THE APPROPRIATE TECHNIQUE FOR YOUR CPU.

ADDRESS	LOCATION	VALUE
001360	DRVADR:	172410
001362	DRVECT:	000124

LOAD AND START THE DIAGNOSTIC IN THE SLAVE PROCESSOR FIRST, USING THE START ADDRESS TABLE LISTED BELOW.

PROCESSOR	START
MASTER	000200
SLAVE	000204

THE PROGRAM WILL IDENTIFY ITSELF AND, IF THE PROCESSOR DOES NOT HAVE A SWITCH REGISTER, THE DIAGNOSTIC WILL PROMPT THE OPERATOR FOR AN INITIAL SOFTWARE SWITCH REGISTER VALUE.

.RUN ZDAVAO
CZDAVAO DAV11-A/B INTERPROCESSOR EXCERISER

SWR = 000000 NEW =

THE PROGRAM WILL AUTOMATICALLY START AS MASTER (LOCATION 200), SO HALT THE PROCESSOR AND LOAD SLAVE START ADDRESS (LOCATION 204), AND RESTART THE CPU USING THE APPROPRIATE METHOD. SLAVE CPU WILL THEN WAIT FOR MASTER TO INITIATE PROGRAM SYNCHRONIZATION.

NEXT, LOAD AND START THE DIAGNOSTIC IN THE MASTER PROCESSOR AS DESCRIBED ABOVE.

5.0 SOFTWARE SWITCH REGISTER

5.1 CONTROL SWITCH SETTINGS

SWITCH	OCTAL	FUNCTION
BIT15=1	100000	HALT ON ERROR
BIT13=1	020000	INHIBIT ERROR TYPEOUTS
BIT10=1	002000	BELL ON ERROR

5.2 CONTROL

IF THE DIAGNOSTIC IS RUN ON A MACHINE WITHOUT A SWITCH REGISTER, THEN A SOFTWARE SWITCH REGISTER IS USED, WHICH ALLOWS THE USER THE SAME SWITCH OPTIONS AS THE HARDWARE SWITCH REGISTER. IF THE HARDWARE SWITCH REGISTER DOES NOT EXIST, OR IF ONE DOES AND IT CONTAINS ALL ONES, THE SOFTWARE SWITCH REGISTER (LOCATION 176) IS USED.

- (1) THE SOFTWARE SWITCH REGISTER "SWREG" (LOCATION 176) CAN BE CHANGED BY USING THE ODT FACILITIES.
- (2) THE SOFTWARE SWITCH REGISTER CAN BE CHANGED UNDER PROGRAM CONTROL BY TYPING THE "CONTROL" AND "G" KEYS. THIS KEYBOARD OPERATION WILL PRINT OUT THE CURRENT CONTENTS AND ACCEPT NEW OCTAL SWITCH REGISTER DATA, TERMINATED WITH A CARRIAGE RETURN.
- (3) ONCE THE ODT MODE HAS BEEN ENTERED DUE TO AN ERROR CONDITION WITH BIT15 SET (HALT ON ERROR), STEP (2) ABOVE IS OF NO VALUE, SO RESORT TO STEP (1), TO ALTER THE SOFTWARE SWITCH REGISTER IF DESIRED BEFORE TYPING "P" (CONTINUE).

6.0 ERRORS

6.1 ERROR COMMENT

ALL ERRORS ARE ACCOMPANIED BY A DESCRIPTIVE COMMENT AS TO THE TYPE OF FAILURE. FURTHER QUALIFICATION OF THE ERROR CAN BE OBTAINED FROM THE PROGRAM COMMENT AT THE ERROR PC OR FROM THE TEST ITSELF.

6.2 ERROR DATA

ERRPC LISTING ADDRESS WHERE THE ERROR WAS DETECTED.
BUSADR DAV11 BUS REGISTER ADDRESS OF CONCERNED OPERATION.
EXPCT DATA VALUE THAT WAS EXPECTED.
RCVD DATA VALUE THAT WAS RECEIVED.
MEMADR MEMORY ADDRESS OF DATA ERROR.

6.3 ERROR RECOVERY

BECAUSE OF THE SYNCHRONIZATION ESTABLISHED WITH THE OTHER COMPUTER, ALL ERROR CONDITIONS FORCE AN AUTOMATIC LOOP ON TEST UNTIL THE ERROR IS ELIMINATED. SOFTWARE SWITCH REGISTER CONTROL SHOULD NOT AFFECT THE ESTABLISHED SYNC.

7.0 PROCESSOR SYNCHRONIZATION

7.1 PROGRAM INITIALIZATION PROTOCOL

WHEN THE PROGRAM IS STARTED IN THE SLAVE PROCESSOR, THE SLAVE WILL FIRST TEST THE MASTER STATUS REGISTER VALUE TO DETERMINE IF THE MASTER IS WAITING. IF NOT, THE SLAVE WILL REMAIN IN AN INFINITE LOOP UNTIL A CHANGE IN MASTER STATUS HAS BEEN OBSERVED, AT WHICH TIME THE FIRST TEST WILL BEGIN. IF THE MASTER HAS BEEN STARTED FIRST, THERE WILL BE NO RESPONSE FROM THE SLAVE AND THE MASTER WILL TIME-OUT, PRINTING THE FOLLOWING MESSAGE, AND REENTER THE WAIT LOOP.

STATUS: MASTER
MASTER CANNOT START COMMUNICATION WITH SLAVE-WAITING FOR DSTATC

ERRPC STATUS
002166 002200

7.2 PROGRAM SYNCHRONIZATION

ALL COMMUNICATION BETWEEN MASTER AND SLAVE IS INTERLOCKED THROUGH DRVCSR STATUS BITS. DURING EACH TEST, THE MASTER CPU WILL INITIATE THE INTERLOCK AND WAIT FOR THE SLAVE TO RESPOND. CONVERSELY, THE SLAVE WILL FIRST WAIT FOR THE MASTER TO SET/RESET THE STATUS BITS, BEFORE RESPONDING. IF SYNCHRONIZATION IS LOST DURING ANY INTERLOCK, EITHER CPU MAY TIME-OUT, AND THE RESULTING MESSAGE WILL BE DISPLAYED:

STUCK WAITING FOR COMPANION INTERLOCK

ERRPC STATUS
XXXXXX XXXXXX

WHERE: ERRPC = LOCATION IN TEST FOLLOWING CALL TO WAIT
STATUS = CURRENT VALUE OF DRVCSR REGISTER

AT THIS POINT THE CPU WILL ATTEMPT TO RESYNCHRONIZE WITH THE OTHER, AND WILL DISPLAY THE FOLLOWING MESSAGE BEFORE RETURNING TO THE INITIALIZATION PROTOCOL SECTION.

*RESYNC...

8.0 EXECUTION TIME

EXECUTION TIME IS ABOUT 1 MINUTE, AND MAY BE FASTER FOR SOME PDP-11 PROCESSORS. NOTE THAT THE "END OF PASS" MESSAGE IS ONLY REPORTED AT THE COMPUTER WHICH WAS STARTED AS THE INITIAL MASTER (START 200).

9.0 PROGRAM DESCRIPTION

9.1 GENERAL

THIS INTERPROCESSOR EXERCISER WAS DESIGNED TO TEST THE I/O ABILITY OF THE DAV11 INTERFACE TO COMMUNICATE BETWEEN TWO COMPUTERS. THE TWO COMPUTERS ARE STARTED AT DIFFERENT ADDRESSES TO ESTABLISH INITIAL SYNCHRONIZATION. THE SLAVE COMPUTER IS STARTED FIRST (START 204), AND THE MASTER COMPUTER IS STARTED SECOND (START 200). THE TERMS "MASTER" AND "SLAVE" SHOULD BE USED LOOSELY AS THE MASTER WILL BECOME THE SLAVE AND THE SLAVE WILL BECOME THE MASTER AS THE PROGRAM ADVANCES. THE COMPUTER STARTED AT ADDRESS 200 WILL ALWAYS REPORT THE "END OF PASS" MESSAGE.

9.2 PROGRAM SEGMENTS

- (1) MTST1 - MASTER SENDS OUT PROGRAM CONTROLLED SINGLE WORDS THROUGH THE DATA BUFFER REGISTER AND EXPECTS THE SLAVE TO ECHO EACH WORD BACK TO THE MASTER VIA THE DATA BUFFER REGISTER.
- (2) MTST2 - MASTER SENDS OUT A "FNCT" BIT CODE IN THE COMMAND STATUS REGISTER AND EXPECTS THE SLAVE TO ECHO EACH CODE IN ITS "FNCT" BITS. THE MASTER WILL READ THE "STAT" BIT CODE FROM THE COMMAND/STATUS REGISTER AND COMPARE IT TO THE CODE WRITTEN.
- (3) MTST3 - MASTER SENDS OUT A 32-WORD DATA BLOCK TO THE SLAVE AND CHECKS FOR PROPER INTERRUPT STATUS, WORD COUNT AND BUFFER ADDRESS AT THE COMPLETION OF THE TRANSFER.
- (4) MTST4 - MASTER SENDS A DATA BLOCK TO THE SLAVE. THE SLAVE IS SET FOR A LARGER BLOCK AND DOES NOT COMPLETE. THE MASTER SETS ATTENTION INTERRUPT. THE SLAVE CHECKS THAT IT GOT THE ATTENTION INTERRUPT.
- (5) STST1 - SLAVE ECHOS ALL CHANGES IN THE DATA BUFFER REGISTER.
- (6) STST2 - SLAVE READS THE "STAT" BIT CODE FROM ITS COMMAND/STATUS REGISTER, CONVERTS THIS CODE AND WRITES IT INTO ITS "FNCT" BITS IN THE COMMAND/STATUS REGISTER.
- (7) STST3 - SLAVE RECEIVES A 32-WORD DATA BLOCK FROM THE MASTER AND CHECKS FOR PROPER INTERRUPT STATUS, WORD COUNT BUFFER ADDRESS AND DATA CONTENT.
- (8) STST4 - SLAVE HAS LARGER WORD COUNT THAN MASTER SENDS. EXPECTS AN ATTENTION INTERRUPT FROM MASTER.

10.0 MAINTENANCE HISTORY

DATE	REVISION		REASON
FEB 79	YW-Z0171-A.0	J. MARTIN	INITIAL VERSION
MAR 81	YW-Z0171-B.0	J. MARTIN	DR11-B REV T COMPATABILITY
NOV 82	CZDAV-A.0	P. HOLSINGER	FIRST RELEASE OF DIAGNOSTIC

424

&

.LIST SEQ,BIN,LOC

426
427
428
429
430
431
432
433
434
435
436
437

.SBTTL OPERATIONAL SWITCH SETTINGS

```

:*
$TN=1
$SWR=122000
:*
:*      SWITCH      USE
:*      -----      -----
:*      15          HALT ON ERROR
:*      13          INHIBIT ERROR TYPEOUTS
:*      10          BELL ON ERROR

```

.SBTTL BASIC DEFINITIONS

```

001100  ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
104000  STACK= 1100
000004  ERROR= EMT
        SCOPE= IOT

```

;*MISCELLANEOUS DEFINITIONS

```

000011  HT= 11          ;;CODE FOR HORIZONTAL TAB
000012  LF= 12          ;;CODE FOR LINE FEED
000015  CR= 15          ;;CODE FOR CARRIAGE RETURN
000200  CRLF= 200       ;;CODE FOR CARRIAGE RETURN-LINE FEED
177776  PS= 177776     ;;PROCESSOR STATUS WORD
177776  PSW= PS
177774  STKLMT= 177774  ;;STACK LIMIT REGISTER
177772  PIRQ= 177772   ;;PROGRAM INTERRUPT REQUEST REGISTER
177570  DSWR= 177570   ;;HARDWARE SWITCH REGISTER
177570  DDISP= 177570  ;;HARDWARE DISPLAY REGISTER

```

;*GENERAL PURPOSE REGISTER DEFINITIONS

```

000000  R0= %0          ;;GENERAL REGISTER
000001  R1= %1          ;;GENERAL REGISTER
000002  R2= %2          ;;GENERAL REGISTER
000003  R3= %3          ;;GENERAL REGISTER
000004  R4= %4          ;;GENERAL REGISTER
000005  R5= %5          ;;GENERAL REGISTER
000006  R6= %6          ;;GENERAL REGISTER
000007  R7= %7          ;;GENERAL REGISTER
000006  SP= R6
000007  PC= R7

```

;*PRIORITY LEVEL DEFINITIONS

```

000000  PR0= 0          ;;PRIORITY LEVEL 0
000040  PR1= 40         ;;PRIORITY LEVEL 1
000100  PR2= 100        ;;PRIORITY LEVEL 2
000140  PR3= 140        ;;PRIORITY LEVEL 3
000200  PR4= 200        ;;PRIORITY LEVEL 4
000240  PR5= 240        ;;PRIORITY LEVEL 5
000300  PR6= 300        ;;PRIORITY LEVEL 6
000340  PR7= 340        ;;PRIORITY LEVEL 7

```

;*''SWITCH REGISTER'' SWITCH DEFINITIONS

```

100000  SW15= 100000
040000  SW14= 40000
020000  SW13= 20000
010000  SW12= 10000

```

```

004000 SW11= 4000
002000 SW10= 2000
001000 SW09= 1000
000400 SW08= 400
000200 SW07= 200
000100 SW06= 100
000040 SW05= 40
000020 SW04= 20
000010 SW03= 10
000004 SW02= 4
000002 SW01= 2
000001 SW00= 1
001000 SW9= SW09
000400 SW8= SW08
000200 SW7= SW07
000100 SW6= SW06
000040 SW5= SW05
000020 SW4= SW04
000010 SW3= SW03
000004 SW2= SW02
000002 SW1= SW01
000001 SW0= SW00
  
```

;*DATA BIT DEFINITIONS (BIT00 TO BIT15)

```

100000 BIT15= 100000
040000 BIT14= 40000
020000 BIT13= 20000
010000 BIT12= 10000
004000 BIT11= 4000
002000 BIT10= 2000
001000 BIT09= 1000
000400 BIT08= 400
000200 BIT07= 200
000100 BIT06= 100
000040 BIT05= 40
000020 BIT04= 20
000010 BIT03= 10
000004 BIT02= 4
000002 BIT01= 2
000001 BIT00= 1
001000 BIT9= BIT09
000400 BIT8= BIT08
000200 BIT7= BIT07
000100 BIT6= BIT06
000040 BIT5= BIT05
000020 BIT4= BIT04
000010 BIT3= BIT03
000004 BIT2= BIT02
000002 BIT1= BIT01
000001 BIT0= BIT00
  
```

;*BASIC "CPU" TRAP VECTOR ADDRESSES

```

000004 ERRVEC= 4 ;; TIME OUT AND OTHER ERRORS
000010 RESVEC= 10 ;; RESERVED AND ILLEGAL INSTRUCTIONS
000014 TBITVEC= 14 ;; "T" BIT
000014 TRTVEC= 14 ;; TRACE TRAP
000014 BPTVEC= 14 ;; BREAKPOINT TRAP (BPT)
  
```

```

000020      IOTVEC= 20      ;; INPUT/OUTPUT TRAP (IOT) **SCOPE**
000024      PWRVEC= 24      ;; POWER FAIL
000030      EMTVEC= 30      ;; EMULATOR TRAP (EMT) **ERROR**
000034      TRAPVEC=34      ;; "TRAP" TRAP
000060      TKVEC= 60      ;; TTY KEYBOARD VECTOR
000064      TPVEC= 64      ;; TTY PRINTER VECTOR
000240      PIRQVEC=240     ;; PROGRAM INTERRUPT REQUEST VECTOR

438
439      106427      MTPS=106427      ;; INSTR EQUATE THAT MOVES BYTE TO PSW
440
441      ;REGISTER BITS
442      000002      FNCT1= 2
443      000004      FNCT2= 4
444      000010      FNCT3= 10
445      004000      DSTATA= 4000
446      002000      DSTATB= 2000
447      001000      DSTATC= 1000
448
449      .SBTTL TRAP CATCHER

      000000      .=0
      ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
      ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
      ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
      000174      000174      .=174
000174      000000      DISPREG: .WORD 0      ;; SOFTWARE DISPLAY REGISTER
000176      000000      SWREG: .WORD 0      ;; SOFTWARE SWITCH REGISTER
450
451      000200      000137      001460      .SBTTL STARTING ADDRESS(ES)
452      000204      000137      001466      JMP @#START1      ;; JUMP TO STARTING ADDRESS OF PROGRAM
453
454
455      ;Q-BUS 'B-EVENT' INTERRUPT HANDLER
456      000100      000100      .=100
457      000102      000340      .WORD 104      ;VECTOR TO LOC 104
458      000104      000002      .WORD 340      ;AT PRIORITY 7
459
      .WORD 2      ;RTI
  
```

460

.SBTTL COMMON TAGS

*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
*USED IN THE PROGRAM.

