



IDENTIFICATION

PRODUCT CODE: AC-9486B-MC  
PRODUCT NAME: CZVSBB0 VS60 INST TST PT2  
DATE: MAY 1979  
MAINTAINER: DIAGNOSTIC ENGINEERING  
AUTHORS: R. SHOOP & R. MOORE

COPYRIGHT (C) 1976, 1979  
DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.

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 DOCUMENT 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.

## 0.0 TABLE OF CONTENTS

---

1.0	ABSTRACT
2.0	REQUIREMENTS
2.1	EQUIPMENT
2.2	STORAGE
3.0	LOADING PROCEDURE
4.0	STARTING PROCEDURE
4.1	STARTING ADDRESS 200
4.2	RESTART ADDRESS 204
4.3	STARTING ADDRESS 210
5.0	SWITCH REGISTER SETTINGS
6.0	ERROR REPORTING
6.1	ERROR COMMENTS
6.2	ERROR DATA
7.0	MISCELLANEOUS
7.1	VS60 BUS/VECTOR ADDRESS MODIFICATION
7.2	XXDP/APT NOTES
7.3	POWER FAIL
7.4	SINGLE VS60 TESTING
8.0	EXECUTION TIME
9.0	PROGRAM TEST DESCRIPTIONS
9.1	GENERAL
9.2	TYPICAL TEST SEQUENCE
10.0	LISTING

## 1.0 ABSTRACT

THIS PROGRAM IS THE SECOND OF A SERIES DESIGNED TO TEST STRICTLY LOGIC FUNCTIONS OF THE VS60 DISPLAY PROCESSOR. THIS DIAGNOSTIC ALL MOST EXCLUSIVELY TESTS THE VS60 GRAPHIC TYPE INSTRUCTIONS. THE BULK OF THE TESTING IS PREFORMED AT FULL SPEED WITH NO MAINTENANCE MODES ENABLED. HOWEVER, ON ODD NUMBERED PASSES, MAINTENANCE MODE SWITCH 4 (SERIAL MODE) IS ENABLED.

\*\*\* REV B INCLUDES ADDITIONAL TESTING FOR VECTOR SCALING AFTER ECO #VT48-0012 IS INSTALLED.  
G.P. MAY '79 \*\*\*

## 2.0 REQUIREMENTS

## 2.1 EQUIPMENT

- A. PDP-11 COMPUTER WITH AT LEAST 8K OF MEMORY
- B. I/O TERMINAL (I.E. ASR33 TTY)
- C. VS60 DISPLAY PROCESSOR

## 2.2 STORAGE

THIS PROGRAM OCCUPIES LOWER 8K OF MEMORY.

## 3.0 LOADING PROCEDURE

NORMAL PROCEDURE FOR LOADING A BINARY PROGRAM INTO MEMORY SHOULD BE FOLLOWED.

## 4.0 STARTING PROCEDURE

## 4.1 STARTING ADDRESS 200

LOADING ADDRESS 200 AND STARTING WILL IDENTIFY THE TEST, INITIALIZE THE SYSTEM, AND BEGIN TESTING.

\*\*\*\*\*  
NOTE: SWITCH REGISTER BIT 10 MUST BE SET TO INDICATE THE PRESENCE OF ECO VT48-0012.  
0 = ECO NOT INSTALLED (DEFAULT).  
1 = ECO IS INSTALLED.  
\*\*\*\*\*

## 4.2 RESTART ADDRESS 204

LOADING ADDRESS 204 AND STARTING WILL INITIALIZE THE SYSTEM AND BEGIN TESTING.

## 4.3 STARTING ADDRESS 210

LOADING ADDRESS 210 AND STARTING IS IDENTICAL TO START 200 EXCEPT THAT EXTERNAL SYNC TIMING WILL BE TESTED.

5.0 SWITCH REGISTER SETTINGS

SWITCH	FUNCTION
SW15 = 1	HALT ON ERROR
SW14 = 1	LOOP ON TEST
SW13 = 1	INHIBIT ERROR TYPEOUTS
SW12 = 1	HALT AT END OF DIAGNOSTIC
SW11 = 1	INHIBIT ITERATIONS - NORMALLY 2000(8) AFTER 1ST PASS
SW10 = 1	SCALING ECO (VT48-0012) IS INSTALLED.
SW09 = 1	LOOP ON ERROR
SW08 = 1	LOOP ON TEST IN SWR<7:0>

6.0 ERROR REPORTING

6.1 ERROR COMMENT

ALL ERRORS ARE ACCOMPANIED WITH AN ENGLISH LANGUAGE DESCRIPTIVE COMMENT AS TO THE TYPE OF FAILURE. FURTHER QUALIFICATION OF THE ERROR CAN BE OBTAINED IF NEEDED FROM THE COMMENT AT THE ERROR PC.

6.2 ERROR DATA

PC LISTING ADDRESS WHERE THE ERROR WAS DETECTED.  
 TSTNUM TEST NUMBER WHERE THE ERROR OCCURRED.  
 BUSADRS DISPLAY BUS ADDRESS WHERE THE RESULTANT DATA WAS EXPECTED  
 EXPCT DATA THAT WAS EXPECTED.  
 RCVD DATA THAT WAS RECEIVED.  
 TSTPC ADDRESS OF TEST WHICH FAILED TO COMPLETE (HUNG)  
 SCALE VALUE OF SCALE (1/4 TO 3 3/4) WHEN ERROR OCCURED  
 STARTVAL X OR Y VALUE THAT WAS SCALED

\*\*\*\*\*  
 NOTE: IF YOU GET SCALING ERRORS (SCALE ER), CHECK  
 THAT SWR BIT 10 = 1 IF ECO VT48-0012 IS INSTALLED.  
 OTHERWISE, SWR BIT 10 SHOULD BE 0 (DEFAULT).  
 \*\*\*\*\*

7.0 MISCELLANEOUS  
-----

7.1 VS60 BUS/VECTOR/PRIORITY ADDRESS MODIFICATION:  
MODIFY LOCATION 1252 (\$VECT1) IF BASE VECTOR PRIORITY/ADDRESS  
IS NOT 100320 (HI BYTE = PRI 4).  
MODIFY LOCATION 1256 (\$BASE) IF BASE REGISTER ADDRESS  
IS NOT 172000.

NOTE: A RESTART IS REQUIRED AFTER THE ABOVE ADDRESS MODIFICATION.  
THE ABOVE LOCATIONS ARE LOCATED IN THE 'APT MAILBOX-ETABLE'.

7.2 XXDP/APT NOTES:  
THIS DIAGNOSTIC IS CHAINABLE UNDER XXDP.  
THIS DIAGNOSTIC INCLUDES THE 'APT' SOFTWARE HOOKS HOWEVER, THEY  
HAVE NOT BEEN TESTED.

\*\*\*\*\*  
THIS DIAGNOSTIC SHOULD NOT BE CHAINED AFTER ECO VT48-0012  
SINCE THE SETTING OF SWR BIT 10 MAY CONFLICT WITH OTHER  
TESTS IN THE CHAIN.  
\*\*\*\*\*

7.3 POWER FAIL:  
A POWER FAILURE WILL CAUSE A RESTART MESSAGE ON POWER UP AT  
WHICH TIME THE PROGRAM IS RESTARTED AT THE BEGINNING.

7.4 SINGLE VS60 TESTING:  
THIS DIAGNOSTIC DOES NOT TEST MULTIPLE VS60'S.

8.0 EXECUTION TIME  
-----

EXECUTION TIME RANGES FROM ABOUT 3 SECONDS WITH NO ITERATIONS  
TO ABOUT 4 MINUTES WITH ITERATIONS ENABLED. AN 'END PASS' MESSAGE  
IS TYPED EACH PASS THRU THE DIAGNOSTIC.

## 9.0 PROGRAM TEST DESCRIPTIONS

-----

### 9.1 GENERAL

THE TABLE OF CONTENTS AT THE BEGINNING OF THE LISTING CONTAINS AT BRIEF DESCRIPTION OF EACH TEST. ADDITIONAL INFORMATION CAN BE OBTAINED FROM THE COMMENTS SUPPLIED AT EACH TEST. THE MAJORITY OF THE TESTS IN THIS DIAGNOSTIC ARE CONCERNED WITH THE GRAPHIC OR DISPLAY INSTRUCTIONS WHERE AT COMPLETION THE RESULTS CAN BE READ FROM THE X OR Y POSITION REGISTERS.

### 9.2 TYPICAL TEST SEQUENCE

TEST FORMAT MIGHT LOOK LIKE LIKE THIS:  
LOCATION 'BUFFER' IS LOADED WITH A GRAPHIC INSTRUCTION OR INSTRUCTIONS THAT WILL PROVIDE THE EXPECTED RESULT. A DISPLAY STOP INSTRUCTION IS LOADED AS THE LAST INSTRUCTION IN 'BUFFER'. '\$BDADR' IS LOADED WITH THE X OR Y REGISTER ADDRESS WHERE THE EXPECTED RESULT IS EXPECTED. LOCATION '\$GDDAT' IS LOADED WITH THE EXPECTED RESULT AND THEN THE DISPLAY IS STARTED AT LOCATION 'BUFFER'. WHEN THE DISPLAY STOP FLAG IS SET THE APPROPRIATE VS60 BUS ADDRESS DEFINED BY '\$BDADR' IS READ AND COMPARED TO '\$GDDAT'. IF AN ERROR IS DETECTED AN ERROR TRAP (EMP INSTR) IS PERFORMED ACCORDING TO THE ITEM NUMBER IN THE '\$ERRIB' (ERROR TABLE). THIS TABLE DIRECTS OR POINTS TO THE PERTINENT 'TYPEOUT' INFORMATION.

## 10.0 LISTING

-----

PROGRAM LISTING FOLLOWS.

```

13 .TITLE CZVSBB -- VS60 INSTRUCTION TEST PART II
(1) : *COPYRIGHT (C) 1979
(1) : *DIGITAL EQUIPMENT CORP.
(1) : *MAYNARD, MASS. 01754
(1) : *
(1) : *PROGRAM BY R. SHOOP
(1) : *
(1) : *THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
(1) : *PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
(1) : *
14 :
15 : REV B -- FIXES SCALING TESTS AFTER ECO VT48-0012.
16 : SET SWR BIT 10 (002000) IF ECO IS INSTALLED.
17 : G.P. MAY '79
18 :
19 :.SBTTL BASIC DEFINITIONS
(1)
(1) : *INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
(1) 001100 STACK= 1100
(1) .EQUIV EMT,ERROR ;;BASIC DEFINITION OF ERROR CALL
(1) .EQUIV IOT,SCOPE ;;BASIC DEFINITION OF SCOPE CALL
(1)
(1) : *MISCELLANEOUS DEFINITIONS
(1) 000011 HT= 11 ;;CODE FOR HORIZONTAL TAB
(1) 000012 LF= 12 ;;CODE FOR LINE FEED
(1) 000015 CR= 15 ;;CODE FOR CARRIAGE RETURN
(1) 000200 CRLF= 200 ;;CODE FOR CARRIAGE RETURN-LINE FEED
(1) 177776 PS= 177776 ;;PROCESSOR STATUS WORD
(1) .EQUIV PS,PSW
(1) 177774 STKLMT= 177774 ;;STACK LIMIT REGISTER
(1) 177772 PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
(1) 177570 DSWR= 177570 ;;HARDWARE SWITCH REGISTER
(1) 177570 DDISP= 177570 ;;HARDWARE DISPLAY REGISTER
(1)
(1) : *GENERAL PURPOSE REGISTER DEFINITIONS
(1) 000000 R0= %0 ;;GENERAL REGISTER
(1) 000001 R1= %1 ;;GENERAL REGISTER
(1) 000002 R2= %2 ;;GENERAL REGISTER
(1) 000003 R3= %3 ;;GENERAL REGISTER
(1) 000004 R4= %4 ;;GENERAL REGISTER
(1) 000005 R5= %5 ;;GENERAL REGISTER
(1) 000006 R6= %6 ;;GENERAL REGISTER
(1) 000007 R7= %7 ;;GENERAL REGISTER
(1) 000006 SP= %6 ;;STACK POINTER
(1) 000007 PC= %7 ;;PROGRAM COUNTER
(1)
(1) : *PRIORITY LEVEL DEFINITIONS
(1) 000000 PR0= 0 ;;PRIORITY LEVEL 0
(1) 000040 PR1= 40 ;;PRIORITY LEVEL 1
(1) 000100 PR2= 100 ;;PRIORITY LEVEL 2
(1) 000140 PR3= 140 ;;PRIORITY LEVEL 3
(1) 000200 PR4= 200 ;;PRIORITY LEVEL 4
(1) 000240 PR5= 240 ;;PRIORITY LEVEL 5
(1) 000300 PR6= 300 ;;PRIORITY LEVEL 6
(1) 000340 PR7= 340 ;;PRIORITY LEVEL 7

```

```
(1) ;*'SWITCH REGISTER' SWITCH DEFINITIONS
(1) 100000 SW15= 100000
(1) 040000 SW14= 40000
(1) 020000 SW13= 20000
(1) 010000 SW12= 10000
(1) 004000 SW11= 4000
(1) 002000 SW10= 2000
(1) 001000 SW09= 1000
(1) 000400 SW08= 400
(1) 000200 SW07= 200
(1) 000100 SW06= 100
(1) 000040 SW05= 40
(1) 000020 SW04= 20
(1) 000010 SW03= 10
(1) 000004 SW02= 4
(1) 000002 SW01= 2
(1) 000001 SW00= 1
(1) .EQUIV SW09,SW9
(1) .EQUIV SW08,SW8
(1) .EQUIV SW07,SW7
(1) .EQUIV SW06,SW6
(1) .EQUIV SW05,SW5
(1) .EQUIV SW04,SW4
(1) .EQUIV SW03,SW3
(1) .EQUIV SW02,SW2
(1) .EQUIV SW01,SW1
(1) .EQUIV SW00,SW0
(1) ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
(1) 100000 BIT15= 100000
(1) 040000 BIT14= 40000
(1) 020000 BIT13= 20000
(1) 010000 BIT12= 10000
(1) 004000 BIT11= 4000
(1) 002000 BIT10= 2000
(1) 001000 BIT09= 1000
(1) 000400 BIT08= 400
(1) 000200 BIT07= 200
(1) 000100 BIT06= 100
(1) 000040 BIT05= 40
(1) 000020 BIT04= 20
(1) 000010 BIT03= 10
(1) 000004 BIT02= 4
(1) 000002 BIT01= 2
(1) 000001 BIT00= 1
(1) .EQUIV BIT09,BIT9
(1) .EQUIV BIT08,BIT8
(1) .EQUIV BIT07,BIT7
(1) .EQUIV BIT06,BIT6
(1) .EQUIV BIT05,BIT5
(1) .EQUIV BIT04,BIT4
(1) .EQUIV BIT03,BIT3
(1) .EQUIV BIT02,BIT2
(1) .EQUIV BIT01,BIT1
(1) .EQUIV BIT00,BIT0
```

```
(1) ;*BASIC "CPU" TRAP VECTOR ADDRESSES
(1) 000004 ERRVEC= 4 ;:TIME OUT AND OTHER ERRORS
(1) 000010 RESVEC= 10 ;:RESERVED AND ILLEGAL INSTRUCTIONS
(1) 000014 TBITVEC=14 ;: "T" BIT
(1) 000014 TRTVEC= 14 ;:TRACE TRAP
(1) 000014 BPTVEC= 14 ;:BREAKPOINT TRAP (BPT)
(1) 000020 IOTVEC= 20 ;:INPUT/OUTPUT TRAP (IOT) **SCOPE**
(1) 000024 PWRVEC= 24 ;:POWER FAIL
(1) 000030 EMTVEC= 30 ;:EMULATOR TRAP (EMT) **ERROR**
(1) 000034 TRAPVEC=34 ;: "TRAP" TRAP
(1) 000060 TKVEC= 60 ;:TTY KEYBOARD VECTOR
(1) 000064 TPVEC= 64 ;:TTY PRINTER VECTOR
(1) 000240 PIRQVEC=240 ;:PROGRAM INTERRUPT REQUEST VECTOR
```

```

21      172000      ABASE= 172000      ; VS60 BASE REGISTER ADDRESS.
22      100320      AVECT1= 100320     ; PRI AND 1ST VECTOR ADDRESS.
23      165400      $SWR= 165400      ; REDEFINE SWITCHES USED.
24
25      .SBTTL  OPERATIONAL SWITCH SETTINGS
(1)      ;*
(1)      ;*      SWITCH      USE
(1)      ;*      -----
(1)      ;*      15      HALT ON ERROR
(1)      ;*      14      LOOP ON TEST
(1)      ;*      13      INHIBIT ERROR TYPEOUTS
(1)      ;*      12      HALT AT END OF DIAGNOSTIC
(1)      ;*      11      INHIBIT ITERATIONS
(1)      ;*      10      SCALING ECO IS INSTALLED
(1)      ;*      9      LOOP ON ERROR
(1)      ;*      8      LOOP ON TEST IN SWR<7:0>
26
27      .SBTTL  TRAP CATCHER
(1)      ;*=0
(1)      000000      ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A '+2,HALT'
(1)      ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
(1)      ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
(1)      ;*=174
(1)      000174      000000      DISPREG: .WORD 0      ;;SOFTWARE DISPLAY REGISTER
(1)      000176      000000      SWREG: .WORD 0      ;;SOFTWARE SWITCH REGISTER
(1)      .SBTTL  STARTING ADDRESS(ES)
(1)      000200      000137      002050      JMP @#BEGIN ;;JUMP TO STARTING ADDRESS OF PROGRAM
28      000204      000137      002460      JMP RESTRT ; 204 = RESTART.
29      000210      000137      002040      JMP BEGINA ; 210 = START W/ EXTERNAL SYNC.
    
```

```

31      .SBTTL ACT11 HOOKS
(1)
(2)      ;:*****
(1)      ;HOOKS REQUIRED BY ACT11
(1)      000214      $SVPC=.      ;SAVE PC
(1)      000046      .=46
(1) 000046 024470      $ENDAD      ;:1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
(1)      000052      .=52
(1) 000052 000000      .WORD 0      ;:2)SET LOC.52 TO ZERO
(1)      000214      .= $SVPC      ;: RESTORE PC
32
33      001000      .=1000
34      .SBTTL APT PARAMETER BLOCK
(1)
(2)      ;:*****
(1)      ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
(2)      ;:*****
(1)      001000      .$X=.      ;:SAVE CURRENT LOCATION
(1)      000024      .=24      ;:SET POWER FAIL TO POINT TO START OF PROGRAM
(1) 000024 000200      200      ;:FOR APT START UP
(1)      000044      .=44      ;:POINT TO APT INDIRECT ADDRESS PNTR.
(1) 000044 001000      $APTHDR ;:POINT TO APT HEADER BLOCK
(1)      001000      .=.$X      ;:RESET LOCATION COUNTER
(2)      ;:*****
(1)      ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
(1)      ;INTERFACE SPEC.
(1)
(1) 001000      $APTHD:
(1) 001000 000000      $HIBTS: .WORD 0      ;:TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
(1) 001002 001176      $MBADR: .WORD $MAIL ;:ADDRESS OF APT MAILBOX (BITS 0-15)
(1) 001004 000012      $STMT: .WORD 10.      ;:RUN TIM OF LONGEST TEST
(1) 001006 000036      $PASTM: .WORD 30.      ;:RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
(1) 001010 000000      $UNITM: .WORD 0      ;:ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
(1) 001012 000032      .WORD $ETEND-$MAIL/2 ;:LENGTH MAILBOX-ETABLE(WORDS)
35
  
```

```

36      .SBTTL COMMON TAGS
(1)
(2)      ;:*****
(1)      ;:THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
(1)      ;:USED IN THE PROGRAM.
(1)
(1)      001100      .=1100
(1)      001100      000000      $CMTAG:      .;START OF COMMON TAGS
(1)      001102      000      $TSTNM: .BYTE 0      .;CONTAINS THE TEST NUMBER
(1)      001103      000      $ERFLG: .BYTE 0      .;CONTAINS ERROR FLAG
(1)      001104      000000      $ICNT:  .WORD 0      .;CONTAINS SUBTEST ITERATION COUNT
(1)      001106      000000      $LPADR: .WORD 0      .;CONTAINS SCOPE LOOP ADDRESS
(1)      001110      000000      $LPERR: .WORD 0      .;CONTAINS SCOPE RETURN FOR ERRORS
(1)      001112      000000      $ERTTL: .WORD 0      .;CONTAINS TOTAL ERRORS DETECTED
(1)      001114      000      $ITEMB: .BYTE 0      .;CONTAINS ITEM CONTROL BYTE
(1)      001115      001      $ERMAX: .BYTE 1      .;CONTAINS MAX. ERRORS PER TEST
(1)      001116      000000      $ERRPC: .WORD 0      .;CONTAINS PC OF LAST ERROR INSTRUCTION
(1)      001120      000000      $GDADR: .WORD 0      .;CONTAINS ADDRESS OF 'GOOD' DATA
(1)      001122      000000      $BDADR: .WORD 0      .;CONTAINS ADDRESS OF 'BAD' DATA
(1)      001124      000000      $GDDAT: .WORD 0      .;CONTAINS 'GOOD' DATA
(1)      001126      000000      $BDDAT: .WORD 0      .;CONTAINS 'BAD' DATA
(1)      001130      000000      .WORD 0      .;RESERVED--NOT TO BE USED
(1)      001132      000000      .WORD 0
(1)      001134      000      $AUTOB: .BYTE 0      .;AUTOMATIC MODE INDICATOR
(1)      001135      000      $INTAG: .BYTE 0      .;INTERRUPT MODE INDICATOR
(1)      001136      000000      .WORD 0
(1)      001140      177570      SWR:      .WORD DSWR      .;ADDRESS OF SWITCH REGISTER
(1)      001142      177570      DISPLAY: .WORD DDISP      .;ADDRESS OF DISPLAY REGISTER
(1)      001144      177560      $TKS:      177560      .;TTY KBD STATUS
(1)      001146      177562      $TKB:      177562      .;TTY KBD BUFFER
(1)      001150      177564      $TPS:      177564      .;TTY PRINTER STATUS REG. ADDRESS
(1)      001152      177566      $TPB:      177566      .;TTY PRINTER BUFFER REG. ADDRESS
(1)      001154      000      $NULL:  .BYTE 0      .;CONTAINS NULL CHARACTER FOR FILLS
(1)      001155      002      $FILLS: .BYTE 2      .;CONTAINS # OF FILLER CHARACTERS REQUIRED
(1)      001156      012      $FILLC: .BYTE 12      .;INSERT FILL CHARS. AFTER A 'LINE FEED'
(1)      001157      000      $TPFLG: .BYTE 0      .;'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
(1)      001.60      000000      $REGAD: .WORD 0      .;CONTAINS THE ADDRESS FROM
(1)      .;WHICH ($REGO) WAS OBTAINED
(3)      001162      000000      $REGO:  .WORD 0      .;CONTAINS (($REGAD)+0)
(3)      001164      000000      $TMPO:  .WORD 0      .;USER DEFINED
(1)      001166      000000      $TIMES: 0      .;MAX. NUMBER OF ITERATIONS
(1)      001170      000000      $ESCAPE:0      .;ESCAPE ON ERROR ADDRESS
(1)      001172      077      $QUES:  .ASCII /?/      .;QUESTION MARK
(1)      001173      015      $CRLF:  .ASCII <15>      .;CARRIAGE RETURN
(1)      001174      000012      $LF:    .ASCIIZ <12>      .;LINE FEED
(2)      ;:*****
(2)      .SBTTL APT MAILBOX-ETABLE
(2)
(3)      ;:*****
(2)      .EVEN
(2)      001176      $MAIL:      .;APT MAILBOX
(2)      001176      000000      $MSGTY: .WORD AMSGTY .;MESSAGE TYPE CODE
(2)      001200      000000      $FATAL: .WORD AFATAL .;FATAL ERROR NUMBER
(2)      001202      000000      $TESTN: .WORD ATESTN .;TEST NUMBER
(2)      001204      000000      $PASS:  .WORD APASS .;PASS COUNT
  
```

(2)	001206	000000	\$DEVCT: .WORD	ADEVCT	::DEVICE COUNT
(2)	001210	000000	\$UNIT: .WORD	AUNIT	::I/O UNIT NUMBER
(2)	001212	000000	\$MSGAD: .WORD	AMSGAD	::MESSAGE ADDRESS
(2)	001214	000000	\$MSGLG: .WORD	AMSGLG	::MESSAGE LENGTH
(2)	001216		\$ETABLE:		::APT ENVIRONMENT TABLE
(2)	001216	000	\$ENV: .BYTE	AENV	::ENVIRONMENT BYTE
(2)	001217	000	\$ENVM: .BYTE	AENVM	::ENVIRONMENT MODE BITS
(2)	001220	000000	\$SWREG: .WORD	ASWREG	::APT SWITCH REGISTER
(2)	001222	000000	\$USWR: .WORD	AUSWR	::USER SWITCHES
(2)	001224	000000	\$CPUOP: .WORD	ACPUOP	::CPU TYPE,OPTIONS
(2)			*		BITS 15-11=CPU TYPE
(2)			*		11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
(2)			*		11/70=06,PDQ=07,Q=10
(2)			*		BIT 10=REAL TIME CLOCK
(2)			*		BIT 9=FLOATING POINT PROCESSOR
(2)			*		BIT 8=MEMORY MANAGEMENT
(2)	001226	000	\$MAMS1: .BYTE	AMAMS1	::HIGH ADDRESS,M.S. BYTE
(2)	001227	000	\$MTYP1: .BYTE	AMTYP1	::MEM. TYPE,BLK#1
(2)			*		MEM.TYPE BYTE -- (HIGH BYTE)
(2)			*		900 NSEC CORE=001
(2)			*		300 NSEC BIPOLAR=002
(2)			*		500 NSEC MOS=003
(2)	001230	000000	\$MADR1: .WORD	AMADR1	::HIGH ADDRESS,BLK#1
(2)			*		MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF 'TYPE' ABOVE
(2)	001232	000	\$MAMS2: .BYTE	AMAMS2	::HIGH ADDRESS,M.S. BYTE
(2)	001233	000	\$MTYP2: .BYTE	AMTYP2	::MEM. TYPE,BLK#2
(2)	001234	000000	\$MADR2: .WORD	AMADR2	::MEM.LAST ADDRESS,BLK#2
(2)	001236	000	\$MAMS3: .BYTE	AMAMS3	::HIGH ADDRESS,M.S.BYTE
(2)	001237	000	\$MTYP3: .BYTE	AMTYP3	::MEM. TYPE,BLK#3
(2)	001240	000000	\$MADR3: .WORD	AMADR3	::MEM.LAST ADDRESS,BLK#3
(2)	001242	000	\$MAMS4: .BYTE	AMAMS4	::HIGH ADDRESS,M.S.BYTE
(2)	001243	000	\$MTYP4: .BYTE	AMTYP4	::MEM. TYPE,BLK#4
(2)	001244	000000	\$MADR4: .WORD	AMADR4	::MEM.LAST ADDRESS,BLK#4
(2)	001246	100320	\$VECT1: .WORD	AVECT1	::INTERRUPT VECTOR#1,BUS PRIORITY#1
(2)	001250	000000	\$VECT2: .WORD	AVECT2	::INTERRUPT VECTOR#2BUS PRIORITY#2
(2)	001252	172000	\$BASE: .WORD	ABASE	::BASE ADDRESS OF EQUIPMENT UNDER TEST
(2)	001254	000000	\$DEVN: .WORD	ADEVN	::DEVICE MAP
(2)	001256	000000	\$CDW1: .WORD	ACDW1	::CONTROLLER DESCRIPTION WORD#1
(2)	001260	000000	\$CDW2: .WORD	ACDW2	::CONTROLLER DESCRIPTION WORD#2
(2)	001262		\$ETEND:		
(2)			.MEXIT		

```

(1) .SBTTL ERROR POINTER TABLE
(1)
(1) ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
(1) ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
(1) ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
(1) ;*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
(1) ;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
(1)
(1) ;*      EM      ;;POINTS TO THE ERROR MESSAGE
(1) ;*      DH      ;;POINTS TO THE DATA HEADER
(1) ;*      DT      ;;POINTS TO THE DATA
(1) ;*      DF      ;;POINTS TO THE DATA FORMAT
(1)
(1) $ERRTB:
(1) ;ITEM 1
(1)
(1) 37 001262
(1) 38
(1) 39 001262 030061 EM1 ;MAINT. SWITCH REGISTER
(1) 40 001264 032323 DH1 ;ERRPC TSTNUM BUSADR EXPCT RCVD
(1) 41 001266 032470 DT1 ;$ERRPC TSTNUM $BDADR $GDDAT $BDDAT
(1) 42 001270 000000 0
(1) 43
(1) 44 ;ITEM 2
(1) 45
(1) 46 001272 030103 EM2 ;LOADING D.P.C. REGISTER <STATIC>
(1) 47 001274 032323 DH1 ;ERRPC TSTNUM BUSADR EXPCT RCVD
(1) 48 001276 032470 DT1 ;$ERRPC TSTNUM $BDADR $GDDAT $BDDAT
(1) 49 001300 000000 0
(1) 50
(1) 51 ;ITEM 3
(1) 52
(1) 53 001302 030137 EM3 ;LOADING RELOCATE REGISTER
(1) 54 001304 032323 DH1 ;ERRPC TSTNUM BUSADR EXPCT RCVD
(1) 55 001306 032470 DT1 ;$ERRPC TSTNUM $BDADR $GDDAT $BDDAT
(1) 56 001310 000000 0
(1) 57
(1) 58 ;ITEM 4
(1) 59
(1) 60 001312 030164 EM4 ;READING D.P.C. WITH RELOCATE SET
(1) 61 001314 032323 DH1 ;ERRPC TSTNUM BUSADR EXPCT RCVD
(1) 62 001316 032470 DT1 ;$ERRPC TSTNUM $BDADR $GDDAT $BDDAT
(1) 63 001320 000000 0
(1) 64
(1) 65 ;ITEM 5
(1) 66
(1) 67 001322 030231 EM5 ;OFFSET INDICATOR FAILED TO SET/CLEAR
(1) 68 001324 032323 DH1 ;ERRPC TSTNUM BUSADR EXPCT RCVD
(1) 69 001326 032470 DT1 ;$ERRPC TSTNUM $BDADR $GDDAT $BDDAT
(1) 70 001330 000000 0
    
```

72			:ITEM 6		
73					
74	001332	030270	EM6	:OFFSET POLARITY BIT FAILED TO SET/CLEAR	
75	001334	032323	DH1	:ERRPC TSTNUM BUSADR EXPCT RCVD	
76	001336	032470	DT1	:\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT	
77	001340	000000	0		
78					
79			:ITEM 7		
80					
81	001342	030332	EM7	:OFFSET INSTRUCTION FAILED TO LOAD DYNAMIC REGISTER	
82	001344	032323	DH1	:ERRPC TSTNUM BUSADR EXPCT RCVD	
83	001346	032470	DT1	:\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT	
84	001350	000000	0		
85					
86			:ITEM 10		
87					
88	001352	030402	EM10	:OFFSET INSTRUCTION FAILED TO LOAD EXTENDED POSITION BIT	
89	001354	032323	DH1	:ERRPC TSTNUM BUSADR EXPCT RCVD	
90	001356	032470	DT1	:\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT	
91	001360	000000	0		
92					
93			:ITEM 11		
94					
95	001362	030460	EM11	:SCALE ERROR	
96	001364	032406	DH11	:ERRPC TSTNUM EXPCT RCVD SCALE STARTVAL	
97	001366	032512	DT11	:\$ERRPC TSTNUM \$GDDAT \$BDDAT \$TMPC \$REGO	
98	001370	000000	0		
99					
100			:ITEM 12		
101					
102	001372	030471	EM12	:ABSOLUTE VECTOR FAILED TO LOAD X OR Y POS REG	
103	001374	032323	DH1	:ERRPC TSTNUM BUSADR EXPCT RCVD	
104	001376	032470	DT1	:\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT	
105	001400	000000	0		
106					
107			:ITEM 13		
108					
109	001402	030544	EM13	:GRAPHPLOT INC REG ER	
110	001404	032323	DH1	:ERRPC TSTNUM BUSADR EXPCT RCVD	
111	001406	032470	DT1	:\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT	
112	001410	000000	0		
113					
114			:ITEM 14		
115					
116	001412	030571	EM14	:LOADING X POSITION REGISTER	
117	001414	032323	DH1	:ERRPC TSTNUM BUSADR EXPCT RCVD	
118	001416	032470	DT1	:\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT	
119	001420	000000	0		

121			:ITEM 15						
122									
123	001422	030613	EM15		:LOADING Y POSITION REGISTER				
124	001424	032323	DH1		:ERRPC TSTNUM BUSADR EXPCT RCVD				
125	001426	032470	DT1		;\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT				
126	001430	000000	0						
127									
128			:ITEM 16						
129									
130	001432	030635	EM16		:INCREMENTING DELTA X OR DELTA Y REGISTERS				
131	001434	032323	DH1		:ERRPC TSTNUM BUSADR EXPCT RCVD				
132	001436	032470	DT1		;\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT				
133	001440	000000	0						
134									
135			:ITEM 17						
136									
137	001442	030701	EM17		:DECREMENTING DELTA X OR DELTA Y REGISTER				
138	001444	032323	DH1		:ERRPC TSTNUM BUSADR EXPCT RCVD				
139	001446	032470	DT1		;\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT				
140	001450	000000	0						
141									
142			:ITEM 20						
143									
144	001452	030745	EM20		:EDGE INDICATOR UPON EXCEEDING AN EDGE				
145	001454	032323	DH1		:ERRPC TSTNUM BUSADR EXPCT RCVD				
146	001456	032470	DT1		;\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT				
147	001460	000000	0						
148									
149			:ITEM 21						
150									
151	001462	031013	EM21		:LOADING CHARACTER REGISTER				
152	001464	032323	DH1		:ERRPC TSTNUM BUSADR EXPCT RCVD				
153	001466	032470	DT1		;\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT				
154	001470	000000	0						
155									
156			:ITEM 22						
157									
158	001472	031034	EM22		:INCORRECT OR UNEXPECTED CHANGE IN X OR Y POSITION REGIS				
159	001474	032323	DH1		:ERRPC TSTNUM BUSADR EXPCT RCVD				
160	001476	032470	DT1		;\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT				
161	001500	000000	0						
162									
163			:ITEM 23						
164									
165	001502	031123	EM23		:WRONG DELTA X - SCALED CHARACTER MODE				
166	001504	032323	DH1		:ERRPC TSTNUM BUSADR EXPCT RCVD				
167	001506	032470	DT1		;\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT				
168	001510	000000	0						

170			:ITEM 24	
171				
172	001512	031164	EM24	:WRONG DELTA Y - SCALED CHARACTER MODE
173	001514	032323	DH1	:ERRPC TSTNUM BUSADR EXPCT RCVD
174	001516	032470	DT1	:\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT
175	001520	000000	0	
176				
177			:ITEM 25	
178				
179	001522	031225	EM25	:WRONG DELTA Y - SCALED CHARACTE MODE ROTATED
180	001524	032323	DH1	:ERRPC TSTNUM BUSADR EXPCT RCVD
181	001526	032470	DT1	:\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT
182	001530	000000	0	
183				
184			:ITEM 26	
185				
186	001532	031276	EM26	:WRONG DELTA X - SCALED CHARACTE MODE ROTATED
187	001534	032323	DH1	:ERRPC TSTNUM BUSADR EXPCT RCVD
188	001536	032470	DT1	:\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT
189	001540	000000	0	
190				
191			:ITEM 27	
192				
193	001542	031347	EM27	:GRAPHPLOT INC FAILED TO SET UP X OR Y POS REG
194	001544	032323	DH1	:ERRPC TSTNUM BUSADR EXPCT RCVD
195	001546	032470	DT1	:\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT
196	001550	000000	0	
197				
198			:ITEM 30	
199				
200	001552	031425	EM30	:BASIC VECTOR FAILED TO LOAD X OR Y POS REG
201	001554	032323	DH1	:ERRPC TSTNUM BUSADR EXPCT RCVD
202	001556	032470	DT1	:\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT
203	001560	000000	0	
204				
205			:ITEM 31	
206				
207	001562	031500	EM31	:BASIC SHORT VECTOR FAILED TO LOAD X OR Y POS REG
208	001564	032323	DH1	:ERRPC TSTNUM BUSADR EXPCT RCVD
209	001566	032470	DT1	:\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT
210	001570	000000	0	
211				
212			:ITEM 32	
213				
214	001572	031561	EM32	:RESET FAILED TO CLEAR A REGISTER
215	001574	032323	DH1	:ERRPC TSTNUM BUSADR EXPCT RCVD
216	001576	032470	DT1	:\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT
217	001600	000000	0	