001100	001100			\$CMTAG:	.=1100	:: START OF COMMON TAGS
001100	000000			\$PASS:	.WORD 0	:: CONTAINS PASS COUNT
001102	000			\$TSTNM:	.BYTE 0	:: CONTAINS THE TEST NUMBER
001103	000			\$ERFLG:	.BYTE 0	:: CONTAINS ERROR FLAG
001104	000000			\$ICNT:	.WORD 0	:: CONTAINS SUBTEST ITERATION COUNT
001106	000000			\$LPADR:	.WORD 0	:: CONTAINS SCOPE LOOP ADDRESS
001110	000000			\$LPERR:	.WORD 0	:: CONTAINS SCOPE RETURN FOR ERRORS
001112	000000			\$ERTTL:	.WORD 0	:: CONTAINS TOTAL ERRORS DETECTED
001114	000			\$ITEMB:	.BYTE 0	:: CONTAINS ITEM CONTROL BYTE
001115	001			\$ERMAX:	.BYTE 1	:: CONTAINS MAX. ERRORS PER TEST
001116	000000			\$ERRPC:	.WORD 0	:: CONTAINS PC OF LAST ERROR INSTRUCTION
001120	000000			\$GDADR:	.WORD 0	:: CONTAINS ADDRESS OF 'GOOD' DATA
001122	000C00			\$BDADR:	.WORD 0	:: CONTAINS ADDRESS OF 'BAD' DATA
001124	000000			\$GDDAT:	.WORD 0	:: CONTAINS 'GOOD' DATA
001126	000000			\$BDDAT:	.WORD 0	:: CONTAINS 'BAD' DATA
001130	000000				.WORD 0	:: RESERVED--NOT TO BE USED
001132	000000				.WORD 0	
001134	000			\$AUTOB:	.BYTE 0	:: AUTOMATIC MODE INDICATOR
001135	000			\$INTAG:	.BYTE 0	:: INTERRUPT MODE INDICATOR
001136	000000				.WORD 0	
001140	177570			\$WR:	.WORD DSWR	:: ADDRESS OF SWITCH REGISTER
001142	177570			\$DISPLAY:	.WORD DDISP	:: ADDRESS OF DISPLAY REGISTER
001144	177560			\$TKS:	177560	:: TTY KBD STATUS
001146	177562			\$TKB:	177562	:: TTY KBD BUFFER
001150	177564			\$TPS:	177564	:: TTY PRINTER STATUS REG. ADDRESS
001152	177566			\$TPB:	177566	:: TTY PRINTER BUFFER REG. ADDRESS
001154	000			\$NULL:	.BYTE 0	:: CONTAINS NULL CHARACTER FOR FILLS
001155	002			\$FILLS:	.BYTE 2	:: CONTAINS # OF FILLER CHARACTERS REQUIRED
001156	012			\$FILLC:	.BYTE 12	:: INSERT FILL CHARS. AFTER A 'LINE FEED'
001157	000			\$TPFLG:	.BYTE 0	:: 'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
001160	207	377	377	\$BELL:	.ASCIZ <207><377><377>	:: CODE FOR BELL
001163	000					
001164	077			\$QUES:	.ASCII /?/	:: QUESTION MARK
001165	015			\$CRLF:	.ASCII <15>	:: CARRIAGE RETURN
001166	012	000		\$LF:	.ASCIZ <12>	:: LINE FEED

.SBTTL ERROR POINTER TABLE

;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
;*LOCATION \$ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
;*NOTE1: IF \$ITEMB IS 0 THE ONLY PERTINENT DATA IS (\$ERRPC).
;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:

;* EM ::POINTS TO THE ERROR MESSAGE
;* DH ::POINTS TO THE DATA HEADER
;* DT ::POINTS TO THE DATA
;* DF ::POINTS TO THE DATA FORMAT

Line	Index	Pointer	Pointer	Description
461	001170			
462	001170	010234	1	:SLAVE DRV11B FAILED TO ECHO DBR CONTENTS :ERRPC BUSADR EXPCT RCVD :\$ERRPC \$BDADR \$GDDAT \$BDDAT
463	001172	011443	EM1	
464	001174	011554	DH1	
465	001176	000000	DT1	
466			0	
467				
468	001200	010327	2	:SLAVE DRV11B FAILED TO ECHO 'STAT'BITS :ERRPC BUSADR EXPCT RCVD :\$ERRPC \$BDADR \$GDDAT \$BDDAT
469	001202	011443	EM2	
470	001204	011554	DH1	
471	001206	000000	DT1	
472			0	
473				
474	001210	010421	3	:FAILED TO INTR ON A 'DATI' :ERRPC BUSADR EXPCT RCVD :\$ERRPC \$BDADR \$GDDAT \$BDDAT
475	001212	011443	EM3	
476	001214	011554	DH1	
477	001216	000000	DT1	
478			0	
479				
480	001220	010473	4	:STATUS ER ON 'DATI' :ERRPC BUSADR EXPCT RCVD :\$ERRPC \$BDADR \$GDDAT \$BDDAT
481	001222	011443	EM4	
482	001224	011554	DH1	
483	001226	000000	DT1	
484			0	
485				
486	001230	010541	5	:WORD COUNT ER ON 'DATI' :ERRPC BUSADR EXPCT RCVD :\$ERRPC \$BDADR \$GDDAT \$BDDAT
487	001232	011443	EM5	
488	001234	011554	DH1	
489	001236	000000	DT1	
490			0	
491				
492	001240	010613	6	:BUFFER ADRS ER ON 'DATI' :ERRPC BUSADR EXPCT RCVD :\$ERRPC \$BDADR \$GDDAT \$BDDAT
493	001242	011443	EM6	
494	001244	011554	DH1	
495	001246	000000	DT1	
496			0	
497				
498	001250	010666	7	:FAILED TO INTR ON A 'DATO' :ERRPC BUSADR EXPCT RCVD :\$ERRPC \$BDADR \$GDDAT \$BDDAT
499	001252	011443	EM7	
500	001254	011554	DH1	
501	001256	000000	DT1	
502			0	

503			;ERROR 10	
504	001260	010735	EM10	;STATUS ER ON 'DATO'
505	001262	011443	DH1	;ERRPC BUSADR EXPCT RCVD
506	001264	011554	DT1	;\$ERRPC \$BDADR \$GDDAT \$BDDAT
507	001266	000000	0	
508				
509			;ERROR 11	
510	001270	011002	EM11	;WORD COUNT ER ON 'DATO'
511	001272	011443	DH1	;ERRPC BUSADR EXPCT RCVD
512	001274	011554	DT1	;\$ERRPC \$BDADR \$GDDAT \$BDDAT
513	001276	000000	0	
514				
515			;ERROR 12	
516	001300	011053	EM12	;BUFFER ADRS ER ON 'DATO'
517	001302	011443	DH1	;ERRPC BUSADR EXPCT RCVD
518	001304	011554	DT1	;\$ERRPC \$BDADR \$GDDAT \$BDDAT
519	001306	000000	0	
520				
521			;ERROR 13	
522	001310	011125	EM13	;DATA ER ON 'DATO'
523	001312	011500	DH2	;ERRPC MEMADR EXPCT RCVD
524	001314	011554	DT1	;\$ERRPC \$BDADR \$GDDAT \$BDDAT
525	001316	000000	0	
526				
527			;ERROR 14	
528	001320	011170	EM14	;NO ATTN INTR
529	001322	011443	DH1	
530	001324	011554	DT1	
531	001326	000000	0	
532				
533			;ERROR 15	
534	001330	011223	EM15	;ATTN STATUS ER
535	001332	011443	DH1	
536	001334	011554	DT1	
537	001336	000000	0	
538				
539			;ERROR 16	
540	001340	011263	EM16	;PROTOCOL INITIALIZATION ERROR -MASTER AND SLAVE
541	001342	011535	DH3	;ERRPC STATUS
542	001344	011566	DT2	;\$ERRPC \$ERRST
543	001346	000000	0	
544				
545			;ERROR 17	
546	001350	011374	EM17	;STUCK WAITING FOR COMPANION INTERLOCK
547	001352	011535	DH3	;ERRPC STATUS
548	001354	011574	DT3	;\$ERRPC \$ERRST
549	001356	000000	0	
550				

```

552 ;DAV11 BASE REGISTER ADDRESS ASSIGNMENT
553
554 001360 172410 DRVADR: 172410 ;MODIFY THIS LOC IF DIFFERENT
555
556 ;DAV11 VECTOR ADDRESS ASSIGNMENT
557
558 001362 000124 DRVECT: 124 ;MODIFY THIS LOC IF DIFFERENT
559
560 ;DAV11 BUS REGISTER ADDRESS POINTERS
561
562 001364 172410 DRVWCR: 172410 ;WORD COUNT
563 001366 172412 DRVBAR: 172412 ;BUFFER ADDRESS
564 001370 172414 DRVCSR: 172414 ;COMMAND/STATUS
565 001372 172416 DRVDBR: 172416 ;DATA BUFFER
566
567 ;DAV11 VECTOR ADDRESS POINTERS
568
569 001374 000124 DRVCT0: 124 ;READY & NEX VECTOR
570 001376 000126 DRVCT2: 126 ;NEW PSW ON INTR
571
572 ;COMMON PROGRAM LOCATION(S)
573
574 001400 000000 STAT: 0 ;STATUS BIT(S) TO TEST
575 001402 000000 INTFLG: 0 ;INTERRUPT FLAG
576 001404 000000 TIME: 0 ;GENERAL PURPOSE COUNTER
577 001406 000000 ABORT: 0 ;PROGRAM ABORT TIMER
578 001410 000000 TEMP: 0 ;TEMPORARY STORAGE
579 001412 000000 $STSTPC: 0 ;PC STORAGE
580 001414 000000 $ERRST: 0 ;CSR STATUS
581 001416 000000 SAVE: 0 ;REG DATA SAVED HERE
582 001420 000000 MSTER: 0 ;0=MASTER START - NON-ZERO=SLAVE START
583 001422 000001 ICOUNT: 1 ;# OF TIMES TO REPEAT ALL TESTS BEFORE END PASS MSG
584
585 001424 177740 XPRAM: -32. ;XMIT WORD COUNT
586 001426 011632 DBUF ;XMIT BUFFER ADRS
587 001430 000103 103 ;XMIT STATUS/CONTROL: INTR ENABLE, FNCT1, GO
588
589 001432 177740 RPRAM: -32. ;RCV WORD COUNT
590 001434 011632 DBUF ;RCV BUFFER ADRS
591 001436 000105 105 ;RCV STATUS/CONTROL: INTR ENABLE, FNCT2, GO
592
593 001440 177740 XPRAM4: -32. ;XMIT WORD COUNT
594 001442 011632 DBUF ;XMIT BUFFER ADDRESS
595 001444 000103 103 ;XMIT STATUS/CONTROL: INTR ENABLE, FNCT1, GO
596
597 001446 177730 RPRAM4: -40. ;RCV WORD COUNT
598 001450 011632 DBUF ;RCV BUFFER ADDRESS
599 001452 000105 105 ;RCV STATUS/CONTRGL: INTR ENABLE, FNCT2, GO
600
601 001454 000000 LSI: 0 ;LSI FLAG
602 001456 000000 ERRFLG: 0 ;ERROR FLAG
603

```

```

605      .SBTTL PROGRAM START
606
607 001460 005037 001420 START1: CLR  MSTER      ;THIS IS MASTER START
608 001464 000403          BR      START      ;SKIP NEXT
609
610 001466 012737 177777 001420 START2: MOV  #-1,MSTER      ;SLAVE START
611
612 001474          START:
613
614      ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
615 001474 012706 001100      MOV  #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
616 001500 005026          CLR  (R6)+          ;;CLEAR MEMORY LOCATION
617 001502 022706 001140      CMP  #SWR,R6      ;;DONE?
618 001506 001374          BNE  -6              ;;LOOP BACK IF NO
619 001510 012706 001100      MOV  #STACK,SP      ;;SETUP THE STACK POINTER
620
621      ;;INITIALIZE A FEW VECTORS
622 001514 012737 006566 000030      MOV  #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
623 001522 012737 000340 000032      MOV  #340,@#EMTVEC+2 ;;LEVEL 7
624 001530 012737 007622 000034      MOV  #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
625 001536 012737 000340 000036      MOV  #340,@#TRAPVEC+2;LEVEL7
626
627      ;;SIZE FOR A HARDWARE SWITCH REGISTER, IF NOT FOUND OR IT IS
628      ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER,
629 001544 013746 000004          MOV  @#ERRVEC,-(SP) ;;SAVE ERROR VECTOR
630 001550 012737 001604 000004      MOV  #64,@#ERRVEC  ;;SET UP ERROR VECTOR
631 001556 012737 177570 001140      MOV  #DSWR,SWR    ;;SETUP FOR A HARDWARE SWITCH REGISTER
632 001564 012737 177570 001142      MOV  #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
633 001572 022777 177777 177340      CMP  #-1,@SWR    ;;TRY TO REFERENCE HARDWARE SWR
634 001600 001012          BNE  66$          ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
635          ;;AND THE HARDWARE SWR IS NOT = -1
636 001602 000403          BR   65$          ;;BRANCH IF NO TIMEOUT
637 001604 012716 001612      64$: MOV  #65$,(SP)    ;;SET UP FOR TRAP RETURN
638 001610 000002          RTI
639 001612 012737 000176 001140      65$: MOV  #SWREG,SWR   ;;POINT TO SOFTWARE SWR
640 001620 012737 000174 001142      MOV  #DISPREG,DISPLAY
641
642      ;CHECK IF 11/03(LSI11)
643 001626 012737 001654 000004      66$: MOV  #70$,@#ERRVEC ;;SET UP ERROR VECTOR
644 001634 012737 000001 001454      MOV  #1,LSI      ;;ASSUME IT'S AN LSI
645 001642 005737 177776          TST  PS          ;;TRY TO ACCESS PSW
646 001646 005037 001454          CLR  LSI        ;;NOT AN LSI
647 001652 000403          BR   75$
648 001654 012716 001662      70$: MOV  #75$,(SP)    ;;SET UP RETURN
649 001660 000002          RTI
650
651 001662 012637 000004      75$: MOV  (SP)+,@#ERRVEC ;;RESTORE ERROR VECTOR
652 001666 012700 001364      MOV  #DRVWCR,R0  ;;SET UP REG ADRS POINTERS
653 001672 013701 001360      MOV  DRVADR,R1  ;;GET BASE ADRS
654

```

```

656
657 001676 010120          SETUP2: MOV    R1,(R0)+      ;LOAD EM
658 001700 062701 000002    ADD     #2,R1          ;
659 001704 022700 001374    CMP     #DRVDBR+2,R0   ;ALL DONE?
660 001710 001372          BNE     SETUP2         ;BR IF NOT
661 001712 012700 001374    MOV     #DRVCT0,R0     ;SET UP DAV11 VECTOR ADRS POINTER
662 001716 013701 001362    MOV     DRVECT,R1     ;GET BASE VECTOR ADRS
663
664 001722 010120          SETUP3: MOV    R1,(R0)+      ;
665 001724 062701 000002    ADD     #2,R1          ;
666 001730 022700 001400    CMP     #DRVCT2+2,R0   ;ALL DONE?
667 001734 001372          BNE     SETUP3         ;BR IF NOT
668
669          ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
670 001736 005227 177777    INC     #-1            ;;FIRST TIME?
671 001742 001007          BNE     67$            ;;BRANCH IF NO
672 001744 104400 010142    TYPE   ,TITLE         ;;TYPE PROGRAM TITLE
673
674          ;;GET VALUE FOR SOFTWARE SWITCH REGISTER
675 001750 023727 001140 000176  CMP     SWR,#SWREG     ;;SOFTWARE SWITCH REG SELECTED?
676 001756 001001          BNE     67$            ;;BRANCH IF NO
677 001760 104405          GTSWR                ;;GET SOFT-SWR SETTINGS
678
679 001762 012706 001100          67$:  MOV     #STACK,SP   ;ALWAYS RESET STACK PTR
680 001766 000005          RESET                ;INITIALIZE DAV11 BEFORE TESTING
681 001770 012737 000001 001422  MOV     #1,ICOUNT     ;1ST TIME DO ALL TEST ONCE - THEN 3000(8)
682

```

```

684      .SBTTL MASTER TESTS
685
686 001776 004737 010072 SYNCST: JSR PC,CPUHI ;NO INTERRUPTS
687 002002 005737 001420          TST  MSTER ;STARTED AS MASTER?
688 002006 001402          BEQ  MINIT1 ;YES: INITIALIZE FOR MASTER TESTS
689 002010 000137 003656          JMP  SINIT1 ;NO: INITIALIZE FOR SLAVE
690
691      ;;*****
692      ;MASTER PROTOCOL INIT-TEST 1
693      ;;*****
694
695 002014 004737 010072 MINIT1: JSR PC,CPUHI ;NO INTERRUPTS
696 002020 012706 001100          MOV  #STACK,SP ;RESET STACK POINTER
697 002024 005077 177340          CLR  @DRVCSR ;CLEAR CSR
698 002030 012737 000010 001406    MOV  #10,ABORT ;INITIALIZE LOOP COUNT
699 002036 005037 001404          1$: CLR  TIME ;INITIALIZE SUBLOOP COUNT
700
701 002042 032777 001000 177320    2$: BIT  #DSTATC,@DRVCSR ;TEST FOR SLAVE SYNC
702 002050 001013          BNE  MTST1 ;YES: START TEST 1
703 002052 005337 001404          DEC  TIME ;DECR SUBLOOP COUNT
704 002056 001371          BNE  2$ ;CONTINUE SUBLOOP UNTIL ZERO
705 002060 005337 001406          DEC  ABORT ;DEC LOOP COUNT
706 002064 001364          BNE  1$ ;CONTINUE LOOP INTIL ZERO
707
708 002066 017737 177276 001414    MOV  @DRVCSR,$ERRST ;SAVE STATUS
709 002074 104016          ERROR+16 ;NOTIFY OPERATOR OF SYNC ERROR
710 002076 104411          RSYNC ;AND RESYNC
711
  
```

```

713 .SBTTL MTST1 TEST THAT SLAVE CAN ECHO THE DBR CORRECTLY
714 :*****
715 :MASTER COMPUTER STARTS HERE FROM PROGRAM START
716 :IT SENDS OUT SINGLE WORDS (FLOATING I/O PTRN) THRU THE
717 :DBR TO THE SLAVE COMPUTER
718 :IT LOOKS FOR THE SLAVE TO ECHO THE DBR WORD CORRECTLY
719 :WITHIN A CERTAIN AMOUNT OF TIME
720 :IF IT FAILS TO RETURN THE WORD AN ERROR IS REPORTED
721 :THIS TEST IS NOT EXITED UNTIL ALL DATA PATTERNS HAVE
722 :BEEN SENT AND RECEIVED CORRECTLY
723 :*****
724 :*****
725 002100 005001 MTST1: CLR R1 ;R1 SAYS FLOAT 0 LEFT WHEN ZERO
726 002102 012700 177776 MOV #-2,R0 ;START WITH #177776 IN R0
727
728 002106 010077 177260 1$: MOV R0,@DRVDBR ;SEND PATTERN OUT
729 002112 010037 001124 MOV R0,$GDDAT ;SAVE IN EXPECTED
730 002116 013737 001372 001122 MOV DRVDBR,$BDADR ;SET UP DBR ADRS
731
732 :*****
733 :START TEST SYNCHRONIZATION
734 :*****
735
736 002124 012777 000002 177236 2$: MOV #FNCT1,@DRVCSR ;TELL SLAVE DATA READY
737
738 002132 012704 001000 MOV #DSTATC,R4 ;LOAD MASK
739 002136 004737 007670 JSR PC,WTNE ;WAIT FOR SLAVE TO ECHO DATA
740
741 002142 004737 005574 3$: JSR PC,DELAY ;WAIT FOR SLAVE
742 002146 017737 177220 001126 MOV @DRVDBR,$BDDAT ;READ IT BACK
743 002154 023737 001124 001126 CMP $GDDAT,$BDDAT ;CORRECT?
744 002162 001402 BEQ 4$ ;BR IF SO
745
746 002164 104001 ERROR+1 ;SLAVE FAILED TO ECHO THE DBR WORD
747 002166 000406 BR 5$ ;LOOP ON ERROR ALWAYS
748
749 002170 005100 4$: COM R0 ;CONVERT PTRN TO FLOATING 1
750 002172 005101 COM R1 ;TIME TO FLOAT LEFT?
751 002174 001003 BNE 5$ ;BR IF NOT
752 002176 006300 ASL R0 ;YES - FLOAT LEFT
753 002200 005200 INC R0 ;KEEP LSB SET
754 002202 103010 BCC 6$ ;BR IF ZERO TO CARRY
755

```

```

757 ;*****
758 ;CLEAR STATUS SYNCHRONIZATION
759 ;*****
760
761 002204 042777 000002 177156 5$: BIC #FNCT1,@DRVCSR ;TELL SLAVE DATA HAS BEEN READ
762
763 002212 012704 001000 MOV #DSTATC,R4 ;LOAD MASK
764 002216 004737 007726 JSR PC,WTEQ ;WAIT FOR SLAVE TO ECHO STATUS CLEAR
765
766 002222 000731 BR 1$ ;CONTINUE WITH NEXT DATA PATTERN
767
768 ;*****
769 ;END OF TEST SYNCHRONIZATION
770 ;*****
771
772 002224 042777 000002 177136 6$: BIC #FNCT1,@DRVCSR ;CLEAR STATUS
773
774 002232 012704 001000 MOV #DSTATC,R4 ;LOAD MASK
775 002236 004737 007726 JSR PC,WTEQ ;WAIT FOR SLAVE TO ECHO STATUS CLEAR
776
777 ;*****
778 ;READ TERMINATOR SYNCHRONIZATION
779 ;*****
780
781 002242 012777 177001 177122 MOV #177001,@DRVDBR ;SET TERMINATOR
782 002250 052777 000002 177112 7$: BIS #FNCT1,@DRVCSR ;TELL SLAVE TO READ DATA
783
784 002256 012704 001000 MOV #DSTATC,R4 ;LOAD MASK
785 002262 004737 007670 JSR PC,WTNE ;WAIT FOR SLAVE TO ECHO DATA READ
786
787 002266 000240 8$: NOP
788 002270 000240 NOP
789

```



```

791 .SBTTL MTST2 TEST THAT SLAVE CAN ECHO THE 'STAT' BITS CORRECTLY
792 :*****
793 :*****
794 :MASTER SENDS OUT A 'FNCT' CODE (1-7) TO THE SLAVE COMPUTER
795 :THE MASTER THEN LOOKS FOR THE SLAVE TO ECHO THE CODE VIA THE
796 :'STAT' BITS WITHIN A CERTAIN AMOUNT OF TIME
797 :IF IT FAILS TO RETURN THE CORRECT CODE AN ERROR IS REPORTED
798 :THIS TEST IS NOT EXITED UNTIL ALL 'FCNT' CODES HAVE BEEN
799 :SENT AND RECEIVED CORRECTLY
      :*****
      :*****

800
801 002272 MTST2:
802 002272 013737 001370 001122      MOV      DRVCSR,$BADDR      ;SET UP CSR ADRS
803 002300 012700 000002              MOV      #2,R0              ;SET UP INITIAL FCNT BIT COUNT
804 002304 012737 001202 001124      MOV      #1202,$GDDAT      ;LOAD EXPECTED
805
806 :*****
807 :                               ;START TEST SYNCHRONIZE
808 :*****
809
810 002312 005077 177052      1$:      CLR      @DRVCSR      ;CLEAR STATUS
811
812 002316 012704 007000              MOV      #7000,R4          ;LOAD MASK
813 002322 004737 007726              JSR      PC,WTEQ           ;WAIT FOR SLAVE TO ECHO STATUS CLEAR
814
815 :*****
816 :                               ;TRANSFER FUNCTION/STATUS SYNCHRONIZE
817 :*****
818
819 002326 010077 177036      2$:      MOV      R0,@DRVCSR      ;SEND FNCT BITS TO SLAVE
820
821 002332 012704 007000              MOV      #7000,R4          ;LOAD MASK
822 002336 004737 007670              JSR      PC,WTNE           ;WAIT FOR SLAVE TO ECHO FNCT BITS
823
824 :
825 :                               PROCESS STATUS
826 002342 017737 177022 001126      3$:      MOV      @DRVCSR,$BDDAT ;READ THE CSR
827 002350 023737 001124 001126      CMP      $GDDAT,$BDDAT    ;CORRECT?
828 002356 001402              BEQ      4$               ;BR IF SO
829
830 002360 104002              ERROR+2
831 002362 000743              BR       MTST2            ;SLAVE FAILED TO ECHO 'STAT' BITS
832 :                               ;LOOP ON ERROR ALWAYS
833 002364 062737 001002 001124      4$:      ADD      #1002,$GDDAT     ;ADVANCE EXPECTED
834 002372 062700 000002              ADD      #2,R0             ;ADVANCE PTRN
835 002376 032700 000020              BIT      #20,R0           ;ALL BEEN DONE?
836 002402 001743              BEQ      1$               ;BR IF NOT
837

```

```

839 ;*****
840 ;END OF TEST SYNCHRONIZATION
841 ;*****
842
843 002404 012777 177002 176760 5$: MOV #177002,@DRVDBR ;SET TERMINATOR
844 002412 005077 176752 CLR @DRVCSR ;CLEAR STATUS
845
846 002416 012704 007050 MOV #7000,R4 ;LOAD MASK
847 002422 004737 007726 JSR PC,WTEQ ;WAIT FOR SLAVE TO ECHO STATUS CLEAR
848
849 ;*****
850 ;SET INTERRUPT TO CLEAR INTERRUPT
851 ;AND ERROR LATCHED BY THIS TEST. (DR11-B)
852 ;*****
853
854 002426 005737 001454 6$: TST LSI ;AN LSI?
855 002432 001021 BNE 8$ ;IF YES
856
857 002434 004537 005534 JSR R5,SETVEC ;SET VECTOR FOR
858 002440 005666 DRVISR ;SERVICE ROUTINE
859
860 002442 004737 010116 7$: JSR PC,CPULO ;ALLOW INTERRUPTS
861 002446 012777 177777 176710 MOV #-1,@DRVWCR ;LOAD WORD COUNT
862 002454 012777 000501 176706 MOV #501,@DRVCSR ;CAUSE INTERRUPT AT SLAVE
863 002462 004737 005574 JSR PC,DELAY ;WAIT FOR POSSIBLE INTERRUPT
864
865 002466 004737 010072 JSR PC,CPUHI ;DISABLE INTERRUPTS
866 002472 004737 005554 JSR PC,RSTVEC ;RESTORE VECTOR
867
868 002476 000240 8$: NOP
869 002500 000240 NOP
870

```

```

872 .SBTTL MTST3 TEST THAT MASTER CAN XMIT A 32 WORD DATA BLOCK TO SLAVE
873 :*****
874 :MASTER XMIT A 32 WORD BLOCK OF DATA TO SLAVE
875 :THEN CHECKS FOR PROPER INTERRUPT STATUS, WC & BA
876 :THE SLAVE CHECKS THE SAME PLUS THE DATA RECEIVED
877 :THE TEST DOES NOT ADVANCE UNTIL A SUCCESSFUL XFER
878 :*****
879
880 002502 004737 010072 M3INI: JSR PC,CPUHI ;NO INTERRUPTS
881 002506 004537 005534 JSR R5,SETVEC ;SET VECTOR FOR
882 002512 005666 DRVISR ;SERVICE ROUTINE
883
884 002514 005037 001456 MTST3: CLR ERRFLG ;CLEAR ERRORS
885 002520 005077 176644 CLR @DRVCSR ;CLEAR STATUS
886 002524 005037 001402 CLR INTFLG ;CLEAR INTR FLAG
887 002530 004737 005574 JSR PC,DELAY ;DELAY FOR SLAVE
888
889 002534 004537 005624 1$: JSR R5,LDBUF ;GO LOAD 'DBUF' WITH DATA PTRM
890 002540 177740 -32. ;DO 32. LOCATIONS
891 002542 012737 100000 001404 MOV #100000,TIME ;SET UP A TIMER VALUE
892 002550 013777 001424 176606 MOV XPRAM,@DRVWCR ;SET UP WORD COUNT
893 002556 013777 001426 176602 MOV XPRAM+2,@DRVBAR ;SET UP BUFFER ADRS
894 002564 004737 010116 JSR PC,CPULO ;ALLOW INTERRUPTS
895 002570 013777 001430 176572 MOV XPRAM+4,@DRVCSR ;SET UP CONTROL: IE, FNCT1 & GO(CLRS CYCLE)
896
897 002576 005737 001402 2$: TST INTFLG ;TEST FOR INTERRUPT
898 002602 001023 BNE 3$ ;IF NE YES
899 002604 005337 001404 DEC TIME ;WAIT FOR INTR
900 002610 001372 BNE 2$ ;UNTIL 0
901
902 ; PROCESS TRANSFER ERROR
903
904 002612 004737 010072 20$: JSR PC,CPUHI ;DISABLE INTERRUPTS
905 002616 017737 176546 001126 MOV @DRVCSR,$BDDAT ;READ CSR
906 002624 042777 000100 176536 BIC #100,@DRVCSR ;DISABLE IE
907 002632 012737 002702 001124 MOV #2702,$GDDAT ;LOAD EXPECTED STATUS:
908 ;STAT B,CYCLE,RDY,IE.FNCT1
909 002640 013737 001370 001122 MOV DRVCSR,$BDADR ;SET UP CSR ADRS
910 002646 104003 ERROR+3 ;DATI FAILED TO CAUSE A WC INTR
911 002650 000466 BR 6$ ;GO RESYNC ON ERROR
912
  
```

```

914 ; PROCESS TRANSFER
915
916 002652 004737 010072 3$: JSR PC,CPUHI ;DISABLE INTERRUPTS
917 002656 017737 176506 001126 MOV @DRVCSR,$BDDAT ;READ STATUS
918 002664 042777 000100 176476 BIC #100,@DRVCSR ;DISABLE IE
919 002672 012737 002702 001124 MOV #2702,$GDDAT ;LOAD EXPECTED STATUS:
920 ;STAT B,,YCLE,RDY,IE,FNCT1
921 002700 023737 001124 001126 CMP $GDDAT,$BDDAT ;CORRECT?
922 002706 001405 BEQ 4$ ;BR IF SO
923
924 002710 013737 001370 001122 MOV DRVCSR,$BDADR ;SET UP CSR ADRS
925 002716 104004 ERROR+4 ;DATI STATUS ERROR
926 002720 000442 BR 6$ ;GO RESYNC ON ERROR
927
928 002722 005037 001124 4$: CLR $GDDAT ;LOAD EXPECTED WC
929 002726 017737 176432 001126 MOV @DRVWCR,$BDDAT ;READ WORD COUNT
930 002734 001405 BEQ 5$ ;BR IF ZERO
931
932 002736 013737 001364 001122 MOV DRVWCR,$BDADR ;SET UP WCR ADRS
933 002744 104005 ERROR+5 ;DATI WORD COUNT ERROR
934 002746 000427 BR 6$ ;GO RESYNC ON ERROR
935
936 002750 013737 001426 001124 5$: MOV XPRAM+2,$GDDAT ;GET STARTING ADRS OF XFER
937 002756 013700 001424 MOV XPRAM,R0 ;GET WC
938 002762 005400 NEG R0 ;GET ACTUAL #
939 002764 006300 ASL R0 ;CONVERT TO WORD
940 002766 060037 001124 ADD R0,$GDDAT ;ADD TO BASE ADRS
941 002772 017737 176370 001126 MOV @DRVBAR,$BDDAT ;READ BAR ADRS
942 003000 042737 000001 001126 BIC #BIT00,$BDDAT ;ELIMINATE LSB
943 003006 023737 001124 001126 CMP $GDDAT,$BDDAT ;CORRECT?
944 003014 001406 BEQ 7$ ;BR IF SO
945
946 003016 013737 001366 001122 MOV DRVBAR,$BDADR ;SET UP BAR ADRS
947 003024 104006 ERROR+6 ;DATI BUFFER ADRS ERROR
948
949 003026 005237 001456 6$: INC ERRFLG ;SET ERROR FLAG
950 003032 004737 005574 7$: JSR PC,DELAY ;WAIT FOR POSSIBLE LSI REFRESH
951
    
```

```

953 ;*****
954 ;END OF TRANSFER SYNCHRONIZATION
955 ;*****
956
957 003036 005077 176326 10$: CLR @DRVCSR ;CLEAR CSR STATUS
958
959 003042 012704 007000 MOV #7000,R4 ;LOAD MASK
960 003046 004737 007726 JSR PC,WTEQ ;WAIT FOR SLAVE TO ECHO STATUS CLEAR
961
962 003052 005737 001456 TST ERRFLG ;ERROR?
963 003056 001404 BEQ 11$ ;IF EQ NO ERROR
964 003060 012777 000004 176302 MOV #FNCT2,@DRVCSR ;ERROR: SET FNCT2
965 003066 000403 BR 12$
966 003070 012777 000002 176272 11$: MOV #FNCT1,@DRVCSR ;NO ERROR: SET FNCT1
967
968 ;*****
969 ;TRANSFER STATUS SYNCHRONIZATION
970 ;*****
971
972 003076 012704 007000 12$: MOV #7000,R4 ;LOAD MASK
973 003102 004737 007670 JSR PC,WTNE ;WAIT FOR SLAVE TO ECHO TRANSFER STATUS
974
975 003106 017737 176256 001416 MOV @DRVCSR,SAVE ;GET CSR VALUE
976 003114 032737 002000 001416 BIT #DSTATB,SAVE ;ERROR?
977 003122 001402 BEQ 13$ ;IF EQ NO
978 003124 005237 001456 INC ERRFLG ;SET ERROR FLAG
979
980 ;*****
981 ;CLEAR STATUS SYNCHRONIZATION
982 ;*****
983
984 003130 005077 176234 13$: CLR @DRVCSR ;CLEAT STATUS
985
986 003134 012704 007000 MOV #7000,R4 ;LOAD MASK
987 003140 004737 007726 JSR PC,WTEQ ;WAIT FOR SLAVE TO ECHO STATUS CLEAR
988
989 ; LOOP ON TEST IF ERROR DETECTED
990
991 003144 005737 001456 TST ERRFLG ;ANY ERRORS?
992 003150 001402 BEQ 14$ ;IF NOT
993
994 003152 000137 002514 JMP MTST3 ;RESTART TEST
995
996 003156 004737 005554 14$: JSR PC,RSTVEC ;GO RESTORE VECTOR
997 003162 000240 NOP
998 003164 000240 NOP
999

```

```

1001      .SBTTL MTST4 SET SLAVE ATTN INTERRUPT
1002      :*****
1003      :*****
1003      :MASTER XMITA A 32 WORD BLOCK TO SLAVE.
1004      :SLAVE SET FOR LARGER TRANSFER.
1005      :      MASTER CHECKS NORMAL COMPLETION.
1006      :MASTER SETS ATTN INTR WITH FNCT3.
1007      :      SLAVE CHECKS FOR TERMINATION.
1008      :*****
1009      :*****
1010      M4INI: JSR      PC,CPUHI      ;DISABLE INTERRUPTS
1011      JSR      R5,SETVEC      ;SET VECTOR FOR
1012      DRVISR      ;SERVICE ROUTINE
1013
1014      MTST4: CLR      ERRFLG      ;CLEAR ERRORS
1015      CLR      @DRVCSR      ;CLEAR STATUS
1016      CLR      INTFLG      ;CLEAR INTR FLAG
1017      JSR      PC,DELAY      ;DELAY FOR SLAVE
1018
1019      1$:  MOV      #100000,TIME      ;SET TIMER
1020      MOV      XPRAM4,@DRVWCR      ;SET WORD COUNT
1021      MOV      XPRAM4+2,@DRVBAR      ;SET BUFFER ADDRESS
1022
1023      :      INITIATE TRANSFER AND WAIT FOR INTERRUPT
1024
1025      JSR      PC,CPULO      ;ALLOW INTERRUPTS
1026      MOV      XPRAM4+4,@DRVCSR      ;SET IE, FNCT1, GO
1027
1028      2$:  TST      INTFLG      ;TEST FOR INTERRUPT
1029      BNE      3$      ;IF NE YES
1030      DEC      TIME      ;WAIT FOR INTR
1031      BNE      2$      ;UNTIL 0
1032
1033      :      PROCESS INTERRUPT ERROR
1034
1035      20$: JSR      PC,CPUHI      ;DISABLE INTERRUPTS
1036      MOV      @DRVCSR,$BDDAT      ;READ CSR
1037      BIC      #100,@DRVCSR      ;DISABLE IE
1038      MOV      #2702,$GDDAT      ;LOAD EXPECTED STATUS:
1039      ;STAT B,CYCLE,RDY,IE, FNCT1
1040      MOV      DRVCSR,$BDADR      ;SET UP CSR ADRS
1041      ERROR+3      ;DATI FAILED TO CAUSE A WC INTR
1042      BR      6$      ;GO RESYNC ON ERROR