219			:ITEM 33		
220					
221	001602	031615	EM33	:EDGE FLAG INTR ER	
222	001604	032323	DH1	:ERRPC TSTNUM BUSADR EXPCT RCVD	
223	001606	032470	DT1	:\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT	
224	001610	000000	0		
225					
226			:ITEM 34		
227					
228	001612	031637	EM34	:INTR PRIORITY FAILURE USING INTERNAL STOP	
229	001614	032323	DH1	:ERRPC TSTNUM BUSADR EXPCT RCVD	
230	001616	032470	DT1	:\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT	
231	001620	000000	0		
232					
233			:ITEM 35		
234					
235	001622	031711	EM35	:SHIFT-OUT BIT SET IN ERROR	
236	001624	032323	DH1	:ERRPC TSTNUM BUSADR EXPCT RCVD	
237	001626	032470	DT1	:\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT	
238	001630	000000	0		
239					
240			:ITEM 36		
241					
242	001632	031744	EM36	:SHIFT-OUT BIT FAILED TO SET	
243	001634	032323	DH1	:ERRPC TSTNUM BUSADR EXPCT RCVD	
244	001636	032470	DT1	:\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT	
245	001640	000000	0		
246					
247			:ITEM 37		
248					
249	001642	032000	EM37	:INTERNAL STOP FLAG FAILED TO SET	
250	001644	032370	DH2	:ERRPC TSTPC	
251	001646	032504	DT2	:\$ERRPC \$LPADR	
252	001650	000000	0		
253					
254			:ITEM 40		
255					
256	001652	032041	EM40	: 'FRAMES PER SECOND' SYNC ER	
257	001654	032323	DH1	:ERRPC TSTNUM BUSADR EXPCT RCVD	
258	001656	032470	DT1	:\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT	
259	001660	000000	0		
260					
261			:ITEM 41		
262					
263	001662	032075	EM41	:LONG VECTOR FAILED TO SET UP X OR Y POS REG	
264	001664	032323	DH1	:ERRPC TSTNUM BUSADR EXPCT RCVD	
265	001666	032470	DT1	:\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT	
266	001670	000000	0		
267					
268			:ITEM 42		
269	001672	032151	EM42	:POINT OR ABS VECTOR WITH OFFSET FAILED TO SET UP X OR Y	
270	001674	032323	DH1	:ERRPC TSTNUM BUSADR EXPCT RCVD	
271	001676	032470	DT1	:\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT	
272	001700	000000	0		
273					
274			:ITEM 43		

275				
276	001702	032245	EM43	:SUPER/SUBSCRIPT CHAR SCALE OR DELTA X OR Y ER
277	001704	032323	DH1	:ERRPC TSTNUM BUSADR EXPCT RCVD
278	001706	032470	DT1	:SERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT
279	001710	000000	0	
280				
281				
282				:COMMON PROGRAM TAGS AND STORAGE LOCATIONS
283				
284	001712	032530	DBUF: BUFFER	:FIRST WORD IN THE DISPLAY BUFFER
285	001714	032532	DBUF1: BUFFER+2	:SECOND WORD
286	001716	032534	DBUF2: BUFFER+4	:THIRD WORD
287	001720	032536	DBUF3: BUFFER+6	:FOURTH WORD
288	001722	032540	DBUF4: BUFFER+10	:FIFTH WORD
289	001724	032542	DBUF5: BUFFER+12	:SIXTH WORD
290	001726	000000	DSAVE: 0	:TEMP REG.
291	001730	000000	DSAVE1: 0	
292	001732	000000	SIZE: 0	:BUFFER SIZE
293	001734	000000	TSTNUM: 0	:CONTAINS TEST NO. ON ERROR
294				
295	001736	000200	BRLEV1: 200	:CONTAINS BR LEVEL TAKEN FROM \$VECT1.
296				
297	001740	000007	CHSZ0: 7.	:CHARACTER SCALE CHARACTER SIZE VALUES
298	001742	000016	CHSZ1: 14.	
299	001744	000025	CHSZ2: 21.	
300	001746	000034	CHSZ3: 28.	
301				
302	001750	000014	LFSZ0: 12.	:CHARACTER SCALE LINE FEED SIZE VALUES
303	001752	000030	LFSZ1: 24.	
304	001754	000056	LFSZ2: 46.	
305	001756	000076	LFSZ3: 62.	
306				
322				

```

324      ;VS-60 ADDRESSES AND VECTORS POINTERS
325
326 001760 172000      DPC:      172000      ;DISPLAY PC REGISTER
327 001762 172002      SREG0:    172002      ;DISPLAY STATUS REGISTER
328 001764 172004      XPOS:     172004      ;X AXIS REGISTER AND GRAPHPLOT REG <READ ONLY>
329 001766 172006      YPOS:     172006      ;Y AXIS REGISTER AND CHARACTER REG <READ ONLY>
330 001770 172010      RLO:      172010      ;RELOCATION REGISTER
331 001772 172012      SREG1:    172012      ;MISC STATUS REGISTER #1
332 001774 172014      XDOFF:    172014      ;X DYNAMIC OFFSET REGISTER
333 001776 172016      YDOFF:    172016      ;Y DYNAMIC OFFSET REGISTER
334 002000 172020      ANAME:    172020      ;ASSOCIATIVE NAME REGISTER
335 002002 172022      CONS:     172022      ;CONSOLE STATUS REGISTER
336 002004 172024      DNAME:    172024      ;DISPLAY NAME REGISTER
337 002006 172026      STKVAL:   172026      ;VALUE OF CURRENT STACK WORD
338 002010 172030      TERMCH:   172030      ;TERMINATE CHARACTER REGISTER
339 002012 172032      STKPT:    172032      ;STACK POINTER REGISTER
340 002014 172034      ZPOS:     172034      ;Z POSITION REGISTER <READ ONLY>
341 002016 172036      ZDOFF:    172036      ;Z DYNAMIC OFFSET REGISTER
342
343 002020 000320      DDONE:    320         ;DISPLAY STOP <DONE> VECTOR
344 002022 000322      DDONE1:   322         ;
345
346 002024 000324      LPVCT:    324         ;DISPLAY LIGHT PEN VECTOR
347 002026 000326      LPVCT1:   326         ;
348
349 002030 000330      TIMEVT:   330         ;DISPLAY TIME-OUT <NXM.> ERROR VECTOR
350 002032 000332      TMEVT1:   332         ; OR 'SHIFT-OUT' VECTOR
351
352 002034 000334      NAMEVT:   334         ;NAME MATCH VECTOR
353 002036 000336      NAMEV1:   336
  
```

```

355 ;SOFTWARE INITIALIZATION ROUTINE
356
358 002040 012737 177777 024356 BEGINA: MOV # -1, EXSYNC ;(CHECK FOR EXT SYNC TMG (START 210)
359 002046 000402 BR BEGIN1 ;SKIP OVER CLR
360 002050 005037 024356 BEGIN: CLR EXSYNC ;DON'T CHECK FOR EXT TMG (START 200)
361 002054 BEGIN1:
(1) .SBTTL INITIALIZE THE COMMON TAGS
(1) ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
(1) 002054 012706 001100 MOV # $CMTAG, R6 ;;FIRST LOCATION TO BE CLEARED
(1) 002060 005026 CLR (R6)+ ;;CLEAR MEMORY LOCATION
(1) 002062 022706 001140 CMP #SWR, R6 ;;DONE?
(1) 002066 001374 DNE -6 ;;LOOP BACK IF NO
(1) 002070 012706 001100 MOV #STACK, SP ;;SETUP THE STACK POINTER
(1) ;;INITIALIZE A FEW VECTORS
(1) 002074 012737 025164 000020 MOV # $SCOPE, @IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
(1) 002102 012737 000340 000022 MOV #340, @IOTVEC+2 ;;LEVEL 7
(1) 002110 012737 025710 000030 MOV # $ERROR, @EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
(1) 002116 012737 000340 000032 MOV #340, @EMTVEC+2 ;;LEVEL 7
(1) 002124 012737 027616 000034 MOV # $TRAP, @TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
(1) 002132 012737 000340 000036 MOV #340, @TRAPVEC+2 ;;LEVEL 7
(1) 002140 012737 026704 000024 MOV # $PWRDN, @PWRVEC ;;POWER FAILURE VECTOR
(1) 002146 012737 000340 000026 MOV #340, @PWRVEC+2 ;;LEVEL 7
(1) 002154 013737 024436 024430 MOV $ENDCT, $EOPCT ;;SETUP END-OF-PROGRAM COUNTER
(1) 002162 005037 001166 CLR $TIMES ;;INITIALIZE NUMBER OF ITERATIONS
(1) 002166 005037 001170 CLR $ESCAPE ;;CLEAR THE ESCAPE ON ERROR ADDRESS
(1) 002172 112737 000001 001115 MOVB #1, $ERMAX ;;ALLOW ONE ERROR PER TEST
(1) 002200 012737 002200 001106 MOV #., $LPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
(1) 002206 012737 002206 001110 MOV #., $LPERR ;;SETUP THE ERROR LOOP ADDRESS
(2) ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
(2) ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
(2) 002214 013746 000004 MOV @ERRVEC, -(SP) ;;SAVE ERROR VECTOR
(2) 002220 012737 002254 000004 MOV #64$, @ERRVEC ;;SET UP ERROR VECTOR
(2) 002226 012737 177570 001140 MOV #DSWR, SWR ;;SETUP FOR A HARDWARE SWITCH REGISTER
(2) 002234 012737 177570 001142 MOV #DDISP, DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
(2) 002242 022777 177777 176670 CMP # -1, @SWR ;;TRY TO REFERENCE HARDWARE SWR
(2) 002250 001012 BNE 66$ ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
(2) ;;AND THE HARDWARE SWR IS NOT -1
(2) 002252 000403 BR 65$ ;;BRANCH IF NO TIMEOUT
(2) 002254 012716 002262 64$: MOV #65$, (SP) ;;SET UP FOR TRAP RETURN
(2) 002260 000002 RTI
(2) 002262 012737 000176 001140 65$: MOV #SWREG, SWR ;;POINT TO SOFTWARE SWR
(2) 002270 012737 000174 001142 MOV #DISPREG, DISPLAY
(2) 002276 012637 000004 66$: MOV (SP)+, @ERRVEC ;;RESTORE ERROR VECTOR
(1)
(2) 002302 005037 001204 CLR $PASS ;;CLEAR PASS COUNT
(2) 002306 132737 000200 001217 BITB #APTSIZE, $ENVM ;;TEST USER SIZE UNDER APT
(2) 002314 001403 BEQ 67$ ;;YES, USE NON-APT SWITCH
(2) 002316 012737 001220 001140 MOV # $SWREG, SWR ;;NO, USE APT SWITCH REGISTER
(2) 002324 67$:
362 002324 013700 001246 SETUP1: MOV $VECT1, R0 ; GET BUS PRIORITY LEVEL.
363 002330 000300 SWAB R0
364 002332 042700 177437 BIL #^C340, R0 ; MASK PSW BITS
365 002336 010037 001736 MOV R0, BRLEV1 ; AND SET BR LEVEL.
366 002342 012700 001760 MOV #DPC, R0 ;SET UP VS60 REG ADRS POINTERS
367 002346 013701 001252 MOV $BASE, R1
368 002352 010120 SETUP2: MOV R1, (R0)+

```

```

369 002354 062701 000002      ADD      #2,R1
370 002360 022700 002020      CMP      #DDONE,R0      ;TEST FOR LAST ADDRESS
371 002364 001372              BNE      SETUP2
372 002366 012700 002020      MOV      #DDONE,R0      ;SET UP VS60 VECTOR ADRS POINTFRS
373 002372 013701 001246      MOV      $VECT1,R1
374 002376 042701 160000      BIC      #160000,R1      ;MASK OF BITS OF PRIORITY
375 002402 010120      SETUP3: MOV      R1,(R0)+
376 002404 062701 000002      ADD      #2,R1
377 002410 022700 002040      CMP      #BEGINA,R0      ;TEST FOR LAST VECTOR
378 002414 001372              BNE      SETUP3
379 002416 004737 024634      JSR      PC,RSTVEC      ;GO SET UP VS60 VECTOR ADRS'S WITH HLTS
380 002422 004737 027124      JSR      PC,$SIZE      ;DETERMINE LAST LOCATION
381 002426 013737 027220      MOV      $LSTAD,SIZE      ;LOAD LAST BUS ADDRESS
382 002434 162737 006000      SUB      #6000,SIZE      ;SAVE LOADERS AND XXDP
383 002442 005737 000042      TST      @#42      ;TEST IF RUNNING CHAIN MODE UNDER XXDP
384 002446 001004              BNE      RESTRT      ;BR IF SO
385 002450 104401              TYPE
386 002452 027666              HEADER
387 002454 104401              TYPE
388 002456 027752      ECOMSG
389 002460 012737 000340      177776 RESTRT: MOV      #340,PSW      ; AND TELL HIM ABOUT THE ECO.
390 002466 012706 001100      MOV      #STACK,SP      ;SET PRIORITY TO HIGHEST LEVEL
391 002472 000005              RESET      ;ALWAYS RESET STACK POINTER
392 002474 105037 001102      CLRB     $STSTM      ;INITILIZE VS60 BEFORE STARTING TESTS
393                                ;RESET TEST NUMBER
  
```

395  
 396  
 397  
 398  
 399  
 (3)  
 (3)  
 (2)  
 (2)  
 400  
 401  
 402  
 403  
 404  
 405  
 406  
 407  
 408  
 409  
 410  
 411  
 412  
 (3)  
 (3)  
 (2)  
 (2)  
 413  
 414  
 415  
 416  
 417  
 418  
 419  
 420  
 421  
 422  
 423  
 424  
 425  
 (3)  
 (3)  
 (2)  
 (2)  
 426  
 427  
 428  
 429  
 430  
 431  
 432  
 433  
 434  
 435  
 436  
 437  
 438

.SBTTL  
 .SBTTL THESE TESTS USE MAINT SWITCH BITS  
 .SBTTL

```

:*****
:*TEST 1      FLOAT A ONE THRU RELOCATE REG
:*****
TST1:  SCOPE
        MOV      #1,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
        MOV      #1$,$LPERR     ;;SET UP SCOPE LOOP ADRS
        MOV      RLO,$BDADR     ;;SET UP RLO ADRS
        MOV      #4000,$GDDAT   ;;SET UP MSB TO A ONE
1$:    MOV      $GDDAT,@RLO     ;;LOAD UP RLO REG
        MOV      @RLO,$BDDAT    ;;READ IT BACK
        CMP      $GDDAT,$BDDAT  ;;IS IT CORRECT?
        BEQ      2$             ;;BR IF OK
        ERROR    3              ;;RELOCATE REG READ/WRITE ER
2$:    ASR      $GDDAT          ;;SHIFT ONE BIT RIGHT
        BNE      1$             ;;BR IF NOT DONE
        CLR      @RLO           ;;INSURE 0 RELOCATE REG BEFORE ADVANCING
    
```

```

:*****
:*TEST 2      TEST THAT ALL MAINT TYPE BITS ARE READ/WRITEABLE
:*****
TST2:  SCOPE
        MOV      #2,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
        MOV      STKPT,$BDADR   ;;SET UP REG 32 ADRS
        MOV      #1$,$LPERR     ;;SET UP SCOPE LOOP ADRS
        MOV      #170000,$GDDAT ;;EXPECT ALL BITS INITIALLY
1$:    MOV      $GDDAT,@STKPT   ;;LOAD UP MAINT BITS
        MOV      @STKPT,$BDDAT  ;;READ THEM BACK
        BIC      #7777,$BDDAT   ;;SAVE ONLY MAINT SWITCHES
        CMP      $GDDAT,$BDDAT  ;;ARE THEY CORRECT?
        BEQ      2$             ;;BR IF SO
        ERROR    1              ;;MAINT SWITCH READ/WRITE ERROR
2$:    SUB      #10000,$GDDAT   ;;GO TO NEXT PATTERN
        BPL      1$             ;;BR IF ALL STATES NOT TESTED
    
```

```

:*****
:*TEST 3      FLOAT A 1 ACROSS THE D.P.C.
:*****
TST3:  SCOPE
        MOV      #3,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
        MOV      DPC,$BDADR     ;;SET UP REG ADRS 00
        MOV      #1$,$LPERR     ;;SET UP SCOPE LOOP ADRS
        MOV      #BIT1,$GDDAT   ;;LOAD EXPECTED
        MOV      #BIT12,@STKPT  ;;SET MAINT SWITCH #1
1$:    MOV      $GDDAT,@DPC     ;;LOAD D.P.C.
        MOV      @DPC,$BDDAT    ;;READ D.P.C. INTO $BDDAT
        CMP      $GDDAT,$BDDAT  ;;COMPARE VALUE EXPECTED TO READ
        BEQ      2$             ;;BR IF SAME
        ERROR    2              ;;FAILED TO LOAD DPC WITH A
        FLOATING 1              ;;FLOATING 1
2$:    ASL      $GDDAT          ;;CHANGE DATA EXPECTED
        BCC      1$             ;;BR IF NOT COMPLETED
    
```

439  
 (3)  
 (3)  
 (2) 002764 000004  
 (1) 002766 012737 000100 001166  
 (2) 002774 012737 000004 001202  
 440 003002 012777 010000 177002  
 441 003010 013737 001760 001122  
 442 003016 005077 176736  
 443 003022 012737 000100 001124  
 444 003030 012700 000001  
 445 003034 010077 176730  
 446 003040 012737 003034 001110  
 447 003046 017737 176706 001126  
 448 003054 023737 001124 001126  
 449 003062 001401  
 450 003064 104004  
 451 003066 062737 000100 001124  
 452 003074 001402  
 453 003076 005200  
 454 003100 000755  
 455 003102 005077 176662  
 456  
 457  
 (3)  
 (3)  
 (2) 003106 000004  
 (2) 003110 012737 000005 001202  
 458 003116 012777 020000 176666  
 459 003124 012737 002000 001124  
 460 003132 012737 114000 032530  
 461 003140 012737 100000 032536  
 462 003146 012777 032530 175604  
 463 003154 012737 010000 032532  
 464 003162 005277 176572  
 465 003166 012737 010000 032534  
 466 003174 005277 176560  
 467 003200 013737 002012 001122  
 468 003206 032777 002000 176576  
 469 003214 001003  
 470 003216 005037 001126  
 471 003222 104005  
 472 003224 005277 176530  
 473 003230 005037 001124  
 474 003234 032777 002000 176550  
 475 003242 001404  
 476 003244 012737 002000 001126  
 477 003252 104005  
 478  
 479  
 480  
 481  
 482  
 483  
 (3)  
 (3)

```

:*****
:*TEST 4 TEST THAT THE RELOCATE REG WILL CORRECTLY SET UP THE DPC
:*****
TST4: SCOPE
MOV #100,$TIMES ;;DO 100 ITERATIONS
MOV #4,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
MOV #BIT12,@STKPT ;;SET MAINT SW 1
MOV DPC,$BDADR ;;SET UP REG 00 ADRS
CLR @DPC ;;INSURE 0 DPC
MOV #100,$GDDAT ;;EXPECT 100 INITIALLY
MOV #1,R0 ;;START WITH 1
1$: MOV R0,@RLO ;;LOAD RELOCATE REG
MOV #1$,$LPERR ;;SET UP SCOPE LOOP ADRS
MOV @DPC,$BDDAT ;;READ RELOCATE REG-SB TIMES 64
CMP $GDDAT,$BDDAT ;;IS IT CORRECT?
BEQ 2$ ;;BR IF SO
ERROR 4 ;;RELOCATE FAILED TO SET UP DPC CORRECTLY
2$: ADD #100,$GDDAT ;;SET UP NEXT EXPECTED VALUE
BEQ 3$ ;;BR IF ALL 15 BITS TESTED
INC R0 ;;SET UP NEXT PATTERN
BR 1$ ;;GO TRY IT
3$: CLR @RLO ;;ZERO RELOCATE BEFORE ADVANCING

:*****
:*TEST 5 TEST THAT THE OFFSET INDICATOR WILL SET & RESET
:*****
TST5: SCOPE
MOV #5,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
MOV #2000,@STKPT ;;SET MAINT SW 2
MOV #2000,$GDDAT ;;EXPECT OFFSET TO BE SET INITIALLY
MOV #114000,BUFFER ;;SET UP OFFSET INSTR
MOV #100000,BUFFER+6 ;;LOAD 'CHAR' INST. (TO RESET INDICATOR)
MOV #BUFFER,@DPC ;;START
MOV #10000,BUFFER+2 ;;X=OFFSET 0
INC @DPC ;;PICK UP X
MOV #10000,BUFFER+4 ;;Y=OFFSET 0
INC @DPC ;;PICK UP Y
MOV STKPT,$BDADR ;;SET UP REG ADRS 32
BIT #2000,@STKPT ;;IS THE OFFSET INDICATOR SET?
BNE 1$ ;;BR IF OK
CLR $BDDAT ;;INDICATE IT WAS NOT SET
ERROR 5 ;;OFFSET INSTR INDICATOR FAILED TO SET
1$: INC @DPC ;;PICK UP CHAR INSTR - SHOULD CLR OFFSET INDICATOR
CLR $GDDAT ;;EXPECT 0
BIT #2000,@STKPT ;;DID THE OFFSET INDICATOR CLR?
BEQ TST6 ;;GO TO NEXT TEST IF OFFSET INDICATOR RESET
MOV #2000,$BDDAT ;;INDICATE IT WAS STILL SET
ERROR 5 ;;OFFSET INSTR INDICATOR FAILED TO CLR

.SBTTL
.SBTTL THESE TEST RUN AT FULL SPEED
.SBTTL

:*****
:*TEST 6 TEST THAT OFFSET INSTR CAN SET & RESET X POLARITY BIT AT REG 36
:*****

```

```

(2) 003254 000004
(2) 003256 012737 000006 001202
484 003264 012777 000040 176520
485 003272 013737 002016 001122
486 003300 012737 100000 001124
487 003306 012737 114000 032530
488 003314 012737 030000 032532
489 003322 012737 010000 032534
490 003330 012737 172000 032536
491 003336 012737 161773 032540
492 003344 004737 024524
493 003350 005777 176442
494 003354 100403
495 003356 005037 001126
496 003362 104006
497 003364 005037 001124
498 003370 042737 020000 032532
499 003376 004737 024552
500 003402 005777 176410
501 003406 100004
502 003410 012737 100000 001126
503 003416 104006
504
505

```

```

TST6: SCOPE
MOV #6,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
MOV #40,@STKPT ;RESET MAINT. BITS & 'TOP OF STACK'
MOV ZDOFF,$BDADR ;SET UP REG 36 ADRS
MOV #100000,$GDDAT ;EXPECT POLARITY BIT TO BE SET INITIALLY
MOV #114000,BUFFER ;SET UP OFFSET INSTR
MOV #30000,BUFFER+2 ;SET UP LD OFFSET POLARITY BIT AT X DATA
MOV #10000,BUFFER+4 ;Y OFFSET=0
MOV #172000,BUFFER+6 ;LD STOP INSTR
MOV #161773,BUFFER+10 ;JUMP TO REPEAT TEST
JSR PC,EXECUTE ;START DISPLAY AND WAIT FOR STOP FLAG
TST @ZDOFF ;IS X POLARITY SET?
BMI 1$ ;BR IF SO
CLR $BDDAT ;SHOW IT WAS NOT SET
ERROR 6 ;OFFSET INSTR FAILED TO SET X POLARITY
1$: CLR $GDDAT ;PREPARE FOR RESET STATE
BIC #20000,BUFFER+2 ;RESET POLARITY AT INSTR LOC
JSR PC,CONT ;RESUME
TST @ZDOFF ;IS X POLARITY RESET?
BPL TST7 ;NEXT TEST IF SO
MOV #100000,$BDDAT ;SHOW IT WAS SET
ERROR 6 ;OFFSET INSTR FAILED TO RESET X POLARITY

```

```

*****
*TEST 7 TEST THAT OFFSET INSTR CAN SET & RESET Y POLARITY BIT AT REG 36
*****

```

```

(2) 003420 000004
(2) 003422 012737 000007 001202
506 003430 013737 002016 001122
507 003436 012737 040000 001124
508 003444 012737 114000 032530
509 003452 012737 010000 032532
510 003460 012737 172000 032536
511 003466 012737 030000 032534
512 003474 012737 161773 032540
513 003502 004737 024524
514 003506 032777 040000 176302
515 003514 001003
516 003516 005037 001126
517 003522 104006
518 003524 005037 001124
519 003530 042737 020000 032534
520 003536 004737 024552
521 003542 032777 040000 176246
522 003550 001404
523 003552 012737 040000 001126
524 003560 104006
525
526

```

```

TST7: SCOPE
MOV #7,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
MOV ZDOFF,$BDADR ;SET UP REG 36 ADRS
MOV #40000,$GDDAT ;EXPECT POLARITY BIT TO BE SET INITIALLY
MOV #114000,BUFFER ;SET UP OFFSET INSTR
MOV #10000,BUFFER+2 ;SET UP X VALUE
MOV #172000,BUFFER+6 ;LOAD 'STOP' INST.
MOV #30000,BUFFER+4 ;SET UP LD OFFSET & POLARITY AT Y DATA
MOV #161773,BUFFER+10 ;JUMP TO REPEAT TEST
JSR PC,EXECUTE ;START DISPLAY AND WAIT FOR STOP FLAG
BIT #40000,@ZDOFF ;IS Y POLARITY BIT SET?
BNE 1$ ;BR IF SO
CLR $BDDAT ;SHOW IT WAS NOT SET
ERROR 6 ;OFFSET INSTR FAILED TO SET Y POLARITY
1$: CLR $GDDAT ;ADVANCE TO RESET STATE
BIC #20000,BUFFER+4 ;RESET POLARITY AT INSTR LOC
JSR PC,CONT ;RESUME
BIT #40000,@ZDOFF ;IS Y POLARITY RESET?
BEQ TST10 ;NEXT TEST IF SO
MOV #40000,$BDDAT ;SHOW IT WAS SET
ERROR 6 ;OFFSET INSTR FAILED TO RESET Y POLARITY

```

```

*****
*TEST 10 TEST THAT OFFSET INSTR CAN LOAD X & Y DYNAMIC OFFSET REG
*****

```

```

(2) 003562 000004
(2) 003564 012737 000010 001202
527 003572 012737 003624 001110
528 003600 012737 116000 032530
529 003606 012737 172000 032536

```

```

TST10: SCOPE
MOV #10,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
MOV #1$,$LPERR ;SET UP SCOPE LOOP ADRS
MOV #116000,BUFFER ;SET UP THE OFFSET INSTR
MOV #172000,BUFFER+6 ;LOAD 'STOP' INST.

```

```

530 003614 012700 004000      MOV      #4000,R0      ;FLOAT A ONE IN X POS
531 003620 012701 173777      MOV      #173777,R1   ;FLOAT A ZERO IN Y POS
532 003624 010037 032532      1$: MOV      R0,BUFFER+2 ;SET UP X POS
533 003630 010137 032534      MOV      R1,BUFFER+4 ;SET UP Y POS
534 003634 042737 170000 032532  BIC      #170000,BUFFER+2 ;PLUS 12 BITS
535 003642 042737 170000 032534  BIC      #170000,BUFFER+4 ;PLUS 12 BITS
536 003650 052737 010000 032532  BIS      #BIT12,BUFFER+2 ;INDICATE AN OFFSET INSTR IN X
537 003656 004737 024524      JSR      PC,EXECUTE   ;START DISPLAY AND WAIT FOR STOP FLAG
538 003662 013737 001774 001122  MOV      XDOFF,$BDADR ;SET UP REG ADRS 14
539 003670 013737 032532 001124  MOV      BUFFER+2,$GDDAT ;LD EXPECTED
540 003676 042737 010000 001124  BIC      #BIT12,$GDDAT ;DON'T WANT OFFSET BIT
541 003704 017737 176064 001126  MOV      @XDOFF,$BDDAT ;READ X DYNAMIC OFFSET REG
542 003712 023737 001124 001126  CMP      $GDDAT,$BDDAT ;CORRECT?
543 003720 001401      BEQ      2$          ;BR IF SO
544 003722 104007      ERROR    7          ;OFFSER INSTR FAILED TO SET UP X DYNAMIC OFFSET REG
545 003724 013737 001776 001122  2$: MOV      YDOFF,$BDADR ;SET UP REG ADRS 16
546 003732 013737 032534 001124  MOV      BUFFER+4,$GDDAT ;LD EXPECTED
547 003740 017737 176032 001126  MOV      @YDOFF,$BDDAT ;READ Y DYNAMIC OFFSET REG
548 003746 023737 001124 001126  CMP      $GDDAT,$BDDAT ;CORRECT?
549 003754 001401      BEQ      3$          ;BR IF SO
550 003756 104007      ERROR    7          ;OFFSET INSTR FAILED TO SET UP Y DYNAMIC OFFSET REG
551 003760 006200      3$: ASR      R0          ;FLOAT PAT RIGHT
552 003762 001403      BEQ      4$          ;BR IF X DONE WITH FLOAT ONE
553 003764 006201      ASR      R1          ;FLOAT PAT RIGHT
554 003766 001406      BEQ      TST11       ;NEXT TEST IF ONES AND ZEROS TESTED
555 003770 000715      BR       1$          ;REPEAT TEST THIS PATTERN
556 003772 012700 173777      4$: MOV      #173777,R0 ;NOW FLOAT A ZERO IN X
557 003776 012701 004000      MOV      #4000,R1    ;NOW FLOAT A ONE IN Y
558 004002 000710      BR       1$          ;REPEAT TEST
559
560

```

\*\*\*\*\*  
 : \*TEST 11 TEST THAT X & Y POSITION REGS CAN BE LOADED CORRECTLY USING POINT MODE 1  
 : \*\*\*\*\*

```

(3)
(3)
(2) 004004 000004      TST11: SCOPE
(2) 004006 012737 000011 001202  MOV      #11,$TESTN   ;SET TEST NUMBER IN APT MAIL BOX
561 004014 005077 175754      CLR      @XDOFF       ;CLR REG 14
562 004020 005077 175752      CLR      @YDOFF       ;CLR REG 16
563 004024 012737 004056 001110  MOV      #1$,$LPERR   ;SET UP SCOPE LOOP
564 004032 012737 116000 032530  MOV      #116000,BUFFER ;LD POINT INSTR
565 004040 012737 172000 032536  MOV      #172000,BUFFER+6 ;LD STOP INSTR
566 004046 012700 004000      MOV      #4000,R0     ;FLOAT A ONE IN X POS
567 004052 012701 173777      MOV      #173777,R1   ;FLOAT A ZERO IN Y POS
568 004056 010037 032532      1$: MOV      R0,BUFFER+2 ;SET UP X POS
569 004062 010137 032534      MOV      R1,BUFFER+4 ;SET UP Y POS
570 004066 042737 170000 032532  BIC      #170000,BUFFER+2 ;12 BITS
571 004074 042737 170000 032534  BIC      #170000,BUFFER+4 ;12 BITS
572 004102 004737 024524      JSR      PC,EXECUTE   ;START
573 004106 013737 001764 001122  MOV      XPOS,$BDADR  ;SET UP REG 04 ADRS
574 004114 013737 032532 001124  MOV      BUFFER+2,$GDDAT ;LD EXPECTED
575 004122 017737 175636 001126  MOV      @XPOS,$BDDAT ;READ X POS
576 004130 017705 175640      MOV      @XDOFF,R5    ;GET HI ORDER X POS
577 004134 004737 025136      JSR      PC,POCONV    ;GO MAKE LP 14 BIT X POS
578 004140 023737 001124 001126  CMP      $GDDAT,$BDDAT ;CORRECT?
579 004146 001401      BEQ      2$          ;BR IF SO
580 004150 104014      ERROR    14         ;POINT INSTR FAILED TO LOAD X POS CORRECTLY
581 004152 013737 001766 001122  2$: MOV      YPOS,$BDADR ;SET UP REG 06 ADRS

```

```

582 004160 013737 032534 001124      MOV    BUFFER+4,$GDDAT ;LD EXPECTED
583 004166 017737 175574 001126      MOV    @YPOS,$BDDAT   ;READ Y POS
584 004174 017705 175576          MOV    @YDOFF,R5      ;GET HI ORDER Y POS
585 004200 004737 025136          JSR    PC,POCONV      ;GO MAKE UP 14 BIT Y POS
586 004204 023737 001124 001126      CMP    $GDDAT,$BDDAT ;CORRECT?
587 004212 001401          BEQ    3$            ;BR IF SO
588 004214 104015          ERROR  15           ;POINT INSTR FAILED TO LOAD Y POS CORRECTLY
589 004216 006200          3$: ASR    R0          ;FLOAT PAT RIGHT
590 004220 001403          BEQ    4$            ;BR IF X DONE WITH FLOAT 1
591 004222 006201          ASR    R1          ;FLOAT PAT RIGTH
592 004224 001406          BEQ    TST12        ;NEXT TEST IF ONES + ZEROS FLOATED
593 004226 000713          BR     1$            ;REPEAT TEST THIS PAT
594 004230 012700 173777          4$: MOV    #173777,R0 ;NOW FLOAT ZERO IN X
595 004234 012701 004000          MOV    #4000,R1     ;NOW FLOAT ONE IN Y
596 004240 000706          BR     1$            ;REPEAT TEST
597
598

```

\*\*\*\*\*  
 (3) \*TEST 12 TEST THAT X & Y POSITION REGS CAN BE LOADED CORRECTLY USING ABS VECTOR  
 (3) \*\*\*\*\*

```

(2) 004242 000004          TST12: SCOPE
(2) 004244 012737 000012 001202      MOV    #12,$TESTN    ;;SET TEST NUMBER IN APT MAIL BOX
599 004252 005077 175516          CLR    @XDOFF        ;CLR REG 14
600 004256 005077 175514          CLR    @YDOFF        ;CLR REG 16
601 004262 012737 004326 001110      MOV    #1$,$LPERR    ;SET UP SCOPE LOOP
602 004270 012700 032530          MOV    #BUFFER,R0    ;SET INSTR ADRS
603 004274 012720 116000          MOV    #116000,(R0)+ ;POINT INSTR
604 004300 005020          CLR    (R0)+         ;X = 0
605 004302 005020          CLR    (R0)+         ;Y = 0
606 004304 012720 146000          MOV    #146000,(R0)+ ;ABSOLUTE VECTOR INSTR
607 004310 022020          CMP    (R0)+,(R0)+   ;SKIP OVER VECTORS PTS
608 004312 012710 172000          MOV    #172000,(R0)  ;SET UP HALT
609 004316 012700 004000          MOV    #4000,R0      ;FLOAT A ONE IN X VECTOR POINT
610 004322 012701 173777          MOV    #173777,R1    ;FLOAT A ZERO IN Y VECTOR POINT
611 004326 010037 032540          1$: MOV    R0,BUFFER+10 ;SET UP X POINT
612 004332 010137 032542          MOV    R1,BUFFER+12  ;SET UP Y POINT
613 004336 042737 170000 032540      BIC    #170000,BUFFER+10 ;12 BITS
614 004344 042737 170000 032542      BIC    #170000,BUFFER+12 ;12 BITS
615 004352 004737 024524          JSR    PC,EXECUTE    ;START
616 004356 013737 001764 001122      MOV    XPOS,$BDADR   ;SET UP REG 04 ADRS
617 004364 013737 032540 001124      MOV    BUFFER+10,$GDDAT ;LD EXPECTED
618 004372 017737 175366 001126      MOV    @XPOS,$BDDAT  ;READ X POS
619 004400 017705 175370          MOV    @XDOFF,R5     ;GET HI ORDER X POS
620 004404 004737 025136          JSR    PC,POCONV     ;GO MAKE UP 14 BIT X POS
621 004410 023737 001124 001126      CMP    $GDDAT,$BDDAT ;CORRECT?
622 004416 001401          BEQ    2$            ;BR IF SO
623 004420 104012          ERROR  12           ;ABSOLUTE VECTOR INSTR FAILED TO LOAD X POS
624 004422 013737 001766 001122      2$: MOV    YPOS,$BDADR  ;SET UP REG 06 ADRS
625 004430 013737 032542 001124      MOV    BUFFER+12,$GDDAT ;LD EXPECTED
626 004436 017737 175324 001126      MOV    @YPOS,$BDDAT  ;READ Y POS
627 004444 017705 175326          MOV    @YDOFF,R5     ;GET HI ORDER Y POS
628 004450 004737 025136          JSR    PC,POCONV     ;GO MAKE UP 14 BIT X POS
629 004454 023737 001124 001126      CMP    $GDDAT,$BDDAT ;CORRECT?
630 004462 001401          BEQ    3$            ;BR IF SO
631 004464 104012          ERROR  12           ;ABSOLUTE VECTOR INSTR FAILED TO LOAD Y POS
632 004466 006200          3$: ASR    R0          ;FLOAT PAT RIGHT
633 004470 001403          BEQ    4$            ;BR IF X DONE WITH FLOAT 1

```

```

634 004472 006201          ASR    R1          ;FLOAT PAT RIGHT
635 004474 001406          BEQ    TST13       ;:NEXT TEST IF ONES + ZEROES FLOATED
636 004476 000713          BR     1$         ;REPEAT TEST THIS PAT
637 004500 012700 173777    4$:  MOV    #173777,R0 ;NOW FLOAT ZERO IN X
638 004504 012701 004000    MOV    #4000,R1   ;NOW FLOAT ONE IN Y
639 004510 000706          BR     1$         ;REPEAT TEST
    
```

```

640
641
(3)
(3)
(2)
    
```

```

:*****
:*TEST 13      TEST THAT X POSITION BITS 10 & 11 WILL SET & RESET VIA OFFSET INSTR
:*****
TST13: SCOPE
    
```

```

(2) 004512 000004          MOV    #13,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
(2) 004514 012737 000013 001202  MOV    #1$,$LPERF ;:SET UP SCOPE LOOP ADRS
642 004522 012737 004600 001110  MOV    XDOFF,$BDADR ;:SET UP REG 14 ADRS
643 004530 013737 001774 001122  MOV    #36000,$GDDAT ;:EXPECT BOTH BITS INITIALLY
644 004536 012737 036000 001124  MOV    #BUFFER,R0  ;:GET INSTR ADRS POINTER
645 004544 012700 032530          MOV    #114000,(R0)+ ;:LOAD OFFSET INSTR
646 004550 012720 114000          MOV    #16000,(R0)+ ;:SET UP MAX X OFFSET
647 004554 012720 016000          MOV    #10000,(R0)+ ;:SET UP Y OFFSET
648 004560 012720 010000          MOV    #114000,(R0)+ ;:LOAD POINT INSTR
649 004564 012720 114000          CLR    (R0)+       ;:X OF 0
650 004570 005020          CLR    (R0)+       ;:Y OF 0
651 004572 005020          MOV    #172000,(R0) ;:LOAD STOP INSTR
652 004574 012710 172000          CLR    @XDOFF     ;:CLR OFFSET REG
653 004600 005077 175170          1$:  JSR    PC,EXECUTE ;:GO START DISPLAY
654 004604 004737 024524          MOV    @XDOFF,$BDAT ;:READ REG 14
655 004610 017737 175160 001126  CMP    $GDDAT,$BDAT ;:ARE THEY CORRECT?
656 004616 023737 001124 001126  BEQ    2$         ;:BR IF SO
657 004624 001401          ERROR 10         ;:OFFSET INSTR FAILED TO LOAD HIGH ORDER X POSITION BIT
658 004626 104010          2$:  SUB    #12000,$GDDAT ;:SET UP FOR NEXT ROUTINE
659 004630 162737 012000 001124  SUB    #2000,BUFFER+2 ;:SET UP NEXT VALUE
660 004636 162737 002000 032532  CMP    #6000,BUFFER+2 ;:ALL VALUES TESTED?
661 004644 022737 006000 032532  BNE    1$         ;:BR IF NOT
662 004652 001352
663
664
    
```

```

(3)
(3)
(2)
    
```

```

:*****
:*TEST 14      TEST THAT Y POSITION BITS 10 & 11 WILL SET & RESET VIA OFFSET INSTR
:*****
TST14: SCOPE
    
```

```

(2) 004654 000004          MOV    #14,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
(2) 004656 012737 000014 001202  MOV    #1$,$LPERF ;:SET UP SCOPE LOOP ADRS
665 004664 012737 004742 001110  MOV    YDOFF,$BDADR ;:SET UP REG 16 ADRS
666 004672 013737 001776 001122  MOV    #36000,$GDDAT ;:EXPECT BOTH BITS INITIALLY
667 004700 012737 036000 001124  MOV    #BUFFER,R0  ;:GET INSTR ADRS POINTER
668 004706 012700 032530          MOV    #114000,(R0)+ ;:LOAD OFFSET INSTR
669 004712 012720 114000          MOV    #10000,(R0)+ ;:SET UP X OFFSET
670 004716 012720 010000          MOV    #16000,(R0)+ ;:SET UP MAX Y OFFSET
671 004722 012720 016000          MOV    #114000,(R0)+ ;:LOAD POINT INSTR
672 004726 012720 114000          CLR    (R0)+       ;:X OF 0
673 004732 005020          CLR    (R0)+       ;:Y OF 0
674 004734 005020          MOV    #172000,(R0) ;:LOAD STOP INSTR
675 004736 012710 172000          CLR    @YDOFF     ;:CLR OFFSET REG
676 004742 005077 175030          1$:  JSR    PC,EXECUTE ;:GO START DISPLAY
677 004746 004737 024524          MOV    @YDOFF,$BDAT ;:READ REG 16
678 004752 017737 175020 001126  CMP    $GDDAT,$BDAT ;:ARE THEY CORRECT?
679 004760 023737 001124 001126  BEQ    2$         ;:BR IF SO
680 004766 001401          ERROR 10         ;:OFFSET INSTR FAILED TO LOAD HIGH ORDER Y POSITION BITS
681 004770 104010
    
```

```

682 004772 162737 012000 001124 2$: SUB #12000,$GDDAT ;SET UP FOR NEXT VALUE
683 005300 162737 002000 032532 SUB #2000,BUFFER+2 ;SET UP NEXT VALUE
684 005006 022737 006000 032532 CMP #6000,BUFFER+2 ;ALL VALUES TESTED?
685 005014 001352 BNE 1$ ;BR IF NOT
686
687
    
```

```

(3) ::*****
(3) :*TEST 15 TEST X AND Y POS BITS 10-13 (+ DIR) USING THE POINT & OFFSET INSTRS
(3) :*****
    
```

```

(2) 005016 000004 TST15: SCOPE
(2) 005020 012737 000015 001202 MOV #15,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
688 005026 012700 032530 MOV #BUFFER,R0 ;:GET ADRS WHERE INSTR'S GO
689 005032 012720 116000 MOV #116000,(R0)+ ;:OFFSET INSTR
690 005036 012720 011000 MOV #11000,(R0)+ ;:X OFFSET OF 1000
691 005042 012720 017000 MOV #17000,(R0)+ ;:Y OFFSET OF 7000
692 005046 012720 116000 MOV #116000,(R0)+ ;:POINT INSTR
693 005052 012720 001000 MOV #1000,(R0)+ ;:X=1000
694 005056 012720 007000 MOV #7000,(R0)+ ;:Y=7000
695 005062 012720 172000 MOV #172000,(R0)+ ;:STOP INSTR
696 005066 012700 010000 MOV #10000,R0 ;:LD EXPECTED X POS
697 005072 012701 070000 MOV #70000,R1 ;:LD EXPECTED Y POS
698 005076 012737 005104 001110 MOV #1$,$LPERR ;:SET UP SCOPE LOOP ADRS
699 005104
    
```

```

(1) 005104 004737 024524 1$: JSR PC,EXECUTE ;:START DISPLAY AND WAIT FOR STOP FLAG
700 005110 013737 001774 001122 MOV XDOFF,$BDADR ;:SET UP REG ADRS 14
701 005116 010037 001124 MOV R0,$GDDAT ;:LD EXPECTED
702 005122 017737 174646 001126 MOV @XDOFF,$BDDAT ;:READ HI X POS BITS
703 005130 042737 007777 001126 BIC #7777,$BDDAT ;:GET RID OF OFFSET
704 005136 023737 001124 001126 CMP $GDDAT,$BDDAT ;:CORRECT?
705 005144 001401 BEQ 2$ ;:BR IF SO
706 005146 104010 ERROR 10 ;:X POS BITS 10-13 FAILED ON POINT/OFFSET
    
```

```

707 005150 013737 001776 001122 2$: MOV YDOFF,$BDADR ;:SET UP REG ADRS 16
708 005156 010137 001124 MOV R1,$GDDAT ;:LD EXPECTED
709 005162 017737 174610 001126 MOV @YDOFF,$BDDAT ;:READ HI Y POS BITS
710 005170 042737 007777 001126 BIC #7777,$BDDAT ;:GET RID OF OFFSET
711 005176 023737 001124 001126 CMP $GDDAT,$BDDAT ;:CORRECT?
712 005204 001401 BEQ 3$ ;:BR IF SO
713 005206 104010 ERROR 10 ;:Y POS 10-13 FAILED ON POINT/OFFSET
714 005210 062737 001000 032532 3$: ADD #1000,BUFFER+2 ;:INCREASE X POINT
715 005216 062737 001000 032540 ADD #1000,BUFFER+10 ;:INCREASE X OFFSET
716 005224 162737 001000 032534 SUB #1000,BUFFER+4 ;:DECREASE Y POINT
717 005232 162737 001000 032542 SUB #1000,BUFFER+12 ;:DECREASE Y OFFSET
718 005240 062700 010000 ADD #10000,R0 ;:UPDATE EXPECTED X
719 005244 162701 010000 SUB #10000,R1 ;:UPDATE EXPECTED Y
720 005250 001315 BNE 1$ ;:BR IF NOT DONE WITH 7 CALCULATIONS
721
722
    
```

```

(3) ::*****
(3) :*TEST 16 TEST ALL POLARITY COMBINATIONS FOR X & Y DELTA AND X & Y OFFSET
(3) :*****
    
```

```

(2) 005252 000004 TST16: SCOPE
(2) 005254 012737 000016 001202 MOV #16,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
723 ;:USE POINT INSTR FIRST THEN REPLACE POINT
724 ;:INSTR WITH ABSOLUTE VECTOR INSTR.
725 005262 012700 032530 MOV #BUFFER,R0 ;:SET UP ADRS WHERE INSTRS GO
726 005266 012720 116000 MOV #116000,(R0)+ ;:OFFSET INSTR
727 005272 012720 010001 MOV #10001,(R0)+ ;:X OFFSET OF 1
728 005276 012720 010003 MOV #10003,(R0)+ ;:Y OFFSET OF 3
    
```

```

729 005302 012720 116000      MOV      #16000,(R0)+    ;POINT INSTR
730 005306 012720 000002      MOV      #2,(R0)+      ;X=2
731 005312 012720 000004      MOV      #4,(R0)+      ;Y=4
732 005316 012720 172000      MOV      #172000,(R0)+ ;STOP INSTR
733 005322 012737 005332 001110  MOV      #1$,SLPERR    ;SET UP SCOPE _LOOP ADRS
734 005330 005003          CLR      R3            ;R3 WILL CONTROL POLARITY OF ALL X & Y DATA
735 005332          1$:
(1) 005332 004737 024524          JSR      PC,EXECUTE    ;START DISPLAY AND WAIT FOR STOP FLAG
736 005336 013737 001764 001122  MOV      XPOS,$BDADR   ;REPORT ER'S AT REG 02
737 005344 013700 032540          MOV      BUFFER+10,R0 ;GET X POINT
738 005350 013701 032532          MOV      BUFFER+2,R1  ;GET X OFFSET
739 005354 004737 025064          JSR      PC,CALPT     ;GO COMPUTE 14 BIT POS
740 005360 017737 174400 001126  MOV      @XPOS,$BDDAT ;READ X POS
741 005366 017705 174402          MOV      @XDOFF,R5   ;READ X OFFSET
742 005372 004737 025136          JSR      PC,POCONV    ;GO MAKE UP 14 BIT POS
743 005376 023737 001124 001126  CMP      $GDDAT,$BDDAT ;CORRECT?
744 005404 001401          BEQ      2$          ;BR IF SO
745 005406 104042          ERROR    42         ;POINT FAILED AT DELTA X & OFFSET X IN BUFFER+2 & +10
746 005410 013737 001766 001122  2$:  MOV      YPOS,$BDADR   ;REPORT ER'S AT REG 04
747 005416 013700 032542          MOV      BUFFER+12,R0 ;GET Y POINT
748 005422 013701 032534          MOV      BUFFER+4,R1  ;GET Y OFFSET
749 005426 004737 025064          JSR      PC,CALPT     ;GO COMPUTE 14 BIT POS
750 005432 017737 174330 001126  MOV      @YPOS,$BDDAT ;READ Y POS
751 005440 017705 174332          MOV      @YDOFF,R5   ;READ Y HI POS
752 005444 004737 025136          JSR      PC,POCONV    ;GO MAKE UP 14 BIT POS
753 005450 023737 001124 001126  CMP      $GDDAT,$BDDAT ;CORRECT?
754 005456 001401          BEQ      3$          ;BR IF SO
755 005460 104042          ERROR    42         ;POINT FAILED AT DELTA Y & OFFSET Y IN BUFFER+4 & +12
756 005462 005203          3$:  INC      R3            ;BUMP POLARITY CONTROL
757 005464 022703 000020          CMP      #20,R3      ;DONE 16 POSIBILITIES?
758 005470 001445          BEQ      9$          ;SEE IF ABS VECTOR WAS TESTED
759 005472 042737 020000 032532  4$:  BIC      #BIT13,BUFFER+2 ;RESET X POL OFFSET
760 005500 032703 000001          BIT      #BIT0,R3    ;WANT -POL X OFFSET?
761 005504 001403          BEQ      5$          ;BR IF NOT
762 005506 052737 020000 032532  BIS      #BIT13,BUFFER+2 ;SET -POL X OFFSET
763 005514 042737 020000 032534  5$:  BIC      #BIT13,BUFFER+4 ;RESET Y POL OFFSET
764 005522 032703 000002          BIT      #BIT1,R3    ;WANT -POL Y OFFSET?
765 005526 001403          BEQ      6$          ;BR IF NOT
766 005530 052737 020000 032534  BIS      #BIT13,BUFFER+4 ;SET -POL Y OFFSET
767 005536 042737 020000 032540  6$:  BIC      #BIT13,BUFFER+10 ;RESET X POL
768 005544 032703 000004          BIT      #BIT2,R3    ;WANT -POL X?
769 005550 001403          BEQ      7$          ;BR IF NOT
770 005552 052737 020000 032540  BIS      #BIT13,BUFFER+10 ;SET -POL X
771 005560 042737 020000 032542  7$:  BIC      #BIT13,BUFFER+12 ;RESET POL Y
772 005566 032703 000010          BIT      #BIT3,R3    ;WANT -POL Y?
773 005572 001403          BEQ      8$          ;BR IF NOT
774 005574 052737 020000 032542  BIS      #BIT13,BUFFER+12 ;SET -POL X
775 005602 000653          8$:  BR      1$          ;REPEAT TEST
776 005604 022737 146000 032536  9$:  CMP      #146000,BUFFER+6 ;JUST DO ABS VEC?
777 005612 001405          BEQ      TST17       ;NEXT TEST IF SO
778 005614 012737 146000 032536  MOV      #146000,BUFFER+6 ;NOW SET UP ASB VEC INSTR
779 005622 005003          CLR      R3            ;RESET POLARITY CONTROL
780 005624 000722          BR      4$          ;REPEAT TEST WITH ABS VEC

```

781  
 782  
 (3) .....  
 : \*TEST 17 TEST THAT RESET WILL CLEAR X & Y POS BITS 10-13

```

(3)
(2) 005626 000004
(1) 005630 012737 000040 001166
(2) 005636 012737 000017 001202
783 005644 012700 032530
784 005650 012720 116000
785 005654 012720 010000
786 005660 012720 010000
787 005664 012720 116000
788 005670 012720 020001
789 005674 012720 020001
790 005700 012710 172000
791 005704 005037 001124
792 005710 004737 024524
793 005714 000005
794 005716 017737 174052 001126
795 005724 001404
796 005726 013737 001774 001122
797 005734 104032
798 005736 017737 174034 001126 1$:
799 005744 001404
800 005746 013737 001776 001122
801 005754 104032
802
803
(3)
(3)
(2) 005756 000004
(1) 005760 012737 000100 001166
(2) 005766 012737 000020 001202
804 005774 012737 006050 001110
805 006002 012703 001777
806 006006 012737 000001 001124
807 006014 013700 001712 1$:
808 006020 012720 116000
809 006024 005020
810 006026 005020
811 006030 012720 110000
812 006034 013720 001124
813 006040 013720 001124
814 006044 012720 172000
815 006050
816 006050 004737 024524 2$:
817 006054 017737 173704 001126
818 006062 023737 001124 001126
819 006070 001404
820 006072 013737 001764 001122
821 006100 104016
822
823 006102 017737 173660 001126 3$:
824 006110 023737 001124 001126
825 006116 001404
826 006120 013737 001766 001122
827 006126 104016
828

```

```

*****
TST17: SCOPE
MOV #40,$TIMES ;:DO 40 ITERATIONS
MOV #17,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
MOV #BUFFER,R0 ;:SET UP ADRS FOR INSTRS
MOV #116000,(R0)+ ;:SET UP OFFSET INSTR
MOV #10000,(R0)+ ;:X OFFSET
MOV #10000,(R0)+ ;:Y OFFSET
MOV #116000,(R0)+ ;:SET UP POINT INSTR
MOV #20001,(R0)+ ;:X POINT
MOV #20001,(R0)+ ;:Y POINT
MOV #172000,(R0) ;:SET UP HALT INSTR
CLR $GDDAT ;:EXPECT ZERO
JSR PC,EXECUTE ;:START
RESET ;:CLR X + Y POS
MOV @XDOFF,$BDDAT ;:DID IT CLR?
BEQ 1$ ;:BR IF SO
MOV XDOFF,$BDADR ;:SET UP REG ADRS 14
ERROR 32 ;:HI ORDER X POS BITS FAILED TO CLEAR
1$: MOV @YDOFF,$BDDAT ;:DID IT CLR?
BEQ TST20 ;:NEXT TEST IF SO
MOV YDOFF,$BDADR ;:SET UP REG ADRS 16
ERROR 32 ;:HI ORDER Y POS BITS FAILED TO CLEAR
*****

```

```

*****
:TST20 TEST THAT LONG VECTOR MODE INCREMENT X AND Y AXIS COUNT 1-1777
*****
TST20: SCOPE
MOV #100,$TIMES ;:DO 100 ITERATIONS
MOV #20,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
MOV #2,$SLPERR ;:SET UP SCOPE LOOP ADRS
MOV #1777,R3 ;:SET UP A COUNTER
MOV #1,$GDDAT ;:PRESET THE COMPARED VALUE
1$: MOV DBUF,R0 ;:SET UP R0
MOV #116000,(0)+ ;:LOAD 'POINT MODE'
CLR (0)+ ;:CLEAR X AXIS
CLR (0)+ ;:CLEAR Y AXIS
MOV #110000,(0)+ ;:LOAD 'LONG VECTOR MODE'
MOV $GDDAT,(0)+ ;:PRESET 'DELTA X AXIS'
MOV $GDDAT,(0)+ ;:PRESET 'DELTA Y AXIS'
MOV #172000,(R0)+ ;:LOAD 'DISPLAY STOP'
2$: JSR PC,EXECUTE ;:START DISPLAY AND WAIT FOR STOP FLAG
MOV @XPOS,$BDDAT ;:READ X POS
CMP $GDDAT,$BDDAT ;:ARE THEY EQUAL ?
BEQ 3$ ;:YES
MOV XPOS,$BDADR ;:SET UP REG ADRS 04
ERROR 16 ;:NO, LONG VECTOR MODE FAILED
3$: MOV @YPOS,$BDDAT ;:READ Y POS
CMP $GDDAT,$BDDAT ;:COMPARE
BEQ 4$ ;:YES
MOV YPOS,$BDADR ;:SET UP REG ADRS 06
ERROR 16 ;:NO, INCREMENT Y AXIS VIA

```

```

829 006130 005237 001124 4$: INC $GDDAT ;INCREMENT EXPECTED VALUE
830 006134 005303 DEC R3 ;FINISHED?
831 006136 001326 BNE 1$ ;;NO, TEST MORE DATA
832
833 ;:*****
(3) ;*TEST 21 TEST THAT LONG VECTOR MODE DECREMENTS X AND Y AXIS COUNT 1-1777
(3) ;:*****
(2) 006140 000004 TST21: SCOPE
(1) 006142 012737 000100 001166 MOV #100,$TIMES ;:DO 100 ITERATIONS
(2) 006150 012737 000021 001202 MOV #21,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
834 006156 012737 006232 001110 MOV #2$, $LPERR ;:SET UP SCOPE LOOP ADRS
835 006164 012703 002000 MOV #2000,R3 ;:SET UP A COUNTER
836 006170 012737 001777 001124 MOV #1777,$GDDAT ;:PRESET THE COMPARED VALUE
837 006176 012705 020001 MOV #20001,R5
838 006202 013700 001712 1$: MOV DBUF,R0 ;:SET UP R0
839 006206 012720 116000 MOV #116000,(0)+ ;:LOAD 'POINT MODE''
840 006212 005020 CLR (0)+ ;:CLEAR X AXIS
841 006214 005020 CLR (0)+ ;:CLEAR Y AXIS
842 006216 012720 110000 MOV #110000,(0)+ ;:LOAD 'LONG VECTOR MODE''
843 006222 010520 MOV R5,(0)+ ;:PRESET 'DELTA X AXIS''
844 006224 010520 MOV R5,(0)+ ;:PRESET 'DELTA Y AXIS''
845 006226 012720 172000 MOV #172000,(R0)+ ;:PRESET 'DISPLAY STOP''
846 006232 2$:
(1) 006232 004737 024524 JSR PC,EXECUTE ;:START DISPLAY AND WAIT FOR STOP FLAG
847
848 006236 017737 173522 001126 MOV @XPOS,$BDDAT ;:READ X AXIS
849 006244 023737 001124 001126 CMP $GDDAT,$BDDAT ;:ARE THEY EQUAL?
850 006252 001404 BEQ 3$ ;:YES
851 006254 013737 001764 001122 MOV XPOS,$BDADR ;:SET UP REG ADRS 04
852 006262 104017 ERROR 17 ;:NO, DECREMENT X AXIS VIA
853
854 006264 017737 173476 001126 3$: MOV @YPOS,$BDDAT ;:READ Y AXIS
855 006272 023737 001124 001126 CMP $GDDAT,$BDDAT ;:ARE THEY EQUAL?
856 006300 001404 BEQ 4$ ;:YES
857 006302 013737 001766 001122 MOV YPOS,$BDADR ;:SET UP REG ADRS 06
858 006310 104017 ERROR 17 ;:NO, DECREMENT Y AXIS VIA
859
860 006312 005205 4$: INC R5 ;:INCREMENT 'DELTA X-Y''
861 006314 005337 001124 DEC $GDDAT ;:DECREMENT EXPECTED VALUE
862 006320 005303 DEC R3 ;:FINISHED?
863 006322 001327 BNE 1$ ;:NO, TEST MORE DATA
864
865 ;:*****
(3) ;*TEST 22 TEST LONG VECTOR FOR ALL VALUES OF SCALE
(3) ;:*****
(2) 006324 000004 TST22: SCOPE
(2) 006326 012737 000022 001202 MOV #22,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
866 006334 012700 032530 MOV #BUFFER,R0 ;:GET 1ST ADRS WHERE INTRS' GO
867 006340 012720 154024 MOV #154024,(R0)+ ;:SCALE OF UNITY
868 006344 012720 116000 MOV #116000,(R0)+ ;:POINT INSTR
869 006350 005020 CLR (R0)+ ;:X = 0
870 006352 005020 CLR (R0)+ ;:Y = 0
871 006354 012720 154021 MOV #154021,(R0)+ ;:SCALE TO BE UPDATED
872 006360 012720 112000 MOV #112000,(R0)+ ;:LONG VECTOR INSTR
873 006364 012720 000100 MOV #100,(R0)+ ;:X = 100
874 006370 012720 000200 MOV #200,(R0)+ ;:Y = 200
    
```

```

875 006374 012710 172000          MOV      #172000,(R0)      ;STOP INSTR
876 006400 012737 000001 001164    MOV      #1,$TMP0        ;SET UP SCALE FOR ER TYPE
877 006406 012701 000020          MOV      #20,R1          ;SET UP X EXPECTED DATA
878 006412 012702 000040          MOV      #40,R2          ;SET UP Y EXPECTED DATA
879 006416 012737 006424 001110    MOV      #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
880 006424 004737 024524          JSR      PC,EXECUTE      ;START DISPLAY
881 006430 013737 001764 001122    1$:     MOV      XPOS,$BDADR   ;SET UP REG ADRS 04
882 006436 010137 001124          MOV      R1,$GDDAT       ;LD EXPECTED
883 006442 012737 000100 001162    MOV      #100,$REGO      ;SET UP X VECTOR FOR ER TYPE
884 006450 017737 173310 001126    MOV      @XPOS,$BDDAT    ;READ X POS REG
885 006456 023737 001124 001126    CMP      $GDDAT,$BDDAT   ;CORRECT?
886 006464 001401                BEQ      2$              ;BR IF SO
887 006466 104011                ERROR    '1              ;X VECTOR POS FAILED TO SCALE
888 006470 013737 001766 001122    2$:     MOV      YPOS,$BDADR   ;SET UP REG ADRS 06
889 006476 010237 001124          MOV      R2,$GDDAT       ;LD EXPECTED
890 006502 012737 000200 001162    MOV      #200,$REGO      ;SET UP Y VECTOR FOR ER TYPE
891 006510 017737 173252 001126    MOV      @YPOS,$BDDAT    ;READ Y POS REG
892 006516 023737 001124 001126    CMP      $GDDAT,$BDDAT   ;CORRECT?
893 006524 001401                BEQ      3$              ;BR IF SO
894 006526 104011                ERROR    '1              ;Y VECTOR POS FAILED TO SCALE
895 006530 005237 001164          3$:     INC      $TMP0          ;ADVANCE SCALE BY 1/4
896 006534 022737 000020 001164    LMP      #20,$TMP0       ;ALL SCALES TESTED?
897 006542 001407                BEQ      TST23           ;NEXT TEST IF SO
898 006544 005237 032540          INC      BUFFER+10       ;BUMP SCALE BY 1/4 AT INSTR LOC
899 006550 062701 000020          ADD      #20,R1          ;BUMP EXPECTED X POS BY 1/4
900 006554 062702 000040          ADD      #40,R2          ;BUMP EXPECTED Y POS BY 1/4
901 006560 000721                BR       1$              ;TRY NEXT SCALE
    
```

```

902
903
(3)
(3)
(2) 006562 000004
(1) 006564 012737 000200 001166    TST23: SCOPE
(2) 006572 012737 000023 001202    MOV      #200,$TIMES     ;;DO 200 ITERATIONS
904 006600 012737 006656 001110    MOV      #23,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
905 006606 012703 000077          MOV      #2$,$LPERR      ;SET UP SCOPE LOOP ADRS
906 006612 012737 000001 001124    MOV      #77,R3          ;SET UP A COUNT LOCATION
907 006620 012704 000201          MOV      #1,$GDDAT       ;SET UP THE COMPARED LOCATION
908 006624 013700 001712          1$:     MOV      #201,R4        ;SET UP 'DELTA X-Y'
909 006630 012720 154024          MOV      DBUF,R0         ;SET UP R0
910 006634 012720 116000          MOV      #154024,(0)+    ;SCALE OF UNITY
911 006640 005020                MOV      #116000,(0)+   ;LOAD 'SET POINT DATA MODE'
912 006642 005020                CLR      (0)+           ;CLEAR X AXIS
913 006644 012720 106000          MOV      #106000,(0)+   ;CLEAR Y AXIS
914 006650 010420                MOV      #106000,(0)+   ;LOAD 'SET SHORT VECTOR MODE'
915 006652 012720 172000          MOV      R4,(0)+        ;PRESET 'DELTA X AND DELTA Y'
916 006656                MOV      #172000,(R0)+  ;LOAD DISPLAY STOP
(1) 006656 004737 024524          2$:     JSR      PC,EXECUTE      ;START DISPLAY AND WAIT FOR STOP FLAG
917
918 006662 017737 173076 001126    MOV      @XPOS,$BDDAT   ;READ X POSITION
919 006670 023737 001124 001126    CMP      $GDDAT,$BDDAT  ;ARE THEY EQUAL
920 006676 001404                BEQ      3$              ;YES
921 006700 013737 001764 001122    MOV      XPOS,$BDADR    ;SET UP REG ADRS 04
922 006706 104016                ERROR    '16            ;INCREMENT X AXIS FAILED USING
923
924 006710 013737 001766 001122    3$:     MOV      YPOS,$BDADR    ;SET UP REG ADRS 06
    
```

```

925 006716 017737 173044 001126      MOV    @YPOS,$BDDAT    ;READ Y POSITION
926 006724 023737 001124 001126      CMP    $GDDAT,$BDDAT  ;ARE THEY EQUAL ?
927 006732 001401                BEQ    4$              ;YES
928 006734 104016                ERROR  16              ;INCREMENT Y AXIS FAILED USING
929
930 006736 062704 000201      4$:   ADD    #201,R4      ;ADD DELTA X-Y
931 006742 005237 001124                INC    $GDDAT          ;INCREMENT EXPECTED VALUE
932 006746 005303                DEC    R3              ;DECREMENT COUNT, FINISHED?
933 006750 001325                BNE    1$              ;:NO, TEST MORE DATA
934
935

```

```

(3)
(3)
(2) 006752 000004
(1) 006754 012737 000200 001166
(2) 006762 012737 000024 001202
936 006770 012737 007042 001110
937 006776 012703 000077
938 007002 012737 001777 001124
939 007010 012704 020301
940 007014 013700 001712
941 007020 012720 116000
942 007024 005020
943 007026 005020
944 007030 012720 106000
945 007034 010420
946 007036 012720 172000
947 007042
(1) 007042 004737 024524
948
949 007046 017737 172712 001126
950 007054 023737 001124 001126
951 007062 001404
952 007064 013737 001764 001122
953 007072 104017
954
955 007074 017737 172666 001126
956 007102 023737 001124 001126
957 007110 001404
958 007112 013737 001766 001122
959 007120 104017
960
961 007122 062704 000201
962 007126 005337 001124
963 007132 005303
964 007134 001327
965
966

```

```

*****
*TEST 24 TEST THAT X AND Y AXIS DECREMENT PROPERLY USING SHORT VECTOR MODE COUNT
*****
TST24: SCOPE

```

```

MOV    #200,$TIMES    ;;DO 200 ITERATIONS
MOV    #24,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
MOV    #2$, $LPERR    ;SET UP SCOPE LOOP ADRS
MOV    #77,R3         ;SET UP A COUNT LOCATION
MOV    #177,$GDDAT    ;SET UP THE COMPARED LOCATION
MOV    #20301,R4      ;PRESET THE 'DELTA X-Y'
1$:   MOV    DBUF,R0   ;SET UP R0
MOV    #116000,(0)+   ;LOAD 'SET POINT DATA MODE''
CLR    (0)+          ;CLEAR X AXIS
CLR    (0)+          ;CLEAR Y AXIS
MOV    #106000,(0)+  ;LOAD 'SET SHORT VECTOR MODE''
MOV    R4,(0)+       ;PRESET 'DELTA X AND DELTA Y''
MOV    #172000,(R0)+ ;LOAD DISPLAY STOP
2$:   JSR    PC,EXECUTE ;START DISPLAY AND WAIT FOR STOP FLAG
MOV    @XPOS,$BDDAT  ;READ X POSITION
CMP    $GDDAT,$BDDAT ;ARE THEY EQUAL
BEQ    3$            ;YES
MOV    XPOS,$BDADR   ;SET UP REG ADRS 04
ERROR  17            ;DECREMENT X AXIS FAILED USING

```

```

3$:   MOV    @YPOS,$BDDAT ;READ Y POSITION
CMP    $GDDAT,$BDDAT  ;ARE THEY EQUAL ?
BEQ    4$            ;YES DECREMENT
MOV    YPOS,$BDADR    ;SET UP REG ADRS 06
ERROR  17            ;DECREMENT Y AXIS FAILED USING
4$:   ADD    #201,R4      ;ADD 'DELTA X-Y''
DEC    $GDDAT         ;DECREMENT EXPECTED VALUE
DEC    R3             ;DECREMENT COUNT, FINISHED?
BNE    1$            ;:NO, TEST MORE DATA

```

```

*****
*TEST 25 TEST SHORT VECTOR AT ORIGIN 1000/1000 FOR ALL VALUES OF SCALE
*****
TST25: SCOPE

```

```

MOV    #25,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
MOV    #BUFFER,R0     ;GET 1ST ADRS WHERE INSTR'S GO
MOV    #154024,(R0)+  ;'LD STATUS C' (SCALE OF UNITY)
MOV    #116000,(R0)+  ;POINT INSTR
MOV    #1000,(R0)+   ;X = 1000

```

```

(3)
(3)
(2) 007136 000004
(2) 007140 012737 000025 001202
967 007146 012700 032530
968 007152 012720 154024
969 007156 012720 116000
970 007162 012720 001000

```

```

971 007166 012720 001000      MOV      #1000,(R0)+      ;Y = 1000
972 007172 012720 154021      MOV      #154021,(R0)+   ;LD STATUS C' (SCALE TO BE UPDATED)
973 007176 012720 106000      MOV      #106000,(R0)+   ;SHORT VECTOR INSTR
974 007202 012720 022104      MOV      #22104,(R0)+    ;X = 10 & Y = 4 IN MINUS DIRECTION
975 007206 012710 172000      MOV      #172000,(R0)    ;LD STOP INSTR
976 007212 012701 000776      MOV      #776,R1         ;SET UP X EXPECTED DATA
977 007216 012702 000777      MOV      #777,R2         ;SET UP Y EXPECTED DATA
978 007222 012737 000001 001164  MOV      #1,$TMP0        ;SET UP SCALE FOR ER TYPE
979 007230 012737 007236 001110  MOV      #1,$SLPERR      ;SET UP SCOPE LOOP ADRS
980 007236 004737 024524      JSR      PC,EXECUTE      ;START DISPLAY
981 007242 013737 001764 001122  MOV      XPOS,$BDDADR    ;SET UP REG ADRS 04
982 007250 010137 001124      MOV      R1,$GDDAT       ;SET UP EXPECTED
983 007254 012737 000010 001162  MOV      #10,$REGO       ;SET UP VECTOR LENGTH FOR ER TYPE
984 007262 017737 172476 001126  MOV      @XPOS,$BDDAT    ;READ X POS REG
985 007270 023737 001124 001126  CMP      $GDDAT,$BDDAT   ;CORRECT?
986 007276 001401              BEQ      2$              ;BR IF SO
987 007300 104011              ERROR    11              ;X VECTOR POS FAILED TO SCALE
988 007302 013737 001766 001122 2$:  MOV      YPOS,$BDDADR    ;SET UP REG ADRS 06
989 007310 010237 001124      MOV      R2,$GDDAT       ;LD EXPECTED
990 007314 012737 000004 001162  MOV      #4,$REGO        ;SET UP VECTOR LENGTH FOR ER TYPE
991 007322 017737 172440 001126  MOV      @YPOS,$BDDAT    ;READ Y POS REG
992 007330 023737 001124 001126  CMP      $GDDAT,$BDDAT   ;CORRECT?
993 007336 001401              BEQ      3$              ;BR IF SO
994 007340 104011              ERROR    11              ;Y VECTOR POS FAILED TO SCALE
995 007342 005237 001164 3$:  INC      $TMP0           ;ADVANCE SCALE BY 1/4
996 007346 022737 000020 001164  CMP      #20,$TMP0       ;ALL SCALES TESTED?
997 007354 001406              BEQ      TST26           ;NEXT TEST IF SO
998 007356 005237 032540      INC      BUFFER+10      ;ADVANCE SCALE BY 1/4 AT ISTR LOC.
999 007362 162701 000002      SUB      #2,R1           ;DECREASE EXPECTED X POS BY 1/4
1000 007366 005302      DEC      R2              ;DECREASE EXPECTED Y POS BY 1/4
1001 007370 000722      BR       1$              ;TRY NEXT SCALE
1002
1003
(3)
(3)
(2) 007372 000004
(1) 007374 012737 000200 001166
(2) 007402 012737 000026 001202
1004 007410 012737 007466 001110
1005 007416 012703 000077
1006 007422 012737 000001 001124
1007 007430 012704 000201
1008 007434 013700 001712 1$:  MOV      DBUF,R0         ;SET UP R0
1009 007440 012720 154024      MOV      #154024,(0)+    ;SCALE OF UNITY
1010 007444 012720 116000      MOV      #116000,(0)+    ;LOAD 'SET POINT DATA MODE'
1011 007450 005020      CLR      (0)+            ;CLEAR X AXIS
1012 007452 005020      CLR      (0)+            ;CLEAR Y AXIS
1013 007454 012720 130000      MOV      #130000,(0)+    ;LOAD 'SET RELATIVE POINT MODE'
1014 007460 010420      MOV      R4,(0)+         ;PRESET 'DELTA X AND DELTA Y'
1015 007462 012720 172000      MOV      #172000,(R0)+   ;LOAD DISPLAY STOP
1016 007466 004737 024524 2$:  JSR      PC,EXECUTE      ;START DISPLAY AND WAIT FOR STOP FLAG
1017
1018 007472 017737 172266 001126  MOV      @XPOS,$BDDAT    ;READ X POSITION
1019 007500 023737 001124 001126  CMP      $GDDAT,$BDDAT   ;ARE THEY EQUAL
1020 007506 001404      BEQ      3$              ;YES

```

\*\*\*\*\*  
 ;\*TST 26 TEST THAT X AND Y AXIS INCREMENT PROPERLY USING RELATIVE POINT MODE COUN  
 ;\*\*\*\*\*

TST26: SCOPE

```

MOV      #200,$TIMES      ;;DO 200 ITERATIONS
MOV      #26,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
MOV      #2,$SLPERR      ;SET UP SCOPE LOOP ADRS
MOV      #77,R3          ;SET UP A COUNT LOCATION
MOV      #1,$GDDAT       ;SET UP THE COMPARED LOCATION
MOV      #201,R4         ;SET UP 'DELTA X-Y'
MOV      DBUF,R0         ;SET UP R0
MOV      #154024,(0)+    ;SCALE OF UNITY
MOV      #116000,(0)+    ;LOAD 'SET POINT DATA MODE'
CLR      (0)+            ;CLEAR X AXIS
CLR      (0)+            ;CLEAR Y AXIS
MOV      #130000,(0)+    ;LOAD 'SET RELATIVE POINT MODE'
MOV      R4,(0)+         ;PRESET 'DELTA X AND DELTA Y'
MOV      #172000,(R0)+   ;LOAD DISPLAY STOP