```

```

1044 ; PROCESS DATA TRANSFER
1045
1046 003330 004737 010072 3$: JSR PC,CPUHI ;DISABLE INTERRUPTS
1047 003334 017737 176030 001126 MOV @DRVCSR,$BDDAT ;READ STATUS
1048 003342 042777 000100 176020 BIC #100,@DRVCSR ;DISABLE IE
1049 003350 012737 002702 001124 MOV #2702,$GDDAT ;LOAD EXPECTED STATUS:
1050 ;STAT B,CYCLE,RDY,IE,FNCT1
1051 003356 023737 001124 001126 CMP $GDDAT,$BDDAT ;CORRECT?
1052 003364 001405 BEQ 4$ ;BR IF SO
1053 003366 013737 001370 001122 MOV DRVCSR,$BDADR ;SET UP CSR ADRS
1054 003374 104004 ERROR+4 ;DATI STATUS ERROR
1055 003376 000442 BR 6$ ;GO RESYNC ON ERROR
1056
1057 003400 005037 001124 4$: CLR $GDDAT ;LOAD EXPECTED WC
1058 003404 017737 175754 001126 MOV @DRVWCR,$BDDAT ;READ WORD COUNT
1059 003412 001405 BEQ 5$ ;BR IF ZERO
1060 003414 013737 001364 001122 MOV DRVWCR,$BDADR ;SET UP WCR ADRS
1061 003422 104005 ERROR+5 ;DATI WORD COUNT ERROR
1062 003424 000427 BR 6$ ;GO RESYNC ON ER
1063
1064 003426 013737 001426 001124 5$: MOV XPRAM+2,$GDDAT ;GET STARTING ADRS OF XFER
1065 003434 013700 001424 MOV XPRAM,R0 ;GET WC
1066 003440 005400 NEG R0 ;GET ACTUAL #
1067 003442 006300 ASL R0 ;CONVERT TO WORD
1068 003444 060037 001124 ADD R0,$GDDAT ;ADD TO BASE ADRS
1069 003450 017737 175712 001126 MOV @DRVBAR,$BDDAT ;READ BAR ADRS
1070 003456 042737 000001 001126 BIC #BIT00,$BDDAT ;ELIMINATE LSB
1071 003464 023737 001124 001126 CMP $GDDAT,$BDDAT ;CORRECT?
1072 003472 001406 BEQ 7$ ;BR IF SO
1073 003474 013737 001366 001122 MOV DRVBAR,$BDADR ;SET UP BAR ADRS
1074 003502 104006 ERROR+6 ;DATI BUFFER ADRS ERROR
1075
1076 003504 005237 001456 6$: INC ERRFLG ;SET ERROR FLAG
1077 003510 052777 000010 175652 7$: BIS #FNCT3,@DRVCSR ;SET ATTENTION
1078
    
```

```

1080 ;*****
1081 ;END OF TRANSFER SYNCHRONIZATION
1082 ;*****
1083
1084 003516 004737 005574 10$: JSR PC,DELAY ;DELAY FOR SLAVE
1085 003522 005077 175642 CLR @DRVCSR ;CLEAR CSR STATUS
1086
1087 003526 012704 007000 MOV #7000,R4 ;LOAD MASK
1088 003532 004737 007726 JSR PC,WTEQ ;WAIT FOR SLAVE TO ECHO STATUS CLEAR
1089
1090 003536 005737 001456 TST ERRFLG ;ERROR?
1091 003542 001404 BEQ 11$ ;IF EQ NO ERROR
1092 003544 012777 000004 175616 MOV #FNCT2,@DRVCSR ;ERROR: SET FNCT2
1093 003552 000403 BR 12$
1094 003554 012777 000002 175606 11$: MOV #FNCT1,@DRVCSR ;NO ERROR: SET FNCT1
1095
1096 ;*****
1097 ;TRANSFER STATUS SYNCHRONIZATION
1098 ;*****
1099
1100 003562 012704 007000 12$: MOV #7000,R4 ;LOAD MASK
1101 003566 004737 007670 JSR PC,WTNE ;WAIT FOR SLAVE TO ECHO TRANSFER STATUS
1102
1103 003572 017737 175572 001416 MOV @DRVCSR,SAVE ;GET CSR VALUE
1104 003600 032737 002000 001416 BIT #DSTATB,SAVE ;ERROR?
1105 003606 001402 BEQ 13$ ;IF EQ NO
1106 003610 005237 001456 INC ERRFLG ;SET ERROR FLAG
1107
1108 ;*****
1109 ;CLEAR STATUS SYNCHRONIZATION
1110 ;*****
1111
1112 003614 005077 175550 13$: CLR @DRVCSR ;CLEAT STATUS
1113
1114 003620 012704 007000 MOV #7000,R4 ;LOAD MASK
1115 003624 004737 007726 JSR PC,WTEQ ;WAIT FOR SLAVE TO ECHO STATUS CLEAR
1116
1117 ; LOOP ON TEST IF ERROR DETECTED
1118
1119 003630 005737 001456 TST ERRFLG ;ANY ERRORS?
1120 003634 001402 BEQ 14$ ;IF NOT
1121
1122 003636 000137 003200 JMP MTST4 ;RESTART TEST
1123
1124 003642 004737 005554 14$: JSR PC,RSTVEC ;GO RESTORE VECTOR
1125 003646 000137 003734 JMP STST1 ;PROCEED TO SLAVE TEST 1
1126 003652 000240 NOP
1127 003654 000240 NOP
1128
    
```



```

1130 .SBTTL SLAVE TESTS
1131 ;:*****
1132 ;SLAVE PROTOCOL INIT-TEST 1
1133 ;:*****
1134
1135 ;* NOTE: THIS SEGMENT EXECUTED ONLY SLAVE START OR RESYNC
1136
1137 003656 004737 010072 SINIT1: JSR PC,CPUHI ;NO INTERRUPTS
1138 003662 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
1139 003666 005077 175476 CLR @DRVCSR ;CLEAR CSR
1140 003672 032777 001000 175470 BIT #DSTATC,@DRVCSR ;TEST DSTATC
1141 003700 001404 BEQ 2$ ;CLEAR: WAIT FOR MASTER
1142
1143 003702 032777 001000 175460 1$: BIT #DSTATC,@DRVCSR ;WAIT FOR CLEAR
1144 003710 001374 BNE 1$ ;
1145
1146 003712 012777 000002 175450 2$: MOV #FNCT1,@DRVCSR ;TELL MASTER TO GO AHEAD
1147 003720 032777 001000 175442 3$: BIT #DSTATC,@DRVCSR ;DID MASTER SET DSTATC?
1148 003726 001774 BEQ 3$ ;NO: WAIT
1149
1150 003730 000137 003750 JMP SL1STR ;DBR DATA IS READY
1151
    
```

```

1153 .SBTTL STST1 RECEIVED MASTER'S DBR DATA/SEND BACK VIA DBR
1154 :*****
1155 :NOW THIS COMPUTER BECOMES THE SLAVE AND ECHOS MASTER'S DBR DATA
1156 :SLAVE COMPUTER STARTS HERE FROM PROGRAM START 204
1157 :*****

1158
1159 003734 005077 175430 STST1: CLR @DRVCSR ;CLEAR FNCT BIT
1160
1161 :*****
1162 :DBR TRANSFER SYNCHRONIZE
1163 :*****
1164
1165 003740 012704 001000 2$: MOV #DSTATC,R4 ;LOAD MASK
1166 003744 004737 007670 JSR PC,WTNE ;WAIT FOR MASTER TO LOAD DBR
1167
1168 ; READ AND PROCESS DBR DATA
1169
1170 003750 017737 175416 001416 SL1STR: MOV @DRVDBR,SAVE ;GET DATA
1171 003756 013777 001416 175406 MOV SAVE,@DRVDBR ;SEND IT BACK
1172 003764 022737 177001 001416 CMP #177001,SAVE ;TEST TERMINATOR?
1173 003772 001410 BEQ 4$ ;IF EQ YES
1174
1175 :*****
1176 :DBR ECHO SYNCHRONIZATION
1177 :*****
1178
1179 003774 052777 000002 175366 3$: BIS #FNCT1,@DRVCSR ;TELL MASTER DONE
1180
1181 004002 012704 001000 MOV #DSTATC,R4 ;LOAD MASK
1182 004006 004737 007726 JSR PC,WTEQ ;WAIT FOR MASTER TO RED DBR
1183
1184 004012 000750 BR STST1 ;GO LOOK FOR MORE DATA
1185
1186 :*****
1187 :END OF TEST SYNCHRONIZATION
1188 :*****
1189
1190 004014 052777 000002 175346 4$: BIS #FNCT1,@DRVCSR ;TELL MASTER DONE
1191 004022 000240 NOP ;
1192 004024 000240 NOP
1193
1194
  
```

```

1196 .SBTTL STST2 RECEIVE MASTER'S 'STAT' BITS/SEND BACK VIA 'FNCT' BITS
1197 :*****
1198 :RECEIVE 'STAT' BITS AND CONVERT TO 'FNCT' BITS AND WRITE TO CSR
1199 :*****

1200 STST2: JSR PC,CPUHI ;DISABLE INTERRUPTS
1201 004026 004737 010072
1202 :*****
1203 :TEST START SYNCHRONIZE
1204 :*****
1205
1206 1$: MOV #7000,R4 ;LOAD MASK
1207 004032 012704 007000 JSR PC,WTEQ ;WAIT FOR MASTER TO START TEST
1208 004036 004737 007726
1209 CLR @DRVCSR ;ECHO MASTER TEST START
1210 004042 005077 175322
1211 : CHECK FOR END OF TEST FLAG
1212
1213 2$: CMP #177002,@DRVDBR ;TERMINATOR?
1214 004046 022777 177002 175316 BEQ 5$ ;IF EQ YES
1215 004054 001416
1216 :*****
1217 :TRANSMIT STATUS SYNCHRONIZATION
1218 :*****
1219
1220 3$: MOV #7000,R4 ;LOAD MASK
1221 004056 012704 007000 JSR PC,WTNE ;WAIT FOR MASTER TO SEND STATUS
1222 004062 004737 007670
1223 MOV @DRVCSR,SAVE ;READ THE CSR
1224 004066 017737 175276 001416 BIC #170777,SAVE ;SAVE ONLY THE STAT BITS
1225 004074 042737 170777 001416
1226 MOV SAVE+1,@DRVCSR ;ECHO WITH FNCT BITS
1227 004102 113777 001417 175260 BR 1$
1228 004110 000750
1229

```

```

1231
1232
1233
1234
1235
1236 004112 005737 001454
1237 004116 001021
1238 004120 004537 005534
1239 004124 005666
1240
1241 004126 004737 010116
1242 004132 012777 177777 175224
1243 004140 012777 000501 175222
1244 004146 004737 005574
1245
1246 004152 004737 010072
1247 004156 004737 005554
1248
1249 004162 000240
1250 004164 000240
1251

```

```

:*****
:SET INTERRUPT TO CLEAR INTERRUPT
:AND ERROR LATCHED BY THIS TEST. (DR11-B)
:*****
5$:   TST     LSI           ;AN LSI?
      BNE     7$           ;IF YES
      JSR     R5,SETVEC   ;SET VECTOR FOR
      DRVISR          ;SERVICE ROUTINE
6$:   JSR     PC,CPULO     ;ALLOW INTERRUPTS
      MOV     #-1,@DRVWCR ;LOAD WORD COUNT
      MOV     #501,@DRVCSR ;CAUSE INTERRUPT IN MASTER
      JSR     PC,DELAY    ;WAIT FOR POSSIBLE INTERRUPT
7$:   JSR     PC,CPUHI    ;DISABLE INTERRUPTS
      JSR     PC,RSTVEC   ;RESTORE VECTOR
      NOP
      NOP

```

```

1253 .SBTTL STST3 RECEIVE A 32 WORD BLOCK/MAKE STATUS & DATA CHECKS
1254 :*****
1255 :*****
1256 :THIS TEST SETS UP TO RECEIVE A 32 WORD BLOCK OF DATA FROM THE MASTER
1257 :THEN CHECKS FOR PROPER INTERRUPT STATUS,WC,BA & DATA
1258 :IF AN ERROR IS DETECTED THE SLAVE TELLS THE MASTER, REPORTS THE
1259 :ERROR,AND RESYNCS ON TEST
1260 :*****
1261 :*****
1262 ; INITIALIZE TRAP VECTOR
1263 S3INI: JSR PC,CPUHI ;DISABLE INTERRUPTS
1264 JSR R5,SETVEC ;SET VECTOR FOR
1265 DRVISR ;SERVICE ROUTINE
1266
1267 ; CLEAR/INITIALIZE TEST PARAMETERS
1268 STST3: CLR ERRFLG ;CLEAR ERRORS
1269 CLR @DRVCSR ;CLEAR STATUS
1270 CLR INTFLG ;CLEAR INTR FLAG
1271
1272 JSR R5,CLRBUF ;GO CLR 'DBUF'
1273 -32. ;DO 32, LOCATIONS
1274
1275 MOV #100C00,TIME ;SET UP A TIMER VALUE
1276 MOV RPRAM,@DRVWCR ;SET UP WORD COUNT
1277 MOV RPRAM+2,@DRVBAR ;SET UP BUFFER ADRS
1278 JSR PC,CPULO ;ALLOW INTERRUPTS
1279
1280 :*****
1281 ;TRANSFER SYNCHRONIZATION
1282 :*****
1283
1284 1$: MOV #DSTATC,R4 ;LOAD MASK
1285 JSR PC,WTNE ;WAIT FOR MASTER TO INITIATE TRANSFER
1286 MOV RPRAM+4,@DRVCSR ;SET UP CONTROL- IE, FNCT2,GO
1287
1288 2$: TST INTFLG ;TEST FOR INTERRUPT
1289 BNE 3$ ;IF NE YES
1290 DEC TIME ;WAIT FOR INTERRUPT
1291 BNE 2$ ;UNTIL ZERO
1292
1293 ; PROCESS INTERRUPT ERROR
1294
1295 20$: JSR PC,CPUHI ;DISABLE INTERRUPTS
1296 MOV @DRVCSR,$BDDAT ;READ CSR
1297 BIC #100,@DRVCSR ;DISABLE IE
1298 MOV #1304,$GDDAT ;LOAD EXPECTED STATUS:
1299 ;STATC,RDY,IE & FNCT 2
1300 MOV DRVCSR,$BDADR ;SET UP CSR ADRS
1301 ERROR+7 ;DATO FAILED TO CAUSE A WC INTR
1302 BR 10$ ;GO RESYNC ON ERROR
1303
1304
  
```

```

1306 ; PROCESS TRANSFER
1307
1308 004342 004737 010072 3$: JSR PC,CPUHI ;DISABLE INTERRUPTS
1309 004346 017737 175016 001126 MOV @DRVCSR,$BDDAT ;READ STATUS
1310 004354 042777 000100 175006 BIC #100,@DRVCSR ;DISABLE IE
1311 004362 012737 001304 001124 MOV #1304,$GDDAT ;LOAD EXPECTED STATUS:
1312 ;STATC,RDY,IE & FNCT 2
1313 004370 023737 001124 001126 CMP $GDDAT,$BDDAT ;CORRECT?
1314 004376 001405 BEQ 4$ ;BR IF SO
1315 004400 013737 001370 001122 MOV DRVCSR,$BDADR ;SET UP CSR ADRS
1316 004406 104010 ERROR+10 ;DATO STATUS ERROR
1317 004410 000474 BR 10$ ;GO RESYNC ON ERROR
1318
1319 004412 005037 001124 4$: CLR $GDDAT ;LOAD EXPECTED WC
1320 004416 017737 174742 001126 MOV @DRVWCR,$BDDAT ;READ WCR
1321 004424 001405 BEQ 5$ ;BR IF SO
1322 004426 013737 001364 001122 MOV DRVWCR,$BDADR ;SET UP WCR ADRS
1323 004434 104011 ERROR+11 ;DATO WORD COUNT ERROR
1324 004436 000461 BR 10$ ;GO RESYNC ON ERROR
1325
1326 004440 013737 001434 001124 5$: MOV RPRAM+2,$GDDAT ;GET STARTING ADRS OF XFER
1327 004446 013700 001432 MOV RPRAM,R0 ;GET WC
1328 004452 005400 NEG R0 ;GET ACTUAL #
1329 004454 006300 ASL R0 ;CONVERT TO WORD
1330 004456 060037 001124 ADD R0,$GDDAT ;ADD TO BASE ADRS
1331 004462 017737 174700 001126 MOV @DRVBAR,$BDDAT ;READ BAR ADRS
1332 004470 042737 000001 001126 BIC #BIT00,$BDDAT ;ELIMINATE LSB
1333 004476 023737 001124 001126 CMP $GDDAT,$BDDAT ;CORRECT?
1334 004504 001405 BEQ 6$ ;BR IF SO
1335 004506 013737 001366 001122 MOV DRVBAR,$BDADR ;SET UP BAR ADRS
1336 004514 104012 ERROR+12 ;DATO BUFFER ADRS ERROR
1337 004516 000431 BR 10$ ;GO RESYNC ON ERROR
1338
1339 004520 013700 001434 6$: MOV RPRAM+2,R0 ;GET BUFFER ADRS
1340 004524 013701 001432 MOV RPRAM,R1 ;GET WORD COUNT
1341 004530 012702 177776 7$: MOV #177776,R2 ;GET 1ST DATA PTRN(FLOATING 0)
1342 004534 005003 CLR R3 ;R3 SAYS WHEN TO SHIFT PTRN
1343 004536 020220 8$: CMP R2,(R0)+ ;COMPARE DATA IN DBUF TO EXPECTED
1344 004540 001011 BNE 9$ ;BR IF DATA ERROR
1345 004542 005201 INC R1 ;COUNT THE WORD COUNT
1346 004544 001420 BEQ 110$ ;BR IF DATA CHECKS DONE
1347
1348 004546 005102 COM R2 ;CONVERT PTRN TO FLOATING 1
1349 004550 005103 COM R3 ;TIME TO SHIFT?
1350 004552 001371 BNE 8$ ;BR IF NOT - GO CK NEXT
1351 004554 006302 ASL R2 ;FLOAT PTRN LEFT
1352 004556 005202 INC R2 ;KEEP LSB SET
1353 004560 103363 BCC 7$ ;GO RESET FLOATING PTRN
1354 004562 000765 BR 8$ ;GO CHECK NEXT
1355

```

```

1357 ; PROCESS DATA ERROR
1358
1359 004564 014037 001126 9$: MOV -(R0), $BDDAT ;GET BAD DATA
1360 004570 010037 001122 MOV R0, $BDADR ;GET MEM ADRS OF DATA ER
1361 004574 010237 001124 MOV R2, $GDDAT ;LOAD EXPECTED DATA
1362 004600 104013 ERROR+13 ;DATO DATA ERROR
1363
1364 004602 005237 001456 10$: INC ERRFLG ;SET ERROR FLAG
1365
1366 ;*****
1367 ;END OF TRANSFER SYNCHRONIZATION
1368 ;*****
1369
1370 004606 012704 007000 110$: MOV #7000, R4 ;LOAD MASK
1371 004612 004737 007726 JSR PC, WTEQ ;WAIT FOR MASTER TO CLEAR CSR
1372
1373 004616 005077 174546 CLR @DRVCSR ;ECHO CSR CLEAR TO MASTER
1374
1375 ;*****
1376 ;TRANSFER STATUS SYNCHRONIZATION
1377 ;*****
1378
1379 004622 012704 007000 11$: MOV #7000, R4 ;LOAD MASK
1380 004626 004737 007670 JSR PC, WTNE ;WAIT FOR MASTER TO POST STATUS
1381 004632 017737 174532 001416 MOV @DRVCSR, SAVE ;READ MASTER STATUS
1382
1383 004640 005737 001456 TST ERRFLG ;TEST LOCAL ERROR
1384 004644 001404 BEQ 111$ ;IF EQ NO ERROR
1385 004646 012777 000004 174514 MOV #FNCT2, @DRVCSR ;ERROR: SET FNCT2
1386 004654 000411 BR 12$ ;CONTINUE
1387
1388 004656 012777 000002 174504 111$: MOV #FNCT1, @DRVCSR ;NO ERROR: SET FNCT1
1389 004664 032737 002000 001416 BIT #DSTATB, SAVE ;CHECK FOR MASTER ERROR
1390 004672 001402 BEQ 12$ ;IF EQ NO ERROR
1391 004674 005237 001456 INC ERRFLG ;YES, FLAG ERROR
1392
1393 ;*****
1394 ;CLEAR STATUS SYNCHRONIZATION
1395 ;*****
1396
1397 004700 012704 007000 12$: MOV #7000, R4 ;LOAD MASK
1398 004704 004737 007726 JSR PC, WTEQ ;WAIT FOR MASTER TO CLEAR TRANSFER STATUS
1399
1400 004710 005077 174454 CLR @DRVCSR ;ECHO MASTER CLEAR TRANSFER STATUS
1401
1402 004714 005737 001456 16$: TST ERRFLG ;ANY ERRORS?
1403 004720 001402 BEQ 17$ ;IF NOT
1404
1405 004722 000137 004200 JMP STST3 ;RESTART TEST
1406
1407 004726 004737 005554 17$: JSR PC, RSTVEC ;GO RESTORE VECTOR
1408 004732 000240 NOP
1409 004734 000240 NOP
1410

```

```

1412      .SBTTL  STST4  GET ATTN INTERRUPT
1413
1414      ;*****
1415      ;*****
1416      ;SETS UP TO RECEIVE A LARGER BUFFER THAN THE MASTER SENDS.
1417      ;MASTER ENDS XFER AND SENDS ATTN INTERRUPT.
1418      ;SLAVE CHECKS FOR ATTN TERMINATION.
1419      ;IF AN ERROR IS DETECTED THE SLAVE TELLS THE MASTER
1420      ;AND RESYNCHS ON TEST.
1421      ;IF ALL OK THEN THIS COMPUTER BECOMES MASTER
1422      ;AND GOES TO MTST1.
1423      ;*****
1424      ;*****
1425      ;
1426      $4INI:  INITIALIZE TRAP VECTOR
1427      JSR    PC,CPUHI      ;DISABLE INTERRUPTS
1428      JSR    R5,SETVEC    ;SET VECTOR FOR
1429      DRVISR                ;SERVICE ROUTINE
1430      ;
1431      STST4:  CLEAR/INITIALIZE TEST PARAMETERS
1432      CLR    ERRFLG        ;CLEAR ERRORS
1433      CLR    INTFLG        ;CLEAR INTERRUPT FLAG
1434      CLR    @DRVCSR      ;CLEAR STATUS
1435      ;
1436      JSR    R5,CLRBUF    ;CLEAR BUFFER
1437      -40.
1438      ;
1439      MOV    #140000,TIME  ;SET TIMER
1440      MOV    RPRAM4,@DRVWCR ;SET WORD COUNT
1441      MOV    RPRAM4+2,@DRVBAR ;SET BUFFER ADDRESS
1442      JSR    PC,CPULO     ;ALLOW INTERRUPTS
1443
1444      ;*****
1445      ;TRANSFER SYNCHRONIZATION
1446      ;*****
1447
1448      1$:  MOV    #7000,R4      ;LOAD MASK
1449      JSR    PC,WTNE        ;WAIT FOR MASTER TO INITIATE TRANSFER
1450      MOV    RPRAM4+4,@DRVCSR ;SET IE, FNCT2, GO
1451
1452      2$:  TST    INTFLG      ;TEST FOR INTERRUPT
1453      BNE   3$              ;IF NE YES
1454      DEC   TIME           ;WAIT FOR INTERRUPT
1455      BNE   2$              ;UNTIL ZERO
1456

```



```

1458 ; PROCESS INTERRUPT ERROR
1459
1460 005052 004737 010072 20$: JSR PC, CPUHI ;DISABLE INTERRUPTS
1461 005056 017737 174306 001126 MOV @DRVCSR, $BDDAT ;READ CSR
1462 005064 042777 000100 174276 BIC #100, @DRVCSR ;DISABLE IE
1463 005072 012737 005304 001124 MOV #5304, $GDDAT ;LOAD EXPECTED STATUS:
1464 ;STATA, STATC, RDY, IE & FNCT2
1465 005100 013737 001370 001122 MOV DRVCSR, $BDADR ;SET UP CSR ADRS
1466 005106 104014 ERROR+14 ;NO ATTN INTR
1467 005110 000442 BR 6$ ;GO RESYNC ON ERROR
1468
1469 005112 004737 010072 3$: JSR PC, CPUHI ;DISABLE INTERRUPTS
1470 005116 017737 174246 001126 MOV @DRVCSR, $BDDAT ;READ STATUS
1471 005124 042777 000100 174236 BIC #100, @DRVCSR ;DISABLE IE
1472 005132 012737 005304 001124 MOV #5304, $GDDAT ;LOAD EXPECTED STATUS:
1473 ;STATA, STATC, RDY, IE & FNCT2
1474 005140 023737 001124 001126 CMP $GDDAT, $BDDAT ;CORRECT?
1475 005146 001405 BEQ 4$ ;BR IF SO
1476 005150 013737 001370 001122 MOV DRVCSR, $BDADR ;SET UP CSR ADRS
1477 005156 104015 ERROR+15 ;ATTN STATUS ERROR
1478 005160 000416 BR 6$ ;GO RESYNC ON ERROR
1479
1480 005162 012737 177770 001124 4$: MOV #-8., $GDDAT ;LOAD EXPECTED WORD COUNT
1481 005170 017737 174170 001126 MOV @DRVWCR, $BDDAT ;READ WCR
1482 005176 023737 001124 001126 CMP $GDDAT, $BDDAT ;CHECK WORD COUNT
1483 005204 001406 BEQ 10$ ;BR IF SO
1484 005206 013737 001364 001122 MOV DRVWCR, $BDADR ;SET UP WCR ADRS
1485 005214 104011 ERROR+11 ;DATA WORD COUNT ERROR
1486
1487 005216 005237 001456 6$: INC ERRFLG ;SET ERROR FLAG
1488
  
```

```

1490 ;*****
1491 ;END OF TRANSFER SYNCHRONIZATION
1492 ;*****
1493
1494 005222 012704 007000 10$: MOV #7000,R4 ;LOAD MASK
1495 005226 004737 007726 JSR PC,WTEQ ;WAIT FOR MASTER TO CLEAR CSR
1496
1497 005232 005077 174132 CLR @DRVCSR ;ECHO CSR CLEAR TO MASTER
1498
1499 ;*****
1500 ;TRANSFER STATUS SYNCHRONIZATION
1501 ;*****
1502
1503 005236 012704 007000 11$: MOV #7000,R4 ;LOAD MASK
1504 005242 004737 007670 JSR PC,WTNE ;WAIT FOR MASTER TO POST STATUS
1505 005246 017737 174116 001416 MOV @DRVCSR,SAVE ;READ MASTER STATUS
1506
1507 005254 005737 001456 TST ERRFLG ;TEST LOCAL ERROR
1508 005260 001404 BEQ 111$ ;IF EQ NO ERROR
1509 005262 012777 000004 174100 MOV #FNCT2,@DRVCSR ;ERROR: SET FNCT2
1510 005270 000411 BR 12$ ;CONTINUE
1511
1512 005272 012777 000002 174070 111$: MOV #FNCT1,@DRVCSR ;NO ERROR: SET FNCT1
1513 005300 032737 002000 001416 BIT #DSTATB,SAVE ;CHECK FOR MASTER ERROR
1514 005306 001402 BEQ 12$ ;IF EQ NO ERROR
1515 005310 005237 001456 INC ERRFLG ;YES, FLAG ERROR
1516
1517 ;*****
1518 ;CLEAR STATUS SYNCHRONIZATION
1519 ;*****
1520
1521 005314 012704 007000 12$: MOV #7000,R4 ;LOAD MASK
1522 005320 004737 007726 JSR PC,WTEQ ;WAIT FOR MASTER TO CLEAR TRANSFER STATUS
1523
1524 005324 005077 174040 CLR @DRVCSR ;ECHO MASTER CLEAR TRANSFER STATUS
1525
1526 ;
1527 005330 005737 001456 13$: IF ERROR DETECTED, LOOP ON TEST
1528 005334 001402 TST ERRFLG ;ANY ERRORS?
1529 BEQ 14$ ;IF EQ NO ERROR
1530 005336 000137 004750 JMP STST4 ;RESTART TEST 4
1531
1532 005342 004737 005554 14$: JSR PC,RSTVEC ;GO RESTORE VECTOR
1533 005346 000240 NOP
1534 005350 000240 NOP
1535
1536 005352 104406 EOPT: CKSWR ;GO CHECK SWR
1537 005354 005737 001420 TST MSTER ;WERE WE STARTED AS MASTER?
1538 005360 001055 BNE EOPTA ;BR IF NOT
1539 005362 005337 001422 DEC ICOUNT ;COUNT TEST PASS
1540 005366 001052 BNE EOPTA ;BR IF NOT DUE FOR 'END PASS' MSG
1541 005370 012737 000370 001422 MOV #370,ICOUNT ;RESET PASS COUNT
1542

```

```

1544 .SBTTL END OF PASS ROUTINE
1545
1546
1547
1548
1549
1550
1551
1552
1553 005376 000240 $EOP: NOP
1554 005400 005037 001102 CLR $STNM ;;ZERO THE TEST NUMBER
1555 005404 005237 001100 INC $PASS ;;INCREMENT THE PASS NUMBER
1556 005410 042737 100000 001100 BIC #100000,$PASS ;;DON'T ALLOW A NEGATIVE NUMBER
1557 005416 005327 DEC (PC)+ ;;LOOP?
1558 005420 000001 $EOPCT: .WORD 1
1559 005422 003022 BGT $DOAGN ;;YES
1560 005424 012737 MOV (PC)+,@(PC)+ ;;RESTORE COUNTER
1561 005426 000001 $ENDCT: .WORD 1
1562 005430 005420 $EOPCT TYPE ;;TYPE 'END PASS #'
1563 005432 104400 $SENDNG
1564 005434 005477 MOV $PASS,-(SP) ;;SAVE $PASS FOR TYPEOUT
1565 005436 013746 001100 TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN
1566 005442 104404 TYPE ;;TYPE A NULL CHARACTER
1567 005444 104400 $ENULL
1568 005446 005474
1569
1570 005450 013700 000042 $GET42: MOV @#42,R0 ;;GET MONITOR ADDRESS
1571 005454 001405 BEQ $DOAGN ;;BRANCH IF NO MONITOR
1572 005456 000005 RESET ;;CLEAR THE WORLD
1573 005460 004710 $ENDAD: JSR PC,(R0) ;;GO TO MONITOR
1574 005462 000240 NOP ;;SAVE ROOM
1575 005464 000240 NOP ;;FOR
1576 005466 000240 NOP ;;ACT11
1577 005470 $DOAGN:
1578 005470 000137 JMP @(PC)+ ;;RETURN
1579
1580 005472 005514 $RTNAD: .WORD EOPTA
1581 005474 377 377 000 $ENULL: .BYTE -1,-1,0 ;;NULL CHARACTER STRING
1582 005477 015 012 105 $ENDNG: .ASCIZ <15><12>/END PASS #/
005502 116 104 040
005505 120 101 123
005510 123 040 043
005513 000

1583 .EVEN
1584
1585 005514 012737 001000 001404 EOPTA: MOV #1000,TIME ;;SET UP A COUNT
1586 005522 005337 001404 EOPTB: DEC TIME ;;STALL FOR OTHER CPU TO BECOME SLAVE
1587 005526 001375 BNE EOPTB ;;UNTIL TIME=0
1588 005530 000137 002100 JMP MTST1 ;;NOW BECOME MASTER
1589

```

```

1591      .SBTTL PROGRAM SUBROUTINES
1592
1593      ;*****
1594      ;THIS ROUTINE SETS THE PRIORITY LEVEL FOR NO INTERRUPT -
1595      ;SETS UP THE DRV11B INTERRUPT TO RETURN ON INTERRUPT
1596      ;TO THE ADDRESS INDICATED ((R5)) BY THE CALL +2
1597      ;*****
1598
1599 005534 004737 010072      SETVEC: JSR    PC,CPUHI      ;DISABLE INTERRUPTS
1600 005540 012577 173630      MOV    (R5)+,@DRVCT0    ;LOAD ISR ADDRESS
1601 005544 012777 000340 173624  MOV    #340,@DRVCT2    ;LOAD ISR PRIORITY
1602 005552 000205      RTS    R5              ;EXIT
1603
1604      ;*****
1605      ;THIS ROUTINE RESTORES THE DRV11B INTERRUPT VECTOR TO A HALT
1606      ;AND RAISES THE PRIORITY LEVEL
1607      ;*****
1608
1609 005554 004737 010072      RSTVEC: JSR   PC,CPUHI    ;DISABLE INTERRUPTS
1610 005560 013777 001376 173606  MOV   DRVCT2,@DRVCT0    ;POINT VECTOR TO HALT
1611 005566 005077 173604      CLR   @DRVCT2          ;SET UP HALT
1612 005572 000207      RTS   PC              ;RETURN
1613
1614      ;*****
1615      ;THIS ROUTINE DELAYS THE PROGRAM TO ALLOW POSSIBLE MEMORY
1616      ;REFRESH PERIODS TO OCCUR IN THE OTHER PROCESSOR WITHOUT LOOSING
1617      ;PROGRAM SYNC.
1618      ;*****
1619
1620 005574 012702 001000      DELAY: MOV   #1000,R2
1621 005600 005302      1$:  DEC   R2
1622 005602 001376      BNE   1$
1623 005604 000207      RTS   PC
1624

```

```

1626
1627
1628
1629
1630
1631
1632
1633
1634 005606 012500
1635 005610 012701 011632
1636 005614 005021
1637 005616 005200
1638 005620 001375
1639 005622 000205
1640
1641
1642
1643
1644
1645
1646
1647 005624 012500
1648 005626 012701 011632
1649 005632 012702 177776
1650 005636 005003
1651 005640 010221
1652 005642 005200
1653 005644 001001
1654 005646 000205
1655 005650 005102
1656 005652 005103
1657 005654 001371
1658 005656 006302
1659 005660 005202
1660 005662 103363
1661 005664 000765
1662
1663
1664
1665
1666
1667 005666 005237 001402
1668 005672 000002
1669

:*****
:THIS ROUTINE CLEARS THE 'DBUF' BEFORE A 'DATI' XFER
:THE # OF LCOATIONS IN 'DBUF' TO BE CLEARED IS SPECIFIED BY
:THE VALUE IN THE CALL +2 - WHEN ALL CLEARED THE RETURN IS TO
:THE CALL +4
:*****
CLRBUF: MOV (R5)+,R0 ;GET THE LOC COUNT
MOV #DBUF,R1 ;GET 1ST ADRS
1$: CLR (R1)+ ;CLR MEM LOC
INC R0 ;COUNT LOC
BNE 1$ ;UNTIL ALL DONE
RTS R5 ;EXIT

:*****
:THIS ROUTINE LOADS 'DBUF' WITH A FLOATING ZERO/ONE PATTERN
:THE # OF LOCATIONS IN 'DBUF' TO BE LOADED IS SPECIFIED BY
:THE VALUE IN THE CALL +2 - WHEN ALL LOADED THE RETURN IS TO CALL +4
:*****
LDBUF: MOV (R5)+,R0 ;GET LOC COUNT
MOV #DBUF,R1 ;GET 1ST ADRS
1$: MOV #177776,R2 ;SET UP FLOATING ZERO PTRN
CLR R3 ;R3 SAYS WHEN TO SHIFT PTRN
2$: MOV R2,(R1)+ ;LOAD MEM WITH PTRN
INC R0 ;COUNT LOC
BNE 3$ ;BR IF MORE
RTS R5 ;EXIT
3$: COM R2 ;CONVERT PTRN TO FLOATING 1
COM R3 ;SHOULD WE SHIFT?
BNE 2$ ;BR IF NOT - THIS WILL BE A FLOATING 1
ASL R2 ;FLOAT ZERO LEFT
INC R2 ;KEEP LSB SET
BCC 1$ ;GO RESET FLOATING PATRN
BR 2$ ;GO LOAD NEXT PATRN

:*****
:THIS ROUTINE WILL INDICATE THAT AN INTERRUPT HAS OCCURED.
:*****
DRVISR: INC INTFLG ;BUMP FLAG
RTI ;AND RETURN
  
```

```

1671          .SBTTL SYSMAC ROUTINES
1672          .SBTTL TYPE ROUTINE
1673
1674          ;*****
1675          ;ROUTINE TO TYPE ASCIZ MESSAGE, MESSAGE MUST TERMINATE WITH A 0 BYTE.
1676          ;THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
1677          ;NOTE1:          $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
1678          ;NOTE2:          $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
1679          ;NOTE3:          $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
1680
1681          ;CALL:
1682          ;1) USING A TRAP INSTRUCTION
1683          ;      TYPE      ,MESADR          ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
1684          ;OR
1685          ;      TYPE
1686          ;      MESADR
1687
1688          $TYPE:  TSTB      $STPFLG          ;; IS THERE A TERMINAL?
1689                  BPL      1$              ;; BR IF YES
1690                  HALT     3$              ;; HALT HERE IF NO TERMINAL
1691                  BR       3$              ;; LEAVE
1692          1$:      MOV      RO,-(SP)         ;; SAVE RO
1693                  MOV      @2(SP),RO      ;; GET ADDRESS OF ASCIZ STRING
1694          2$:      MOVB     (RO)+,-(SP)     ;; PUSH CHARACTER TO BE TYPED ONTO STACK
1695                  BNE      4$              ;; BR IF IT ISN'T THE TERMINATOR
1696                  TST      (SP)+         ;; IF TERMINATOR POP IT OFF THE STACK
1697          60$:     MOV      (SP)+,RO      ;; RESTORE RO
1698          3$:      ADD      #2,(SP)        ;; ADJUST RETURN PC
1699                  RTI
1700          4$:      CMPB     #HT,(SP)       ;; BRANCH IF <HT>
1701                  BEQ      8$
1702                  CMPB     #CRLF,(SP)     ;; BRANCH IF NOT <CRLF>
1703                  BNE      5$
1704                  TST      (SP)+         ;; POP <CR><LF> EQUIV
1705                  TYPE
1706                  $CRLF
1707          5$:      CLRB     $CHARCNT       ;; CLEAR CHARACTER COUNT
1708                  BR       2$              ;; GET NEXT CHARACTER
1709          5$:      JSR      PC,$TYPEC      ;; GO TYPE THIS CHARACTER
1710          6$:      CMPB     $FILLC,(SP)+   ;; IS IT TIME FOR FILLER CHARS?
1711                  BNE      2$              ;; IF NO GO GET NEXT CHAR.
1712                  MOV      $NULL,-(SP)    ;; GET # OF FILLER CHARS, NEEDED
1713                  AND      THE NULL CHAR.
1714          7$:      DECB     1(SP)          ;; DOES A NULL NEED TO BE TYPED?
1715                  BLT      6$              ;; BR IF NO--GO POP THE NULL OFF OF STACK
1716                  JSR      PC,$TYPEC      ;; GO TYPE A NULL
1717                  DECB     $CHARCNT       ;; DO NOT COUNT AS A COUNT
1718                  BR       7$              ;; LOOP
1719
1688 005674 105737 001157
1689 005700 100002
1690 005702 000000
1691 005704 000407
1692 005706 010046
1693 005710 017600 000002
1694 005714 112046
1695 005716 001005
1696 005720 005726
1697 005722 012600
1698 005724 062716 000002
1699 005730 000002
1700 005732 122716 000011
1701 005736 001430
1702 005740 122716 000200
1703 005744 001006
1704 005746 005726
1705 005750 104400
1706 005752 001165
1707 005754 105037 006110
1708 005760 000755
1709 005762 004737 006044
1710 005766 123726 001156
1711 005772 001350
1712 005774 013746 001154
1713
1714 006000 105366 000001
1715 006004 002770
1716 006006 004737 006044
1717 006012 105337 006110
1718 006016 000770
1719
  
```

```

1721 ;HORIZONTAL TAB PROCESSOR
1722
1723 006020 112716 000040 8$: MOVB #' ,(SP) ;REPLACE TAB WITH SPACE
1724 006024 004737 006044 9$: JSR PC,$TYPEC ;;TYPE A SPACE
1725 006030 132737 000007 006110 BITB #7,$CHARCNT ;;BRANCH IF NOT AT
1726 006036 001372 BNE 9$ ;;TAB STOP
1727 006040 005726 TST (SP)+ ;;POP SPACE OFF STACK
1728 006042 000724 BR 2$ ;;GET NEXT CHARACTER
1729 006044 105777 173100 $TYPEC: TSTB @$TPS ;;WAIT UNTIL PRINTER IS READY
1730 006050 100375 BPL $TYPEC
1731 006052 116677 000002 173072 MOVB 2(SP),@$TPB ;;LOAD CHAR TO BE TYPED INTO DATA REG.
1732 006060 122766 000015 000002 CMPB #CR,2(SP) ;;IS CHARACTER A CARRIAGE RETURN?
1733 006066 001003 BNE 1$ ;;BRANCH IF NO
1734 006070 105037 006110 CLRB $CHARCNT ;;YES--CLEAR CHARACTER COUNT
1735 006074 000406 BR $TYPEX ;;EXIT
1736 006076 122766 000012 000002 1$: CMPB #LF,2(SP) ;;IS CHARACTER A LINE FEED?
1737 006104 001402 BEQ $TYPEX ;;BRANCH IF YES
1738 006106 105227 INCB (PC)+ ;;COUNT THE CHARACTER
1739 006110 000000 $CHARCNT: .WORD 0 ;;CHARACTER COUNT STORAGE
1740 006112 000207 $TYPEX: RTS PC
1741