```

```

1021 007510 013737 001764 001122      MOV    XPOS,$BDADR      ;SET UP REG ADRS 04
1022 007516 104016                      ERROR  16              ;INCREMENT X AXIS FAILED USING
1023
1024 007520 017737 172242 001126 3$:  MOV    @YPOS,$BDDAT    ;READ Y POSITION
1025 007526 023737 001124 001126      CMP    $GDDAT,$BDDAT   ;ARE THEY EQUAL ?
1026 007534 001404                      BEQ    4$              ;YES
1027 007536 013737 001766 001122      MOV    YPOS,$BDADR     ;SET UP REG ADRS 06
1028 007544 104016                      ERROR  16              ;INCREMENT Y AXIS FAILED USING
1029
1030 007546 062704 000201 4$:  ADD    #201,R4          ;ADD DELTA X-Y
1031 007552 005237 001124              INC    $GDDAT          ;INCREMENT EXPECTED VALUE
1032 007556 005303                      DEC    R3              ;DECREMENT COUNT, FINISHED?
1033 007560 001325                      BNE   1$              ;;NO, TEST MORE DATA
1034
1035
(3)
(3)
(2) 007562 000004
(1) 007564 012737 000200 001166
(2) 007572 012737 000027 001202
1036 007600 012737 007652 001110
1037 007606 012703 000077
1038 007612 012737 001777 001124
1039 007620 012704 020301
1040 007624 013700 001712 1$:  MOV    DBUF,R0          ;SET UP R0
1041 007630 012720 116000              MOV    #116000,(0)+    ;LOAD 'SET POINT DATA MODE'
1042 007634 005020                      CLR    (0)+            ;CLEAR X AXIS
1043 007636 005020                      CLR    (0)+            ;CLEAR Y AXIS
1044 007640 012720 130000              MOV    #130000,(0)+    ;LOAD 'SET RELATIVE POINT MODE'
1045 007644 010420                      MOV    R4,(0)+         ;PRESET 'DELTA X AND DELTA Y'
1046 007646 012720 172000              MOV    #172000,(R0)+   ;LOAD DISPLAY STOP
1047 007652
(1) 007652 004737 024524 2$:  JSR    PC,EXECUTE      ;START DISPLAY AND WAIT FOR STOP FLAG
1048
1049 007656 017737 172102 001126      MOV    @XPOS,$BDDAT    ;READ X POSITION
1050 007664 023737 001124 001126      CMP    $GDDAT,$BDDAT   ;ARE THEY EQUAL
1051 007672 001404                      BEQ    3$              ;YES
1052 007674 013737 001764 001122      MOV    XPOS,$BDADR     ;SET UP REG ADRS 04
1053 007702 104017                      ERROR  17              ;DECREMENT X AXIS FAILED USING
1054
1055 007704 017737 172056 001126 3$:  MOV    @YPOS,$BDDAT    ;READ Y POSITION
1056 007712 023737 001124 001126      CMP    $GDDAT,$BDDAT   ;ARE THEY EQUAL ?
1057 007720 001404                      BEQ    4$              ;YES DECREMENT
1058 007722 013737 001766 001122      MOV    YPOS,$BDADR     ;SET UP REG ADRS 06
1059 007730 104017                      ERROR  17              ;DECREMENT Y AXIS FAILED USING
1060
1061 007732 062704 000201 4$:  ADD    #201,R4          ;ADD 'DELTA X-Y'
1062 007736 005337 001124              DEC    $GDDAT          ;DECREMENT EXPECTED VALUE
1063 007742 005303                      DEC    R3              ;DECREMENT COUNT, FINISHED?
1064 007744 001327                      BNE   1$              ;;NO, TEST MORE DATA
1065
1066
(3)
(3)
(2) 007746 000004
(2) 007750 012737 000030 001202

```

\*\*\*\*\*  
 : \*TEST 27 TEST THAT X AND Y AXIS DECREMENT PROPERLY USING RELATIVE POINT MODE COUN  
 : \*\*\*\*\*

```

TST27: SCOPE
MOV    #200,$TIMES      ;;DO 200 ITERATIONS
MOV    #27,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
MOV    #25,$LPERR      ;SET UP SCOPE LOOP ADRS
MOV    #77,R3          ;SET UP A COUNT LOCATION
MOV    #1777,$GDDAT    ;SET UP THE COMPARED LOCATION
MOV    #20301,R4       ;PRESET THE 'DELTA X-Y'
1$:  MOV    DBUF,R0          ;SET UP R0
MOV    #116000,(0)+    ;LOAD 'SET POINT DATA MODE'
CLR    (0)+            ;CLEAR X AXIS
CLR    (0)+            ;CLEAR Y AXIS
MOV    #130000,(0)+    ;LOAD 'SET RELATIVE POINT MODE'
MOV    R4,(0)+         ;PRESET 'DELTA X AND DELTA Y'
MOV    #172000,(R0)+   ;LOAD DISPLAY STOP
2$:  JSR    PC,EXECUTE      ;START DISPLAY AND WAIT FOR STOP FLAG
MOV    @XPOS,$BDDAT    ;READ X POSITION
CMP    $GDDAT,$BDDAT   ;ARE THEY EQUAL
BEQ    3$              ;YES
MOV    XPOS,$BDADR     ;SET UP REG ADRS 04
ERROR  17              ;DECREMENT X AXIS FAILED USING
3$:  MOV    @YPOS,$BDDAT    ;READ Y POSITION
CMP    $GDDAT,$BDDAT   ;ARE THEY EQUAL ?
BEQ    4$              ;YES DECREMENT
MOV    YPOS,$BDADR     ;SET UP REG ADRS 06
ERROR  17              ;DECREMENT Y AXIS FAILED USING
4$:  ADD    #201,R4          ;ADD 'DELTA X-Y'
DEC    $GDDAT          ;DECREMENT EXPECTED VALUE
DEC    R3              ;DECREMENT COUNT, FINISHED?
BNE   1$              ;;NO, TEST MORE DATA

```

\*\*\*\*\*  
 : \*TEST 30 TEST RELATIVE POINT FOR ALL VALUES OF SCALE  
 : \*\*\*\*\*

```

TST30: SCOPE
MOV    #30,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX

```

```

1067 007756 012700 032530      MOV      #BUFFER,R0      ;GET 1ST ADRS WHERE INSTR'S GO
1068 007762 012720 154024      MOV      #154024,(R0)+   ;LD STATUS C' (SCALE OF UNITY)
1069 007766 012720 116000      MOV      #116000,(R0)+   ;POINT INSTR
1070 007772 012720 001000      MOV      #1000,(R0)+     ;X = 1000
1071 007776 012720 001000      MOV      #1000,(R0)+     ;Y = 1000
1072 010002 012720 154021      MOV      #154021,(R0)+   ;LD STATUS C' (SCALE TO BE UPDATED)
1073 010006 012720 130000      MOV      #130000,(R0)+   ;RELATIVE POINT INSTR
1074 010012 012720 001010      MOV      #1010,(R0)+     ;X = 4 & Y = 10
1075 010016 012710 172000      MOV      #172000,(R0)    ;LD STOP INSTR
1076 010022 012701 001001      MOV      #1001,R1        ;SET UP X EXPECTED DATA
1077 010026 012702 001002      MOV      #1002,R2        ;SET UP Y EXPECTED DATA
1078 010032 012737 000001 001164  MOV      #1,$TMP0        ;SET UP SCALE FOR ER TYPE
1079 010040 012737 010046 001110  MOV      #1$,SLPERR      ;SET UP SCOPE LOOP ADRS
1080 010046 004737 024524      JSR      PC,EXECUTE      ;START DISPLAY
1081 010052 013737 001764 001122  MOV      XPOS,$BDADR     ;SET UP REG ADRS 74
1082 010060 010137 001124      MOV      R1,$GDDAT       ;LD EXPECTED
1083 010064 012737 000004 001162  MOV      #4,$REG0        ;SET UP X POINT FOR ER TYPE
1084 010072 017737 171666 001126  MOV      @XPOS,$BDDAT    ;READ X POS REG
1085 010100 023737 001124 001126  CMP      $GDDAT,$BDDAT   ;CORRECT?
1086 010106 001401              BEQ      2$              ;BR IF SO
1087 010110 104011              ERROR      11            ;X POINT POS FAILED TO SCALE
1088 010112 013737 001766 001122 2$: MOV      YPOS,$BDADR     ;SET UP REG ADRS 06
1089 010120 010237 001124      MOV      R2,$GDDAT       ;LD EXPECTED
1090 010124 012737 000010 001162  MOV      #10,$REG0       ;SET UP Y POINT FOR ER TYPE
1091 010132 017737 171630 001126  MOV      @YPOS,$BDDAT    ;READ Y POS REG
1092 010140 023737 001124 001126  CMP      $GDDAT,$BDDAT   ;CORRECT?
1093 010146 001401              BEQ      3$              ;BR IF SO
1094 010150 104011              ERROR      11            ;Y POINT POS FAILED TO SCALE
1095 010152 005237 001164 3$: INC      $TMP0          ;ADVANCE SCALE BY 1/4
1096 010156 022737 000020 001164  CMP      #20,$TMP0       ;ALL SCALES TESTED?
1097 010164 001406              BEQ      TST31           ;NEXT TEST IF SO
1098 010166 005237 032540      INC      BUFFER+10       ;ADVANCE SCALE BY 1/4 AT INSTR LOC.
1099 010172 005201              INC      R1              ;INCREASE EXPECTED X POS BY 1/4
1100 010174 062702 000002      ADD      #2,R2           ;INCREASE EXPECTED Y POS BY 1/4
1101 010200 000722              BR       1$              ;TRY NEXT SCALE
1102
1103

```

\*\*\*\*\*  
 : \*TEST 31 TEST POINT INSTR (10 BITS) FOR ALL VALUES OF SCALE  
 : \*\*\*\*\*

```

(2) 010202 000004      TST31: SCOPE
(1) 010204 012737 000004 001166  MOV      #4,$TIMES      ;;DO 4 ITERATIONS
(2) 010212 012737 000031 001202  MOV      #31,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
1104 010220 012700 032530      MOV      #BUFFER,R0     ;SET UP ADRS WHERE INSTR'S GO
1105 010224 012720 154020      MOV      #154020,(R0)+  ;LD SCALE INSTR
1106 010230 012720 116000      MOV      #116000,(R0)+  ;LD POINT INSTR
1107 010234 005020      CLR      (R0)+          ;X = 0
1108 010236 005020      CLR      (R0)+          ;Y = 0
1109 010240 012720 172000      MOV      #172000,(R0)+  ;LD STOP INSTR
1110 010244 005037 001164      CLR      $TMP0         ;SET UP SCALE FOR ER TYPE
1111 010250 005000      CLR      R0            ;START WITH 0 POINT
1112 010252 005037 001162      CLR      $REG0         ;PUT POINT IN $REG0 FOR ER TYPE
1113 010256 005001      CLR      R1            ;START WITH 0 SCALE
1114 010260 012737 010266 001110  MOV      #1$,SLPERR     ;SET UP SCOPE LOOP ADRS
1115 010266 004737 024674      JSR      PC,MULSCL     ;GO MUL POINT BY SCALE - RESULT IN $GDDAT
1116 010272 005077 171476      CLR      @XDOFF        ;INSURE ZERO X OFFSET
1117 010276 005077 171474      CLR      @YDOFF        ;INSURE ZERO Y OFFSET

```

```

1118 010302 004737 024524 JSR PC,EXECUTE ;START DISPLAY
1119 010306 013737 001764 001122 MOV XPOS,$BDADR ;SET UP REG ADRS 04
1120 010314 017737 171444 001126 MOV @XPOS,$BDDAT ;READ LOW ORDER X POS BITS
1121 010322 017705 171446 MOV @XDOFF,R5 ;READ HI ORDER X POS BITS
1122 010326 004737 025136 JSR PC,POCONV ;GO MAKE UP 14 BIT X POINT POS
1123 010332 023737 001124 001126 CMP $GDDAT,$BDDAT ;CORRECT?
1124 010340 001401 BEQ 2$ ;BR IF SO
1125 010342 104011 ERROR 11 ;X POINT POS FAILED TO SCALE
1126 010344 013737 001766 001122 2$: MOV YPOS,$BDADR ;SET UP REG ADRS 06
1127 010352 017737 171410 001126 MOV @YPOS,$BDDAT ;READ LOW ORDER Y POS BITS
1128 010360 017705 171412 MOV @YDOFF,R5 ;READ HI ORDER Y POS BITS
1129 010364 004737 025136 JSR PC,POCONV ;GO MAKE UP 14 BIT Y POINT POS
1130 010370 023737 001124 001126 CMP $GDDAT,$BDDAT ;CORRECT?
1131 010376 001401 BEQ 3$ ;BR IF SO
1132 010400 104011 ERROR 11 ;Y POINT POS FAILED TO SCALE
1133 010402 005237 001162 3$: INC $REGO ;UPDATE POINT FOR ER TYPE
1134 010406 005200 INC R0 ;UPDATE POINT BY 1
1135 010410 022700 002000 CMP #2000,R0 ;DONE 10 BITS THIS SCALE?
1136 010414 001405 BEQ 5$ ;BR IF SO
1137 010416 010037 032534 4$: MOV R0,BUFFER+4 ;SET UP X POINT
1138 010422 010037 032536 MOV R0,BUFFER+6 ;SET UP Y POINT
1139 010426 000717 BR 1$ ;REPEAT TEST SAME SCALE AT POINT +1
1140 010430 005037 001162 5$: CLR $REGO ;RESET POINT FOR ER TYPE
1141 010434 005000 CLR R0 ;RESET POINT
1142 010436 005237 001164 INC $TMP0 ;BUMP SCALE BY 1/4
1143 010442 005237 032530 INC BUFFER ;NOW AT INSTR LOC
1144 010446 005201 INC R1 ;NOW AT R1
1145 010450 022701 000020 CMP #20,R1 ;ALL SCALES TESTED?
1146 010454 001360 BNE 4$ ;REPEAT TEST AT NEW SCALE

```

```

1147
1148
(3)
(3)

```

```

(2) 010456 000004 TST32: SCOPE
(2) 010460 012737 000032 001202 MOV #32,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
1149 010466 012700 032530 MOV #BUFFER,R0 ;:GET 1ST ADRS WHERE INSTR'S GO
1150 010472 012720 154024 MOV #154024,(R0)+ ;:SCALE OF UNITY
1151 010476 012720 116000 MOV #116000,(R0)+ ;:POINT INSTR
1152 010502 005020 CLR (R0)+ ;:X = 0
1153 010504 005020 CLR (R0)+ ;:Y = 0
1154 010506 012720 154021 MOV #154021,(R0)+ ;:SCALE TO BE UPDATED
1155 010512 012720 146000 MOV #146000,(R0)+ ;:ABSOLUTE VECTOR INSTR
1156 010516 012720 000400 MOV #400,(R0)+ ;:X = 400
1157 010522 012720 000200 MOV #200,(R0)+ ;:Y = 200
1158 010526 012710 172000 MOV #172000,(R0) ;:STOP INSTR
1159 010532 012737 000001 001164 MOV #1,$TMP0 ;:SET UP SCALE OF ER TYPE
1160 010540 012701 000100 MOV #100,R1 ;:SET UP X EXPECTED DATA
1161 010544 012702 000040 MOV #40,R2 ;:SET UP Y EXPECTED DATA
1162 010550 012737 010556 001110 MOV #1$,$LPERR ;:SET UP SCOPE LOOP ADRS
1163 010556 004737 024524 1$: JSR PC,EXECUTE ;:START DISPLAY
1164 010562 013737 001764 001122 MOV XPOS,$BDADR ;:SET UP REG ADRS 04
1165 010570 010137 001124 MOV R1,$GDDAT ;:LD EXPECTED
1166 010574 012737 000400 001162 MOV #400,$REGO ;:SET UP X VECTOR FOR ER TYPE
1167 010602 017737 171156 001126 MOV @XPOS,$BDDAT ;:READ X POS REG
1168 010610 023737 001124 001126 CMP $GDDAT,$BDDAT ;:CORRECT?
1169 010616 001401 BEQ 2$ ;:BR IF SO

```

```

1170 010620 104011          ERROR 11          ;X VECTOR POS FAILED TO SCALE
1171 010622 013737 001766 001122 2$:  MOV    YPOS,$BDADR ;SET UP REG ADRS 06
1172 010630 010237 001124          MOV    R2,$GDDAT  ;LD EXPECTED
1173 010634 012737 000200 001162  MOV    #200,$REGO  ;SET UP Y VECTOR FOR ER TYPE
1174 010642 017737 171120 001126  MOV    @YPOS,$BDDAT ;READ Y POS REG
1175 010650 023737 001124 001126  CMP    $GDDAT,$BDDAT ;CORRECT?
1176 010656 001401          BEQ    3$          ;BR IF SO
1177 010660 104011          ERROR 11          ;Y VECTOR POS FAILED TO SCALE
1178 010662 005237 001164          INC    $TMP0       ;ADVANCE SCALE BY 1/4
1179 010666 022737 000020 001164 3$:  CMP    #20,$TMP0   ;ALL SCALES TESTED?
1180 010674 001407          BEQ    TST33       ;:NEXT TEST IF SO
1181 010676 005237 032540          INC    BUFFER+10   ;BUMP SCALE AT INSTR LOC BY 1/4
1182 010702 062701 000100          ADD    #100,R1     ;BUMP EXPECTED X POS BY 1/4
1183 010706 062702 000040          ADD    #40,R2     ;BUMP EXPECTED Y POS BY 1/4
1184 010712 000721          BR     1$          ;TRY NEXT SCALE
1185
1186

```

(3) \*\*\*\*\*  
 \*TEST 33 USE GRAPHPLOT X MODE TO TEST Y AXIS IS INCREMENTED BY 'GRAPHPLOT INCREME  
 (3) \*\*\*\*\*

```

(2) 010714 000004          TST33: SCOPE
(1) 010716 012737 000400 001156  MOV    #400,$TIMES ;:DO 400 ITERATIONS
(2) 010724 012737 000033 001202  MOV    #33,$TESTN  ;:SET TEST NUMBER IN APT MAIL BOX
1187 010732 013737 001766 001122  MOV    YPOS,$BDADR ;:SET UP REG ADRS 06
1188 010740 012703 000077          MOV    #77,R3     ;:SET UP EXECUTION COUNTER
1189 010744 012737 000001 001124  MOV    #1,$GDDAT  ;:SET UP COMPARED DATA
1190 010752 012737 174101 001726  MOV    #174101,DSAVE ;:SET UP BASIC 'LOAD STATUS B'
1191 010760 013700 001712          MOV    DBUF,R0    ;:SET UP R0
1192 010764 012720 154024          MOV    #154024,(R0)+ ;LD STATUS C - VECTOR SCALE OF UNITY
1193 010770 012720 116000          MOV    #116000,(0)+ ;LOAD 'POINT MODE'
1194 010774 005020          CLR    (0)+       ;CLEAR X AXIS
1195 010776 005020          CLR    (0)+       ;CLEAR Y AXIS
1196 011000 013720 001726          MOV    DSAVE,(0)+ ;LOAD 'SET STATUS B'
1197 011004 012720 120000          MOV    #120000,(0)+ ;LOAD 'SET GRAPHPLOT X MODE'
1198 011010 005020          CLR    (0)+       ;LOAD 'Y GRAPHPLOT DATA'
1199 011012 012720 172000          MOV    #172000,(R0)+ ;LOAD DISPLAY STOP
1200 011016 004737 024524          JSR    PC,EXECUTE ;START DISPLAY AND WAIT FOR STOP FLAG
1201
1202 011022 017737 170740 001126  MOV    @YPOS,$BDDAT ;READ Y AXIS
1203 011030 023737 001124 001126  CMP    $GDDAT,$BDDAT ;COMPARE TO EXPECTED VALUE
1204 011036 001402          BEQ    2$          ;ARE THEY EQUAL?
1205 011040 104027          ERROR 27          ;GRAPHPLOT INC FAILED TO SET UP X POS REG
1206 011042 000406          BR     TST34       ;:
1207 011044 005237 001726          2$:  INC    DSAVE
1208 011050 005237 001124          INC    $GDDAT     ;INCREMENT THE STATUS B COUNT
1209 011054 005303          DEC    R3         ;DECREMENT THE EXECUTION COUNT
1210 011056 001340          BNE   1$          ;:TEST MORE DATA
1211
1212

```

(3) \*\*\*\*\*  
 \*TEST 34 USE GRAPHPLOT Y MODE TO TEST X AXIS IS INCREMENTED BY 'GRAPHPLOT INCREME  
 (3) \*\*\*\*\*

```

(2) 011060 000004          TST34: SCOPE
(1) 011062 012737 000400 001166  MOV    #400,$TIMES ;:DO 400 ITERATIONS
(2) 011070 012737 000034 001202  MOV    #34,$TESTN  ;:SET TEST NUMBER IN APT MAIL BOX
1213 011076 013737 001764 001122  MOV    XPOS,$BDADR ;:SET UP REG ADRS 04
1214 011104 012703 000077          MOV    #77,R3     ;:SET UP EXECUTION COUNTER
1215 011110 012737 000001 001124  MOV    #1,$GDDAT  ;:SET UP COMPARED DATA

```

```

1216 011116 012737 174101 001726      MOV      #174101,DSAVE      ;SET UP BASIC 'LOAD STATUS B'
1217 011124 013700 001712      1$:     MOV      DBUF,R0        ;SET UP R0
1218 011130 012720 116000      MOV      #116000,(0)+     ;LOAD 'POINT MODE'
1219 011134 005020      CLR      (0)+             ;CLEAR X AXIS
1220 011136 005020      CLR      (0)+             ;CLEAR Y AXIS
1221 011140 013720 001726      MOV      DSAVE,(0)+      ;LOAD 'SET STATUS B'
1222 011144 012720 124000      MOV      #124000,(0)+    ;LOAD 'SET GRAPHPLOT Y MODE'
1223 011150 005020      CLR      (0)+             ;LOAD 'Y GRAPHPLOT DATA'
1224 011152 012720 172000      MOV      #172000,(R0)+   ;LOAD DISPLAY STOP
1225 011156 004737 024524      JSR      PC,EXECUTE      ;START DISPLAY AND WAIT FOR STOP FLAG
1226
1227 011162 017737 170576 001126      MOV      @XPOS,$BDDAT    ;READ X AXIS
1228 011170 042737 176000 001126      BIC      #176000,$BDDAT  ;MASK TO BITS 0-9
1229 011176 023737 001124 001126      CMP      $GDDAT,$BDDAT  ;COMPARE TO EXPECTED VALUE
1230 011204 001402      BEQ      2$              ;ARE THEY EQUAL?
1231 011206 104027      ERROR   27              ;GRAPHPLOT INC FAILED TO SET UP Y POS REG
1232 011210 000416      BR       TST35          ;;
1233
1234 011212 005237 001726      2$:     INC      DSAVE          ;INCREMENT THE STATUS B COUNT
1235 011216 005237 001124      INC      $GDDAT         ;DECREMENT THE EXECUTION COUNT
1236 011222 005303      DEC      R3              ;TEST MORE DATA
1237 011224 001337      BNE      1$
1238
1239
1240 011226 012737 174100 032530      MOV      #174100,BUFFER
1241 011234 012737 172000 032532      MOV      #172000,BUFFER+2
1242 011242 004737 024524      JSR      PC,EXECUTE      ;START DISPLAY AND WAIT FOR STOP FLAG
1243
1244

```

\*\*\*\*\*  
 ;\*TEST 35 TEST THAT GRAPHPLOT X WILL SET UP X & Y FOR ALL VALUES OF SCALE  
 \*\*\*\*\*

```

(3)
(3)
(2) 011246 000004      TST35: SCOPE
(2) 011250 012737 000035 001202      MOV      #35,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
1245 011256 012700 032530      MOV      #BUFFER,R0      ;GET 1ST ADRS WHERE INSTR'S GO
1246 011262 012720 154021      MOV      #154021,(R0)+   ;'LD STATUS C' (VECTOR SCALE TO BE UPDATED)
1247 011266 012720 174120      MOV      #174120,(R0)+   ;'LD STATUS B' (GRAPH PLOT INC OF 20)
1248 011272 012720 116000      MOV      #116000,(R0)+   ;POINT INSTR
1249 011276 005020      CLR      (R0)+           ;X OF ZERO
1250 011300 005020      CLR      (R0)+           ;Y OF ZERO
1251 011302 012720 120000      MOV      #120000,(R0)+   ;GRAPH PLOT X INSTR
1252 011306 012720 000100      MOV      #100,(R0)+     ;X DATA OF 100
1253 011312 012710 172000      MOV      #172000,(R0)   ;STOP INSTR
1254 011316 012701 000004      MOV      #4,R1          ;SET UP EXPECTED Y GRAPH PLOT INC VALUE
1255 011322 012702 000020      MOV      #20,R2         ;SET UP EXPECTED X GRAPH PLOT INC VALUE
1256 011326 012737 000001 001164      MOV      #1,$TMFO       ;SET UP SCALE FOR ER TYPE
1257 011334 012737 011342 001110      MOV      #1,$LPERR      ;SET UP SCOPE LOOP ADRS
1258 011342 004737 024524      1$:     JSR      PC,EXECUTE      ;START DISPLAY
1259 011346 013737 001764 001122      MOV      XPOS,$BDADR    ;SET UP REG ADRS 04
1260 011354 010237 001124      MOV      R2,$GDDAT     ;LD EXPECTED
1261 011360 012737 000100 001162      MOV      #100,$REGO     ;SET UP X DATA FOR ER TYPE
1262 011366 017737 170372 001126      MOV      @XPOS,$BDDAT  ;READ X POS REG
1263 011374 042737 176000 001126      BIC      #176000,$BDDAT ;ONLY WANT XPOS BITS
1264 011402 023737 001124 001126      CMP      $GDDAT,$BDDAT ;CORRECT?
1265 011410 001401      BEQ      2$              ;BR IF SO
1266 011412 104011      ERROR   11              ;X GRAPHPLOT DATA FAILED TO SCALE
1267 011414 013737 001766 001122 2$:     MOV      YPOS,$BDADR    ;SET UP REG ADRS 06

```

```

1268 011422 010137 001124      MOV      R1,$GDDAT      ;LD EXPECTED
1269 011426 012737 000020 001162  MOV      #20,$REGO      ;SET UP Y GRAPHPLOT INC FOR ER TYPE
1270 011434 017737 170326 001126  MOV      @YPOS,$BDDAT   ;READ Y POS REG
1271 011442 023737 001124 001126  CMP      $GDDAT,$BDDAT  ;CORRECT?
1272 011450 001401          BEQ      3$             ;BR IF SO
1273 011452 104011          ERROR    11            ;Y GRAPHPLOT INC FAILED TO SCALE
1274 011454 005237 001164      3$: INC      $TMP0         ;ADVANCE SCALE BY 1/4
1275 011460 022737 000020 001164  CMP      #20,$TMP0     ;ALL SCALE VALUES TESTED?
1276 011466 001407          BEQ      TST36         ;NEXT TEST IF SO
1277 011470 005237 032530      INC      BUFFER        ;ADVANCE SCALE BY 1/4 AT INSTR LOC.
1278 011474 062702 000020      ADD      #20,R2        ;BUMP EXPECTED X GRAPHPLOT BY 1/4
1279 011500 062701 000004      ADD      #4,R1         ;BUMP EXPECTED Y GRAPHPLOT INC BY 1/4
1280 011504 000716          BR       1$            ;TRY NEXT SCALE
1281
1282
(3)
(3)
(2) 011506 000004          TST36: SCOPE
(2) 011510 012737 000036 001202  MOV      #36,$TESTN    ;;SET TEST NUMBER IN APT MAIL BOX
1283 011516 012700 032530      MOV      #BUFFER,R0    ;GET 1ST ADRS WHERE INSTR'S GO
1284 011522 012720 154021      MOV      #154021,(R0)+ ;LD STATUS C' (VECTOR SCALE TO BE UPDATED)
1285 011526 012720 174120      MOV      #174120,(R0)+ ;LD STATUS B' (GRAPH PLOT INC OF 20)
1286 011532 012720 116000      MOV      #116000,(R0+ ;POINT INSTR
1287 011536 005020      CLR      (R0)+         ;X OF ZERO
1288 011540 005020      CLR      (R0)+         ;Y OF ZERO
1289 011542 012720 124000      MOV      #124000,(R0)+ ;GRAPH PLOT Y INSTR
1290 011546 012720 000100      MOV      #100,(R0)+   ;Y DATA OF 100
1291 011552 012710 172000      MOV      #172000,(R0) ;STOP INSTR
1292 011556 012701 000004      MOV      #4,R1        ;SET UP EXPECTED X GRAPH PLOT INC VALUE
1293 011562 012702 000020      MOV      #20,R2       ;SET UP EXPECTED Y GRAPH PLOT DATA
1294 011566 012737 000001 001164  MOV      #1,$TMP0     ;SET UP SCALE FOR ER TYPE
1295 011574 012737 011602 001110  MOV      #1$,$LPERR    ;SET UP SCOPE LOOP ADRS
1296 011602 004737 024524      1$: JSR      PC,EXECUTE  ;START DISPLAY
1297 011606 013737 001766 001122  MOV      YPOS,$BADDR   ;SET UP REG ADRS 06
1298 011614 010237 001124      MOV      R2,$GDDAT    ;LD EXPECTED
1299 011620 012737 000100 001162  MOV      #100,$REGO    ;SET UP DATA FOR ER TYPE
1300 011626 017737 170134 001126  MOV      @YPOS,$BDDAT  ;READ Y POS REG
1301 011634 023737 001124 001126  CMP      $GDDAT,$BDDAT ;CORRECT?
1302 011642 001401          BEQ      2$             ;BR IF SO
1303 011644 104011          ERROR    11            ;Y GRAPHPLOT DATA FAILED TO SCALE
1304 011646 013737 001764 001122  2$: MOV      XPOS,$BADDR  ;SET UP REG ADRS 04
1305 011654 010137 001124      MOV      R1,$GDDAT    ;LD EXPECTED
1306 011660 012737 000020 001162  MOV      #20,$REGO    ;SET UP X GRAPH PLOT INC FOR ER TYPE
1307 011666 017737 170072 001126  MOV      @XPOS,$BDDAT  ;READ X POS REG
1308 011674 042737 176000 001126  BIC      #176000,$BDDAT ;SAVE ONLY X POS BITS
1309 011702 023737 001124 001126  CMP      $GDDAT,$BDDAT ;CORRECT?
1310 011710 001401          BEQ      3$             ;BR IF SO
1311 011712 104011          ERROR    11            ;X GRAPHPLOT INC FAILED TO SCALE
1312 011714 005237 001164      3$: INC      $TMP0         ;ADVANCE SCALE BY 1/4
1313 011720 022737 000020 001164  CMP      #20,$TMP0     ;ALL SCALE VALUES TESTED?
1314 011726 001407          BEQ      TST37         ;NEXT TEST IF SO
1315 011730 005237 032530      INC      BUFFER        ;ADVANCE SCALE BY 1/4 AT INSTR LOC.
1316 011734 062702 000020      ADD      #20,R2        ;BUMP EXPECTED Y GRAPH PLOT BY 1/4
1317 011740 062701 000004      ADD      #4,R1         ;BUMP EXPECTED X GRAPH PLOT INC BY 1/4
1318 011744 000716          BR       1$            ;TRY NEXT SCALE
1319

```

```

1320          ;*****
(3)          ;*TEST 37      TEST BASIC VECTOR (GRAPH X & Y) AT ORIGIN 1000/1000
(3)          ;*****
(2) 011746 000004 TST37: SCOPE
(2) 011750 012737 000037 001202      MOV #37,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
1321 011756 012737 012034 001110      MOV #1$, $LPERR    ;;SET UP SCOPE LOOP ADRS
1322 011764 012700 032530              MOV #BUFFER,R0     ;;SET UP INSTR ADRS
1323 011770 012720 174100              MOV #174100,(R0)+  ;;SET GRAPH PLOT INC TO 0
1324 011774 012720 154024              MOV #154024,(R0)+  ;'LD STATUS C' (VECTOR SCALE TO UNITY)
1325 012000 012720 116000              MOV #116000,(R0)+  ;;SET UP POINT INSTR
1326 012004 012720 001000              MOV #1000,(R0)+    ;X = 1000
1327 012010 012720 001000              MOV #1000,(R0)+    ;Y = 1000
1328 012014 012720 122000              MOV #122000,(R0)+  ;BASIC VECTOR INSTR (GRAPH X)
1329 012020 012720 036400              MOV #36400,(R0)+   ;VECTOR LENGTH OF 400
1330 012024 012710 172000              MOV #172000,(R0)   ;STOP INSTR
1331 012030 012700 012206              MOV #TBL2R,R0      ;SET UP ADRS OF EXPECTED X & Y POS VALUES
1332 012034 004737 024524              JSR PC,EXECUTE     ;START
1333 012040 013737 001764 001122      MOV XPOS,$BDADR    ;SET UP REG ADRS 04
1334 012046 011037 001124              MOV (R0),$GDDAT    ;LD EXPECTED
1335 012052 017737 167706 001126      MOV @XPOS,$BDDAT   ;READ X POS
1336 012060 042737 176000 001126      BIC #176000,$BDDAT ;RID TRASH
1337 012066 023737 001124 001126      CMP $GDDAT,$BDDAT  ;CORRECT?
1338 012074 001401                      BEQ 2$             ;BR IF SO
1339 012076 104030                      ERROR 30           ;BASIC VECTOR FAILED TO SET UP X POS
1340 012100 013737 001766 001122      MOV YPOS,$BDADR    ;SET UP REG ADRS 06
1341 012106 016037 000002 001124      MOV 2(R0),$GDDAT   ;LD EXPECTED
1342 012114 017737 167646 001126      MOV @YPOS,$BDDAT   ;READ Y POS
1343 012122 042737 176000 001126      BIC #176000,$BDDAT ;RID TRASH
1344 012130 023737 001124 001126      CMP $GDDAT,$BDDAT  ;CORRECT?
1345 012136 001401                      BEQ 3$             ;BR IF SO
1346 012140 104030                      ERROR 30           ;BASIC VECTOR FAILED TO SET UP Y POS
1347 012142 022020                      CMP (R0)+,(R0)+    ;MOV ADRS PTR TO NEXT EXPECTED X & Y POS
1348 012144 162737 004000 032544      SUB #4000,BUFFER+14 ;SET UP NEXT DIRECTION
1349 012152 100730                      BPL 1$             ;REPEAT TEST IF THIS DIRECTION NOT TESTED
1350 012154 062737 004000 032542      ADD #4000,BUFFER+12 ;NOW SET UP BASIC VECTOR (GRAPH Y)
1351 012162 032737 010000 032542      BIT #10000,BUFFER+12 ;HAS IT BEEN DONE ALREADY?
1352 012170 001426                      BEQ TST40          ;NEXT TEST IF SO
1353 012172 012737 036400 032544      MOV #36400,BUFFER+14 ;RESET DIRECTION TO PATH 7
1354 012200 012700 012206              MOV #TBL2R,R0      ;RESET ADRS PTR OF RESULTS
1355 012204 000713                      BR 1$              ;REPEAT TEST
1356
1357 012206 001400          TBL2R: 1400          ;EXPECT X & Y BASIC VECTOR POS RESULTS
1358 012210 000400          400
1359 012212 001000          1000
1360 012214 000400          400
1361 012216 000400          400
1362 012220 000400          400
1363 012222 000400          400
1364 012224 001000          1000
1365 012226 000400          400
1366 012230 001400          1400
1367 012232 001000          1000
1368 012234 001400          1400
1369 012236 001400          1400
1370 012240 001400          1400
1371 012242 001400          1400
    
```

```

1372 012244 001000          1000
1373
1374
(3)
(3)
(2) 012246 000004
(2) 012250 012737 000040 001202
1375 012256 012700 032530
1376 012262 012720 154024
1377 012266 012720 116000
1378 012272 012720 001000
1379 012276 012720 001000
1380 012302 012720 154021
1381 012306 012720 122000
1382 012312 012720 036200
1383 012316 012720 172000
1384 012322 012701 001040 1$:
1385 012326 012702 000740
1386 012332 012737 000001 001164
1387 012340 012737 012354 001110
1388 012346 012737 000200 001162
1389 012354 004737 024524 2$:
1390 012360 013737 001764 001122
1391 012366 010137 001124
1392 012372 017737 167366 001126
1393 012400 023737 001124 001126
1394 012406 001401
1395 012410 104011
1396 012412 013737 001766 001122 3$:
1397 012420 010237 001124
1398 012424 017737 167336 001126
1399 012432 023737 001124 001126
1400 012440 001401
1401 012442 104011
1402 012444 005237 001164 4$:
1403 012450 022737 000020 001164
1404 012456 001407
1405 012460 005237 032540
1406 012464 062701 000040
1407 012470 162702 000040
1408 012474 000727
1409 012476 062737 004000 032542 5$:
1410 012504 032737 010000 032542
1411 012512 001404
1412 012514 012737 154021 032540
1413 012522 000677
1414
1415
(3)
(3)
(2) 012524 000004
(2) 012526 012737 000041 001202
1416 012534 012700 032530
1417 012540 012720 154024
1418 012544 012720 116000
*419 012550 012720 00100C