```

```

1743 .SBTTL BINARY TO OCTAL (ASCII) AND TYPE
1744 :*****
1745 :THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
1746 :OCTAL (ASCII) NUMBER AND TYPE IT.
1747 :$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
1748 :CALL:
1749 :       MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
1750 :       TYPOS   ;;CALL FOR TYPEOUT
1751 :       .BYTE  N              ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
1752 :       .BYTE  M              ;;M=1 OR 0
1753 :                               ;;1=TYPE LEADING ZEROS
1754 :                               ;;0=SUPPRESS LEADING ZEROS
1755 :
1756 :$TYPEON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
1757 :$TYPOS OR $TYPOC
1758 :CALL:
1759 :       MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
1760 :       TYPON   ;;CALL FOR TYPEOUT
1761 :
1762 :$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
1763 :CALL:
1764 :       MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
1765 :       TYPOC   ;;CALL FOR TYPEOUT
1766 :
1767 006114 017646 000000      $TYPOS: MOV     @ (SP),-(SP)      ;;PICKUP THE MODE
1768 006120 116637 000001 006337  MOVB    1(SP), $OFILL      ;;LOAD ZERO FILL SWITCH
1769 006126 112637 006341      MOVB    (SP)+, $OMODE+1    ;;NUMBER OF DIGITS TO TYPE
1770 006132 062716 000002      ADD     #2,(SP)          ;;ADJUST RETURN ADDRESS
1771 006136 000406      BR      $TYPON
1772 006140 112737 000001 006337 $TYPOC: MOVB    #1, $OFILL      ;;SET THE ZERO FILL SWITCH
1773 006146 112737 000006 006341  MOVB    #6, $OMODE+1      ;;SET FOR SIX(6) DIGITS
1774 006154 112737 000005 006336 $TYPON: MOVB    #5, $OCNT      ;;SET THE ITERATION COUNT
1775 006162 010346      MOV     R3,-(SP)         ;;SAVE R3
1776 006164 010446      MOV     R4,-(SP)         ;;SAVE R4
1777 006166 010546      MOV     R5,-(SP)         ;;SAVE R5
1778 006170 113704 006341      MOVB    $OMODE+1,R4      ;;GET THE NUMBER OF DIGITS TO TYPE
1779 006174 005404      NEG     R4
1780 006176 062704 000006      ADD     #6,R4            ;;SUBTRACT IT FOR MAX, ALLOWED
1781 006202 110437 006340      MOVB    R4, $OMODE        ;;SAVE IT FOR USE
1782 006206 113704 006337      MOVB    $OFILL,R4        ;;GET THE ZERO FILL SWITCH
1783 006212 016605 000012      MOV     12(SP),R5        ;;PICKUP THE INPUT NUMBER
1784 006216 005003      CLR     R3                ;;CLEAR THE OUTPUT WORD
1785

```



```

1787 006220 006105      1$:  ROL  R5      ;;ROTATE MSB INTO 'C'
1788 006222 000404      BR   3$      ;;GO DO MSB
1789 006224 006105      2$:  ROL  R5      ;;FORM THIS DIGIT
1790 006226 006105      ROL  R5
1791 006230 006105      ROL  R5
1792 006232 010503      MOV  R5,R3
1793 006234 006103      3$:  ROL  R3      ;;GET LSB OF THIS DIGIT
1794 006236 105337 006340  DECB $OMODE  ;;TYPE THIS DIGIT?
1795 006242 100016      BPL  7$      ;;BR IF NO
1796 006244 042703 177770  BIC  #177770,R3  ;;GET RID OF JUNK
1797 006250 001002      BNE  4$      ;;TEST FOR 0
1798 006252 005704      TST  R4      ;;SUPPRESS THIS 0?
1799 006254 001403      BEQ  5$      ;;BR IF YES
1800 006256 005204      4$:  INC  R4      ;;DON'T SUPPRESS ANYMORE 0'S
1801 006260 052703 000060  BIS  #'0,R3    ;;MAKE THIS DIGIT ASCII
1802 006264 052703 000040  5$:  BIS  #' ,R3  ;;MAKE ASCII IF NOT ALREADY
1803 006270 110337 006334  MOVB R3,8$    ;;SAVE FOR TYPING
1804 006274 104400      TYPE  ;;GO TYPE THIS DIGIT
1805 006276 006334      8$
1806 006300 105337 006336  7$:  DECB $OCNT  ;;COUNT BY 1
1807 006304 003347      BGT  2$      ;;BR IF MORE TO DO
1808 006306 002402      BLT  6$      ;;BR IF DONE
1809 006310 005204      INC  R4      ;;INSURE LAST DIGIT ISN'T A BLANK
1810 006312 000744      BR   2$      ;;GO DO THE LAST DIGIT
1811 006314 012605      6$:  MOV  (SP)+,R5  ;;RESTORE R5
1812 006316 012604      MOV  (SP)+,R4  ;;RESTORE R4
1813 006320 012603      MOV  (SP)+,R3  ;;RESTORE R3
1814 006322 016666 000002 000004  MOV  2(SP),4(SP)  ;;SET THE STACK FOR RETURNING
1815 006330 012616      MOV  (SP)+,(SP)
1816 006332 000002      RTI
1817 006334 000      8$:  .BYTE 0      ;;RETURN
1818 006335 000      .BYTE 0      ;;STORAGE FOR ASCII DIGIT
1819 006336 000      $OCNT: .BYTE 0  ;;TERMINATOR FOR TYPE ROUTINE
1820 006337 000      $OFILL: .BYTE 0  ;;OCTAL DIGIT COUNTER
1821 006340 000000      $OMODE: .WORD 0  ;;ZERO FILL SWITCH
1822

```

```

1824 .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
1825 :*****
1826 :THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
1827 :SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT, DEPENDING ON WHETHER THE
1828 :NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
1829 :BEFORE THE FIRST DIGIT OF THE NUMBER, LEADING ZEROS WILL ALWAYS BE
1830 :REPLACED WITH SPACES.
1831 :CALL:
1832 :      MOV      NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
1833 :      TYPDS      ;;GO TO THE ROUTINE
1834
1835 $TYPDS:
1836 006342 010046      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
1837 006344 010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
1838 006345 010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
1839 006350 010346      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
1840 006352 010546      MOV      R5,-(SP)      ;;PUSH R5 ON STACK
1841 006354 012746 020200      MOV      #20200,-(SP)  ;;SET BLANK SWITCH AND SIGN
1842 006360 016605 000020      MOV      20(SP),R5    ;;GET THE INPUT NUMBER
1843 006364 100004      BPL      1$           ;;BR IF INPUT IS POS.
1844 006366 005405      NEG      R5           ;;MAKE THE BINARY NUMBER POS.
1845 006370 112766 000055 000001      MOV      #'-,1(SP)    ;;MAKE THE ASCII NUMBER NEG.
1846 006376 005000      CLR      R0           ;;ZERO THE CONSTANTS INDEX
1847 006400 012703 006556      MOV      #$DBLK,R3    ;;SETUP THE OUTPUT POINTER
1848 006404 112723 000040      MOV      #' ,(R3)+    ;;SET THE FIRST CHARACTER TO A BLANK
1849 006410 005002      CLR      R2           ;;CLEAR THE BCD NUMBER
1850 006412 016001 006546      MOV      $DTBL(R0),R1 ;;GET THE CONSTANT
1851 006416 160105      SUB      R1,R5        ;;FORM THIS BCD DIGIT
1852 006420 002402      BLT      4$           ;;BR IF DONE
1853 006422 005202      INC      R2           ;;INCREASE THE BCD DIGIT BY 1
1854 006424 000774      BR      3$
1855 006426 060105      ADD      R1,R5        ;;ADD BACK THE CONSTANT
1856 006430 005702      TST      R2           ;;CHECK IF BCD DIGIT=0
1857 006432 001002      BNE      5$           ;;FALL THROUGH IF 0
1858 006434 105716      TSTB    (SP)          ;;STILL DOING LEADING 0'S?
1859 006436 100407      BMI      7$           ;;BR IF YES
1860 006440 106316      ASLB    (SP)          ;;MSD?
1861 006442 103003      BCC      6$           ;;BR IF NO
1862 006444 116663 000001 177777      MOV      1(SP),-1(R3) ;;YES--SET THE SIGN
1863 006452 052702 000060      BIS      #'0,R2      ;;MAKE THE BCD DIGIT ASCII
1864

```

```

1866 006456 052702 000040      7$:  BIS      #' ,R2      ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
1867 006462 110223             MOVB     R2,(R3)+    ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
1868 006464 005720             TST      (R0)+      ;;JUST INCREMENTING
1869 006466 020027 000010     CMP      R0,#10     ;;CHECK THE TABLE INDEX
1870 006472 002746             BLT      2$         ;;GO DO THE NEXT DIGIT
1871 006474 003002             BGT      8$         ;;GO TO EXIT
1872 006476 010502             MOV      R5,R2     ;;GET THE LSD
1873 006500 000764             BR       6$         ;;GO CHANGE TO ASCII
1874 006502 105726             8$:  TSTB     (SP)+    ;;WAS THE LSD THE FIRST NON-ZERO?
1875 006504 100003             BPL      9$         ;;BR IF NO
1876 006506 116663 177777 177776 MOVB     -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
1877 006514 105013             9$:  CLRB     (R3)    ;;SET THE TERMINATOR
1878 006516 012605             MOV      (SP)+,R5  ;;POP STACK INTO R5
1879 006520 012603             MOV      (SP)+,R3  ;;POP STACK INTO R3
1880 006522 012602             MOV      (SP)+,R2  ;;POP STACK INTO R2
1881 006524 012601             MOV      (SP)+,R1  ;;POP STACK INTO R1
1882 006526 012600             MOV      (SP)+,R0  ;;POP STACK INTO R0
1883 006530 104400             TYPE                    ;;NOW TYPE THE NUMBER
1884 006532 006556             $DBLK
1885 006534 016666 000002 000004 MOV      2(SP),4(SP) ;;ADJUST THE STACK
1886 006542 012616             MOV      (SP)+,(SP)
1887 006544 000002             RTI                    ;;RETURN TO USER
1888 006546 023420             $DTBL: 10000.
1889 006550 001750             1000.
1890 006552 000144             100.
1891 006554 000012             10.
1892 006556             $DBLK: .BLKW 4
1893
  
```

```

1895 .SBTTL ERROR HANDLER ROUTINE
1896
1897 *****
1898 :THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
1899 :SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
1900 :AND GO TO SWRCK ON ERROR
1901 :THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
1902 :SW15=1 HALT ON ERROR
1903 :SW13=1 INHIBIT ERROR TYPEOUTS
1904 :SW10=1 BELL ON ERROR
1905 :CALL
1906
1907 $ERROR: ERROR N ;;ERROR=EMT AND N=ERROR ITEM NUMBER
1908 006566 104406 CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
1909 006570 104406 CKSWR ;;GO LOOK FOR SWR CHANGE
1910 006572 105237 001103 7$: INCB $ERFLG ;;SET THE ERROR FLAG
1911 006576 001775 BEQ 7$ ;;DON'T LET THE FLAG TO TO ZERO
1912 006600 013777 001102 172334 MOV $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
1913 006606 032777 002000 172324 BIT #BIT10,@SWR ;;BELL ON ERROR?
1914 006614 001402 BEQ 1$ ;;NO - SKIP
1915 006616 104400 TYPE ;;RING BELL
1916 006620 001160 $BELL
1917 006622 005237 001112 1$: INC $ERTTL ;;COUNT THE NUMBER OF ERRORS
1918 006626 011637 001116 MOV (SP),$ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
1919 006632 162737 000002 001116 SUB #2,$ERRPC
1920 006640 117737 172252 001114 MOVB @$ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
1921 006646 032777 020000 172264 BIT #BIT13,@SWR ;;SKIP TYPEOUT IF SET
1922 006654 001004 BNE 20$ ;;SKIP TYPEOUTS
1923 006656 004737 006702 JSR PC,SWRCK ;;GO TO USER ERROR ROUTINE
1924 006662 104400 TYPE
1925 006664 001165 $CRLF
1926 006666 20$:
1927 006666 005777 172246 2$: TST @SWR ;;HALT ON ERROR
1928 006672 100002 BPL 3$ ;;SKIP IF CONTINUE
1929 006674 000000 HALT ;;HALT ON ERROR!
1930 006676 104406 CKSWR ;;TEST FOR CHANGE IN SOFT/SWR
1931 006700 3$:
1932 006700 000002 RTI ;;RETURN
1933 *****
1934 :GO TYPE ERROR
1935 :GO UPDATE SOFTWARE SWR IF 'CNTRL/G'
1936 *****
1937
1938 006702 004737 006712 SWRCK: JSR PC,$ERRTYP ;GO TYPE ERROR
1939 006706 104406 CKSWR ;GO LOOK FOR SWR CHANGE
1940 006710 000207 RTS PC ;RETURN TO ERROR HANDLER
1941

```

```

1943 .SBTTL ERROR MESSAGE TYPEOUT ROUTINE
1944 :*****
1945 :THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
1946 :ERROR IS TO BE REPORTED, IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
1947 :AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
1948
1949 006712 $ERRTYP:
1950 006712 104400 TYPE ;:"CARRIAGE RETURN" & "LINE FEED"
1951 006714 001165 $CRLF
1952 006716 010046 MOV RO,-(SP) ;:SAVE RO
1953 006720 005000 CLR RO ;:PICKUP THE ITEM INDEX
1954 006722 153700 001114 BISB @#$ITEMB,RO
1955 006726 001004 BNE 1$ ;:IT ITEM NUMBER IS ZERO, JUST
1956 ;:TYPE THE PC OF THE ERROR
1957 006730 013746 001116 MOV $ERRPC,-(SP) ;:SAVE $ERRPC FOR TYPEOUT
1958 ;:ERROR ADDRESS
1959 006734 104401 TYPOC ;:GO TYPE--OCTAL ASCII (ALL DIGITS)
1960 006736 000426 BR 6$ ;:GET OUT
1961 006740 005300 1$: DEC RO ;:ADJUST THE INDEX SO THAT IT WILL
1962 006742 006300 ASL RO ;:WORK FOR THE ERROR TABLE
1963 006744 006300 ASL RO
1964 006746 006300 ASL RO
1965 006750 062700 001170 ADD #$ERRTB,RO ;:FORM TABLE POINTER
1966 006754 012037 006764 MOV (RO)+,2$ ;:PICKUP "ERROR MESSAGE" POINTER
1967 006760 001404 BEQ 3$ ;:SKIP TYPEOUT IF NO POINTER
1968 006762 104400 TYPE ;:TYPE THE "ERROR MESSAGE"
1969 006764 000000 2$: .WORD 0 ;:"ERROR MESSAGE" POINTER GOES HERE
1970 006766 104400 TYPE ;:"CARRIAGE RETURN" & "LINE FEED"
1971 006770 001165 $CRLF
1972 006772 012037 007002 3$: MOV (RO)+,4$ ;:PICKUP "DATA HEADER" POINTER
1973 006776 001404 BEQ 5$ ;:SKIP TYPEOUT IF 0
1974 007000 104400 TYPE ;:TYPE THE "DATA HEADER"
1975 007002 000000 4$: .WORD 0 ;:"DATA HEADER" POINTER GOES HERE
1976 007004 104400 TYPE ;:"CARRIAGE RETURN" & "LINE FEED"
1977 007006 001165 $CRLF
1978 007010 011000 5$: MOV (RO),RO ;:PICKUP "DATA TABLE" POINTER
1979 007012 001004 BNE 7$ ;:GO TYPE THE DATA
1980 007014 012600 6$: MOV (SP)+,RO ;:RESTORE RO
1981 007016 104400 TYPE ;:"CARRIAGE RETURN" & "LINE FEED"
1982 007020 001165 $CRLF
1983 007022 000207 RTS PC ;:RETURN
1984 007024
1985 007024 013046 7$: MOV @ (RO)+,-(SP) ;:SAVE @ (RO)+ FOR TYPEOUT
1986 007026 104401 TYPOC ;:GO TYPE--OCTAL ASCII (ALL DIGITS)
1987 007030 005710 TST (RO) ;:IS THERE ANOTHER NUMBER?
1988 007032 001770 BEQ 6$ ;:BR IF NO
1989 007034 104400 TYPE ;:TYPE TWO(2) SPACES
1990 007036 007042 8$
1991 007040 000771 BR 7$ ;:LOOP
1992 007042 040 040 000 8$: .ASCIZ / / ;:TWO (2) SPACES
1993 .EVEN
1994

```

```

1996 .SBTTL TTY INPUT ROUTINE
1997 .ENABL LSB
1998
1999
2000
2001
2002
2003
2004
2005
2006 007046 022737 000176 001140 $CKSWR: CMP #SWREG,SWR ;;IS THE SOFT-SWR SELECTED?
2007 007054 001074 BNE 15$ ;;BRANCH IF NO
2008 007056 105777 172062 TSTB @$TKS ;;CHAR THERE?
2009 007062 100071 BPL 15$ ;;IF NO, DON'T WAIT AROUND
2010 007064 117746 172056 MOVB @$TKB,-(SP) ;;SAVE THE CHAR
2011 007070 042716 177600 BIC #^C177,(SP) ;;STRIP-OFF THE ASCII
2012 007074 C22726 000007 CMP #7,(SP)+ ;;IS IT A CONTROL G?
2013 007100 001062 BNE 15$ ;;NO, RETURN TO USER
2014 007102 123727 001134 000001 CMPB $AUTOB,#1 ;;ARE WE RUNNING IN AUTO-MODE?
2015 007110 001456 BEQ 15$ ;;BRANCH IF YES
2016 007112 104400 TYPE ;;ECHO THE CONTROL-G (-G)
2017 007114 007573 $CNTLG
2018 007116 104400 $GTSWR: TYPE ;;TYPE CURRENT CONTENTS
2019 007120 007600 $MSWR
2020 007122 013746 000176 MOV SWREG,-(SP) ;;SAVE SWREG FOR TYPEOUT
2021 007126 104401 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2022 007130 104400 TYPE ;;PROMPT FOR NEW SWR
2023 007132 007611 $MNEW
2024 007134 005046 19$: CLR -(SP) ;;CLEAR COUNTER
2025 007136 005046 CLR -(SP) ;;THE NEW SWR
2026 007140 105777 172000 7$: TSTB @$TKS ;;CHAR THERE?
2027 007144 100375 BPL 7$ ;;IF NOT TRY AGAIN
2028 007146 117746 171774 MOVB @$TKB,-(SP) ;;PICK UP CHAR
2029 007152 042716 177600 BIC #^C177,(SP) ;;MAKE IT 7-BIT ASCII
2030 007156 021627 000025 9$: CMP (SP),#25 ;;IS IT A CONTROL-U?
2031 007162 001005 BNE 10$ ;;BRANCH IF NOT
2032 007164 104400 TYPE ;;YES,ECHO CONTROL-U (^U)
2033 007166 007566 $CNTLU
2034 007170 062706 000006 20$: ADD #6,SP ;;IGNORE PREVIOUS INPUT
2035 007174 000757 BR 19$ ;;LET'S TRY IT AGAIN
2036 007176 021627 000015 10$: CMP (SP),#15 ;;IS IT A <CR>?
2037 007202 001022 BNE 16$ ;;BRANCH IF NO
2038 007204 005766 000004 TST 4(SP) ;;YES,IS IT THE FIRST CHAR?
2039 007210 001403 BEQ 11$ ;;BRANCH IF YES
2040 007212 016677 000002 171720 MOV 2(SP),@SWR ;;SAVE NEW SWR
2041 007220 062706 000006 11$: ADD #6,SP ;;CLEAR UP STACK
2042 007224 104400 14$: TYPE ;;ECHO <CR> AND <LF>
2043 007226 001165 $CRLF
2044 007230 123727 001135 000001 CMPB $INTAG,#1 ;;RE-ENABLE TTY KBD INTERRUPTS?
2045 007236 001003 BNE 15$ ;;BRANCH IF NOT
2046 007240 012777 000100 171676 MOV #100,@TKS ;;RE-ENABLE TTY KBD INTERRUPTS
2047 007246 000002 15$: RTI ;;RETURN
2048

```

2050	007250	004737	006044	16\$:	JSR	PC,\$TYPEC	::ECHO CHAR
2051	007254	021627	000060		CMP	(SP),#60	::CHAR < 0?
2052	007260	002420			BLT	18\$::BRANCH IF YES
2053	007262	021627	000067		CMP	(SP),#67	::CHAR > 7?
2054	007266	003015			BGT	18\$::BRANCH IF YES
2055	007270	042726	000060		BIC	#60,(SP)+	::STRIP-OFF ASCII
2056	007274	005766	000002		TST	2(SP)	::IS THIS THE FIRST CHAR
2057	007300	001403			BEQ	17\$::BRANCH IF YES
2058	007302	006316			ASL	(SP)	::NO, SHIFT PRESENT
2059	007304	006316			ASL	(SP)	::CHAR OVER TO MAKE
2060	007306	006316			ASL	(SP)	::ROOM FOR NEW ONE.
2061	007310	005266	000002	17\$:	INC	2(SP)	::KEEP COUNT OF CHAR
2062	007314	056616	177776		BIS	-2(SP),(SP)	::SET IN NEW CHAR
2063	007320	000707			BR	7\$::GET THE NEXT ONE
2064	007322	104400		18\$:	TYPE		::TYPE ?<CR><LF>
2065	007324	001164			\$QUES		
2066	007326	000720			BR	20\$::SIMULATE CONTROL^U
2067				.DSABL	LSB		
2068							

```

2070
2071
2072
2073
2074
2075
2076
2077
2078
2079 007330 011646
2080 007332 016666 000004 000002
2081 007340 105777 171600
2082 007344 100375
2083 007346 117766 171574 000004
2084 007354 042766 177600 000004
2085 007362 026627 000004 000023
2086 007370 001013
2087 007372 105777 171550
2088 007376 100375
2089 007400 117746 171542
2090 007404 042716 177600
2091 007410 022627 000021
2092 007414 001366
2093 007416 000750
2094 007420 026627 000004 000140
2095 007426 002407
2096 007430 026627 000004 000175
2097 007436 003003
2098 007440 042766 000040 000004
2099 007446 000002
2100

:*****
:THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
:CALL:
:      RDCHR          ;;INPUT A SINGLE CHARACTER FROM THE TTY
:      RETURN HERE   ;;CHARACTER IS ON THE STACK
:                   ;;WITH PARITY BIT STRIPPED OFF
:
$RDCHR: MOV      (SP),-(SP)      ;;PUSH DOWN THE PC
        MOV      4(SP),2(SP)    ;;SAVE THE PS
1$:     TSTB     @$TKS          ;;WAIT FOR
        BPL      1$             ;;A CHARACTER
        MOVB     @$TKB,4(SP)    ;;READ THE TTY
        BIC      #^C<177>,4(SP) ;;GET RID OF JUNK IF ANY
        CMP      4(SP),#23     ;;IS IT A CONTROL-S?
        BNE      3$             ;;BRANCH IF NO
2$:     TSTB     @$TKB          ;;WAIT FOR A CHARACTER
        BPL      2$             ;;LOOP UNTIL ITS THERE
        MOVB     @$TKB,-(SP)    ;;GET CHARACTER
        BIC      #^C177,(SP)   ;;MAKE IT 7-BIT ASCII
        CMP      (SP)+,#21     ;;IS IT A CONTROL-Q?
        BNE      2$             ;;IF NOT DISCARD IT
        BR       1$             ;;YES, RESUME
3$:     CMP      4(SP),#140     ;;IS IT UPPER CASE?
        BLT      4$             ;;BRANCH IF YES
        CMP      4(SP),#175    ;;IS IT A SPECIAL CHAR?
        BGT      4$             ;;BRANCH IF YES
        BIC      #40,4(SP)     ;;MAKE IT UPPER CASE
4$:     RTI
        ;;GU BACK TO USER
  
```



```

2102
2103
2104
2105
2106
2107
2108 007450 010346
2109 007452 012703 007556
2110 007456 022703 007566
2111 007462 101405
2112 007464 104407
2113 007466 112613
2114 007470 122713 000177
2115 007474 001003
2116 007476 104400
2117 007500 001164
2118 007502 000763
2119 007504 111337 007554
2120 007510 104400
2121 007512 007554
2122 007514 122723 000015
2123 007520 001356
2124 007522 105063 177777
2125 007526 104400
2126 007530 001166
2127 007532 012603
2128 007534 011646
2129 007536 016666 000004 000002
2130 007544 012766 007556 000004
2131 007552 000002
2132 007554 000
2133 007555 000
2134 007556
2135 007566 136 125 015
      007571 012 000
2136 007573 136 107 015
      007576 012 000
2137 007600 015 012 123
      007603 127 122 040
      007606 075 040 000
2138 007611 040 040 116
      007614 105 127 040
      007617 075 040 000
2139
2140
  
```

```

:*****
:THIS ROUTINE WILL INPUT A STRING FROM THE TTY
:CALL:
:      RDLIN          ;; INPUT A STRING FROM THE TTY
:      RETURN HERE   ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
:                   ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
:                   ;; SAVE R3
:RDLIN: MOV R3, -(SP) ;; GET ADDRESS
1$: MOV $TTYIN, R3   ;; BUFFER FULL?
2$: CMP $TTYIN+8., R3 ;; BR IF YES
      BLOS 4$        ;; GO READ ONE CHARACTER FROM THE TTY
      RDCHR          ;; GET CHARACTER
10$: MOV (SP)+, (R3) ;; IS IT A RUBOUT
      CMPB #177, (R3) ;; SKIP IF NOT
      BNE 3$         ;; TYPE A '?'
4$: TYPE           ;; CLEAR THE BUFFER AND LOOP
      $QUES         ;; ECHO THE CHARACTER
3$: BR 1$          ;; CHECK FOR RETURN
      MOV (R3), 9$   ;; LOOP IF NOT RETURN
      TYPE          ;; CLEAR RETURN (THE 15)
      9$           ;; TYPE A LINE FEED
      CMPB #15, (R3)+
      BNE 2$        ;; RESTORE R3
      CLRB -1(R3)   ;; ADJUST THE STACK AND PUT ADDRESS OF THE
                       ;; FIRST ASCII CHARACTER ON IT
      TYPE          ;; RETURN
      $LF          ;; STORAGE FOR ASCII CHAR, TO TYPE
9$: .BYTE 0        ;; TERMINATOR
      .BYTE 0      ;; RESERVE 8 BYTES FOR TTY INPUT
$TTYIN: .BLKB 8.  ;; CONTROL 'U'
$CNTLU: .ASCIZ /^U/<15><12>
$CNTLG: .ASCIZ /^G/<15><12> ;; CONTROL 'G'
$MSWR: .ASCIZ <15><12>/SWR = /
$MNEW: .ASCIZ / NEW = /
.EVEN
  
```

2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187

007622 010046
007624 016600 000002
007630 005740
007632 111000
007634 006300
007636 016000 007644
007642 000200

007644 005674
007646 006140
007650 006114
007652 006154
007654 006342
007656 007116
007660 007046
007662 007330
007664 007450
007666 010004

104400
104401
104402
104403
104404
104405
104406
104407
104410
104411

.SBTTL TRAP DECODER

 :THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
 :AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
 :OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
 :GO TO THAT ROUTINE.

```
$TRAP:  MOV    RO, -(SP)           ;;SAVE RO
        MOV    2(SP), RO         ;;GET TRAP ADDRESS
        TST   -(RO)             ;;BACKUP BY 2
        MOVB  (RO), RO          ;;GET RIGHT BYTE OF TRP
        ASL   RO                ;;POSITION FOR INDEXING
        MOV   $TRPAD(RO), RO     ;;INDEX TO TABLE
        RTS   RO                ;;GO TO ROUTINE
```

.SBTTL TRAP TABLE

:THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
 :BY THE "TRAP" INSTRUCTION.

: ROUTINE
 :-----

```
$TRPAD: $TYPE  ;;CALL=TYPE      TRAP+0(104400)  TTY TYPEOUT ROUTINE
        $TYPOC ;;CALL=TYPOC    TRAP+1(104401)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
        $TYPOS ;;CALL=TYPOS    TRAP+2(104402)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
        $TYPON ;;CALL=TYPON    TRAP+3(104403)  TYPE OCTAL NUMBER (AS PER LAST CALL)
        $TYPDS ;;CALL=TYPDS    TRAP+4(104404)  TYPE DECIMAL NUMBER (WITH SIGN)
        $GTSWR ;;CALL=GTSWR    TRAP+5(104405)  GET SOFT-SWR SETTING
        $CKSWR ;;CALL=CKSWR    TRAP+6(104406)  TEST FOR CHANGE IN SOFT-SWR
        $RDCHR ;;CALL=RDCHR    TRAP+7(104407)  TTY TYPEIN CHARACTER ROUTINE
        $RDLIN ;;CALL=RDLIN    TRAP+10(104410) TTY TYPEIN STRING ROUTINE
        RESYNC ;;CALL=RESYNC   TRAP+11(104411) PROGRAM RESYNC
```

TYPE=TRAP+0
 TYPOC=TRAP+1
 TYPOS=TRAP+2
 TYPON=TRAP+3
 TYPDS=TRAP+4
 GTSWR=TRAP+5
 CKSWR=TRAP+6
 RDCHR=TRAP+7
 RDLIN=TRAP+10
 RESYNC=TRAP+11

```

2189      .SBTTL INTERBOARD COMMUNICATION SUBROUTINES
2190      ;:*****
2191      ;:                               ;WTNE
2192      ;:*****
2193
2194 007670 WTNE:      MOV      #4,ABORT      ;WAIT FOR BITS SET
2195 007670 012737 000004 001406      ;SETUP ABORT TIMER
2196
2197 007676 005037 001404      10$:      CLR      TIME      ;CLEAR SUB LOOP COUNTER
2198      ;INNER LOOP
2199 007702 030477 171462      15$:      BIT      R4,@DRVCSR      ;WAIT FOR BITS SET
2200 007706 001035      BNE      WTEXI      ;YES: CONTINUE
2201 007710 005337 001404      DEC      TIME      ;DECR SUB LOOP COUNTER
2202 007714 001372      BNE      15$      ;CONTINUE UNTIL ZERO
2203      ;OUTER LOOP
2204 007716 005337 001406      DEC      ABORT      ;DECR MAIN LOOP COUNTER
2205 007722 001365      BNE      10$      ;CONTINUE UNTIL ZERO
2206
2207 007724 000417      BR      WTABO      ;TIMEOUT: ABORT PROGRAM
2208
2209      ;:*****
2210      ;:                               ;WTEQ
2211      ;:*****
2212
2213 007726 WTEQ:      MOV      #4,ABORT      ;WAIT FOR BITS CLEAR
2214 007726 012737 000004 001406      ;SETUP ABORT TIMER
2215      ;INNER LOOP
2216 007734 005037 001404      10$:      CLR      TIME      ;CLEAR SUB LOOP COUNTER
2217 007740 030477 171424      15$:      BIT      R4,@DRVCSR      ;WAIT FOR BITS CLEAR
2218 007744 001416      BEQ      WTEXI      ;YES: CONTINUE
2219 007746 005337 001404      DEC      TIME      ;DECR SUB LOOP COUNTER
2220 007752 001372      BNE      15$      ;CONTINUE UNTIL ZERO
2221      ;OUTER LOOP
2222 007754 005337 001406      DEC      ABORT      ;DECR MAIN LOOP COUNTER
2223 007760 001365      BNE      10$      ;CONTINUE UNTIL ZERO
2224
2225 007762 000400      BR      WTABO      ;TIMEOUT: ABORT PROGRAM
2226
2227      ;SYNC LOST, RESTART
2228 007764 011637 001412 001414 WTABO:  MOV      (SP), $STPC      ;SAVE PC
2229 007770 017737 171374      MOV      @DRVCSR, $ERRST      ;SAVE CSR
2230 007776 104017      ERROR+17      ;ALERT OPERATOR THAT WE'RE STUCK
2231 010000 104411      RSYNC      ;AND RESYNC
2232
2233      ;PROGRAM SYNCHRONIZED
2234 010002 000207 WTEXI:  RTS      PC      ;RETURN
2235

```

```

2237 .SBTTL INTERPROCESSOR PROGRAM RESYNCHRONIZATION ROUTINE
2238 ;:*****
2239 ;:RESYNC
2240 ;:*****
2241
2242 RESYNC:
2243 010004 JSR PC,CPUHI ;
2244 010004 004737 010072 MOV #STACK,SP ;:INITIALIZE STACK POINTER
2245 010010 012706 001100 TST MSTER ;:WAS THIS CPU STARTED AS MASTER?
2246 010014 005737 001420 BEQ 20$ ;:YES: SETUP DELAY FOR MASTER
2247 010020 001404 BR #14,ABORT ;:SLAVE CPU: WAIT FOR MASTER TO
2248 010022 012737 000014 001406 10$: WTSYNC ;:REACH ITS RESYNC ROUTINE
2249 010030 000403
2250 010032 012737 000030 001406 20$: MOV #30,ABORT ;:MASTER CPU: WAIT FOR SLAVE
2251
2252 WTSYNC:
2253 010040 005037 001404 10$: CLR TIME ;:CLEAR SUB LOOP COUNTER
2254 010044 005337 001404 15$: DEC TIME ;:DECR SUB LOOP COUNTER
2255 010050 001375 BNE 15$ ;:CONTINUE UNTIL ZERO
2256 010052 005337 001406 DEC ABORT ;:DECR MAIN LOOP COUNTER
2257 010056 001370 BNE 10$ ;:CONTINUE UNTIL ZERO
2258
2259 010060 000005 RESET ;:RESET PROCESSOR
2260 010062 104400 010217 TYPE ,SYNMSG ;:TYPE THE MESSAGE
2261 010066 000137 001776 JMP SYNCST ;:AND RESTART
2262
2263 ;:*****
2264 ;:CPUHI
2265 ;:*****
2266 CPUHI: 010072 005737 001454 TST LSI ;:IS THIS CPU AN LSI?
2267 010076 001403 BEQ 1$ ;:NO: UNIBUS
2268 010100 106427 000340 MTPS #340 ;:SET HIGHEST PRIORITY (LSI)
2269 010104 000207 RTS PC
2270 010106 012737 000340 177776 1$: MOV #340,PS ;:SET PRI07, IN UNIBUS
2271 010114 000207 RTS PC
2272
2273 ;:*****
2274 ;:CPULO
2275 ;:*****
2276 010116 005737 001454 CPULO: TST LSI ;:IS THIS CPU AN LSI?
2277 010122 001403 BEQ 1$ ;:NO: UNIBUS
2278 010124 106427 000000 MTPS #0 ;:SET LOWEST PRIORITY (LSI)
2279 010130 000207 RTS PC
2280 010132 012737 000000 177776 1$: MOV #0,PS ;:SET PRI00, IN UNIBUS
2281 010140 000207 RTS PC
2282

```

```

2284 .SBTTL ASCII MESSAGES
2285 .NLIST BEX
2286
2287 010142 200 TITLE: .ASCII <CRLF>
2288 010143 103 132 104 .ASCII \CZDAVAO DAV11-A/B INTERPROCESSOR EXERCISER\
2289 010215 200 000 .ASCIIZ <CRLF>
2290
2291 010217 200 052 122 SYNMSG: .ASCIIZ <CRLF>/*RESYNC.../<CRLF>
2292
2293 010234 124 105 123 EM1: .ASCII /TEST 1 STATUS: MASTER/<CRLF>
2294 010265 123 114 101 .ASCIIZ /SLAVE FAILED TO ECHO DBR CONTENTS/
2295 010327 124 105 123 EM2: .ASCII /TEST 2 STATUS: MASTER/<CRLF>
2296 010360 123 114 101 .ASCIIZ /SLAVE FAILED TO ECHO 'STAT' BITS/
2297 010421 123 124 101 EM3: .ASCII /STATUS: MASTER/<CRLF>
2298 010440 106 101 111 .ASCIIZ /FAILED TO INTR ON A 'DATI'/
2299 010473 123 124 101 EM4: .ASCII /STATUS: MASTER/<CRLF>
2300 010512 123 124 101 .ASCIIZ /STATUS ERROR ON 'DATI'/
2301 010541 123 124 101 EM5: .ASCII /STATUS: MASTER/<CRLF>
2302 010560 127 117 122 .ASCIIZ /WORD COUNT ERROR ON 'DATI'/
2303 010613 123 124 101 EM6: .ASCII /STATUS: MASTER/<CRLF>
2304 010632 102 125 106 .ASCIIZ /BUFFER ADRS ERROR ON 'DATI'/
2305 010666 123 124 101 EM7: .ASCII /STATUS: SLAVE/<CRLF>
2306 010704 106 101 111 .ASCIIZ /FAILED TO INTR ON 'DATO'/
2307 010735 123 124 101 EM10: .ASCII /STATUS: SLAVE/<CRLF>
2308 010753 123 124 101 .ASCIIZ /STATUS ERROR ON 'DATO'/
2309 011002 123 124 101 EM11: .ASCII /STATUS: SLAVE/<CRLF>
2310 011020 127 117 122 .ASCIIZ /WORD COUNT ERROR ON 'DATO'/
2311 011053 123 124 101 EM12: .ASCII /STATUS: SLAVE/<CRLF>
2312 011071 102 125 106 .ASCIIZ /BUFFER ADRS ERROR ON 'DATO'/
2313 011125 123 124 101 EM13: .ASCII /STATUS: SLAVE/<CRLF>
2314 011143 104 101 124 .ASCIIZ /DATA ERROR ON 'DATO'/
2315 011170 123 124 101 EM14: .ASCII /STATUS: SLAVE/<CRLF>
2316 011206 116 117 040 .ASCIIZ /NO ATTN INTR/
2317 011223 123 124 101 EM15: .ASCII /STATUS: SLAVE/<CRLF>
2318 011241 101 124 124 .ASCIIZ /ATTN STATUS ERROR/
2319 011263 123 124 101 EM16: .ASCII /STATUS: MASTER/<CRLF>
2320 011302 103 101 116 .ASCII ANNOT START COMMUNICATION WITH SLAVE-/
2321 011350 127 101 111 .ASCIIZ / WAITING FOR DSTATC/<CRLF>
2322 011374 123 124 125 EM17: .ASCIIZ / LUCK WAITING FOR COMPANION INTERLOCK/<CRLF>
2323
2324 011443 105 122 122 DH1: .ASCIIZ /ERRPC BUSADR EXPCT RCVD/
2325 011500 105 122 122 DH2: .ASCIIZ /ERRPC MEMADR EXPCT RCVD/
2326 011535 105 122 122 DH3: .ASCIIZ /ERRPC DRVCSR/
2327 .EVEN
2328
2329 011554 001116 001122 001124 DT1: $ERRPC,$BDADR,$GDDAT,$BDDAT,0
2330 011566 001116 001414 000000 DT2: $ERRPC,$ERRST,0
2331 011574 001412 001414 000000 DT3: $TSTPC,$ERRST,0
2332