```

```

*****
*TEST 40 TEST BASIC VECTOR (GRAPH X & Y) AT ORIGIN 1000/1000, PATH 7 FOR ALL VALU
*****
TST40: SCOPE
MOV #40,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
MOV #BUFFER,R0 ;;GET 1ST ADRS WHERE INSTR'S GO
MOV #154024,(R0)+ ;;LD STATUS C'' (VECTOR SCALE OF UNITY)
MOV #116000,(R0)+ ;;POINT INSTR
MOV #1000,(R0)+ ;;X = 1000
MOV #1000,(R0)+ ;;Y = 1000
MOV #154021,(R0)+ ;;LD STATUS C'' (VECTOR SCALE TO BE UPDATED)
MOV #122000,(R0)+ ;;BASIC VECTOR INSTR (GRAPH X)
MOV #36200,(R0)+ ;;VECTOR LENGTH OF 200 - PATH 7
MOV #172000,(R0)+ ;;STOP INSTR
1$: MOV #1040,R1 ;;SET UP X EXPECTED DATA
MOV #740,R2 ;;SET UP Y EXPECTED DATA
MOV #1,$TMP0 ;;SET UP SCALE FOR ER TYPE
MOV #2,$SLPERR ;;SET UP SCOPE LOOP ADRS
MOV #200,$REGO ;;SET UP VECTOR LENGTH FOR ER TYPE
2$: JSR PC,EXECUTE ;;START DISPLAY
MOV XPOS,$BADDR ;;SET UP REG ADRS 04
MOV R1,$GDDAT ;;LD EXPECTED
MOV @XPOS,$BDDAT ;;READ X POS REG
CMP $GDDAT,$BDDAT ;;CORRECT?
BEQ 3$ ;;BR IF SO
ERROR 11 ;;X VECTOR POS FAILED TO SCALE
3$: MOV YPOS,$BADDR ;;SET UP REG ADRS 06
MOV R2,$GDDAT ;;LD EXPECTED
MOV @YPOS,$BDDAT ;;READ Y POS REG
CMP $GDDAT,$BDDAT ;;CORRECT?
BEQ 4$ ;;BR IF SO
ERROR 11 ;;Y VECTOR POS FAILED TO SCALE
4$: INC $TMP0 ;;ADVANCE SCALE BY 1/4
CMP #20,$TMP0 ;;ALL SCALE VALUES TESTED?
BEQ 5$ ;;BR IF SO
INC BUFFER+10 ;;ADVANCE SCALE BY 1/4 AT INSTR LOC.
ADD #40,R1 ;;BUMP EXPECTED X POS BY 1/4
SUB #40,R2 ;;DECREASE EXPECTED Y POS BY 1/4
BR 2$ ;;TRY NEXT SCALE
5$: ADD #4000,BUFFER+12 ;;NOW SET UP BASIC VECTOR (GRAPH Y)
BIT #10000,BUFFER+12 ;;HAS IT BEEN DONE ALREADY?
BEQ TST41 ;;NEXT TEST IF SO
MOV #154021,BUFFER+10 ;;RESTORE SCALE INSTR TO 1/4
BR 1$ ;;REPEAT TEST

```

```

*****
*TEST 41 TEST BASIC SHORT VECTOR AT ORIGIN 1000/1000
*****
TST41: SCOPE
MOV #41,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
MOV #BUFFER,R0 ;;SET UP INSTR ADRS PTR
MOV #154024,(R0)+ ;;LD STATUS C'' (VECTOR SCALE TO UNITY)
MOV #116000,(R0)+ ;;POINT INSTR
MOV #1000,(R0)+ ;;X OF 1000

```

```

1420 012554 012720 001000      MOV      #1000,(R0)+      ;Y OF 1000
1421 012560 012720 136000      MOV      #136000,(R0)+  ;BASIC SHORT VECTOR INSTR
1422 012564 012720 000177      MOV      #177,(R0)+    ;LOW BYTE, 17 COUNTS, PATH 7 INITIALLY
1423 012570 012720 172000      MOV      #172000,(R0)+ ;STOP INSTR
1424 012574 005001      CLR      R1            ;0 = LOW BYTE, 1 = HI BYTE UNDER TEST
1425 012576 012700 012764      MOV      #TBL3R,R0     ;SET UP ADRS PTR OF X & Y POS RESULTS
1426 012602 012737 012610 001110 1$:      MOV      #1$, $LPERR   ;SET UP SCOPE LOOP ADRS
1427 012610 004737 024524      JSR      PC,EXECUTE    ;START
1428 012614 013737 001764 001122      MOV      XPOS,$BDADR   ;SET UP REG ADRS 04
1429 012622 011037 001124      MOV      (R0),$GDDAT   ;SET UP EXPECTED
1430 012626 017737 167132 001126      MOV      @XPOS,$BDDAT  ;READ X POS
1431 012634 042737 176000 001126      BIC      #176000,$BDDAT ;RID TRASH
1432 012642 023737 001124 001126      CMP      $GDDAT,$BDDAT ;CORRECT?
1433 012650 001401      BEQ      2$           ;BR IF SO
1434 012652 104031      ERROR   31           ;X POS FAILED ON SHORT VECTOR
1435 012654 013737 001766 001122 2$:      MOV      YPOS,$BDADR   ;SET UP REG ADRS 06
1436 012662 016037 000002 001124      MOV      2(R0),$GDDAT  ;SET UP EXPECTED
1437 012670 017737 167072 001126      MOV      @YPOS,$BDDAT  ;READ Y POS
1438 012676 042737 176000 001126      BIC      #176000,$BDDAT ;RID TRASH
1439 012704 023737 001124 001126      CMP      $GDDAT,$BDDAT ;CORRECT?
1440 012712 001401      BEQ      3$           ;BR IF SO
1441 012714 104031      ERROR   31           ;Y POS FAILED ON SHORT VECTOR
1442 012716 022020      3$:      CMP      (R0)+,(R0)+   ;MOVE ADRS PTR TO NEXT EXPECTED X & Y POS
1443 012720 005701      TST      R1           ;LOW BYTE?
1444 012722 001013      BNE      4$           ;BR IF HI BYTE
1445 012724 162737 000020 032542      SUB      #20,BUFFER+12 ;SET UP NEXT VECTOR PATH
1446 012732 100326      BPL      1$           ;BR IF ALL DIRECTIONS NOT TESTED - LOW BYTE
1447 012734 012737 037600 032542      MOV      #37600,BUFFER+12 ;SET UP HI BYTE NOW
1448 012742 012700 012764      MOV      #TBL3R,R0     ;RESET PTR OF EXPECTED X & Y POS RESULTS
1449 012746 005201      INC      R1           ;R1 NON-ZERO MEANS HI BYTE
1450 012750 000717      BR       1$           ;NOW TEST HI BYTE
1451 012752 162737 004000 032542 4$:      SUB      #4000,BUFFER+12 ;SET UP NEXT VECTOR PATH
1452 012760 100421      BMI     TST42         ;NEXT TEST IF HI BYTE TESTED ALSO
1453 012762 000712      BR       1$           ;BR TO TEST NEXT HI BYTE DIRECTION
1454
1455 012764 001017      TBL3R: 1017          ;TABLE CONTAINS EXPECTED X & Y POS RESULTS FOR BASIC SHO
1456 012766 000761      761
1457 012770 001000      1000
1458 012772 000761      761
1459 012774 000761      761
1460 012776 000761      761
1461 013000 000761      761
1462 013002 001000      1000
1463 013004 000761      761
1464 013006 001017      1017
1465 013010 001000      1000
1466 013012 001017      1017
1467 013014 001017      1017
1468 013016 001017      1017
1469 013020 001017      1017
1470 013022 001000      1000
1471
1472
(3)
(3)
(2) 013024 000004

```

```

:*****
:*TEST 42      TEST BASIC SHORT VECTOR AT ORIGIN 1000/1000, PATH 3 FOR ALL VALUES OF SC
:*****
TST42: SCOPE

```

```

(2) 013026 012737 000042 001202 MOV #42,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
1473 013034 012700 032530 MOV #BUFFER,R0 ;;GET 1ST ADRS WHERE INSTR'S GO
1474 013040 012720 154024 MOV #154024,(R0)+ ;:'LD STATUS C' (VECTOR SCALE OF UNITY)
1475 013044 012720 116000 MOV #116000,(R0)+ ;:POINT INSTR
1476 013050 012720 001000 MOV #1000,(R0)+ ;:X = 1000
1477 013054 012720 001000 MOV #1000,(R0)+ ;:Y = 1000
1478 013060 012720 154021 MOV #154021,(R0)+ ;:'LD STATUS C' (VECTOR SCALE TO BE UPDA ED)
1479 013064 012720 136000 MOV #136000,(R0)+ ;:BASIC SHORT VECTOR INSTR
1480 013070 012720 000070 MOV #70,(R0)+ ;:LOW BYTE, 10 COUNTS, PATH 3
1481 013074 012720 172000 MOV #172000,(R0)+ ;:LD STOP INSTR
1482 013100 012701 000776 MOV #776,R1 ;:SET UP X EXPECTED DATA
1483 013104 012702 001002 MOV #1002,R2 ;:SET UP Y EXPECTED DATA
1484 013110 012737 000001 001164 MOV #1,$TMP0 ;:SET UP SCALE FOR ER TYPE
1485 013116 012737 000010 001162 MOV #10,$REGO ;:SET UP VECTOR LENGTH FOR ER TYPE
1486 013124 012737 013132 001110 MOV #1$,$LPERR ;:SET UP SCOPE LOOP ADRS
1487 013132 004737 024524 1$: JSR PC,EXECUTE ;:START DISPLAY
1488 013136 013737 001764 001122 MOV XPOS,$BDADR ;:SET UP REG ADRS 04
1489 013144 010137 001124 MOV R1,$GDDAT ;:LD EXPECTED
1490 013150 017737 166610 001126 MOV @XPOS,$BDDAT ;:READ X POS REG
1491 013156 023737 001124 001126 CMP $GDDAT,$BDDAT ;:CORRECT?
1492 013164 001401 BEQ 2$ ;:BR IF SO
1493 013166 104011 ERROR 11 ;:X VECTOR POS FAILED TO SCALE
1494 013170 013737 001766 001122 2$: MOV YPOS,$BDADR ;:SET UP REG ADRS 06
1495 013176 010237 001124 MOV R2,$GDDAT ;:LD EXPECTED
1496 013202 017737 166560 001126 MOV @YPOS,$BDDAT ;:READ Y POS REG
1497 013210 023737 001124 001126 CMP $GDDAT,$BDDAT ;:CORRECT?
1498 013216 001401 BEQ 3$ ;:BR IF SO
1499 013220 104011 ERROR 11 ;:Y VECTOR POS FAILED TO SCALE
1500 013222 005237 001164 3$: INC $TMP0 ;:ADVANCE SCALE BY 1/4
1501 013226 022737 000020 001164 CMP #20,$TMP0 ;:ALL SCALE VALUES TESTED?
1502 013234 001407 BEQ TST43 ;:NEXT TEST IF SO
1503 013236 005237 032540 INC BUFFER+10 ;:ADVANCE SCALE BY 1/4 AT INSTR LOC.
1504 013242 162701 000002 SUB #2,R1 ;:DECREASE EXPECTED X POS BY 1/4
1505 013246 062702 000002 ADD #2,R2 ;:BUMP EXPECTED Y POS BY 1/4
1506 013252 000727 BR 1$ ;:TRY NEXT SCALE
1507
1508 ;:*****
(3) ;:*TEST 43 TEST VECTOR SCALE SUMMING LOGIC USING BASIC VECTOR MODE
(3) ;:*****
(2) 013254 000004 TST43: SCOPE
(2) 013256 012737 000043 001202 MOV #43,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
1509 ;:DRAW A SERIES OF 8 VECTORS, 31 UNITS LONG. TEST THAT SUMMER
1510 ;:ADDS AND SUBTRACTS CORRECTLY FOR EACH SCALE VALUE.
1511 ;:OMIT IF SCALING ECO (# ) IS NOT INSTALLED (SWR BIT 10).
1512 ;:
1513 013264 032777 002000 165646 BIT #SW10,@SWR
1514 013272 001002 BNE .+6
1515 013274 000137 013562 JMP 4$ ;:EXIT IF SWR10 NOT SET.
1516 013300 012700 032530 MOV #BUFFER,R0 ;:SET DISPLAY CODE IN BUFFER.
1517 013304 012720 154024 MOV #154024,(R0)+ ;:SCALE UNITY (1).
1518 013310 012720 116000 MOV #116000,(R0)+ ;:POINT.
1519 013314 005020 CLR (R0)+ ;:
1520 013316 005020 CLR (R0)+ ;: X 0
1521 013320 012720 154021 MOV #154021,(R0)+ ;: Y 0
1522 013324 012720 122000 MOV #122000,(R0)+ ;:SCALE 1/4 (VARIABLE).
1523 013330 012701 000010 MOV #8.,R1 ;:BASIC VECTOR MODE.

```

```

1524 013334 012720 006037 1$: MOV #6037,(R0)+ ; DIRECTION 1, LENGTH 31...
1525 013340 005301 DEC R1
1526 013342 001374 BNE 1$ ; ...8 TIMES.
1527 013344 012720 172000 MOV #172000,(R0)+ ; STOP AND TEST X/Y.
1528 013350 012701 000010 MOV #8,R1
1529 013354 012720 026037 2$: MOV #26037,(R0)+ ; DIRECTION 5, LENGTH 31...
1530 013360 005301 DEC R1
1531 013362 001374 BNE 2$ ; ...8 TIMES.
1532 013364 012720 172000 MOV #172000,(R0)+ ; STOP AND TEST X/Y AGAIN.
1533
1534 013370 012737 013414 001110 MOV #3$, $LPERR ; SET SCOPE LOOP ADDRESS.
1535 013376 012737 000001 001164 MOV #1, $TMP0 ; SCALE FACTOR FOR ERROR OUTPUT.
1536 013404 005037 001162 CLR $REG0 ; START VAL IS 0.
1537 013410 012701 000076 MOV #62.,R1 ; 31. X 8. X 1/4 = UNIT LENGTH.
1538 013414 010137 001124 3$: MOV R1,$GDDAT ; SET EXPECTED END POINT.
1539 013420 004737 024524 JSR PC,EXECUTE ; EXECUTE 8 VECTORS (POS,POS).
1540 013424 017737 166334 001126 MOV @XPOS,$BDDAT
1541 013432 023737 001124 001126 CMP $GDDAT,$BDDAT
1542 013440 001401 BEQ .+4
1543 013442 104011 ERROR 11 ; X SCALE (ADDING) FAILED.
1544 013444 017737 166316 001126 MOV @YPOS,$BDDAT
1545 013452 023737 001124 001126 CMP $GDDAT,$BDDAT
1546 013460 001401 BEQ .+4
1547 013462 104011 ERROR 11 ; Y SCALE (ADDING) FAILED.
1548
1549 013464 005037 001124 CLR $GDDAT ; RESET EXPECTED X/Y.
1550 013470 004737 024552 JSR PC,CONT ; EXECUTE 8 VECTORS (NEG,NEG).
1551 013474 017737 166264 001126 MOV @XPOS,$BDDAT
1552 013502 023737 001124 001126 CMP $GDDAT,$BDDAT
1553 013510 001401 BEQ .+4
1554 013512 104011 ERROR 11 ; X SCALE (SUBTRACTING) FAILED.
1555 013514 017737 166246 001126 MOV @YPOS,$BDDAT
1556 013522 023737 001124 001126 CMP $GDDAT,$BDDAT
1557 013530 001401 BEQ .+4
1558 013532 104011 ERROR 11 ; Y SCALE (SUBTRACTING) FAILED.
1559
1560 013534 023727 001164 000017 CMP $TMP0,#17
1561 013542 001407 BEQ TST44 ;:EXIT WHEN ALL SCALES DONE
1562 013544 005237 001164 INC $TMP0 ; OTHERWISE, NEXT SCALE VALUE...
1563 013550 005237 032540 INC BUFFER+10 ;...IN DISPLAY CODE TOO.
1564 013554 062701 000076 ADD #62.,R1 ; NEXT EXPECTED END POINT.
1565 013560 000715 BR 3$ ; AND CONTINUE.
1566
1567
1568
(3)
(3)
(2) 013562 000004
(2) 013564 012737 000044 001202
1569 013572 013737 001762 001122
1570 013600 013700 001712
1571 013604 012720 154024
1572 013610 012720 116000
1573 013614 012720 001777
1574 013620 005020
1575 013622 012720 172000

```

```

:*****
: *TEST 44 TEST THAT EXCEEDING +X AXIS SETS EDGE INDICATOR
:*****

```

```

TST44: SCOPE
MOV #44,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
MOV $REG0,$BDDADR ;SET UP REG ADRS 02
MOV DBUF,R0
MOV #154024,(R0)+ ;SCALE OF UNITY
MOV #116000,(R0)+ ;LOAD POINT
MOV #1777,(R0)+ ;LOAD MAX X
CLR (R0)+ ;LOAD Y
MOV #172000,(R0)+ ;LOAD STOP

```

```

1576 013626 012720 110000      MOV      #110000,(0)+      ;LOAD LONG VECTOR
1577 013632 012720 000001      MOV      #1,(0)+          ;LOAD DELTA X
1578 013636 005020              CLR      (0)+             ;LOAD DELTA Y
1579 013640 012720 172000      MOV      #172000,(0)+     ;LOAD STOP
1580 013644 004737 024524      JSR      PC,EXECUTE       ;START DISPLAY AND WAIT FOR STOP FLAG
1581
1582 013650 005037 001124      CLR      $GDDAT           ;CLEAR EXPECTED
1583 013654 017737 166102 001126  MOV      @SREG0,$BDDAT    ;READ STATUS
1584 013662 032737 000040 001126  BIT      #40,$BDDAT       ;TEST BIT 5
1585 013670 001402              BEQ      1$
1586 013672 104020              ERROR   20                ;EDGE INDICATOR SET IN ERROR
1587 013674 000432              BR      TST45             ;;
1588
1589
1590 013676              1$:
(1) 013676 004737 024552      JSR      PC,CONT          ;RESUME DISPLAY AND WAIT FOR STOP FLAG
1591 013702 012737 000040 001124  MOV      #BIT5,$GDDAT     ;LOAD EXPECTED
1592 013710 017737 166046 001126  MOV      @SREG0,$BDDAT    ;READ STATUS
1593 013716 032737 000040 001126  BIT      #40,$BDDAT       ;TEST BIT 5
1594 013724 001002              BNE     2$
1595 013726 104020              ERROR   20                ;EDGE INDICATOR FAILED TO SET
1596 013730 000414              BR      TST45             ;;
1597
1598 013732              2$:
(1) 013732 004737 024524      JSR      PC,EXECUTE       ;START DISPLAY AND WAIT FOR STOP FLAG
1599 013736 005037 001124      CLR      $GDDAT           ;CLEAR EXPECTED
1600 013742 017737 166014 001126  MOV      @SREG0,$BDDAT    ;READ STATUS
1601 013750 032737 000040 001126  BIT      #40,$BDDAT       ;TEST BIT 5
1602 013756 001401              BEQ     TST45             ;BR IF EQUAL
1603 013760 104020              ERROR   20                ;EDGE INDICATOR FAILED TO CLEAR
1604
1605 *****
(3) *TEST 45      TEST THAT EXCEEDING -X AXIS SETS EDGE INDICATOR
(3) *****
(2) TST45: SCOPE
(2) 013762 000004
1606 013764 012737 000045 001202  MOV      #45,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
1607 013772 013737 001762 001122  MOV      SREG0,$BDDADR   ;SET UP REG ADRS 02
1608 014000 013700 001712      MOV      DBUF,R0
1609 014004 012720 116000      MOV      #116000,(0)+    ;LOAD POINT
1610 014010 005020              CLR      (0)+            ;LOAD MAX X
1611 014012 005020              CLR      (0)+            ;LOAD Y
1612 014014 012720 172000      MOV      #172000,(R0)+   ;LOAD STOP DISPLAY
1613 014020 012720 110000      MOV      #110000,(0)+    ;LOAD LONG VECTOR
1614 014024 012720 020001      MOV      #20001,(0)+     ;LOAD DELTA X
1615 014030 005020              CLR      (0)+            ;LOAD DELTA Y
1616 014032 012720 172000      MOV      #172000,(0)+    ;LOAD STOP
1617 014036 004737 024524      JSR      PC,EXECUTE       ;START DISPLAY AND WAIT FOR STOP FLAG
1618 014042 004737 024552      JSR      PC,CONT          ;RESUME DISPLAY AND WAIT FOR STOP FLAG
1619
1619 014046 012737 000040 001124  MOV      #BIT5,$GDDAT     ;LOAD EXPECTED
1620 014054 017737 165702 001126  MOV      @SREG0,$BDDAT    ;READ STATUS
1621 014062 032737 000040 001126  BIT      #40,$BDDAT       ;TEST BIT 5
1622 014070 001002              BNE     2$
1623 014072 104020              ERROR   20                ;EDGE INDICATOR FAILED TO SET
1624 014074 000414              BR      TST46             ;;
1625

```

```

1626 014076 004737 024524 2$: JSR PC,EXECUTE ;START DISPLAY AND WAIT FOR STOP FLAG
(1) 014076 005037 001124 CLR $GDDAT ;CLEAR EXPECTED
1627 014102 017737 165650 001126 MOV @SREG0,$BDDAT ;READ STATUS
1628 014106 032737 000040 001126 BIT #40,$BDDAT ;TEST BIT 5
1629 014114 001401 BEQ TST46 ;BR IF EQUAL
1630 014122 104020 ERROR 20 ;EDGE INDICATOR FAILED TO CLEAR
1631 014124
1632
1633
(3)
(3)
(2) 014126 000004
(2) 014130 012737 000046 001202 TST46: SCOPE
1634 014136 013737 001762 001122 MOV #46,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
1635 014144 013700 001712 MOV SREG0,$BDADR ;SET UP REG ADRS 02
1636 014150 012720 116000 MOV DBUF,R0
1637 014154 005020 CLR #116000,(0)+ ;LOAD POINT
1638 014156 012720 001777 MOV (0)+ ;LOAD X
1639 014162 012720 110000 MOV #1777,(0)+ ;LOAD MAX Y
1640 014166 005020 CLR #110000,(0)+ ;LOAD LONG VECTOR
1641 014170 012720 000001 MOV (0)+ ;LOAD DELTA X
1642 014174 012720 172000 MOV #1,(0)+ ;LOAD DELTA Y
1643 014200 004737 024524 JSR PC,EXECUTE ;START DISPLAY AND WAIT FOR STOP FLAG
1644
1645 014204 012737 000040 001124 MOV #BITS,$GDDAT ;LOAD EXPECTED
1646 014212 017737 165544 001126 MOV @SREG0,$BDDAT ;READ STATUS
1647 014220 033737 001124 001126 BIT $GDDAT,$BDDAT ;TEST BIT 5
1648 014226 001001 BNE TST47 ;BR IF SET
1649 014230 104020 ERROR 20 ;EDGE FLAG FAILED TO SET
1650
1651
1652
(3)
(3)
(2) 014232 000004
(1) 014234 012737 000040 001166 TST47: SCOPE
(2) 014242 012737 000047 001202 MOV #40,$TIMES ;;DO 40 ITERATIONS
1653 014250 013737 001762 001122 MOV #47,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
1654 014256 013700 001712 MOV SREG0,$BDADR ;SET UP REG ADRS 02
1655 014262 012720 116000 MOV DBUF,R0
1656 014266 005020 CLR #116000,(0)+ ;LOAD POINT
1657 014270 005020 CLR (0)+ ;LOAD X
1658 014272 012720 110000 MOV (0)+ ;LOAD Y
1659 014276 005020 CLR #110000,(0)+ ;LOAD LONG VECTOR
1660 014300 012720 020001 MOV (0)+ ;LOAD DELTA X
1661 014304 012720 172000 MOV #20001,(0)+ ;LOAD DELTA Y
1662 014310 004737 024524 JSR #172000,(0)+ ;LOAD STOP
1663 JSR PC,EXECUTE ;START DISPLAY AND WAIT FOR STOP FLAG
1664 014314 012737 000040 001124 MOV #BITS,$GDDAT ;LOAD EXPECTED
1665 014322 017737 165434 001126 MOV @SREG0,$BDDAT ;READ STATUS
1666 014330 032737 000040 001126 BIT #40,$BDDAT ;TEST BIT 5
1667 014336 001001 BNE 1$ ;BR IF SET
1668 014340 104020 ERROR 20 ;EDGE FLAG FAILED TO SET
1669 014342 000005 1$: RESET ;CLR EDGE INDICATOR
1670 014344 005037 001124 CLR $GDDAT ;EXPECT ZERO
1671 014350 017737 165406 001126 MOV @SREG0,$BDDAT ;READ REG 02
    
```

```

1672 014356 042737 177737 001126      BIC    #177737,$BDDAT ;CLR JUNK
1673 014364 001401                      BEQ    TST50           ;;NEXT TEST IF CLRED
1674 014366 104032                      ERROR  32             ;RESET FAILED TO CLEAR EDGE INDICATOR
1675
1676
1677                                     ;*****
(3)                                     ;*TEST 50      TEST EDGE INDICATOR & EDGE INTR GOING-ON TO OFF, OFF TO OFF & OFF TO ON
(3)                                     ;*****
(2) 014370 000004                        TST50: SCOPE
(1) 014372 012737 000100 001166        MOV    #100,$TIMES    ;;DO 100 ITERATIONS
(2) 014400 012737 000050 001202        MOV    #50,$TESTN    ;;SET TEST NUMBER IN APT MAIL BOX
1678                                     ;DO ABOVE TEST ON ALL FOUR SCREEN EDGES USING LONG VECTOR
1679                                     ;SEQUENCE AS FOLLOWS - VECTOR OFF SCREEN DELTA 1377 FROM
1680                                     ;POINT 1000/1000, VECTOR THRU CORNER OF SCREEN DELTA 1377
1681                                     ;TO EQUIVALENT POINT IN ADJACENT AXIS, THEN VECTOR ON SCREEN
1682                                     ;DELTA 1377 TO POINT OF ORIGIN 1000/1000
1683 014406 012737 000340 177776        MOV    #340,$PSW     ;HIGHEST PRIORITY
1684 014414 012777 014674 165402        MOV    #2,$@LPVCT   ;INTR RETURN ADRS
1685 014422 012777 000340 165376        MOV    #340,@LPVCT1 ;HIGHEST PRIORITY ON INTR
1686 014430 012700 032530                 MOV    #BUFFER,R0   ;STARTING ADRS OF INSTRS
1687 014434 012720 176060                 MOV    #176060,(R0)+ ;LD STATUS B' INSTR (ENAB EDGE FLAG INTR)
1688 014440 012720 116000                 MOV    #116000,(R0)+ ;POINT INSTR
1689 014444 012720 001000                 MOV    #1000,(R0)+  ;X=1000
1690 014450 012720 001000                 MOV    #1000,(R0)+  ;Y=1000
1691 014454 012720 112000                 MOV    #112000,(R0)+ ;LONG VECTOR INSTR
1692 014460 012720 001377                 MOV    #1377,(R0)+  ;X
1693 014464 005020                         CLR    (R0)+         ;Y
1694 014466 012720 021377                 MOV    #21377,(R0)+ ;-X
1695 014472 012720 001377                 MOV    #1377,(R0)+  ;Y
1696 014476 005020                         CLR    (R0)+         ;X
1697 014500 012720 021377                 MOV    #21377,(R0)+ ;-Y
1698 014504 005020                         CLR    (R0)+         ;X
1699 014506 012720 001377                 MOV    #1377,(R0)+  ;Y
1700 014512 012720 021377                 MOV    #21377,(R0)+ ;-X
1701 014516 012720 021377                 MOV    #21377,(R0)+ ;-Y
1702 014522 012720 001377                 MOV    #1377,(R0)+  ;X
1703 014526 005020                         CLR    (R0)+         ;Y
1704 014530 012720 021377                 MOV    #21377,(R0)+ ;-X
1705 014534 005020                         CLR    (R0)+         ;Y
1706 014536 012720 001377                 MOV    #1377,(R0)+  ;X
1707 014542 012720 021377                 MOV    #21377,(R0)+ ;-Y
1708 014546 005020                         CLR    (R0)+         ;X
1709 014550 012720 001377                 MOV    #1377,(R0)+  ;Y
1710 014554 005020                         CLR    (R0)+         ;X
1711 014556 012720 021377                 MOV    #21377,(R0)+ ;-Y
1712 014562 012720 001377                 MOV    #1377,(R0)+  ;X
1713 014566 012720 001377                 MOV    #1377,(R0)+  ;Y
1714 014572 012720 021377                 MOV    #21377,(R0)+ ;-X
1715 014576 005020                         CLR    (R0)+         ;Y
1716 014600 012710 172000                 MOV    #172000,(R0) ;STOP
1717 014604 012700 015214                 MOV    #TBL4R,R0   ;ADRS OF EXPECTED X & Y POS VALUES
1718 014610 005002                         CLR    R2          ;CLR SOFTWARE TIMER
1719 014612 012704 163636                 MOV    #163636,R4  ;THE EXPECTED EDGE INDICATOR
1720 014616 012703 000171                 MOV    #171,R3     ;CONTROLLED BY R3 & R4
1721 014622 012777 032530 165130         MOV    #BUFFER,@DPC ;START
1722 014630 005037 177776                 CLR    PSW         ;ALLOW INT TO OCCUR
    
```

```

1723 014634 005302          1$: DEC R2 ;THIS COUNTER SHOULD NEVER UNDERFLOW
1724 014636 001376          BNE 1$ ;WAIT FOR INTR
1725 014640 013737 001762 001122 MOV SREG0,$BDADR ;SET UP REG 02 ADRS
1726 014646 012737 000004 001124 MOV #4,$GDDAT ;EXPECT EDGE FLAG
1727 014654 017737 165102 001126 MOV @SREG0,$BDDAT ;READ REG 02
1728 014662 042737 177773 001126 BIC #177773,$BDDAT ;SAVE ONLY EDGE FLAG
1729 014670 104033          ERROR 33 ;EDGE FLAG FAILED TO INTERRUPT
1730 014672 000545          BR 11$ ;LOOK FOR LOOP ON TEST
1731 014674 022626          2$: CMP (SP)+,(SP)+ ;RETURN HERE ON INTR - FIX STACK SINCE NO RTI
1732 014676 013737 001762 001122 MOV SREG0,$BDADR ;SET REG ADRS 02
1733 014704 012737 000004 001124 MOV #4,$GDDAT ;EXPECT EDGE FLAG
1734 014712 000241          CLC ;ZERO CARRY
1735 014714 006003          ROR R3 ;SHIFT HI WORD
1736 014716 006004          ROR R4 ;SHIFT LO WORD
1737 014720 103003          BCC 3$ ;BR IF NOT EXPECTING EDGE INDICATOR
1738 014722 052737 000040 001124 BIS #40,$GDDAT ;EXPECT EDGE INDICATOR ALSO
1739 014730 017737 165026 001126 3$: MOV @SREG0,$BDDAT ;READ REG 02
1740 014736 042737 177733 001126 BIC #177733,$BDDAT ;SAVE EDGE BITS ONLY
1741 014744 023737 001124 001126 CMP $GDDAT,$BDDAT ;CORRECT?
1742 014752 001402          BEQ 4$ ;BR IF SO
1743 014754 104033          ERROR 33 ;EDGE FLAG OR INDICATOR FAILED TO SET ON INTERRUPT
1744 014756 000513          BR 11$ ;LOOK FOR LOOP ON TEST
1745 014760 012037 001124          4$: MOV (R0)+,$GDDAT ;LD EXPECTED X POS
1746 014764 013737 001764 001122 MOV XPOS,$BDADR ;SET UP REG 04 ADRS
1747 014772 017737 164766 001126 MOV @XPOS,$BDDAT ;READ X POS
1748 015000 017705 164770          MOV @XDOFF,R5 ;GET HI X POS
1749 015004 004737 025136          JSR PC,POCONV ;MAKE SURE NOT GREATER THAN 10 BITS
1750 015010 023737 001124 001126 CMP $GDDAT,$BDDAT ;IS IT CORRECT?
1751 015016 001402          BEQ 5$ ;BR IF SO
1752 015020 104033          ERROR 33 ;X POS INCORRECT ON INTR
1753 015022 000471          BR 11$ ;LOOK FOR LOOP ON TEST
1754 015024 012037 001124          5$: MOV (R0)+,$GDDAT ;LD EXPECTED Y POS
1755 015030 013737 001766 001122 MOV YPOS,$BDADR ;SET UP REG 06 ADRS
1756 015036 017737 164724 001126 MOV @YPOS,$BDDAT ;READ Y POS
1757 015044 017705 164726          MOV @YDOFF,R5 ;GET HI Y POS
1758 015050 004737 025136          JSR PC,POCONV ;MAKE SURE NOT GREATER THAN 02 BITS
1759 015054 023737 001124 001126 CMP $GDDAT,$BDDAT ;IS IT CORRECT?
1760 015062 001402          BEQ 6$ ;BR IF SO
1761 015064 104033          ERROR 33 ;Y POS INCORRECT ON INTR
1762 015066 000447          BR 11$ ;LOOK FOR LOOP ON TEST
1763 015070 005710          6$: TST (R0) ;HAVE 24 INTRS BEEN DONE? (18 OFF-6 ON SCREEN)
1764 015072 100405          BMI 7$ ;BR IF SO
1765 015074 005037 177776          CLR PSW ;ALLOW INTR ON RESUME
1766 015100 005277 164654          INC @DPC ;RESUME
1767 015104 000653          BR 1$ ;RETURN TO COUNTER
1768 015106 042737 000020 032530 7$: BIC #20,BUFFER ;SET UP FOR NO EDGE FLAG INTR ENAB
1769 015114 012777 015150 164702 MOV #9$,@LPVCT ;SET UP INTR RETURN ADRS - NONE EXPECTED
1770 015122 013737 001762 001122 MOV SREG0,$BDADR ;SET UP REG ADRS 02
1771 015130 012777 032530 164622 MOV #BUFFER,@DPC ;START
1772 015136 012702 001000          MOV #1000,R2 ;SET COUNTER
1773 015142 005302          8$: DEC R2 ;COUNT
1774 015144 001376          BNE 8$ ;ALLOW TIME TO COMPLETE
1775 015146 000401          BR 10$ ;SKIP OVER INTR RETURN
1776 015150 022626          9$: CMP (SP)+,(SP)+ ;FIX STACK IF UNEXPECTED INTR OCCURED
1777 015152 012737 100000 001124 10$: MOV #100000,$GDDAT ;EXPECT STOP ONLY
1778 015160 017737 164576 001126 MOV @SREG0,$BDDAT ;READ REG 02
    
```

```

1779 015166 042737 077773 001126      BIC    #77773,$BDDAT ;SAVE STOP + EDGE FLAG ONLY
1780 015174 023737 001124 001126      CMP    $GDDAT,$BDDAT ;CORRECT?
1781 015202 001401                      BEQ    11$           ;BR IF SO
1782 015204 104045                      ERROR  45           ;UNEXPECTED EDGE INTR OR EDGE FAILED TO CLR
1783 015206 004737 024634      11$:  JSR    PC,RSTVEC   ;RESTORE VECTOR
1784 015212 000461                      BR     TST51        ;;TO NEXT TEST

```

```

1785
1786 015214 001000      TBL4R: 1000      ;TABLE CONTAINS EXPCTED X & Y POS RESULTS ON EA. EDGE INTR
1787 015216 001000      1000
1788 015220 001777      1777
1789 015222 001000      1000
1790 015224 001777      1777
1791 015226 001400      1400
1792 015230 001400      1400
1793 015232 001777      1777
1794 015234 001000      1000
1795 015236 001777      1777
1796 015240 001000      1000
1797 015242 001000      1000
1798 015244 001000      1000
1799 015246 001000      1000
1800 015250 001000      1000
1801 015252 001777      1777
1802 015254 000400      400
1803 015256 001777      1777
1804 015260 000000      0
1805 015262 001377      1377
1806 015264 000000      0
1807 015266 001000      1000
1808 015270 001000      1000
1809 015272 001000      1000
1810 015274 001000      1000
1811 015276 001000      1000
1812 015300 000000      0
1813 015302 001000      1000
1814 015304 000000      0
1815 015306 000401      401
1816 015310 000401      401
1817 015312 000000      0
1818 015314 001000      1000
1819 015316 000000      0
1820 015320 001000      1000
1821 015322 001000      1000
1822 015324 001000      1000
1823 015326 001000      1000
1824 015330 001000      1000
1825 015332 000000      0
1826 015334 001377      1377
1827 015336 000000      0
1828 015340 001777      1777
1829 015342 000400      400
1830 015344 001777      1777
1831 015346 001000      1000
1832 015350 001000      1000
1833 015352 001000      1000
1834 015354 100000      100000      ;TERMINATOR

```

1835  
1836  
(3)  
(3)  
(2) 015356 000004  
(1) 015360 012737 000100 001166  
(2) 015366 012737 000051 001202  
1837 015374 013737 001766 001122  
1838 015402 012737 015434 001110  
1839 015410 012777 100000 164274  
1840 015416 005037 001124  
1841 015422 005037 001726  
1842 015426 012737 172000 032534  
1843 015434 013777 001726 164252  
1844 015442 004737 024524  
1845  
1846 015446 017737 164314 001126  
1847 015454 042737 001777 001126  
1848 015462 023737 001124 001126  
1849 015470 001402  
1850 015472 104021  
1851 015474 000415  
1852  
1853 015476 062737 002000 001124  
1854 015504 005237 001726  
1855 015510 022737 000016 001726  
1856 015516 001767  
1857 015520 022737 000100 001726  
1858 015526 001342  
1859  
1860  
(3)  
(3)  
(2) 015530 000004  
(2) 015532 012737 000052 001202  
1861 015540 012777 116000 164144  
1862 015546 012777 001000 164140  
1863 015554 012777 001000 164134  
1864 015562 012777 100000 164130  
1865 015570 005077 164126  
1866 015574 012777 172000 164122  
1867 015602 004737 024524  
1868  
1869 015606 017737 164152 001126  
1870 015614 012737 001000 001124  
1871 015622 023737 001124 001126  
1872 015630 001405  
1873 015632 013737 001764 001122  
1874 015640 104022  
1875 015642 000416  
1876  
1877 015644 017737 164116 001126  
1878 015652 042737 176000 001126  
1879 015660 023737 001124 001126  
1880 015666 001404  
1881 015670 013737 001766 001122

```
*****  
*TEST 51 LOAD CHARACTER REGISTER WITH #0 THRU #77  
*****  
TST51: SCOPE  
MOV #100,$TIMES ;:DO 100 ITERATIONS  
MOV #51,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX  
MOV YPOS,$BDADR ;:SET UP REG ADRS 06  
MOV #1,$SLPERR ;:LOAD RETURN  
MOV #100000,@DBUF ;:LOAD 'CHARACTER MODE'  
CLR $GDDAT ;:LOAD 'NULL' CHARACTER  
CLR DSAVE ;:CLEAR EXPECT CHARACTER  
MOV #172000,BUFFER+4 ;:LOAD DISPLAY STOP  
1$: MOV DSAVE,@DBUF1 ;:LOAD BUFFER  
JSR PC,EXECUTE ;:START DISPLAY AND WAIT FOR STOP FLAG  
  
MOV @YPOS,$BDDAT ;:READ CHARACTER REG.  
BIC #1777,$BDDAT ;:MASK TO BITS 10-15  
CMP $GDDAT,$BDDAT ;:COMPARE EXPCT TO RCVD  
BEQ 2$ ;:BR IF EQUAL  
ERROR 21 ;:ERROR, CHARACTER REGISTER LOADED IN ERROR  
BR TST52 ;:BR UPON ERROR  
  
2$: ADD #2000,$GDDAT ;:UPDATE EXPECTED VALUE  
INC DSAVE ;:UPDATE CHARACTER VALUE  
CMP #16,DSAVE ;:TEST FOR 'SHIFT-OUT'  
BEQ 2$ ;:BR BACK AND GET ANOTHER CHAR  
CMP #100,DSAVE ;:TEST FOR LAST VALID CHAR <6 BITS>  
BNE 1$ ;:BR BACK IF NOT DONE  
  
*****  
*TEST 52 TEST THAT 'NULL' DOES NOT CHANGE X OR Y AXIS  
*****  
TST52: SCOPE  
MOV #52,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX  
MOV #116000,@DBUF ;:POINT MODE  
MOV #1000,@DBUF1 ;:LOAD BUFFER  
MOV #1000,@DBUF2 ;:1000,1000  
MOV #100000,@DBUF3 ;:LOAD 'CHARACTER MODE'  
CLR @DBUF4 ;:NULL CHARACTER  
MOV #172000,@DBUF5 ;:LOAD DISPLAY STOP  
JSR PC,EXECUTE ;:START DISPLAY AND WAIT FOR STOP FLAG  
  
1$: MOV @XPOS,$BDDAT ;:READ X AXIS  
MOV #1000,$GDDAT ;:LOAD EXPECTED  
CMP $GDDAT,$BDDAT ;:COMPARE EXPCT TO RCVD  
BEQ 2$ ;:YES  
MOV XPOS,$BDADR ;:SET UP REG ADRS 04  
ERROR 22 ;:'NULL' CHARACTER CHANGED X AXIS  
BR TST53 ;:  
  
2$: MOV @YPOS,$BDDAT ;:READ Y AXIS  
BIC #176000,$BDDAT ;:MASK TO BITS 0-9  
CMP $GDDAT,$BDDAT ;:COMPARE EXPCT TO RCVD  
BEQ TST53 ;:BR IF EQUAL  
MOV YPOS,$BDADR ;:SET UP REG ADRS 06
```

```
1882 015676 104022          ERROR 22          ;'NULL' CHARACTER CHANGED Y AXIS
1883
1884          ;*****
          ;*TEST 53          CHAR SCALE 1/2 - NOT ROTATED - TEST DELTA X
          ;*****
          TST53: SCOPE
          (2) 015700 000004          MOV #53,$TESTN          ;;SET TEST NUMBER IN APT MAIL BOX
          (2) 015702 012737 000053 001202 MOV #BUFFER,R0          ;LOAD BUFFER POINTER
1885 015710 012700 032530 MOV #155200,(R0)+          ;LOAD 'STATUS C NO ROT. & CHAR SCALE 0'
1886 015714 012720 155200 MOV #116000,(R0)+          ;LOAD 'PCINT MODE'
1887 015720 012720 116000 MOV #1000,(R0)+          ;X = 1000
1888 015724 012720 001000 MOV #1000,(R0)+          ;Y = 1000
1889 015730 012720 001000 MOV #100000,(R0)+          ;CHAR MODE
1890 015734 012720 100000 MOV #40,(R0)+          ; 'SPACE' & 'NULL'
1891 015740 012720 000040 MOV #172000,(R0)+          ;LOAD DISPLAY STOP
1892 015744 012720 172000 JSR PC,EXECUTE          ;START DISPLAY AND WAIT FOR STOP FLAG
1893 015750 004737 024524
1894
1895 015754 017737 164004 001126 MOV @XPOS,$BDDAT          ;READ X POSITION
1896 015762 042737 176000 001126 BIC #176000,$BDDAT          ;MASK GRAPH INC.
1897 015770 012737 001000 001124 MOV #1000,$GDDAT          ;LOAD BASE
1898 015776 063737 001740 001124 ADD CHSZ0,$GDDAT          ;LOAD EXPECTED DATA
1899 016004 023737 001124 001126 CMP $GDDAT,$BDDAT          ;COMPARE RESULT
1900 016012 001404          BEQ 1$          ;;BR IF DELTA X IS CORRECT
1901 016014 013737 001764 001122 MOV XPOS,$BDADR          ;SET UP REG ADRS 04
1902 016022 104023          ERROR 23          ;INCORRECT DELTA X WITH CHAR SCALE OF 1/2
1903 016024 017737 163736 001126 1$: MOV @YPOS,$BDDAT          ;READ Y AXIS
1904 016032 042737 176000 001126 BIC #176000,$BDDAT          ;MASK TOP BITS
1905 016040 012737 001000 001124 MOV #1000,$GDDAT          ;LOAD EXPECTED
1906 016046 023737 001124 001126 CMP $GDDAT,$BDDAT          ;TEST IF EQUAL
1907 016054 001404          BEQ TST54          ;;BR IF SAME
1908 016056 013737 001766 001122 MOV YPOS,$BDADR          ;SET UP REG ADRS 06
1909 016064 104022          ERROR 22          ;Y AXIS CHANGED IN ERROR
1910
1911          ;*****
          ;*TEST 54          CHAR SCALE 1 - NOT ROTATED - TEST DELTA X
          ;*****
          TST54: SCOPE
          (2) 016066 000004          MOV #54,$TESTN          ;;SET TEST NUMBER IN APT MAIL BOX
          (2) 016070 012737 000054 001202 MOV #BUFFER,R0          ;LOAD BUFFER POINTER
1912 016076 012700 032530 MOV #155240,(R0)+          ;LOAD 'STATUS C NO ROT. & CHAR SCALE 1'
1913 016102 012720 155240 MOV #116000,(R0)+          ;LOAD 'POINT MODE'
1914 016106 012720 116000 MOV #1000,(R0)+          ;X = 1000
1915 016112 012720 001000 MOV #1000,(R0)+          ;Y = 1000
1916 016116 012720 001000 MOV #100000,(R0)+          ;CHAR MODE
1917 016122 012720 100000 MOV #40,(R0)+          ; 'SPACE' & 'NULL'
1918 016126 012720 000040 MOV #172000,(R0)+          ;LOAD DISPLAY STOP
1919 016132 012720 172000 JSR PC,EXECUTE          ;START DISPLAY AND WAIT FOR STOP FLAG
1920 016136 004737 024524
1921
1922 016142 017737 163616 001126 MOV @XPOS,$BDDAT          ;READ X POSITION
1923 016150 042737 176000 001126 BIC #176000,$BDDAT          ;MASK GRAPH INC.
1924 016156 012737 001000 001124 MOV #1000,$GDDAT          ;LOAD BASE
1925 016164 063737 001742 001124 ADD CHSZ1,$GDDAT          ;LOAD EXPECTED DATA
1926 016172 023737 001124 001126 CMP $GDDAT,$BDDAT          ;COMPARE RESULT
1927 016200 001404          BEQ TST55          ;;BR IF DELTA X IS CORRECT
1928 016202 013737 001764 001122 MOV XPOS,$BDADR          ;SET UP REG ADRS 04
1929 016210 104023          ERROR 23          ;INCORRECT DELTA X WITH CHAR SCALE OF 1
```

```
1930
1931
(3)
(3)
(2) 016212 000004
(2) 016214 012737 000055 001202
1932 016222 012700 032530
1933 016226 012720 155300
1934 016232 012720 116000
1935 016236 012720 001000
1936 016242 012720 001000
1937 016246 012720 100000
1938 016252 012720 000040
1939 016256 012720 172000
1940 016262 004737 024524
1941
1942 016266 017737 163472 001126
1943 016274 042737 176000 001126
1944 016302 012737 001000 001124
1945 016310 063737 001744 001124
1946 016316 023737 001124 001126
1947 016324 001404
1948 016326 013737 001766 001122
1949 016334 104023
1950
1951
(3)
(3)
(2) 016336 000004
(2) 016340 012737 000056 001202
1952 016346 013737 001764 001122
1953 016354 012700 032530
1954 016360 012720 155340
1955 016364 012720 116000
1956 016370 012720 001000
1957 016374 012720 001000
1958 016400 012720 100000
1959 016404 012720 000040
1960 016410 012720 172000
1961 016414 004737 024524
1962
1963 016420 017737 163340 001126
1964 016426 042737 176000 001126
1965 016434 012737 001000 001124
1966 016442 063737 001746 001124
1967 016450 023737 001124 001126
1968 016456 001401
1969 016460 104023
1970
1971
(3)
(3)
(2) 016462 000004
(2) 016464 012737 000057 001202
1972 016472 013737 001766 001122
1973 016500 012700 032530
```

```
*****
*TEST 55 CHAR SCALE 1 1/2 - NOT ROTATED - TEST DELTA X
*****
TST55: SCOPE
MOV #55,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
MOV #BUFFER,RO ;LOAD BUFFER POINTER
MOV #155300,(RO)+ ;LOAD 'STATUS C NO ROT. & CHAR SCALE 1 1/2'
MOV #116000,(RO)+ ;LOAD 'POINT MODE'
MOV #1000,(RO)+ ;X = 1000
MOV #1000,(RO)+ ;Y = 1000
MOV #100000,(RO)+ ;CHAR MODE
MOV #40,(RO)+ ;'SPACE' & 'NULL'
MOV #172000,(RO)+ ;LOAD DISPLAY STOP
JSR PC,EXECUTE ;START DISPLAY AND WAIT FOR STOP FLAG

MOV @XPOS,$BDDAT ;READ X POSITION
BIC #176000,$BDDAT ;MASK GRAPH INC.
MOV #1000,$GDDAT ;LOAD BASE
ADD CHS22,$GDDAT ;LOAD EXPECTED DATA
CMP $GDDAT,$BDDAT ;COMPARE RESULT
BEQ TST56 ;;BR IF DELTA X IS CORRECT
MOV YPOS,$BDADR ;SET UP REG ADRS 06
ERROR 23 ;INCORRECT DELTA X WITH CHAR SCALE OF 1 1/2
```

```
1951
(3)
(3)
(2) 016336 000004
(2) 016340 012737 000056 001202
1952 016346 013737 001764 001122
1953 016354 012700 032530
1954 016360 012720 155340
1955 016364 012720 116000
1956 016370 012720 001000
1957 016374 012720 001000
1958 016400 012720 100000
1959 016404 012720 000040
1960 016410 012720 172000
1961 016414 004737 024524
1962
1963 016420 017737 163340 001126
1964 016426 042737 176000 001126
1965 016434 012737 001000 001124
1966 016442 063737 001746 001124
1967 016450 023737 001124 001126
1968 016456 001401
1969 016460 104023
1970
1971
(3)
(3)
(2) 016462 000004
(2) 016464 012737 000057 001202
1972 016472 013737 001766 001122
1973 016500 012700 032530
```

```
*****
*TEST 56 CHAR SCALE 2 - NOT ROTATED - TEST DELTA X
*****
TST56: SCOPE
MOV #56,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
MOV XPOS,$BDADR ;SET UP REG ADRS 04
MOV #BUFFER,RO ;LOAD BUFFER POINTER
MOV #155340,(RO)+ ;LOAD 'STATUS C NO ROT. & CHAR SCALE 2'
MOV #116000,(RO)+ ;LOAD 'POINT MODE'
MOV #1000,(RO)+ ;X = 1000
MOV #1000,(RO)+ ;Y = 1000
MOV #100000,(RO)+ ;CHAR MODE
MOV #40,(RO)+ ;'SPACE' & 'NULL'
MOV #172000,(RO)+ ;LOAD DISPLAY STOP
JSR PC,EXECUTE ;START DISPLAY AND WAIT FOR STOP FLAG

MOV @XPOS,$BDDAT ;READ X POSITION
BIC #176000,$BDDAT ;MASK GRAPH INC.
MOV #1000,$GDDAT ;LOAD BASE
ADD CHS23,$GDDAT ;LOAD EXPECTED DATA
CMP $GDDAT,$BDDAT ;COMPARE RESULT
BEQ TST57 ;;BR IF DELTA X IS CORRECT
MOV YPOS,$BDADR ;SET UP REG ADRS 06
ERROR 23 ;INCORRECT DELTA X WITH CHAR SCALE OF 2
```

```
1971
(3)
(3)
(2) 016462 000004
(2) 016464 012737 000057 001202
1972 016472 013737 001766 001122
1973 016500 012700 032530
```

```
*****
*TEST 57 CHAR SCALE 1/2 - NOT ROTATED - TEST DELTA Y
*****
TST57: SCOPE
MOV #57,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
MOV YPOS,$BDADR ;SET UP REG ADRS 06
MOV #BUFFER,RO ;LOAD BUFFER POINTER
```

```

1974 016504 012720 155200      MOV      #155200,(R0)+      ;LOAD 'STATUS C NO ROT. & CHAR SCALE 1/2''
1975 016510 012720 116000      MOV      #116000,(R0)+      ;LOAD 'POINT MODE''
1976 016514 012720 001000      MOV      #1000,(R0)+        ;X = 1000
1977 016520 012720 001000      MOV      #1000,(R0)+        ;Y = 1000
1978 016524 012720 100000      MOV      #100000,(R0)+      ;CHAR MODE
1979 016530 012720 000012      MOV      #12,(R0)+         ; 'LF' & 'NULL''
1980 016534 012720 172000      MOV      #172000,(R0)+     ;LOAD DISPLAY STOP
1981 016540 004737 024524      JSR      PC,EXECUTE        ;START DISPLAY AND WAIT FOR STOP FLAG
1982
1983 016544 017737 163216 001126      MOV      @YPOS,$BDDAT      ;READ Y POSITION
1984 016552 042737 176000 001126      BIC      #176000,$BDDAT    ;MASK GRAPH INC.
1985 016560 012737 001000 001124      MOV      #1000,$GDDAT     ;LOAD BASE
1986 016566 163737 001750 001124      SUB      LFSZ0,$GDDAT     ;LOAD EXPECTED DATA
1987 016574 023737 001124 001126      CMP      $GDDAT,$BDDAT    ;COMPARE RESULT
1988 016602 001401                BEQ      TST60             ;;BR IF DELTA Y IS CORRECT
1989 016604 104024                ERROR    24               ;INCORRECT DELTA Y WITH CHAR SCALE OF 1/2
1990
1991

```

```

(3)
(3)
(2) 016606 000004
(2) 016610 012737 000060 001202
1992 016616 013737 001766 001122
1993 016624 012700 032530
1994 016630 012720 155240
1995 016634 012720 116000
1996 016640 012720 001000
1997 016644 012720 001000
1998 016650 012720 100000
1999 016654 012720 000012
2000 016660 012720 172000
2001 016664 004737 024524
2002
2003 016670 017737 163072 001126      MOV      @YPOS,$BDDAT      ;READ Y POSITION
2004 016676 042737 176000 001126      BIC      #176000,$BDDAT    ;MASK GRAPH INC.
2005 016704 012737 001000 001124      MOV      #1000,$GDDAT     ;LOAD BASE
2006 016712 163737 001752 001124      SUB      LFSZ1,$GDDAT     ;LOAD EXPECTED DATA
2007 016720 023737 001124 001126      CMP      $GDDAT,$BDDAT    ;COMPARE RESULT
2008 016726 001401                BEQ      TST61             ;;BR IF DELTA Y IS CORRECT
2009 016730 104024                ERROR    24               ;INCORRECT DELTA Y WITH CHAR SCALE OF 1
2010
2011

```

```

(3)
(3)
(2) 016732 000004
(2) 016734 012737 000061 001202
2012 016742 013737 001766 001122
2013 016750 012700 032530
2014 016754 012720 155300
2015 016760 012720 116000
2016 016764 012720 001000
2017 016770 012720 001000
2018 016774 012720 100000
2019 017000 012720 000012
2020 017004 012720 172000
2021 017010 004737 024524

```

```

2022
2023 017014 017737 162746 001126      MOV      @YPOS,$BDDAT      ;READ Y POSITION
2024 017022 042737 176000 001126      BIC      #176000,$BDDAT    ;MASK GRAPH INC.
2025 017030 012737 001000 001124      MOV      #1000,$GDDAT     ;LOAD EXPECTED
2026 017036 163737 001754 001124      SUB      LFSZ2,$GDDAT     ;LOAD EXPECTED DATA
2027 017044 023737 001124 001126      CMP      $GDDAT,$BDDAT    ;COMPARE RESULT
2028 017052 001401                BEQ      TST62             ;;BR IF DELTA Y IS CORRECT
2029 017054 104024                ERROR    24               ;INCORRECT DELTA Y WITH CHAR SCALE OF 1 1/2
    
```

```

2030
2031      ;*****
      ;*TEST 62      CHAR SCALE 2 - NOT ROTATED - TEST DELTA Y
      ;*****
      TST62:  SCOPE
      (3)
      (3)
      (2) 017056 000004                MOV      #62,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
      (2) 017060 012737 000062 001202    MOV      YPOS,$BDADR     ;SET UP REG ADRS 06
2032 017066 013737 001766 001122    MOV      #BUFFER,R0      ;LOAD BUFFER POINTER
2033 017074 012700 032530                MOV      #155340,(R0)+   ;LOAD 'STATUS C NO ROT. & CHAR SCALE 2''
2034 017100 012720 155340                MOV      #116000,(R0)+   ;LOAD 'POINT MODE''
2035 017104 012720 116000                MOV      #1000,(R0)+     ;X = 1000
2036 017110 012720 001000                MOV      #1000,(R0)+     ;Y = 1000
2037 017114 012720 001000                MOV      #100000,(R0)+   ;CHAR MODE
2038 017120 012720 100000                MOV      #12,(R0)+       ; 'LF' & 'NULL''
2039 017124 012720 000012                MOV      #172000,(R0)+   ;LOAD DISPLAY STOP
2040 017130 012720 172000                JSR      PC,EXECUTE      ;START DISPLAY AND WAIT FOR STOP FLAG
2041 017134 004737 024524
2042
2043 017140 017737 162622 001126      MOV      @YPOS,$BDDAT    ;READ Y POSITION
2044 017146 042737 176000 001126      BIC      #176000,$BDDAT  ;MASK GRAPH INC.
2045 017154 012737 001000 001124      MOV      #1000,$GDDAT    ;LOAD EXPECTED
2046 017162 163737 001756 001124      SUB      LFSZ3,$GDDAT    ;LOAD EXPECTED DATA
2047 017170 023737 001124 001126      CMP      $GDDAT,$BDDAT  ;COMPARE RESULT
2048 017176 001401                BEQ      TST63             ;;BR IF DELTA Y IS CORRECT
2049 017200 104024                ERROR    24               ;INCORRECT DELTA Y WITH CHAR SCALE OF 2
    
```

```

2050
2051      ;*****
      ;*TEST 63      CHAR SCALE 1/2 - ROTATED - TEST DELTA Y
      ;*****
      TST63:  SCOPE
      (3)
      (3)
      (2) 017202 000004                MOV      #63,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
      (2) 017204 012737 000063 001202    MOV      YPOS,$BDADR     ;SET UP REG ADRS 06
2052 017212 013737 001766 001122    MOV      #BUFFER,R0      ;LOAD BUFFER POINTER
2053 017220 012700 032530                MOV      #155600,(R0)+   ;LOAD 'STATUS C ROT. & CHAR SCALE 1/2''
2054 017224 012720 155600                MOV      #116000,(R0)+   ;LOAD 'POINT MODE''
2055 017230 012720 116000                MOV      #1000,(R0)+     ;X = 1000
2056 017234 012720 001000                MOV      #1000,(R0)+     ;Y = 1000
2057 017240 012720 001000                MOV      #100000,(R0)+   ;CHAR MODE
2058 017244 012720 100000                MOV      #40,(R0)+       ; 'SPACE' & 'NULL''
2059 017250 012720 000040                MOV      #172000,(R0)+   ;LOAD DISPLAY STOP
2060 017254 012720 172000                JSR      PC,EXECUTE      ;START DISPLAY AND WAIT FOR STOP FLAG
2061 017260 004737 024524
2062
2063 017264 017737 162476 001126      MOV      @YPOS,$BDDAT    ;READ Y POSITION
2064 017272 042737 176000 001126      BIC      #176000,$BDDAT  ;MASK GRAPH INC.
2065 017300 012737 001000 001124      MOV      #1000,$GDDAT    ;LOAD BASE
2066 017306 063737 001740 001124      ADD      CHSZ0,$GDDAT    ;LOAD EXPECTED DATA
2067 017314 023737 001124 001126      CMP      $GDDAT,$BDDAT  ;COMPARE RESULT
2068 017322 001401                BEQ      1$               ;;BR IF DELTA Y IS CORRECT
2069 017324 104025                ERROR    25               ;INCORRECT DELTA Y WITH CHAR SCALE OF 1/2
    
```

```

2070 017326 017737 162432 001126 1$:  MOV @XPOS,$BDDAT ;READ X AXIS
2071 017334 042737 176000 001126    BIC #176000,$BDDAT ;MASK TOP BITS
2072 017342 012737 001000 001124    MOV #1000,$GDDAT ;LOAD EXPECTED
2073 017350 023737 001124 001126    CMP $GDDAT,$BDDAT ;TEST IF EQUAL
2074 017356 001401                BEQ TST64 ;:BR IF SAME
2075 017360 104022                ERROR 22 ;X AXIS CHANGED IN ERROR
2076
2077
(3)
(3)
(2) 017362 000004                TST64: SCOPE
(2) 017364 012737 000064 001202    MOV #64,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
2078 017372 013737 001766 001122    MOV YPOS,$BDDADR ;SET UP REG ADRS 06
2079 017400 012700 032530                MOV #BUFFER,R0 ;LOAD BUFFER POINTER
2080 017404 012720 155640                MOV #155640,(R0)+ ;LOAD 'STATUS C ROT. & CHAR SCALE 1'
2081 017410 012720 116000                MOV #116000,(R0)+ ;LOAD 'POINT MODE'
2082 017414 012720 001000                MOV #1000,(R0)+ ;X = 1000
2083 017420 012720 001000                MOV #1000,(R0)+ ;Y = 1000
2084 017424 012720 100000                MOV #100000,(R0)+ ;CHAR MODE
2085 017430 012720 000040                MOV #40,(R0)+ ; 'SPACE' & 'NULL'
2086 017434 012720 172000                MOV #172000,(R0)+ ;LOAD DISPLAY STOP
2087 017440 004737 024524                JSR PC,EXECUTE ;START DISPLAY AND WAIT FOR STOP FLAG
2088
2089 017444 017737 162316 001126    MOV @YPOS,$BDDAT ;READ Y POSITION
2090 017452 042737 176000 001126    BIC #176000,$BDDAT ;MASK GRAPH INC.
2091 017460 012737 001000 001124    MOV #1000,$GDDAT ;LOAD BASE
2092 017466 063737 001742 001124    ADD CHS21,$GDDAT ;LOAD EXPECTED DATA
2093 017474 023737 001124 001126    CMP $GDDAT,$BDDAT ;COMPARE RESULT
2094 017502 001401                BEQ TST65 ;:BR IF DELTA Y IS CORRECT
2095 017504 104025                ERROR 25 ;INCORRECT DELTA Y WITH CHAR SCALE OF 1
2096
2097

```

```

(3)
(3)
(2) 017506 000004                TST65: SCOPE
(2) 017510 012737 000065 001202    MOV #65,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
2098 017516 013737 001766 001122    MOV YPOS,$BDDADR ;SET UP REG ADRS 06
2099 017524 012700 032530                MOV #BUFFER,R0 ;LOAD BUFFER POINTER
2100 017530 012720 155700                MOV #155700,(R0)+ ;LOAD 'STATUS C ROT. & CHAR SCALE 1 1/2'
2101 017534 012720 116000                MOV #116000,(R0)+ ;LOAD 'POINT MODE'
2102 017540 012720 001000                MOV #1000,(R0)+ ;X = 1000
2103 017544 012720 001000                MOV #1000,(R0)+ ;Y = 1000
2104 017550 012720 100000                MOV #100000,(R0)+ ;CHAR MODE
2105 017554 012720 000040                MOV #40,(R0)+ ; 'SPACE' & 'NULL'
2106 017560 012720 172000                MOV #172000,(R0)+ ;LOAD DISPLAY STOP
2107 017564 004737 024524                JSR PC,EXECUTE ;START DISPLAY AND WAIT FOR STOP FLAG
2108
2109 017570 017737 162172 001126    MOV @YPOS,$BDDAT ;READ Y POSITION
2110 017576 042737 176000 001126    BIC #176000,$BDDAT ;MASK GRAPH INC.
2111 017604 012737 001000 001124    MOV #1000,$GDDAT ;LOAD BASE DATA
2112 017612 063737 001744 001124    ADD CHS22,$GDDAT ;LOAD EXPECTED DATA
2113 017620 023737 001124 001126    CMP $GDDAT,$BDDAT ;COMPARE RESULT
2114 017626 001401                BEQ TST66 ;:BR IF DELTA Y IS CORRECT
2115 017630 104025                ERROR 25 ;INCORRECT DELTA Y WITH CHAR SCALE OF 1 1/2
2116
2117

```

::\*\*\*\*\*

```

(3) ;*TEST 66 CHAR SCALE 2 - ROTATED - TEST DELTA Y
(3) ;*****
(2) 017632 000004 TST66: SCOPE
(2) 017634 012737 000066 001202 MOV #66,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
2118 017642 013737 001766 001122 MOV YPOS,$BDADR ;SET UP REG ADRS 06
2119 017650 012700 032530 MOV #BUFFER,RC ;LOAD BUFFER POINTER
2120 017654 012720 155740 MOV #155740,(R0)+ ;LOAD 'STATUS C ROT. & CHAR SCALE 2''
2121 017660 012720 116000 MOV #116000,(R0)+ ;LOAD 'POINT MODE''
2122 017664 012720 001000 MOV #1000,(R0)+ ;X = 1000
2123 017670 012720 001000 MOV #1000,(R0)+ ;Y = 1000
2124 017674 012720 100000 MOV #100000,(R0)+ ;CHAR MODE
2125 017700 012720 000040 MOV #40,(R0)+ ; 'SPACE' & 'NULL''
2126 017704 012720 172000 MOV #172000,(R0)+ ;LOAD DISPLAY STOP
2127 017710 004737 024524 JSR PC,EXECUTE ;START DISPLAY AND WAIT FOR STOP FLAG
2128
2129 017714 017737 162046 001126 MOV @YPOS,$BDDAT ;READ Y POSITION
2130 017722 042737 176000 001126 BIC #176000,$BDDAT ;MASK GRAPH INC.
2131 017730 012737 001000 001124 MOV #1000,$GDDAT ;LOAD BASE
2132 017736 063737 001746 001124 ADD CHS23,$GDDAT ;LOAD EXPECTED DATA
2133 017744 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE RESULT
2134 017752 001401 BEQ TST67 ;;BR IF DELTA Y IS CORRECT
2135 017754 104025 ERROR 25 ;INCORRECT DELTA Y WITH CHAR SCALE OF 2
2136
2137 ;*****

```

```

(3) ;*TEST 67 CHAR SCALE 1/2 - ROTATED - TEST DELTA X
(3) ;*****
(2) 017756 000004 TST67: SCOPE
(2) 017760 012737 000067 001202 MOV #67,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
2138 017766 013737 001764 001122 MOV XPOS,$BDADR ;SET UP REG ADRS 04
2139 017774 012700 032530 MOV #BUFFER,RC ;LOAD BUFFER POINTER
2140 020000 012720 155600 MOV #155600,(R0)+ ;LOAD 'STATUS C ROT. & CHAR SCALE 1/2''
2141 020004 012720 116000 MOV #116000,(R0)+ ;LOAD 'POINT MODE''
2142 020010 012720 001000 MOV #1000,(R0)+ ;X = 1000
2143 020014 012720 001000 MOV #1000,(R0)+ ;Y = 1000
2144 020020 012720 100000 MOV #100000,(R0)+ ;CHAR MODE
2145 020024 012720 000012 MOV #12,(R0)+ ; 'LF' & 'NULL''
2146 020030 012720 172000 MOV #172000,(R0)+ ;LOAD DISPLAY STOP
2147 020034 004737 024524 JSR PC,EXECUTE ;START DISPLAY AND WAIT FOR STOP FLAG
2148
2149 020040 017737 161720 001126 MOV @XPOS,$BDDAT ;READ X POSITION
2150 020046 042737 176000 001126 BIC #176000,$BDDAT ;MASK GRAPH INC.
2151 020054 012737 001000 001124 MOV #1000,$GDDAT ;LOAD BASE
2152 020062 063737 001750 001124 ADD LFSZ0,$GDDAT ;LOAD EXPECTED DATA
2153 020070 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE RESULT
2154 020076 001401 BEQ TST70 ;;BR IF DELTA X IS CORRECT
2155 020100 104026 ERROR 26 ;INCORRECT DELTA X WITH CHAR SCALE OF 1/2
2156
2157 ;*****

```

```

(3) ;*TEST 70 CHAR SCALE 1 - ROTATED - TEST DELTA X
(3) ;*****
(2) 020102 000004 TST70: SCOPE
(2) 020104 012737 000070 001202 MOV #70,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
2158 020112 013737 001764 001122 MOV XPOS,$BDADR ;SET UP REG ADRS 04
2159 020120 012700 032530 MOV #BUFFER,RC ;LOAD BUFFER POINTER
2160 020124 012720 155640 MOV #155640,(R0)+ ;LOAD 'STATUS C ROT. & CHAR SCALE 1''
2161 020130 012720 116000 MOV #116000,(R0)+ ;LOAD 'POINT MODE''

```

```
2162 020134 012720 001000      MOV      #1000,(R0)+      ;X = 1000
2163 020140 012720 001000      MOV      #1000,(R0)+      ;Y = 1000
2164 020144 012720 100000      MOV      #100000,(R0)+    ;CHAR MODE
2165 020150 012720 000012      MOV      #12,(R0)+       ; 'LF' & 'NULL'
2166 020154 012720 172000      MOV      #172000,(R0)+   ;LOAD DISPLAY STOP
2167 020160 004737 024524      JSR      PC,EXECUTE      ;START DISPLAY AND WAIT FOR STOP FLAG
2168
2169 020164 017737 161574 001126  MOV      @XPOS,$BDDAT    ;READ X POSITION
2170 020172 042737 176000 001126  BIC      #176000,$BDDAT  ;MASK GRAPH INC.
2171 020200 012737 001000 001124  MOV      #1000,$GDDAT    ;LOAD BASE
2172 020206 063737 001752 001124  ADD      LFSZ1,$GDDAT    ;LOAD EXPECTED DATA
2173 020214 023737 001124 001126  CMP      $GDDAT,$BDDAT  ;COMPARE RESULT
2174 020222 001401                BEQ      TST71           ;;BR IF DELTA X IS CORRECT
2175 020224 104026                ERROR    26             ;INCORRECT DELTA X WITH CHAR SCALE OF 1
2176
2177
(3)
(3)
(2) 020226 000004
(2) 020230 012737 000071 001202  TST71: SCOPE
2178 020236 013737 001764 001122  MOV      #71,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
2179 020244 012700 032530                MOV      XPOS,$BDADR    ;SET UP REG ADRS 04
2180 020250 012720 155700                MOV      #BUFFER,R0     ;LOAD BUFFER POINTER
2181 020254 012720 116000                MOV      #155700,(R0)+  ;LOAD 'STATUS C ROT. & CHAR SCALE 1 1/2'
2182 020260 012720 001000                MOV      #116000,(R0)+  ;LOAD 'POINT MODE'
2183 020264 012720 001000                MOV      #1000,(R0)+    ;X = 1000
2184 020270 012720 100000                MOV      #1000,(R0)+    ;Y = 1000
2185 020274 012720 000012                MOV      #100000,(R0)+  ;CHAR MODE
2186 020274 012720 000012                MOV      #12,(R0)+      ; 'LF' & 'NULL'
2187 020300 012720 172000                MOV      #172000,(R0)+  ;LOAD DISPLAY STOP
2188 020304 004737 024524                JSR      PC,EXECUTE      ;START DISPLAY AND WAIT FOR STOP FLAG
2189 020310 017737 161450 001126  MOV      @XPOS,$BDDAT    ;READ X POSITION
2190 020316 042737 176000 001126  BIC      #176000,$BDDAT  ;MASK GRAPH INC.
2191 020324 012737 001000 001124  MOV      #1000,$GDDAT    ;LOAD BASE
2192 020332 063737 001754 001124  ADD      LFSZ2,$GDDAT    ;LOAD EXPECTED DATA
2193 020340 023737 001124 001126  CMP      $GDDAT,$BDDAT  ;COMPARE RESULT
2194 020346 001401                BEQ      TST72           ;;BR IF DELTA X IS CORRECT
2195 020350 104026                ERROR    26             ;INCORRECT DELTA X WITH CHAR SCALE OF 1 1/2
2196
2197
(3)
(3)
(2) 020352 000004
(2) 020354 012737 000072 001202  TST72: SCOPE
2198 020362 013737 001764 001122  MOV      #72,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
2199 020370 012700 032530                MOV      XPOS,$BDADR    ;SET UP REG ADRS 04
2200 020374 012720 155740                MOV      #BUFFER,R0     ;LOAD BUFFER POINTER
2201 020400 012720 116000                MOV      #155740,(R0)+  ;LOAD 'STATUS C ROT. & CHAR SCALE 2'
2202 020404 012720 001000                MOV      #116000,(R0)+  ;LOAD 'POINT MODE'
2203 020410 012720 001000                MOV      #1000,(R0)+    ;X = 1000
2204 020414 012720 100000                MOV      #1000,(R0)+    ;Y = 1000
2205 020420 012720 000012                MOV      #100000,(R0)+  ;CHAR MODE
2206 020424 012720 000012                MOV      #12,(R0)+      ; 'LF' & 'NULL'
2207 020424 012720 172000                MOV      #172000,(R0)+  ;LOAD DISPLAY STOP
2208 020430 004737 024524                JSR      PC,EXECUTE      ;START DISPLAY AND WAIT FOR STOP FLAG
2209 020434 017737 161324 001126  MOV      @XPOS,$BDDAT    ;READ X POSITION
```

```

2210 020442 042737 176000 001126 BIC #176000,$BDDAT ;MASK GRAPH INC.
2211 020450 012737 001000 001124 MOV #1000,$GDDAT ;LOAD BASE
2212 020456 063737 001756 001124 ADD LFSZ3,$GDDAT ;LOAD EXPECTED DATA
2213 020464 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE RESULT
2214 020472 001401 BEQ TST73 ;:BR IF DELTA X IS CORRECT
2215 020474 104026 ERROR 26 ;INCORRECT DELTA X WITH CHAR SCALE OF 2
2216
2217