```

2334
2335
2336
2337
2338 011602
2339
2340
2341
2342
2343 011632 000000
2344 000001

```
.SBTTL PROGRAM AREAS  
:*****  
: 'PATCH' IS A PATCH AREA FOR TEMPORARY CODE.  
:*****  
PATCH: .BLKW 12.  
  
:*****  
: 'DBUF' IS THE WORKING AREA IN MEM FOR ALL NPR OPERATIONS.  
:*****  
DBUF: 0 ;1ST ADRS OF DATA BUFFER  
.END
```


CZDAVAO DAV11 INTERPROC EXER MACRO M1200 06-DEC-82 12:16 PAGE L 5
CZDAVA.P11 06-DEC-82 12:20 SYMBOL TABLE 67-2

SEQ 0063

. ABS. 011634 000
ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 12735 WORDS (50 PAGES)

DYNAMIC MEMORY: 14114 WORDS (54 PAGES)

ELAPSED TIME: 00:00:38

ZDAVAO.BIN/DS:GBL/EN:ABS:AMA,ZDAVAO/CR:SYM/-SP=[6,1]SYSDGN,[6,4]ZDAVAO.P11

ZDAVAO		CREATED BY MACRO ON 6-DEC-82 AT 12:16		PAGE 1		M 5				CREF	V01	SEQ 0064
CZDAVA.P11		06-DEC-82 12:20		SYMBOL CROSS REFERENCE								
SYMBOL	VALUE	REFERENCES										
ABORT	001406	#24-577	*27-698	*27-705	*64-2195	*64-2204	*64-2214	*64-2222	*65-2247	*65-2250		
		*65-2256										
BIT0	= 000001	#21-437										
BIT0G	= 000001	#21-437	21-437	33-942	36-1070	43-1332						
BIT01	= 000002	#21-437	21-437									
BIT02	= 000004	#21-437	21-437									
BIT03	= 000010	#21-437	21-437									
BIT04	= 000020	#21-437	21-437									
BIT05	= 000040	#21-437	21-437									
BIT06	= 000100	#21-437	21-437									
BIT07	= 000200	#21-437	21-437									
BIT08	= 000400	#21-437	21-437									
BIT09	= 001000	#21-437	21-437									
BIT1	= 000002	#21-437										
BIT10	= 002000	#21-437	57-1913									
BIT11	= 004000	#21-437										
BIT12	= 010000	#21-437										
BIT13	= 020000	#21-437	57-1921									
BIT14	= 040000	#21-437										
BIT15	= 100000	#21-437										
BIT2	= 000004	#21-437										
BIT3	= 000010	#21-437										
BIT4	= 000020	#21-437										
BIT5	= 000040	#21-437										
BIT6	= 000100	#21-437										
BIT7	= 000200	#21-437										
BIT8	= 000400	#21-437										
BIT9	= 001000	#21-437										
BPTVEC	= 000014	#21-437										
CKSWR	= 104406	47-1536	57-1908	57-1909	57-1930	57-1939	#63-2183					
CLRBUF	005606	42-1273	45-1436	#50-1634								
CPUHI	010072	27-686	27-695	31-865	32-880	32-904	33-916	35-1010	35-1035	36-1046		
		38-1137	40-1201	41-1246	42-1263	42-1296	43-1308	45-1426	46-1460	46-1469		
		49-1599	49-1609	65-2243	#65-2266							
CPULO	010116	31-860	32-894	35-1025	41-1241	42-1279	45-1442	#65-2276				
CR	= 000015	#21-437	52-1732									
CRLF	= 000200	#21-437	51-1702	66-2287	66-2289	66-2291	66-2291	66-2293	66-2295	66-2297		
		66-2299	66-2301	66-2303	66-2305	66-2307	66-2309	66-2311	66-2313	66-2315		
		66-2317	66-2319	66-2321	66-2322							
DBUF	011632	24-586	24-590	24-594	24-598	50-1635	50-1648	#67-2343				
DDISP	= 177570	#21-437	22-460	25-632								
DELAY	005574	28-741	31-863	32-887	33-950	35-1017	37-1084	41-1244	#49-1620			
DH1	011443	23-463	23-469	23-475	23-481	23-487	23-493	23-499	23-505	23-511		
		23-517	23-529	23-535	#66-2324							
DH2	011500	23-523	#66-2325									
DH3	011535	23-541	23-547	#66-2326								
DISPLA	001142	#22-460	*25-632	*25-640	57-1912							
DISPRE	000174	#21-449	25-640									
DRVADR	001360	#24-554	25-653									
DRVBAR	001366	#24-563	32-893	33-941	33-946	35-1021	36-1069	36-1073	42-1278	43-1331		
		43-1335	45-1441									
DRVCSR	001370	#24-564	27-697	27-701	27-708	28-736	29-761	29-772	29-782	30-802		

SYMBOL	VALUE	REFERENCES	30-810	30-819	30-826	31-844	31-862	32-885	32-895	32-905	32-906
			30-810	30-819	30-826	31-844	31-862	32-885	32-895	32-905	32-906
			32-909	33-917	33-918	33-924	34-957	34-964	34-966	34-975	34-984
			35-1015	35-1026	35-1036	35-1037	35-1040	36-1047	36-1048	36-1053	36-1077
			37-1085	37-1092	37-1094	37-1103	37-1112	38-1139	38-1140	38-1143	38-1146
			38-1147	39-1159	39-1179	39-1190	40-1210	40-1224	40-1227	41-1243	42-1270
			42-1287	42-1297	42-1298	42-1301	43-1309	43-1310	43-1315	44-1373	44-1381
			44-1385	44-1388	44-1400	45-1434	45-1450	46-1461	46-1462	46-1465	46-1470
			46-1471	46-1476	47-1497	47-1505	47-1509	47-1512	47-1524	64-2199	64-2217
			64-2229								
DRVCTO	001374	#24-569	26-661	49-1600	49-1610						
DRVCT2	001376	#24-570	26-666	49-1601	49-1610	49-1611					
DRVDBR	001372	#24-565	26-659	28-728	28-730	28-742	29-781	31-843	39-1170	39-1171	
		40-1214									
DRVECT	001362	#24-558	26-662								
DRVISR	005666	31-858	32-882	35-1012	41-1239	42-1265	45-1428	#50-1667			
DRVWCR	001364	#24-562	25-652	31-861	32-892	33-929	33-932	35-1020	36-1058	36-1060	
		41-1242	42-1277	43-1320	43-1322	45-1440	46-1481	46-1484			
DSTAT A	= 004000	#21-445									
DSTAT B	= 002000	#21-446	34-976	37-1104	44-1389	47-1513					
DSTAT C	= 001000	#21-447	27-701	28-738	29-763	29-774	29-784	38-1140	38-1143	38-1147	
		39-1165	39-1181	42-1285							
DSWR	= 177570	#21-437	22-460	25-631							
DT1	011554	23-464	23-470	23-476	23-482	23-488	23-494	23-500	23-506	23-512	
		23-518	23-524	23-530	23-536	#66-2329					
DT2	011566	23-542	#66-2330								
DT3	011574	23-548	#66-2331								
EMTVEC	= 000030	#21-437	*25-622	*25-623							
EM1	010234	23-462	#66-2293								
EM10	010735	23-504	#66-2307								
EM11	011002	23-510	#66-2309								
EM12	011053	23-516	#66-2311								
EM13	011125	23-522	#66-2313								
EM14	011170	23-528	#66-2315								
EM15	011223	23-534	#66-2317								
EM16	011263	23-540	#66-2319								
EM17	011374	23-546	#66-2322								
EM2	010327	23-468	#66-2295								
EM3	010421	23-474	#66-2297								
EM4	010473	23-480	#66-2299								
EM5	010541	23-486	#66-2301								
EM6	010613	23-492	#66-2303								
EM7	010666	23-498	#66-2305								
EOPT	005352	#47-1536									
EOPTA	005514	47-1538	47-1540	48-1580	#48-1585						
EOPTB	005522	#48-1586	48-1587								
ERRFLG	001456	#24-602	*32-884	*33-949	34-962	*34-978	34-991	*35-1014	*36-1076	37-1090	
		*37-1106	37-1119	*42-1269	*44-1364	44-1383	*44-1391	44-1402	*45-1432	*46-1487	
		47-1507	*47-1515	47-1527							
ERROR	= 104000	#21-437	27-709	28-746	30-830	32-910	33-925	33-933	33-947	35-1041	
		36-1054	36-1061	36-1074	42-1302	43-1316	43-1323	43-1336	44-1362	46-1466	
		46-1477	46-1485	64-2230							
ERRVEC	= 000004	#21-437	25-629	*25-630	*25-643	*25-651					

ZDAVAO
CZDAVA.P11

CREATED BY MACRO ON 6-DEC-82 AT 12:16
06-DEC-82 12:20 SYMBOL CROSS REFERENCE

PAGE 3

B 6

CREF

V01

SEQ 0066

SYMBOL	VALUE	REFERENCES								
FNCT1	= 000002	#21-442	28-736	29-761	29-772	29-782	34-966	37-1094	38-1146	39-1179
		39-1190	44-1388	47-1512						
FNCT2	= 000004	#21-443	34-964	37-1092	44-1385	47-1509				
FNCT3	= 000010	#21-444	36-1077							
GNS	= *****	21-449	21-449							
GTSWR	= 104405	26-677	#63-2182							
HT	= 000011	#21-437	51-1700							
ICOUNT	001422	#24-583	*26-681	*47-1539	*47-1541					
INTFLG	001402	#24-575	*32-886	32-897	*35-1016	35-1028	*42-1271	42-1289	*45-1433	45-1452
		*50-1667								
IOTVEC	= 000020	#21-437								
LDBUF	005624	32-889	#50-1647							
LF	= 000012	#21-437	52-1736							
LSI	001454	#24-601	*25-644	*25-646	31-854	41-1236	65-2266	65-2276		
MINIT1	002014	27-688	#27-695							
MSTER	001420	#24-582	*25-607	*25-610	27-687	47-1537	65-2245			
MTPS	= 106427	#21-439								
MTST1	002100	27-702	#28-725	48-1588						
MTST2	002272	#30-801	30-831							
MTST3	002514	#32-884	34-994							
MTST4	003200	#35-1014	37-1122							
M3INI	002502	#32-880								
M4INI	003166	#35-1010								
PATCH	011602	#67-2338								
PIRQ	= 177772	#21-437								
PIRQVE	= 000240	#21-437								
PRO	= 000000	#21-437								
PR1	= 000040	#21-437								
PR2	= 000100	#21-437								
PR3	= 000140	#21-437								
PR4	= 000200	#21-437								
PR5	= 000240	#21-437								
PR6	= 000300	#21-437								
PR7	= 000340	#21-437								
PS	= 177776	#21-437	21-437	25-645	*65-2270	*65-2280				
PSW	= 177776	#21-437								
PWRVEC	= 000024	#21-437								
RDCHR	= 104407	62-2112	#65-2184							
RDLIN	= 104410	#63-2185								
RESVEC	= 000010	#21-437								
RESYNC	010004	63-2175	#65-2242							
RPRAM	001432	#24-589	42-1277	42-1278	42-1287	43-1326	43-1327	43-1339	43-1340	
RPRAM4	001446	#24-597	45-1440	45-1441	45-1450					
RSTVEC	005554	31-866	34-996	37-1124	41-1247	44-1407	47-1532	#49-1609		
RSYNC	= 104411	27-710	#63-2186	64-2231						
R6	=%000006	#21-437	21-437	*25-615	*25-616	25-617				
R7	=%000007	#21-437	21-437							
SAVE	001416	#24-581	*34-975	34-976	*37-1103	37-1104	*39-1170	39-1171	39-1172	*40-1224
		*40-1225	40-1227	*44-1381	44-1389	*47-1505	47-1513			
SCOPE	= 000004	#21-437								
SETUP2	001676	#26-657	26-660							
SETUP3	001722	#26-664	26-667							

SYMBOL	VALUE	REFERENCES								
TITLE	010142	26-672	#66-2287							
TKVEC	= 000060	#21-437								
TPVEC	= 000064	#21-437								
TRAPVE	= 000034	#21-437	*25-624	*25-625						
TRTVEC	= 000014	#21-437								
TYPDS	= 104404	48-1566	#63-2181							
TYPE	= 104400	26-672	48-1563	48-1567	51-1705	54-1804	56-1883	57-1915	57-1924	58-1950
		58-1968	58-1970	58-1974	58-1976	58-1981	58-1989	59-2016	59-2018	59-2022
		59-2032	59-2042	60-2064	62-2116	62-2120	62-2125	#63-2177	65-2260	
		58-1959	58-1986	59-2021	#63-2178					
TYPOC	= 104401									
TYPON	= 104403	#63-2180								
TYPOS	= 104402	#63-2179								
WTABO	007764	64-2207	64-2225	#64-2228						
WTEQ	007726	29-764	29-775	30-813	31-847	34-960	34-987	37-1088	37-1115	39-1182
		40-1208	44-1371	44-1398	47-1495	47-1522	#64-2213			
WTEXI	010002	64-2200	64-2218	#64-2234						
WTNE	007670	28-739	29-785	30-822	34-973	37-1101	39-1166	40-1222	42-1286	44-1380
		45-1449	47-1504	#64-2194						
WTSYNC	010040	65-2248	#65-2252							
XPRAM	001424	#24-585	32-892	32-893	32-895	33-936	33-937	36-1064	36-1065	
XPRAM4	001440	#24-593	35-1020	35-1021	35-1026					
\$AUTOB	001134	#22-460	59-2014							
\$BDADR	001122	#22-460	*28-730	*30-802	*32-909	*33-924	*33-932	*33-946	*35-1040	*36-1053
		*36-1060	*36-1073	*42-1301	*43-1315	*43-1322	*43-1335	*44-1360	*46-1465	*46-1476
		*46-1484	66-2329							
\$BDDAT	001126	#22-460	*28-742	28-743	*30-826	30-827	*32-905	*33-917	33-921	*33-929
		*33-941	*33-942	33-943	*35-1036	*36-1047	36-1051	*36-1058	*36-1069	*36-1070
		36-1071	*42-1297	*43-1309	43-1313	*43-1320	*43-1331	*43-1332	43-1333	*44-1359
		*46-1461	*46-1470	46-1474	*46-1481	46-1482	66-2329			
\$BELL	001160	#22-460	57-1916							
\$CHARC	006110	*51-1707	*51-1717	52-1725	*52-1734	#52-1739				
\$CKSWR	007046	#59-2006	63-2172							
\$CMTAG	001100	#22-460	25-615							
\$CM3	= 000000	#22-460	22-460							
\$CNTLG	007573	59-2017	#62-2136							
\$CNTLU	007566	59-2033	#62-2135							
\$CRLF	001165	#22-460	51-1706	57-1925	58-1951	58-1971	58-1977	58-1982	59-2043	
\$DBLK	006556	55-1847	56-1884	#56-1892						
\$DOAGN	005470	48-1559	48-1571	#48-1577						
\$DTBL	006546	55-1850	#56-1888							
\$ENDAD	005460	#48-1573								
\$ENDCT	005426	#48-1561								
\$ENDNG	005477	48-1564	#48-1582							
\$ENULL	005474	48-1568	#48-1581							
\$EOP	005376	#48-1553								
\$EOPCT	005420	#48-1558	48-1562							
\$ERFLG	001103	#22-460	*57-1910							
\$ERMAX	001115	#22-460								
\$ERROR	006566	25-622	#57-1907							
\$ERRPC	001116	#22-460	*57-1918	*57-1919	57-1920	58-1957	66-2329	66-2330		
\$ERRST	001414	#24-580	*27-708	*64-2229	66-2330	66-2331				
\$ERRTS	001170	#23-460	58-1965							

SYMBOL	VALUE	REFERENCES								
\$ERRTY	006712	57-1938	#58-1949							
\$ERTTL	001112	#22-460	*57-1917							
\$FILLC	001156	#22-460	51-1710							
\$FILLS	001155	#22-460								
\$GDADR	001120	#22-460								
\$GDDAT	001124	#22-460	*28-729	28-743	*30-804	30-827	*30-833	*32-907	*33-919	33-921
			*33-928	*33-936	*33-940	33-943	*35-1038	*36-1049	36-1051	*36-1057
			*36-1068	36-1071	*42-1299	*43-1311	43-1313	*43-1319	*43-1326	*43-1330
			*44-1361	*46-1463	*46-1472	46-1474	*46-1480	46-1482	66-2329	43-1333
\$GET42	005450	#48-1570								
\$GTSWR	007116	#59-2018	63-2171							
\$ICNT	001104	#22-460								
\$INTAG	001135	#22-460	59-2044							
\$ITEMB	001114	#22-460	*57-1920	58-1954						
\$LF	001166	#22-460	62-2126							
\$LPADR	001106	#22-460								
\$LPERR	001110	#22-460								
\$MNEW	007611	59-2023	#62-2138							
\$MSWR	007600	59-2019	#62-2137							
\$NULL	001154	#22-460	51-1712							
\$OCNT	006336	*53-1774	*54-1806	#54-1819						
\$OFILL	006337	*53-1768	*53-1772	53-1782	#54-1820					
\$OMODE	006340	*53-1769	*53-1773	53-1778	*53-1781	*54-1794	#54-1821			
\$PASS	001100	#22-460	*48-1555	*48-1556	48-1565					
\$QUES	001164	#22-460	60-2065	62-2117						
\$RDCHR	007330	#61-2079	63-2173							
\$RDLIN	007450	#62-2108	63-2174							
\$RTNAD	005472	#48-1580								
\$SWR	= 122000	#21-430	22-460	22-460	22-460					
\$TKB	001146	#22-460	59-2010	59-2028	61-2083	61-2087	61-2089			
\$TKS	001144	#22-460	59-2008	59-2026	59-2046	61-2081				
\$TN	= 000001	#21-429								
\$TPB	001152	#22-460	52-1731							
\$TPFLG	001157	#22-460	51-1688							
\$TPS	001150	#22-460	52-1729							
\$TRAP	007622	25-624	#63-2150							
\$TRPAD	007644	63-2155	#63-2165							
\$TSTNM	001102	#22-460	*48-1554	57-1912						
\$TSTPC	001412	#24-579	*64-2228	66-2331						
\$TTYIN	007556	62-2109	62-2110	62-2130	#62-2134					
\$TYPDS	006342	#55-1835	63-2170							
\$TYPE	005674	#51-1688	63-2166							
\$TYPEC	006044	51-1709	51-1716	52-1724	#52-1729	52-1730	60-2050			
\$TYPEX	006112	52-1735	52-1737	#52-1740						
\$TYPOC	006140	#53-1772	63-2167							
\$TYPON	006154	53-1771	#53-1774	63-2169						
\$TYPOS	006114	#53-1767	63-2168							