```

\*\*\*\*\*  
 \*TEST 73 TEST THAT SUPERSCRIPT ON & OFF CHANGES CHAR SCALE & DELTA X & Y(REPEAT R  
 \*\*\*\*\*

```

(3)
(3)
(2) 020476 000004 TST73: SCOPE
(2) 020500 012737 000073 001202 MOV #73,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
2218 020506 012700 032530 MOV #BUFFER,R0 ;LOAD ADRS POINTER
2219 020512 012720 155340 MOV #155340,(R0)+ ;CHAR SCALE AND ROTATE TO BE UPDATED
2220 020516 012720 116000 MOV #116000,(R0)+ ;POINT MODE
2221 020522 012720 001000 MOV #1000,(R0)+ ;X=1000
2222 020526 012720 001000 MOV #1000,(R0)+ ;Y=1000
2223 020532 012720 102000 MOV #102000,(R0)+ ;CHAR MODE
2224 020536 012720 000021 MOV #21,(R0)+ ;SUPERSCRIPT ON - LATER SUPERSCRIPT ESCAPE
2225 020542 012720 172000 MOV #172000,(R0)+ ;STOP
2226 020546 012700 021102 MOV #CSCL,R0 ;R0 POINTS TO 'LD STATUS C' INSTRS FOR SCALE
2227 020552 012703 021114 MOV #XPOSR,R3 ;R3 POINTS TO EXPECTED X DELTA
2228 020556 012704 021130 MOV #YPOSR,R4 ;R4 POINTS TO EXPECTED Y DELTA
2229 020562 012705 021066 MOV #CSCLR,R5 ;R5 POINTS TO EXPECTED CHAR SCALE
2230 020566 012702 000003 MOV #3,R2 ;R2 WHEN ZERO SAYS TO SET UP SUPERSCRIPT ESCAPE
2231 020572 012701 000001 MOV #1,R1 ;R1=0 SAYS ROTATE, R1=-1 SAYS NEXT TEST
2232 020576 012737 020604 001110 MOV #1$,$LPERR ;SET UP SCOPE LOOP ADRS
2233 020604 1$:
(1) 020604 004737 024524 JSR PC,EXECUTE ;START DISPLAY AND WAIT FOR STOP FLAG
2234 020610 013737 001772 001122 MOV SREG1,$BDADR ;SET UP REG 12 ADRS
2235 020616 011537 001124 MOV (R5),$GDDAT ;LD EXPECTED CHAR SCALE
2236 020622 017737 161144 001126 MOV @SREG1,$BDDAT ;READ CHAR SCALE
2237 020630 042737 176377 001126 BIC #176377,$BDDAT ;SAVE ONLY CHAR SCALE
2238 020636 023737 001124 001126 CMP $GDDAT,$BDDAT ;CORRECT?
2239 020644 001401 BEQ 2$ ;BR IF SO
2240 020646 104043 ERROR 43 ;SUPERSCRIPT FAILED TO CHANGE CHAR SCALE
2241 020650 013737 001764 001122 2$: MOV XPOS,$BDADR ;SET UP REG ADRS 04
2242 020656 011337 001124 MOV (R3),$GDDAT ;LD EXPECTED DELTA X
2243 020662 017737 161076 001126 MOV @XPOS,$BDDAT ;READ X DELTA
2244 020670 042737 176000 001126 BIC #176000,$BDDAT ;SAVE X ONLY
2245 020676 023737 001124 001126 CMP $GDDAT,$BDDAT ;CORRECT?
2246 020704 001401 BEQ 3$ ;BR IF SO
2247 020706 104043 ERROR 43 ;SUPERSCRIPT DELTA X ER AT CHAR SCALE IN REG 12
2248 020710 013737 001766 001122 3$: MOV YPOS,$BDADR ;SET UP REG ADS 06
2249 020716 011437 001124 MOV (R4),$GDDAT ;LD EXPECTED Y DELTA
2250 020722 017737 161040 001126 MOV @YPOS,$BDDAT ;READ DELTA Y
2251 020730 042737 176000 001126 BIC #176000,$BDDAT ;SAVE ONLY Y
2252 020736 023737 001124 001126 CMP $GDDAT,$BDDAT ;CORRECT?
2253 020744 001401 BEQ 4$ ;BR IF SO
2254 020746 104043 ERROR 43 ;SUPERSCRIPT DELTA Y ER AT CHAR SCALE IN REG 12
2255 020750 005302 4$: DEC R2 ;TIME TO TURN ON SUPER ESCAPE?
2256 020752 001003 BNE 5$ ;BR IF NOT OR ON ALREADY
2257 020754 012737 000023 032542 MOV #23,BUFFER+12 ;NOW SET UP SUPER ESCAPE
2258 020762 062703 000002 5$: ADD #2,R3 ;UPDATE X DELTA PTR
2259 020766 062704 000002 ADD #2,R4 ;UPDATE Y DELTA PTR
2260 020772 062705 000002 ADD #2,R5 ;UPDATE SCALE PTR

```

```

2261 020776 012037 032530      MOV      (R0)+,BUFFER      ;SET UP NEXT CHAR SCALE AT INSTR LOC
2262 021002 005701              TST      R1                ;THIS PASS ROTATED?
2263 021004 001003              BNE      6$                ;BR IF NOT
2264 021006 052737 000400 032530  BIS      #400,BUFFER      ;YES - SET CHAR ROTATE BIT AT INSTR LOC
2265 021014 005715              TST      (R5)              ;HAVE WE DONE SUPER AND SUPER ESCAPE?
2266 021016 100272              BPL      1$                ;BR IF NOT
2267 021020 005301              DEC      R1                ;WILL REPEAT TEST ROTATED NOW
2268 021022 100464              BMI      TST74             ;NEXT TEST IF ROTATED SUPERSCRIPT ALREADY DONE
2269 021024 012705 021066      MOV      #CSCLR,R5        ;RESET EXPECTED CHAR SCALE PTR
2270 021030 012702 000003      MOV      #3,R2            ;RESET R3 - TELLS WHEN TO SWITCH TO SUPER ESCAPE
2271 021034 012737 000021 032542  MOV      #21,BUFFER+12    ;RESET SUPER IN FILE
2272 021042 012703 021144      MOV      #XPOSRR,R3       ;R3 POINTS TO EXPECTED X DELTA WHEN ROTATED
2273 021046 012704 021160      MOV      #YPOSRR,R4       ;R4 POINTS TO EXPECTED Y DELTA WHEN ROTATED
2274 021052 012700 021102      MOV      #CSCL,R0         ;RESET PTR TO 'LD STATUS C' INSTRS (SCALE)
2275 021056 012737 155740 032530  MOV      #155740,BUFFER   ;SET UP CHAR SCALE INSTR THIS TIME ROTATED
2276 021064 000647              BR       1$                ;REPEAT TEST ROTATED NOW
2277
2278 021066 001000              CSCLR:   1000              ;TABLE OF EXPECTED CHAR SCALE RESULTS
2279 021070 000400              400
2280 021072 000000              0
2281 021074 000400              400
2282 021076 001000              1000
2283 021100 001400              1400
2284
2285 021102 155300              CSCL:    155300           ;TABLE OF CHAR SCALE INSTRS
2286 021104 155240              155240
2287 021106 155200              155200
2288 021110 155240              155240
2289 021112 155300              155300
2290
2291 021114 000775              XPOSRR:  775              ;EXPECTED SUPERSCRIPT DELTA X RESULTS
2292 021116 000774              774
2293 021120 000776              776
2294 021122 001002              1002
2295 021124 001000              1000
2296 021126 001000              1000
2297
2298
2299 021130 001030              YPOSRR:  1030            ;EXPECTED SUPERSCRIPT DELTA Y RESULTS
2300 021132 001022              1022
2301 021134 001011              1011
2302 021136 000767              767
2303 021140 000756              756
2304 021142 000750              750
2305
2306 021144 000750              XPOSRR:  750              ;EXPECTED SUPERSCRIPT DELTA X ROTATED RESULTS
2307 021146 000756              756
2308 021150 000767              767
2309 021152 001011              1011
2310 021154 001022              1022
2311 021156 001030              1030
2312
2313 021160 001000              YPOSRR:  1000            ;EXPECTED SUPERSCRIPT DELTA Y ROTATED RESULTS
2314 021162 000774              774
2315 021164 000776              776
2316 021166 001002              1002
    
```

```

2317 021170 001000          1000
2318 021172 001000          1000
2319
2320
(3)
(3)
(2) 021174 000004
(2) 021176 012737 000074 001202
2321 021204 012700 032530
2322 021210 012720 155340
2323 021214 012720 116000
2324 021220 012720 001000
2325 021224 012720 001000
2326 021230 012720 102000
2327 021234 012720 000022
2328 021240 012720 172000
2329 021244 012700 021102
2330 021250 012703 021564
2331 021254 012704 021600
2332 021260 012705 021066
2333 021264 012702 000003
2334 021270 012701 000001
2335 021274 012737 021302 001110
2336 021302
(1) 021302 004737 024524
2337 021306 013737 001772 001122
2338 021314 011537 001124
2339 021320 017737 160446 001126
2340 021326 042737 176377 001126
2341 021334 023737 001124 001126
2342 021342 001401
2343 021344 104043
2344 021346 013737 001764 001122 2$:
2345 021354 011337 001124
2346 021360 017737 160400 001126
2347 021366 042737 176000 001126
2348 021374 023737 001124 001126
2349 021402 001401
2350 021404 104043
2351 021406 013737 001766 001122 3$:
2352 021414 011437 001124
2353 021420 017737 160342 001126
2354 021426 042737 176000 001126
2355 021434 023737 001124 001126
2356 021442 001401
2357 021444 104043
2358 021446 005302
2359 021450 001003
2360 021452 012737 000024 032542
2361 021460 062703 000002 5$:
2362 021464 062704 000002
2363 021470 062705 000002
2364 021474 012037 032530
2365 021500 005701
2366 021502 001003
2367 021504 052737 000400 032530

```

```

*****
*TEST 74 TEST THAT SUBSCRIPT ON & OFF CHANGES CHAR SCALE & DELTA X & Y(REPEAT ROT
*****
TST74: SCOPE
MOV #74,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
MOV #BUFFER,R0 ;:LOAD POINTER ADRS
MOV #155340,(R0)+ ;:CHAR SCALE AND ROTATE TO BE UPDATAED
MOV #116000,(R0)+ ;:POINT MODE
MOV #1000,(R0)+ ;:X=1000
MOV #1000,(R0)+ ;:Y=1000
MOV #102000,(R0)+ ;:CHAR MODE
MOV #22,(R0)+ ;:SUBSCRIPT ON - LATER SUBSCRIPT ESCAPE
MOV #172000,(R0)+ ;:STOP
MOV #CSCL,R0 ;:R0 POINTS TO 'LD STATUS C' INSTRS FOR SCALE
MOV #XPOSR1,R3 ;:R3 POINTS TO EXPECTED X DELTA RESULTS
MOV #YPOSR1,R4 ;:R4 POINTS TO EXPECTED Y DELTA RESULTS
MOV #CSCLR,R5 ;:R5 POINTS TO EXPECTED CHAR SCALE
MOV #3,R2 ;:R2 WHEN 0 SAYS TO SET UP SUB ESCAPE
MOV #1,R1 ;:R1=0 SAYS ROTATE, R1=-1 SAYS NEXT TEST
MOV #1$, $LPERR ;:SET UP SCOPE LOOP ADRS

1$: JSR PC,EXECUTE ;:START DISPLAY AND WAIT FOR STOP FLAG
MOV SREG1,$BDADR ;:SET UP REG ADRS 12
MOV (R5),$GDDAT ;:LD EXPECTED CHAR SCALE
MOV @SREG1,$BDDAT ;:READ CHAR SCALE
BIC #176377,$BDDAT ;:SAVE ONLY SCALE
CMP $GDDAT,$BDDAT ;:CORRECT?
BEQ 2$ ;:BR IF SO
ERROR 43 ;:SUB SCRIPT FAILED TO CHANGE CHAR SCALE
MOV XPOS,$BDADR ;:SET UP REG ADRS 04
MOV (R3),$GDDAT ;:LD EXPECTED X DELTA
MOV @XPOS,$BDDAT ;:READ DELTA X
BIC #176000,$BDDAT ;:SAVE ONLY DELTA X
CMP $GDDAT,$BDDAT ;:CORRECT?
BEQ 3$ ;:BR IF SO
ERROR 43 ;:SUBSCRIPT DELTA X ER AT CHAR SCALE IN REG 12
MOV YPOS,$BDADR ;:SET UP REG ADRS 06
MOV (R4),$GDDAT ;:LD EXPECTED DELTA Y
MOV @YPOS,$BDDAT ;:READ DELTA Y
BIC #176000,$BDDAT ;:SAVE ONLY DELTA Y
CMP $GDDAT,$BDDAT ;:CORRECT?
BEQ 4$ ;:BR IF SO
ERROR 43 ;:SUBSCRIPT DELTA Y ER AT CHAR SCALE AT REG 12
DEC R2 ;:TIME TO SWITCH TO SUB ESCAPE?
BNE 5$ ;:BR IF NOT OR ON ALREADY
MOV #24,BUFFER+12 ;:NOE SET UP FOR SUB ESCAPE
ADD #2,R3 ;:UPDATE X DELTA RESULT PTR
ADD #2,R4 ;:UPDATE Y DELTA RESULT PTR
ADD #2,R5 ;:UPDATE CHAR SCALE RESULT PTR
MOV (R0)+,BUFFER ;:SET UP NEXT CHAR SCALE AT INSTR LOC
TST R1 ;:THIS PASS ROTATED?
BNE 6$ ;:BR IF NOT
BIS #400,BUFFER ;:YES - SET CHAR ROTATE BIT AT INSTR LOC

```

```

2368 021512 005715      6$:  TST      (R5)          ;HAVE WE DONE SUB AND SUB ESCAPE?
2369 021514 100272      BPL      1$             ;BR IF NOT
2370 021516 005301      DEC      R1             ;WILL REPEAT TEST NOW ROTATED
2371 021520 100451      BMI      TST75         ;:NEXT TEST IF ROTATED SUBSCRIPT ALREADY DONE
2372 021522 012705 021066  MOV      #CSCLR,R5     ;RESET EXPECTED CHAR SCALE PTR
2373 021522 012702 000003  MOV      #3,R2         ;RESET R2 - TELLS WHEN TO SWITCH TO SUB ESCAPE
2374 021532 012737 000022 032542  MOV      #22,BUFFER+12 ;RESET SUB IN FILE
2375 021540 012703 021614  MOV      #XPORR1,R3    ;R3 POINTS TO EXPECTED X DELTA ROTATED
2376 021544 012704 021630  MOV      #YPORR1,R4    ;R4 POINTS TO EXPECTED Y DELTA ROTATED
2377 021550 012700 021102  MOV      #CSCL,R0      ;RESET PTR TO 'LD STATUS C' INSTRS (SCALE)
2378 021554 012737 155740 032530  MOV      #155740,BUFFER ;SET CHAR SCALE INSTR THIS TIME ROTATED
2379 021562 000647      BR       1$             ;REPEAT TEST CAHR ROTATED
    
```

```

2380
2381 021564 000775      XPOSR1: 775           ;EXPECTED SUBSCRIPT DELTA X RESULTS
2382 021566 000774      774
2383 021570 000776      776
2384 021572 001002      1002
2385 021574 001000      1000
2386 021576 001000      1000
    
```

```

2387
2388 021600 000771      YPOSR1: 771           ;EXPECTED SUBSCRIPT DELTA Y RESULTS
2389 021602 000772      772
2390 021604 000775      775
2391 021606 001003      1003
2392 021610 001006      1006
2393 021612 001010      1010
    
```

```

2394
2395 021614 001007      XPORR1: 1007         ;EXPECTED SUBSCRIPT DELTA X ROTATED RESULTS
2396 021616 001006      1006
2397 021620 001003      1003
2398 021622 000775      775
2399 021624 000772      772
2400 021626 000770      770
    
```

```

2401
2402 021630 001000      YPORR1: 1000         ;EXPECTED SUBSCRIPT DELTA Y ROTATED RESULTS
2403 021632 000774      774
2404 021634 000776      776
2405 021636 001002      1002
2406 021640 001000      1000
2407 021642 001000      1000
    
```

```

2408
2409
(3)
(3)
(2) 021644 000004
(2) 021646 012737 000075 001202
2410 021654 013737 001764 001122
2411 021662 012700 032530
2412 021666 012720 155240
2413 021672 012720 116000
2414 021676 012720 001000
2415 021702 012720 001000
2416 021706 012720 100000
2417 021712 012720 000015
2418 021716 012720 172000
2419 021722 004737 024524
    
```

```

:*****
:*TEST 75 TEST THAT 'CR' DOES CHANGE X AND DOES NOT CHANGE Y AXIS
:*****
TST75: SCOPE
MOV #75,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
MOV XPOS,$BDADR ;:SET UP REG ADRS 04
MOV #BUFFER,R0 ;:LOAD BUFFER POINTER
MOV #155240,(R0)+ ;:LOAD CHAR SIZE 1
MOV #116000,(R0)+ ;:POINT MODE
MOV #1000,(R0)+ ;:LOAD BUFFER
MOV #1000,(R0)+ ;
MOV #100000,(R0)+ ;:LOAD 'CHARACTER MODE'
MOV #15,(R0)+ ;:LOAD 'CR'
MOV #172000,(R0)+ ;:LOAD DISPLAY STOP
JSR PC,EXECUTE ;:START DISPLAY AND WAIT FOR STOP FLAG
    
```

```

2420
2421 021726 017737 160032 001126 1$:  MOV @XPOS,$BDDAT ;READ X AXIS
2422 021734 005037 001124          CLR $GDDAT ;LOAD EXPECTED
2423 021740 023737 001124 001126  CMP $GDDAT,$BDDAT ;COMPARE EXPCT TO RCVD
2424 021746 001402          BEQ 2$ ;YES
2425 021750 104022          ERROR 22 ;'CR' CHARACTER FAILED TO CHANGED X AXIS CORRECTLY
2426 021752 000421          BR TST76 ;:
2427
2428 021754 017737 160006 001126 2$:  MOV @YPOS,$BDDAT ;READ Y AXIS
2429 021762 042737 176000 001126  BIC #176000,$BDDAT ;MASK TO BITS 0-9
2430 021770 012737 001000 001124  MOV #1000,$GDDAT ;LOAD EXPECTED
2431 021776 023737 001124 001126  CMP $GDDAT,$BDDAT ;COMPARE EXPCT TO RCVD
2432 022004 001404          BEQ TST76 ;:BR IF EQUAL
2433 022006 013737 001766 001122  MOV YPOS,$BDADR ;SET UP REG ADRS 06
2434 022014 104022          ERROR 22 ;'CR' CHARACTER CHANGED Y AXIS
2435
2436 ;:*****
(3) ;*TEST 76 TEST THAT 'BS' (NOT ROTATED) DOES CHANGE X BUT NOT Y AXIS
(3) ;:*****
(2) TST76: SCOPE
(2) 022016 000004          MOV #76,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
2437 022020 012737 000076 001202  MOV #BUFFER,R0
2438 022026 012700 032530          MOV #155240,(R0)+ ;LOAD NO ROT CHAR SIZE - 1
2439 022032 012720 155240          MOV #116000,(R0)+ ;POINT MODE
2440 022042 012720 001000          MOV #1000,(R0)+ ;LOAD BUFFER
2441 022046 012720 001000          MOV #1000,(R0)+ ;1000,1000
2442 022052 012720 100000          MOV #100000,(R0)+ ;LOAD 'CHARACTER MODE'
2443 022056 012720 000010          MOV #10,(R0)+
2444 022062 012720 172000          MOV #172000,(R0)+ ;LOAD DISPLAY STOP
2445 022066 004737 024524          JSR PC,EXECUTE ;START DISPLAY AND WAIT FOR STOP FLAG
2446 022072 017737 157666 001126 1$:  MOV @XPOS,$BDDAT ;READ X AXIS
2447 022100 012737 001000 001124  MOV #1000,$GDDAT ;LOAD EXPECTED
2448 022106 163737 001742 001124  SUB CHSZ1,$GDDAT ;ADJUST EXPECTED
2449 022114 023737 001124 001126  CMP $GDDAT,$BDDAT ;COMPARE
2450 022122 001405          BEQ 2$ ;YES
2451 022124 013737 001766 001122  MOV YPOS,$BDADR ;SET UP REG ADRS 06
2452 022132 104022          ERROR 22 ;'BS' CHARACTER FAILED TO CHANGED X AXIS CORRECTLY
2453 022134 000435          BR TST77 ;:
2454
2455 022136 017737 157624 001126 2$:  MOV @YPOS,$BDDAT ;READ Y AXIS
2456 022144 042737 176000 001126  BIC #176000,$BDDAT ;MASK TO BITS 0-9
2457 022152 012737 001000 001124  MOV #1000,$GDDAT ;LOAD EXPECTED
2458 022160 023737 001124 001126  CMP $GDDAT,$BDDAT ;COMPARE EXPCT TO RCVD
2459 022166 001405          BEQ 3$ ;YES
2460 022170 013737 001764 001122  MOV XPOS,$BDADR ;SET UP REG ADRS 04
2461 022176 104022          ERROR 22 ;'BS' CHARACTER CHANGED Y AXIS
2462 022200 000413          BR TST77 ;:
2463
2464 ;:TEST THAT 'SHIFT-OUT' STATUS BIT IS NOT SET
2465
2466 022202 017737 157554 001126 3$:  MOV @SREG0,$BDDAT ;READ STATUS
2467 022210 032737 000100 001126  BIT #100,$BDDAT
2468 022216 001404          BEQ TST77 ;:BR IF EQUAL
2469 022220 013737 001762 001122  MOV SREG0,$BDADR ;SET UP REG ADRS 02
2470 022226 104035          ERROR 35 ;SHIFT OUT STATUS BIT IS SET
2471
    
```

```

2472      ;:*****
(3)      ;*TEST 77      TEST THAT 'BS' (ROTATED) DOES CHANGE Y BUT NOT X AXIS
(3)      ;:*****
(2) 022230 000004      TST77: SCOPE
(2) 022232 012737 000077 001202      MOV      #77,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
2473 022240 012700 032530      MOV      #BUFFER,R0
2474 022244 012720 155640      MOV      #155640,(R0)+ ;LOAD ROTATE CHSIZE - 1
2475 022250 012720 116000      MOV      #116000,(R0)+ ;POINT MODE
2476 022254 012720 001000      MOV      #1000,(R0)+ ;LOAD BUFFER
2477 022260 012720 001000      MOV      #1000,(R0)+ ;1000,1000
2478 022264 012720 100000      MOV      #100000,(R0)+ ;LOAD 'CHARACTER MODE'
2479 022270 012720 000010      MOV      #10,(R0)+
2480 022274 012720 172000      MOV      #172000,(R0)+ ;LOAD DISPLAY STOP
2481 022300 004737 024524      JSR      PC,EXECUTE      ;START DISPLAY AND WAIT FOR STOP FLAG
2482 022304 017737 157456 001126 1$:      MOV      @YPOS,$BDDAT ;READ Y AXIS
2483 022312 042737 176000 001126      BIC      #176000,$BDDAT
2484 022320 012737 001000 001124      MOV      #1000,$GDDAT ;LOAD EXPECTED
2485 022326 163737 001742 001124      SUB      CHSZ1,$GDDAT ;ADJUST EXPECTED
2486 022334 023737 001124 001126      CMP      $GDDAT,$BDDAT ;COMPARE
2487 022342 001405      BEQ      2$ ;YES
2488 022344 013737 001766 001122      MOV      YPOS,$BDADR ;SET UP REG ADRS 06
2489 022352 104022      ERROR   22 ;'BS' (ROTATED) CHARACTER FAILED TO 'CHANGED Y AXIS CORR
2490 022354 000435      BR      TST100      ;;
2491
2492 022356 017737 157402 001126 2$:      MOV      @XPOS,$BDDAT ;READ X AXIS
2493 022364 042737 176000 001126      BIC      #176000,$BDDAT ;MASK TO BITS 0-9
2494 022372 012737 001000 001124      MOV      #1000,$GDDAT ;LOAD EXPECTED
2495 022400 023737 001124 001126      CMP      $GDDAT,$BDDAT ;COMPARE EXPCT TO RCVD
2496 022406 00 05      BEQ      3$ ;YES
2497 022410 013737 001764 001122      MOV      XPOS,$BDADR ;SET UP REG ADRS 04
2498 022416 104022      ERROR   22 ;'BS' (ROTATED) CHARACTER CHANGED Y AXIS
2499 022420 000413      BR      TST100      ;;
2500
2501      ;TEST THAT 'SHIFT-OUT' STATUS BIT IS NOT SET
2502
2503 022422 017737 157334 001126 3$:      MOV      @SREG0,$BDDAT ;READ STATUS
2504 022430 032737 000100 001126      BIT      #100,$BDDAT
2505 022436 001404      BEQ      TST100 ;:BR IF EQUAL
2506 022440 013737 001762 001122      MOV      SREG0,$BDADR ;SET REG ADRS 02
2507 022446 104035      ERROR   35 ;SHIFT OUT STATUS BIT IS SET
2508
2509      ;:*****
(3)      ;*TEST 100      TEST THAT 'SHIFT-OUT' GENERATES A STATUS BIT
(3)      ;:*****
(2) 022450 000004      TST100: SCOPE
(1) 022452 012737 000100 001166      MOV      #100,$TIMES ;:DO 100 ITERATIONS
(2) 022460 012737 000100 001202      MOV      #100,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
2510      ;SHIFT-OUT <LOW BYTE>, FOLLOWED BY CODE 77 <HIGH BYTE>
2511 022466 012737 000340 177776      MOV      #340,$PSW ;RAISE PRIORITY
2512 022474 012700 032530      MOV      #BUFFER,R0 ;LOAD BUFFER POINTER
2513 022500 012720 172000      MOV      #172000,(R0)+ ;LOAD 'ST'
2514 022504 012720 116000      MOV      #116000,(R0)+ ;POINT MODE
2515 022510 012720 001000      MOV      #1000,(R0)+
2516 022514 012720 001000      MOV      #1000,(R0)+ ;1000,1000
2517 022520 012720 100000      MOV      #100000,(R0)+ ;LOAD 'CHARACTER MODE'
2518 022524 012720 037416      MOV      #37416,(R0)+ ;'SHIFT-OUT' IN LOW BYTE #77 IN HIGH BYTE
    
```

```

2519 022530 012720 000017      MOV      #17,(R0)+          ;LOAD 'SHIFT-IN'
2520 022534 012720 172000      MOV      #172000,(R0)+     ;LOAD DISPLAY STOP
2521 022540 004737 024524      JSR      PC,EXECUTE        ;START DISPLAY AND WAIT FOR STOP FLAG
2522 022544 012737 000001 024632  MOV      #1,NOFLAG         ;ENABLE NO FLAG EXIT
2523 022552 004737 024552      JSR      PC,CONT           ;RESUME DISPLAY AND WAIT FOR STOP FLAG
2524 022556 012737 176000 001124  MOV      #176000,$GDDAT
2525 022564 017737 157176 001126  MOV      @YPOS,$BDDAT      ;READ CHARACTER REG
2526 022572 042737 001777 001126  BIC      #1777,$BDDAT      ;MASK TO BITS 10-15
2527 022600 023737 001124 001126  CMP      $GDDAT,$BDDAT     ;COMPARE EXPCT TO RCVD
2528 022606 001405      BEQ      1$
2529 022610 013737 001766 001122  MOV      YPOS,$BDADR       ;SET UP REG ADRS 06
2530 022616 104021      ERROR   21                 ;CHARACTER REGISTER IN ERROR
2531 022620 000477      BR       TST101            ;;
2532
2533 022622 017737 157134 001126 1$:  MOV      @SREG0,$BDDAT     ;READ STATUS REGISTER
2534 022630 012737 000100 001124  MOV      #BIT6,$GDDAT      ;LOAD EXPECTED
2535 022636 032737 000100 001126  BIT      #100,$BDDAT
2536 022644 001005      BNE      2$
2537 022646 013737 001762 001122  MOV      SREG0,$BDADR      ;SET UP REG ADRS 02
2538 022654 104036      ERROR   36                 ;SHIFT OUT STATUS BIT FAILED TO SET
2539 022656 000460      BR       TST101            ;;
2540
2541 022660 017737 157100 001126 2$:  MOV      @XPOS,$BDDAT      ;READ X POS
2542 022666 012737 001000 001124  MOV      #1000,$GDDAT      ;LOAD EXPECTED
2543 022674 023737 001124 001126  CMP      $GDDAT,$BDDAT     ;COMPARE EXPCT TO RCVD
2544 022702 001405      BEQ      3$
2545 022704 013737 001764 001122  MOV      XPOS,$BDADR       ;SET UP REG ADRS 04
2546 022712 104022      ERROR   22                 ;SHIFT-OUT CHARACTER CHANGED X AXIS
2547 022714 000441      BR       TST101            ;;
2548
2549 022716 017737 157044 001126 3$:  MOV      @YPOS,$BDDAT      ;READ Y POS
2550 022724 042737 176000 001126  BIC      #176000,$BDDAT     ;MASK
2551 022732 012737 001000 001124  MOV      #1000,$GDDAT      ;LOAD EXPECTED
2552 022740 023737 001124 001126  CMP      $GDDAT,$BDDAT     ;COMPARE EXPCT TO RCVD
2553 022746 001404      BEQ      4$                ;;BR IF EQUAL
2554 022750 013737 001766 001122  MOV      YPOS,$BDADR       ;SET UP REG ADRS 06
2555 022756 104022      ERROR   22                 ;SHIFT-OUT CHARACTER CHANGED Y AXIS
2556
2557 022760      4$:  JSR      PC,EXECUTE        ;START DISPLAY AND WAIT FOR STOP FLAG
(1) 022760 004737 024524      MOV      @SREG0,$BDDAT     ;READ STATUS
2558 022764 017737 156772 001126  CLR      $GDDAT            ;CLEAR EXPECTED
2559 022772 005037 001124      BIT      #BIT6,$BDDAT      ;TEST IF SHIFT-OUT SET
2560 022776 032737 000100 001126  BEQ      5$                ;;BR IF ZERO
2561 023004 001404      MOV      SREG0,$BDADR      ;SET UP REG ADRS 02
2562 023006 013737 001762 001122  ERROR   35                 ;SHIFT-OUT STATUS BIT FAILED TO CLEAR
2563 023014 104035      5$:  RESET
2564 023016 000005
2565
2566      ;*****
(3)      ;*TEST 101 TEST THAT 'SHIFT-OUT' DOES NOT GENERATE A STATUS BIT ON CODE 0-37
(3)      ;*****
(2) 023020 000004      TST101: SCOPE
(1) 023022 012737 000100 001166      MOV      #100,$TIMES      ;;DO 100 ITERATIONS
(2) 023030 012737 000101 001202      MOV      #101,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
2567      ;('SHIFT-OUT' FOLLOWED BY CODE 0 THRU 37 EXCEPT #17)
2568
    
```

```

2569 023036 012737 000016 001726      MOV      #16,DSAVE
2570 023044 012700 032530      MOV      #BUFFER,R0      ;LOAD POINTER
2571 023050 012720 100000      MOV      #100000,(R0)+   ;SET 'CHAR' MODE
2572 023054 012720 000016      MOV      #16,(R0)+      ;LOAD 'SHIFT-OUT' INTO THE LOW BYTE
2573 023060 012720 172000      MOV      #172000,(R0)+  ;LOAD STOP
2574 023064 013777 001726 156622 1$:  MOV      DSAVE,@DBUF1   ;LOAD DISPLAY BUFFER
2575 023072 004737 024524      JSR      PC,EXECUTE      ;START DISPLAY AND WAIT FOR STOP FLAG
2576
2577 023076 005037 001124      CLR      $GDDAT
2578 023102 017737 156654 001126      MOV      @SREG0,$BDDAT  ;READ STATUS
2579 023110 032737 000100 001126      BIT      #100,$BDDAT    ;TEST FOR SHIFT BIT
2580 023116 001405      BEQ      2$
2581 023120 013737 001762 001122      MOV      SREG0,$BADDR   ;SET UP REG ADRS 02
2582 023126 104035      ERROR    35             ;SHIFT STATUS BIT SET IN ERROR
2583 023130 000412      BR       TST102         ;;
2584
2585 023132 105237 001727 2$:  INCB     DSAVE+1
2586 023136 122737 000017 001727      CMPB     #17,DSAVE+1    ;TEST FOR #17 (SHIFT-IN)
2587 023144 001772      BEQ      2$             ;;BR AND TRY AGAIN
2588 023146 122737 000040 001727      CMPB     #40,DSAVE+1
2589 023154 001343      BNE     1$             ;;BR BACK IF NOT #40
2590
2591
(3)
(3)
(2) 023156 000004
(2) 023160 012737 000102 001202
2592
2593
2594 023166 012737 000340 177776      MOV      #340,PSW
2595 023174 012777 100000 156510      MOV      #100000,@DBUF  ;LOAD SET 'CHAR' MODE
2596 023202 012777 007000 156504      MOV      #7000,@DBUF1  ;LOAD 'SHIFT-OUT' INTO THE HIGH BYTE
2597 023210 012777 000040 156500      MOV      #40,@DBUF2    ;LOAD A SHIFT-OUT CHARACTER IN THE NEXT
2598                                     ;WORD <LOW BYTE>
2599 023216 012777 172000 156474      MOV      #172000,@DBUF3 ;LOAD STOP
2600 023224 012737 000001 024632      MOV      #1,NOFLAG     ;SET NO FLAG EXIT
2601 023232 004737 024524      JSR      PC,EXECUTE      ;START DISPLAY AND WAIT FOR STOP FLAG
2602
2603 023236 012737 000100 001124      MOV      #BIT6,$GDDAT   ;LOAD EXPECTED
2604 023244 017737 156512 001126      MOV      @SREG0,$BDDAT  ;READ STATUS
2605 023252 032737 000100 001126      BIT      #100,$BDDAT    ;TEST THE STATUS REGISTER
2606 023260 001005      BNE     1$
2607 023262 013737 001762 001122      MOV      SREG0,$BADDR   ;SET UP REG ADRS 02
2608 023270 104036      ERROR    36             ;SHIFT-OUT IN THE HIGH BYTE FAILED TO SET STATUS BIT
2609 023272 000421      BR       TST103         ;;
2610
2611 023274 017737 156466 001126 1$:  MOV      @YPOS,$BDDAT   ;READ Y POS
2612 023302 042737 001777 001126      BIC      #1777,$BDDAT   ;MASK TO BITS 15-10
2613 023310 012737 100000 001124      MOV      #BIT15,$GDDAT  ;LOAD EXPECTED
2614 023316 023737 001124 001126      CMP      $GDDAT,$BDDAT  ;COMPARE EXPCT TO RCVD
2615 023324 001404      BEQ     TST103         ;;BR IF EQUAL
2616 023326 013737 001766 001122      MOV      YPOS,$BADDR    ;SET UP REG ADRS 06
2617 023334 104021      ERROR    21             ;CHARACTER REGISTER IN ERROR AFTER A
2618                                     ;'SHIFT-OUT' <HIGH BYTE> FOLLOWED BY
2619                                     ; #40 <LOW BYTE NEXT WORD>
2620

```

```

2621
2622
(3)
(3)
(2) 023336 000004
(1) 023340 012737 000040 001166
(2) 023346 012737 000103 001202
2623 023354 012737 000340 177776
2624 023362 013737 001762 001122
2625 023370 012737 100000 032530
2626 023376 012737 007000 032532
2627 023404 012737 000040 032534
2628 023412 012737 172000 032536
2629 023420 012737 000001 024632
2630 023426 004737 024524
2631 023432 000005
2632 023434 005037 001124
2633 023440 017737 156316 001126
2634 023446 042737 177677 001126
2635 023454 001404
2636 023456 013737 001762 001122
2637 023464 104032
2638
2639
(3)
(3)
(2) 023466 000004
(2) 023470 012737 000104 001202
2640 023476 012737 000340 177776
2641 023504 012777 023574 156306
2642 023512 012737 173400 032530
2643 023520 013777 001712 156232
2644 023526 013737 001762 001122
2645 023534 012737 172000 001124
2646 023542 013700 001736
2647 023546 162700 000040
2648 023552 010037 177776
2649 023556 000240
2650 023560 000240
2651 023562 017737 156174 001126
2652 023570 104034
2653 023572 000406
2654
2655 023574 012716 023602 1$: MOV #2$, (SP) ; OK, FIX STACKED PC...
2656 023600 000002 RTI ; ...AND DISMISS INTERRUPT.
2657 023602 013777 002022 156210 2$: MOV DDONE1, @DDONE ; RESET VECTOR
2658
2659
(3)
(3)
(2) 023610 000004
(2) 023612 012737 000105 001202
2660 023620 013737 001762 001122
2661 023626 012737 172000 001124
2662 023634 012737 000340 177776
2663 023642 012777 023704 156150

```

```

:*****
:*TEST 103 TEST THAT RESET WILL CLEAR 'SHIFT-OUT' STATUS BIT
:*****
TST103: SCOPE

```

```

MOV #40, $TIMES ;:DO 40 ITERATIONS
MOV #103, $TESTN ;:SET TEST NUMBER IN APT MAIL BOX
MOV #340, $PSW ;:RAISE PRIORITY
MOV $REG0, $BDADR ;:SET UP REG 02 ADRS
MOV #100000, $BUFFER ;:LD CHAR INSTR
MOV #7000, $BUFFER+2 ;:SET UP FOR SHIFT OUT
MOV #40, $BUFFER+4 ;:SET UP SHIFT OUT CHAR
MOV #172000, $BUFFER+6 ;:LD STOP INSTR
MOV #1, $NOFLAG ;:SET NO FLAG EXIT
JSR PC, EXECUTE ;:START DISPLAY
RESET ;:CLR SHIFT OUT
CLR $GDDAT ;:EXPECT 0
MOV @SREG0, $BDDAT ;:READ REG 02
BIC #177677, $BDDAT ;:SAVE ONLY SHIFT-OUT
BEQ TST104 ;:NEXT TEST IF CLEARED
MOV $REG0, $BDADR ;:SET UP REG ADRS 02
ERROR 32 ;:RESET FAILED TO CLR SHIFT-OUT

```

```

:*****
:*TEST 104 INTERRUPT PRIORITY TEST (DEVICE LEVEL -1)
:*****
TST104: SCOPE

```

```

MOV #104, $TESTN ;:SET TEST NUMBER IN APT MAIL BOX
MOV #340, $PSW ;:RAISE CPU STAT.
MOV #1$, @DDONE ;:SET INTERRUPT ADDRESS.
MOV #173400, $BUFFER ;:SET 'STOP AND INTERRUPT' INSTR
MOV $BUF, @DPC ;:LOAD THE DISPLAY P.C.
MOV $REG0, $BDADR ;:SET UP REG ADRS 02
MOV #172000, $GDDAT ;:EXPECT STOP, MODE & INTENSITY BITS
MOV BRLEV1, R0 ;:GET BUS PRIORITY.
SUB #40, R0 ;:MAKE ONE LOWER
MOV R0, $PSW ;:LOWER PRIORITY TO DEVICE LEVEL -1
NOP ;:*** HERE IF NO INTERRUPT ***
NOP
MOV @SREG0, $BDDAT ;:GET REG 02
ERROR 34 ;:NO STOP INTERRUPT ON BR LEVEL INDICATED -1
BR TST105 ;:BR TO NEXT TEST

```

```

:*****
:*TEST 105 INTERRUPT PRIORITY TEST (DEVICE LEVEL)
:*****
TST105: SCOPE

```

```

MOV #105, $TESTN ;:SET TEST NUMBER IN APT MAIL BOX
MOV $REG0, $BDADR ;:SET UP REG ADRS 02
MOV #172000, $GDDAT ;:EXPECT STOP, MODE & INTENSITY BITS
MOV #340, $PSW ;:RAISE CPU STAT.
MOV #1$, @DDONE ;:SET INTERRUPT ADDRESS.

```

```

2664 023650 012737 173400 032530      MOV      #173400,BUFFER ; SET 'STOP AND INTERRUPT' INSTR
2665 023656 013777 001712 156074      MOV      DBUF,@DPC
2666 023664 000240                      NOP
2667 023666 000240                      NOP
2668 023670 013737 001736 177776      MOV      BRLEV1,PSW      ;LOWER PRIORITY TO DEVICE LEVEL
2669 023676 000240                      NOP      ; *** SHOULD NOT INTERRUPT ***
2670 023700 000240                      NOP
2671 023702 000410                      BR       3$              ; IT DIDN'T, CONTINUE.
2672
2673 023704 012716 023712          1$:    MOV      #2$(,SP)      ; IT DID, FIX PC...
2674 023710 000002                      RTI      ;...DISMISS...
2675 023712 017737 156044 001126  2$:    MOV      @SREG0,$BDDAT ;GIVE THEM REG 02
2676 023720 104034                      ERROR   34              ;VS60 INTERRUPTED ON THE WRONG BR LEVEL
2677 023722 000425                      BR       TST106        ;:NEXT TEST
2678
2679 023724 012777 023754 156066  3$:    MOV      #4$,@DDONE    ; CHANGE INTERRUPT ADDRESS.
2680 023732 005037 177776                      CLR      PSW           ;LOWER PRIORITY LEVEL
2681 023736 000240                      NOP      ; *** SHOULD INTERRUPT ***
2682 023740 000240                      NOP
2683 023742 017737 156014 001126      MOV      @SREG0,$BDDAT ; DIDN'T, GIVE THEM REG 02
2684 023750 104034                      ERROR   34              ;VS60 STOP FAILED TO INTERRUPT AFTER LOWERING PRIORITY
2685 023752 000411                      BR       TST106        ;:NEXT TEST
2686
2687 023754 012716 023762          4$:    MOV      #5$(,SP)      ; OK, FIX PC...
2688 023760 000002                      RTI      ;...DISMISS...
2689 023762 012737 000340 177776  5$:    MOV      #340,PSW     ;RAISE TO HIGHEST PRIORITY BEFORE ADVANCING
2690 023770 013777 002022 156022      MOV      DDONE1,@DDONE ; AND RESET DONE VECTOR.
2691
2692
2693 (3)
2694 (3)
2695 (2) 023776 000004
2696 (1) 024000 012737 000100 001166
2697 (2) 024006 012737 000106 001202
2698 024014 013737 001766 001122
2699 024022 005077 155764
2700 024026 012737 100000 032530
2701 024034 012737 077577 032532
2702 024042 012737 172000 032534
2703 024050 004737 024524
2704
2705 RESET
2706 CLR      $GDDAT      ;CLEAR EXPECTED
2707 MOV      @YPOS,$BDDAT ;READ CHAR REG
2708 BIC      #1777,$BDDAT
2709 BEQ      TST107      ;:BR IF CLEARED
2710 ERROR   32          ;RESET 'INIT' FAILED TO CLEAR CHARACTER REGISTER
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000

```

```

2710 024126 005037 001124 CLR $GDDAT ;N/A
2711 024132 005037 001126 CLR $BDDAT ;N/A
2712 024136 012700 032530 MOV #BUFFER,R0 ;SET UP ADRS FOR INSTRS
2713 024142 012720 170000 MOV #170000,(R0)+ ;LD STATUS A (SYNC INSTR)
2714 024146 012720 172000 MOV #172000,(R0)+ ;LD STOP INSTR
2715 024152 012720 170000 MOV #170000,(R0)+ ;LD ANOTHER LD STATUS A INSTR
2716 024156 012720 172000 MOV #172000,(R0)+ ;STOP INSTR
2717 024162 012702 024346 MOV #TMSAV,R2 ;SET UP ADRS FOR SYNC COUNTS
2718 024166 005000 1$: CLR R0 ;CLR SYNC TIMER
2719 024170 012777 032530 155562 MOV #BUFFER,@DPC ;START
2720 024176 005200 2$: INC R0 ;COUNT SYNC TIMER
2721 024200 001413 4$: BEQ 4$ ;SHOULD NEVER OVFL0 - NO SYNC TMG
2722 024202 005777 155554 TST @SREG0 ;DONE SET?
2723 024206 100373 2$: BPL 2$ ;BR IF NOT
2724 024210 005000 CLR R0 ;CLR TIMER
2725 024212 005277 155542 INC @DPC ;RESUME - NOW NPR REQ IN SYNC WITH INTERNAL OSC
2726 024216 005200 3$: INC R0 ;COUNT TIMER
2727 024220 001403 4$: BEQ 4$ ;SHOULD NEVER OVFL0 - NO SYNC TIMING
2728 024222 005777 155534 TST @SREG0 ;HAS STOP BEEN PICKED UP YET?
2729 024226 100373 3$: BPL 3$ ;BR IF NOT
2730 024230 010022 4$: MOV R0,(R2)+ ;SAVE TIMER RESULTS
2731 024232 062737 000004 032534 ADD #4,BUFFER+4 ;SET UP NEXT SYNC
2732 024240 062737 000004 032530 ADD #4,BUFFER ;SET UP NEXT SYNC
2733 024246 022737 170020 032530 CMP #170020,BUFFER ;ALL SYNCs TESTED?
2734 024254 001344 1$: BNE 1$ ;BR IF NOT
2735 024256 013701 024352 5$: MOV TMSAV+4,R1 ;40 FPS CNT TO R1
2736 024262 163701 024346 SUB TMSAV,R1 ;SUB FULL SPEED CNT
2737 024266 162701 000020 SUB #20,R1 ;MAKE SURE DIFF IS GREATER THAN 20
2738 024272 100001 6$: BPL 6$ ;BR IF 40 FPS SLOWER THAN FULL SPEED
2739 024274 104040 40 ERROR 40 ;40 FPS SYNC ER
2740 024276 013701 024350 6$: MOV TMSAV+2,R1 ;30 FPS CNT TO R1
2741 024302 163701 024352 SUB TMSAV+4,R1 ;SUB 40 FPS CNT
2742 024306 162701 000020 SUB #20,R1 ;MAKE SURE DIFF IS GREATER THAN 20
2743 024312 100001 7$: BPL 7$ ;BR IF 30 FPS SLOWER THAN 40 FPS
2744 024314 104040 40 ERROR 40 ;30 FPS SYNC ERROR
2745 024316 005737 024356 7$: TST EXSYNC ;START 210 SAYS LOOK FOR EXT SYNC TMG
2746 024322 001404 8$: BEQ 8$ ;BR IF NOT
2747 024324 005737 024354 TST TMSAV+6 ;LOOK FOR NON-ZERO TMG CNT
2748 024330 001013 BNE TST110 ;NEXT TEST IF TIMING THERE
2749 024332 104040 40 ERROR 40 ;EXT SYNC FAILED TO TRIGGER DISPLAY
2750 024334 005737 024354 8$: TST TMSAV+6 ;LOOK FOR ZERO TMG CNT
2751 024340 001407 BEQ TST110 ;NEXT TEST IF TIMING NOT FOUND
2752 024342 104040 40 ERROR 40 ;EXT SYNC TIMING FOUND WHEN NOT EXPECTED
2753 024344 000405 BR TST110 ;LOOK FOR LOOP ON TEST SW
2754
2755 024346 000000 TMSAV: 0 ;FULL SPEED CNT
2756 024350 000000 0 ;30 FPS CNT
2757 024352 000000 0 ;40 FPS CNT
2758 024354 000000 0 ;EXTERNAL CNT
2759
2760 024356 000000 EXSYNC: 0 ;ZERO(DON'T EXPECT EXT TMG), NON-ZERO(EXPECT EXT TMG)
2761
2762
(3) ;*****
(3) ;*TEST 110 TEST FOR HALT AT END OF DIAGNOSTIC (SW12 SET)
(2) 024360 000004 ;*****
TST110: SCOPE
    
```

(2)	024362	012737	000110	001202	MOV	#110,\$TESTN	::SET TEST NUMBER IN APT MAIL BOX
2763	024370	032777	010000	154542	BIT	#SW12,@SWR	: IS SW12 SET ?
2764	024376	001401			BEG	\$EOP	:BR IF NOT
2765	024400	000000			HALT		:COMPLETED PASS - SW12 SAYS HALT

2767  
 (1)  
 (2)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1) 024402  
 (1) 024402 000004  
 (1) 024404 005037 001102  
 (1) 024410 005037 001166  
 (1) 024414 005237 001204  
 (1) 024420 042737 100000 001204  
 (1) 024426 005327  
 (1) 024430 000001  
 (1) 024432 003022  
 (1) 024434 012737  
 (1) 024436 000001  
 (1) 024440 024430  
 (1) 024442 104401 024507  
 (2) 024446 013746 001204  
 (2) 024452 104405  
 (1) 024454 104401 024504  
 (1) 024460 013700 000042  
 (1) 024464 001405  
 (1) 024466 000005  
 (1) 024470 004710  
 (1) 024472 000240  
 (1) 024474 000240  
 (1) 024476 000240  
 (1) 024500  
 (1) 024500 000137  
 (1) 024502 002460  
 (1) 024504 377 377 000  
 (1) 024507 015 042412 042116

```

.SBTTL END OF PASS ROUTINE

:*****
:*INCREMENT THE PASS NUMBER ($PASS)
:*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
:*TYPE 'END PASS #XXXXX' (WHERE XXXXX IS A DECIMAL NUMBER)
:*IF THERES A MONITOR GO TO IT
:*IF THERE ISN'T JUMP TO RESTRT

$EOP:
  SCOPE
  CLR $TSTNM ;;ZERO THE TEST NUMBER
  CLR $TIMES ;;ZERO THE NUMBER OF ITERATIONS
  INC $PASS ;;INCREMENT THE PASS NUMBER
  BIC #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER
  DEC (PC)+ ;;LOOP?
$EOPCT: .WORD 1
  BGT $DOAGN ;;YES
  MOV (PC)+,@(PC)+ ;;RESTORE COUNTER
$ENDCT: .WORD 1
  TYPE $EOPCT
  MOV $SENDMG ;;TYPE 'END PASS #'
  TYPDS $PASS,-(SP) ;;SAVE $PASS FOR TYPEOUT
  TYPE $ENULL ;;GO TYPE--DECIMAL ASCII WITH SIGN
  MOV @#42,R0 ;;TYPE A NULL CHARACTER
  BEQ $DOAGN ;;GET MONITOR ADDRESS
  RESET ;;BRANCH IF NO MONITOR
  JSR PC,(R0) ;;CLEAR THE WORLD
  NOP ;;GO TO MONITOR
  NOP ;;SAVE ROOM
  NOP ;;FOR
  NOP ;;ACT11
$DOAGN:
  JMP @(PC)+ ;;RETURN
$RTNAD: .WORD RESTRT
$ENULL: .BYTE -1,-1,0 ;;NULL CHARACTER STRING
$SENDMG: .ASCIZ <15><12>/END PASS #/

```

```
2769      ;:*****  
2770      ;SUBROUTINE TO START DISPLAY AND WAIT FOR STOP FLAG  
2771      ;:*****  
2772 024524 EXECUTE:  
2773 024524 032737 000001 001204      BIT      #BIT0,$PASS      ;ODD # PASS?  
2774 024532 001403                      BEQ      EXEC          ;BR IF NOT  
2775 024534 052777 100000 155250      BIS      #BIT15,@STKPT ;RUN IN SERIAL MODE ON ODD # PASSES  
2776 024542 012777 032530 155210 EXEC:  MOV      #BUFFER,@DPC      ;START DISPLAY  
2777 024550 000403                      BR EXECRS  
2778  
2779      ;:*****  
2780      ;SUBROUTINE TO CONTINUE DISPLAY AND WAIT FOR STOP FLAG  
2781      ;:*****  
2782  
2783 024552 012777 000001 155200 CONT:  MOV      #1,@DPC          ;RESUME DISPLAY  
2784 024560 010046 EXECRS: MOV      R0,-(SP)      ;SAVE R0  
2785 024562 012700 000200              MOV      #BIT7,R0      ;LOAD DELAY  
2786 024566 005777 155170 1$:      TST      @SREG0      ;WAIT FOR STOP FLAG  
2787 024572 100410                      BMI      2$          ;BR IF SET  
2788 024574 005300                      DEC      R0          ;DELAY  
2789 024576 001373                      BNE      1$          ;BR IF NOT DONE  
2790 024600 005737 024632              TST      NOFLAG      ;VALID NO STOP FLAG CONDITION ?  
2791 024604 001003                      BNE      2$          ;BR IF YES  
2792 024606 012600                      MOV      (SP)+,R0     ;RESTORE R0 IN CASE OF LOOP ON ER SW  
2793 024610 104037                      ERROR    37          ;STOP FLAG FAILED TO SET  
2794 024612 000401                      BR       3$          ;R0 ALREADY RESTORED  
2795 024614 012600 2$:      MOV      (SP)+,R0     ;RESTORE R0  
2796 024616 005037 024632 3$:      CLR      NOFLAG      ;ENSURE RESET FLAG  
2797 024622 042777 100000 155162      BIC      #BIT15,@STKPT ;RETURN WITH MAINT SW 4 OFF  
2798 024630 000207                      RTS      PC          ;EXIT  
2799  
2800 024632 000000 NOFLAG: 0  
2801  
2802      ;:*****  
2803      ;THIS CODE RESETS ALL VS60 VECTORS WITH HALTS  
2804      ;:*****  
2805 024634 012737 000340 177776 RSTVEC: MOV      #340,PSW      ;RESET PRIORITY TO HIGHEST LEVEL  
2806 024642 012701 000004              MOV      #4,R1        ;SET UP VECTOR LOCATION COUNT  
2807 024646 013700 001246              MOV      $VECT1,R0    ;GET 1ST VS60 VECTOR ADRS  
2808 024652 042700 160000              BIC      #160000,R0   ;STRIP PSW BITS.  
2809 024656 010010 1$:      MOV      R0,(R0)      ;SET UP ADRS TO HALT  
2810 024660 062720 000002              ADD      #2,(R0)+     ;POINT IT TO HALT  
2811 024664 005020                      (LR     (R0)+        ;SET UP HALT  
2812 024666 005301                      DEC      R1          ;COUNT THIS VECTOR RESTORE  
2813 024670 001372                      BNE      1$          ;BR IF MORE TO RESTORE  
2814 024672 000207                      RTS      PC          ;RETURN IF ALL DONE  
2815  
2816      ;:*****  
2817      ;ROUTINE TO MULTIPLY AN X OR Y POINT BY A SCALE FACTOR.  
2818      ;SAVE SCALED POINT VALUE IN $GDDAT.  
2819      ;*** TWO VERSION SUPPLIED ***  
2820      ;*** FOR BEFORE AND AFTER SCALING ECO. ***  
2821      ;*** SWR BIT 10 = 1 IF ECO IS INSTALLED ***  
2822      ;*** G.P. MAY '9 ***  
2823      ;:*****  
2824 024674 010146 MULSCL: MOV      R1,-(SP)      ;SAVE SCALE ON STACK
```

```

2825 024676 010046      MOV      R0,-(SP)      ;SAVE POINT ON STACK
2826 024700 001001      BNE      1$
2827 024702 005001      CLR      R1           ; SET EXP POINT = 0...
2828 024704 005701      1$:     TST      R1
2829 024706 001461      BEQ      11$         ;...AND EXIT IF EITHER IS 0.
2830
2831 024710 032777 002000 154222      BIT      #SW10,@SWR
2832 024716 001411      BEQ      2$           ; BR IF SCALING ECO NOT IN.
2833
2834      ; USE THE FOLLOWING AFTER SCALING ECO (#???) IS INSTALLED.
2835      ; SCALE VALUE IS X 2^2 (I.E. 3 = 3/4, 4 = 1, ETC).
2836
2837 024720 010046      MOV      R0,-(SP)      ; POINT...
2838 024722 010146      MOV      R1,-(SP)      ;...TIMES SCALE [2^2]...
2839 024724 004737 027504      JSR      PC,$MULT
2840 024730 012601      MOV      (SP)+,R1      ;... TO R1 [2^2].
2841 024732 005726      TST      (SP)+        ; DISCARD HI HALF OF PRODUCT.
2842 024734 006201      ASR      R1           ; RESCALE RESULT.
2843 024736 006201      ASR      R1
2844 024740 000444      BR       11$         ; AND EXIT.
2845
2846      ; USE THE FOLLOWING BEFORE SCALING ECO.
2847
2848 024742 005003      2$:     CLR      R3           ;CLR UNITY MULTIPLIER
2849 024744 010002      MOV      R0,R2        ;PUT POINT IN R2 ALSO
2850 024746 005004      CLR      R4           ;CLR LSB ROUNDING INDICATOR
2851 024750 162701 000004      3$:     SUB      #4,R1      ;SUBTRACT ALL UNITY TIMES
2852 024754 002405      BLT      4$           ;BR IF NOW TO FRACTIONAL PART
2853 024756 005203      INC      R3           ;RECORD IN UNITY MULTIPLIER
2854 024760 005701      TST      R1           ;DOES SCALE HAVE FRACTIONAL PART?
2855 024762 001372      BNE      3$          ;BR IF FRACTION EXISTS
2856 024764 005000      CLR      R0           ;NO FRACTIONAL PART EXISTS
2857 024766 000421      BR       7$          ;GO MUL POINT BY SCALE
2858 024770 006200      4$:     ASR      R0           ;DIVIDE BY TWO
2859 024772 103001      BCC      5$          ;BR IF NO CARRY
2860 024774 005204      INC      R4           ;SAVE LSB SHIFT OUT FOR ROUNDING AT 1/4 OR 3/4
2861 024776 062701 000002      5$:     ADD      #2,R1      ;COUNT DIVISION BY TWO
2862 025002 001413      BEQ      7$          ;GO MUL POINT BY SCALE IF FRACT PART IS 1/2
2863 025004 003006      BGT      6$          ;BR IF FRACT PART IS 3/4
2864 025006 006200      ASR      R0           ;ADJUST FRACTIONAL PART TO 1/4
2865 025010 103010      BCC      7$          ;GO MUL POINT BY SCALE IF NO LSB CARRY OUT
2866 025012 005704      TST      R4           ;WAS THER A PREVIOUS CARRY OUT?
2867 025014 001406      BEQ      7$          ;BR IF NOT - GO MUL POINT BY SCALE
2868 025016 005200      INC      R0           ;IF SO - TWO LSB CARRIES ROUNDS UP TO 1
2869 025020 000404      BR       7$          ;GO MUL POINT BY SCALE
2870 025022 060400      6$:     ADD      R4,R0      ;ALWAYS ROUND UP IF LSB CARRY ON 1ST SHIFT ON 3/4 FRAC
2871 025024 006202      ASR      R2           ;MAKE UP 1/4 FRACTION IN R2
2872 025026 006202      ASR      R2
2873 025030 060200      ADD      R2,R0        ;ADD 1/4 TO 1/2 FOR 3/4'S
2874 025032 011601      7$:     MOV      (SP),R1     ;GET POINT VALUE TO R1
2875 025034 005303      8$:     DEC      R3           ;NOW COUNT UNITY TIMES
2876 025036 001404      BEQ      10$         ;BR IF ALL WHOLE UNITS ADDED IN
2877 025040 100402      BMI      9$          ;BR IF ONLY FRACTIONAL PART EXISTS
2878 025042 061601      ADD      (SP),R1     ;ADD IN ANOTHER UNITY
2879 025044 000773      BR       8$          ;GO LOOK FOR ANOTHER TIMES
2880 025046 005001      9$:     CLR      R1           ;THERE IS ONLY A FRACTIONAL PART

```

```

2881 025050 060001      10$:  ADD    R0,R1          ;ADD IN FRACTIONAL PART
2882
2883                    ;: COMMON EXIT FROM EITHER SCALING FUNCTION ABOVE.
2884
2885 025052 010137 001124 11$:  MOV    R1,$GDDAT       ;SAVE COMPUTED POINT.
2886 025056 012600          MOV    (SP)+,R0         ;RESTORE R0
2887 025060 012601          MOV    (SP)+,R1         ;RESTORE R1
2888 025062 000207          RTS     PC              ;EXIT
2889
2890                    ;:*****
2891                    ;: THIS ROUTINE ADDS AN X OR Y POINT TO THE RESPECTIVE X OR
2892                    ;: Y DYNAMIC OFFSET AND LEAVES THE 14 BIT RESULT IN $GDDAT.
2893                    ;: R0 CONTAINS THE POINT DATA & R1 CONTAINS THE OFFSET DATA
2894                    ;: WHEN ENTERING THIS ROUTINE.
2895                    ;:*****
2896
2897 025064 032700 020000  CALPT: BIT    #BIT13,R0     ;POINT NEG?
2898 025070 001403          BEQ    POS1            ;BR IF NOT
2899 025072 042700 020000  BIC    #BIT13,R0     ;CLR SIGN BIT
2900 025076 005400          NEG    R0             ;MAKE NEG
2901 025100 032701 020000  POS1:  BIT    #BIT13,R1     ;OFFSET NEG?
2902 025104 001404          BEQ    POS2            ;BR IF NOT
2903 025106 042701 030000  BIC    #BIT13!BIT12,R1 ;CLR SIGN & OFFSET BITS
2904 025112 005401          NEG    R1             ;MAKE NEG
2905 025114 000402          BR     OVER1          ;SKIP OVER NEXT INSTR
2906 025116 042701 010000  POS2:  BIC    #BIT12,R1     ;DONT WANT OFFSET BIT
2907 025122 060100  OVER1:  ADD    R1,R0         ;MAKE UP CALCULATED POINT VALUE
2908 025124 042700 140000  BIC    #140000,R0     ;LIMIT TO 14 BITS
2909 025130 010037 001124  MOV    R0,$GDDAT     ;SAVE POINT RESULT IN $GDDAT
2910 025134 000207          RTS     PC              ;RETURN
2911
2912                    ;:*****
2913                    ;: THIS ROUTINE MAKES UP A 14 BIT X OR Y POINT LENGTH WHICH IS LEFT IN $BDDAT
2914                    ;:*****
2915 025136 042737 176000 001126  POCONV: BIC    #176000,$BDDAT ;SAVE ONLY X OR Y POSITION BITS
2916 025144 042705 007777          BIC    #7777,R5       ;SAVE ONLY X OR Y HI ORDER POSITION BITS
2917 025150 000241          CLC                    ;CLR CARRY BEFORE ROTATE
2918 025152 006005          ROR    R5              ;ROTATE RIGHT 2 PLACES
2919 025154 006005          ROR    R5              ;
2920 025156 060537 001126  ADD    R5,$BDDAT     ;MAKE UP 14 BIT X OR Y POINT LENGTH
2921 025162 000207          RTS     PC              ;EXIT
    
```

```

2923      .SBTTL  SCOPE HANDLER ROUTINE
(1)
(2)      ;*****
(1)      ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
(1)      ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
(1)      ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
(1)      ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
(1)      ;*SW14=1      LOOP ON TEST
(1)      ;*SW11=1      INHIBIT ITERATIONS
(1)      ;*SW09=1      LOOP ON ERROR
(1)      ;*SW08=1      LOOP ON TEST IN SWR<7:0>
(1)      ;*CALL
(1)      ;*      SCOPE      ;;SCOPE=IOT
(1)
(1)      $SCOPE:
(1)      025164 032777 040000 153746 1$: BIT      #BIT14,@SWR      ;;LOOP ON PRESENT TEST?
(1)      025172 001114      BNE      $OVER      ;;YES IF SW14=1
(1)      ;*****START OF CODE FOR THE XOR TESTER*****
(1)      025174 000416      $XTSTR: BR      6$      ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
(1)      ;THIS INSTRUCTION TO A 'NOP' (NOP=240)
(1)      025176 013746 000004      MOV      @ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
(1)      025202 012737 025222 000004      MOV      #5$,@ERRVEC  ;;SET FOR TIMEOUT
(1)      025210 005737 177060      TST      @177060      ;;TIME OUT ON XOR?
(1)      025214 012637 000004      MOV      (SP)+,@ERRVEC  ;;RESTORE THE ERROR VECTOR
(1)      025220 000463      BR      $SVLAD      ;;GO TO THE NEXT TEST
(1)      025222 022626      5$: CMP      (SP)+,(SP)+  ;;CLEAR THE STACK AFTER A TIME OUT
(1)      025224 012637 000004      MOV      (SP)+,@ERRVEC  ;;RESTORE THE ERROR VECTOR
(1)      025230 000423      BR      7$      ;;LOOP ON THE PRESENT TEST
(1)      025232      6$:;*****END OF CODE FOR THE XOR TESTER*****
(1)      025232 032777 000400 153700      BIT      #BIT08,@SWR  ;;LOOP ON SPEC. TEST?
(1)      025240 001404      BEQ      2$      ;;BR IF NO
(1)      025242 127737 153672 001102      CMPB    @SWR,$TSTNM  ;;ON THE RIGHT TEST? SWR<7:0>
(1)      025250 001465      BEQ      $OVER      ;;BR IF YES
(1)      025252 105737 001103      2$: TSTB    $ERFLG      ;;HAS AN ERROR OCCURRED?
(1)      025256 001421      BEQ      3$      ;;BR IF NO
(1)      025260 123737 001115 001103      CMPB    $ERMAX,$ERFLG  ;;MAX. ERRORS FOR THIS TEST OCCURRED?
(1)      025266 101015      BHI      3$      ;;BR IF NO
(1)      025270 032777 001000 153642      BIT      #BIT09,@SWR  ;;LOOP ON ERROR?
(1)      025276 001404      BEQ      4$      ;;BR IF NO
(1)      025300 013737 001110 001106      7$: MOV      $LPERR,$LPADR  ;;SET LOOP ADDRESS TO LAST SCOPE
(1)      025306 000446      BR      $OVER
(1)      025310 105037 001103      4$: CLRB    $ERFLG      ;;ZERO THE ERROR FLAG
(1)      025314 005037 001166      CLR      $TIMES      ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
(1)      025320 000415      BR      1$      ;;ESCAPE TO THE NEXT TEST
(1)      025322 032777 004000 153610      3$: BIT      #BIT11,@SWR  ;;INHIBIT ITERATIONS?
(1)      025330 001011      BNE      1$      ;;BR IF YES
(1)      025332 005737 001204      TST      $PASS      ;;IF FIRST PASS OF PROGRAM
(1)      025336 001406      BEQ      1$      ;;INHIBIT ITERATIONS
(1)      025340 005237 001104      INC      $ICNT      ;;INCREMENT ITERATION COUNT
(1)      025344 023737 001166 001104      CMP      $TIMES,$ICNT  ;;CHECK THE NUMBER OF ITERATIONS MADE
(1)      025352 002024      BGE      $OVER      ;;BR IF MORE ITERATION REQUIRED
(1)      025354 012737 000001 001104      1$: MOV      #1,$ICNT    ;;REINITIALIZE THE ITERATION COUNTER
(1)      025362 013737 025440 001166      MOV      $MXCNT,$TIMES  ;;SET NUMBER OF ITERATIONS TO DO
(1)      025370 105237 001102      $SVLAD: INCB   $TSTNM    ;;COUNT TEST NUMBERS
(1)      025374 113737 001102 001202      MOVB    $TSTNM,$TESTN  ;;SET TEST NUMBER IN APT MAILBOX
(1)      025402 011637 001106      MOV      (SP),$LPADR   ;;SAVE SCOPE LOOP ADDRESS
  
```

```

(1) 025406 011637 001110      MOV      (SP), $LPERR      ;;SAVE ERROR LOOP ADDRESS
(1) 025412 005037 001170      CLR      $ESCAPE          ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
(1) 025416 112737 000001 001115  MOVVB   #1, $ERMAX        ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
(1) 025424 013777 001102 153510 $OVER:  MOV      $TSTNM, @DISPLAY ;;DISPLAY TEST NUMBER
(1) 025432 013716 001106      MOV      $LPADR, (SP)     ;;FUDGE RETURN ADDRESS
(1) 025436 000002      RTI                      ;;FIXES PS
(1) 025440 003720      $MXCNT: 2000.           ;;MAX. NUMBER OF ITERATIONS
2924 .SBTTL  APT COMMUNICATIONS ROUTINE
(1)
(2)
*****
(1) 025442 112737 000001 025706 $ATY1:  MOVVB   #1, $FFLG      ;;TO REPORT FATAL ERROR
(1) 025450 112737 000001 025704 $ATY3:  MOVVB   #1, $MFLG      ;;TO TYPE A MESSAGE
(1) 025456 000403      BR      $ATYC
(1) 025460 112737 000001 025706 $ATY4:  MOVVB   #1, $FFLG      ;;TO ONLY REPORT FATAL ERROR
(1) 025466      $ATYC:
(3) 025466 010046      MOV      R0, -(SP)        ;;PUSH R0 ON STACK
(3) 025470 010146      MOV      R1, -(SP)        ;;PUSH R1 ON STACK
(1) 025472 105737 025704      TSTB    $MFLG            ;;SHOULD TYPE A MESSAGE?
(1) 025476 001450      BEQ     5$               ;;IF NOT: BR
(1) 025500 122737 000001 001216  CMPB    #APTENV, $ENV     ;;OPERATING UNDER APT?
(1) 025506 001031      BNE     3$               ;;IF NOT: BR
(1) 025510 132737 000100 001217  BITB    #APTPOOL, $ENVM   ;;SHOULD SPOOL MESSAGES?
(1) 025516 001425      BEQ     3$               ;;IF NOT: BR
(1) 025520 017600 000004      MOV     @4(SP), R0        ;;GET MESSAGE ADDR.
(1) 025524 062766 000002 000004  ADD     #2, 4(SP)         ;;BUMP RETURN ADDR.
(1) 025532 005737 001176      1$:    TST     $MSGTYPE      ;;SEE IF DONE W/ LAST XMISSION?
(1) 025536 001375      BNE     1$               ;;IF NOT: WAIT
(1) 025540 010037 001212      MOV     R0, $MSGAD       ;;PUT ADDR IN MAILBOX
(1) 025544 105720      2$:    TSTB    (R0)+         ;;FIND END OF MESSAGE
(1) 025546 001376      BNE     2$
(1) 025550 163700 001212      SUB     $MSGAD, R0        ;;SUB START OF MESSAGE
(1) 025554 006200      ASR     R0               ;;GET MESSAGE LNGTH IN WORDS
(1) 025556 010037 001214      MOV     R0, $MSGGLGT     ;;PUT LENGTH IN MAILBOX
(1) 025562 012737 000004 001176  MOV     #4, $MSGTYPE     ;;TELL APT TO TAKE MSG.
(1) 025570 000413      BR      5$
(1) 025572 017637 000004 025616  3$:    MOV     @4(SP), 4$      ;;PUT MSG ADDR IN JSR LINKAGE
(1) 025600 062766 000002 000004  ADD     #2, 4(SP)         ;;BUMP RETURN ADDRESS
(3) 025606 013746 177776      MOV     177776, -(SP)    ;;PUSH 177776 ON STACK
(1) 025612 004737 027222      JSR     PC, $TYPE        ;;CALL TYPE MACRO
(1) 025616 000000      4$:    .WORD   0
(1) 025620      5$:
(1) 025620 105737 025706      10$:   TSTB    $FFLG            ;;SHOULD REPORT FATAL ERROR?
(1) 025624 001416      BEQ     12$              ;;IF NOT: BR
(1) 025626 005737 001216      TST     $ENV             ;;RUNNING UNDER APT?
(1) 025632 001413      BEQ     12$              ;;IF NOT: BR
(1) 025634 005737 001176      11$:   TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
(1) 025640 001375      BNE     11$              ;;IF NOT: WAIT
(1) 025642 017637 000004 001200  MOV     @4(SP), $FATAL   ;;GET ERROR #
(1) 025650 062766 000002 000004  ADD     #2, 4(SP)         ;;BUMP RETURN ADDR.
(1) 025656 005237 001176      INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
(1) 025662 105037 025706      12$:   CLRB    $FFLG            ;;CLEAR FATAL FLAG
(1) 025666 105037 025705      CLRB    $LFLG           ;;CLEAR LOG FLAG
(1) 025672 105037 025704      CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
(3) 025676 012601      MOV     (SP)+, R1        ;;POP STACK INTO R1
(3) 025700 012600      MOV     (SP)+, R0        ;;POP STACK INTO R0
(1) 025702 000207      RTS     PC               ;;RETURN

```

```

(1) 025704 000 $MFLG: .BYTE 0 ;;MESSG. FLAG
(1) 025705 000 $LFLG: .BYTE 0 ;;LOG FLAG
(1) 025706 000 $FFLG: .BYTE 0 ;;FATAL FLAG
(1) 025710 .EVEN
(1) 000200 APTSIZE=200
(1) 000001 APTENV=001
(1) 000100 APTSPOOL=100
(1) 000040 APTCSUP=040
2925 .SBTTL ERROR HANDLER ROUTINE
(1)
(2)
(1) *****
(1) *THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
(1) *SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
(1) *AND GO TO $ERRTYP ON ERROR
(1) *THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
(1) *SW15=1 HALT ON ERROR
(1) *SW13=1 INHIBIT ERROR TYPEOUTS
(1) *SW09=1 LOOP ON ERROR
(1) *CALL
(1) * ERROR N ;;ERROR=EMT AND N-ERROR ITEM NUMBER
(1) $ERROR:
(2) 025710 113737 001102 001734 MOV $STNM,TSTNUM
(1) 025716 105237 001103 7$: INCB $ERFLG ;;SET THE ERROR FLAG
(1) 025722 001775 BEQ 7$ ;;DON'T LET THE FLAG GO TO ZERO
(1) 025724 013777 001102 153210 MOV $STNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
(1) 025732 005237 001112 INC $ERTTL ;;INC THE ERROR COUNT
(1) 025736 011637 001116 MOV (SP),$ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
(1) 025742 162737 000002 001116 SUB #2,$ERRPC
(1) 025750 117737 153142 001114 MOV @ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
(1) 025756 032777 020000 153154 BIT #BIT13,@SWR ;;SKIP TYPEOUT IF SET
(1) 025764 001004 BNE 20$ ;;SKIP TYPEOUTS
(1) 025766 004737 026076 JSR PC,$ERRTYP ;;GO TO USER ERROR ROUTINE
(1) 025772 104401 001173 TYPE $CRLF
(1) 025776 20$:
(1) 025776 122737 000001 001216 CMPB #APTENV,$ENV ;;RUNNING IN APT MODE
(1) 026004 001007 BNE 2$ ;;NO,SKIP APT ERROR REPORT
(1) 026006 113737 001114 026020 MOV $ITEMB,21$ ;;SET ITEM NUMBER AS ERROR NUMBER
(1) 026014 004737 025460 JSR PC,$ATY4 ;;REPORT FATAL ERROR TO APT
(1) 026020 000 21$: .BYTE 0
(1) 026021 000 .BYTE 0
(1) 026022 000777 22$: BR 22$ ;;APT ERROR LOOP
(1) 026024 005777 153110 2$: TST @SWR ;;HALT ON ERROR
(1) 026030 100001 BPL 3$ ;;SKIP IF CONTINUE
(1) 026032 000000 HALT ;;HALT ON ERROR!
(1) 026034 032777 001000 153076 3$: BIT #BIT09,@SWR ;;LOOP ON ERROR SWITCH SET?
(1) 026042 001402 BEQ 4$ ;;BR IF NO
(1) 026044 013716 001110 MOV $LPERR,(SP) ;;FUDGE RETURN FOR LOOPING
(1) 026050 005737 001170 4$: TST $ESCAPE ;;CHECK FOR AN ESCAPE ADDRESS
(1) 026054 001402 BEQ 5$ ;;BR IF NONE
(1) 026056 013716 001170 MOV $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
(1) 026062 5$:
(1) 026062 022737 024470 000042 CMP #SENDAD,@#42 ;;ACT-11 AUTO-ACCEPT?
(1) 026070 001001 BNE 6$ ;;BRANCH IF NO
(1) 026072 000000 HALT ;;YES
(1) 026074 6$:

```

(1) 026074 000002 RTI ;:RETURN  
2926







```

(1) 026552 105716          TSTB    (SP)          ;;STILL DOING LEADING 0'S?
(1) 026554 100407          BMI     7$           ;;BR IF YES
(1) 026556 106316          5$:    ASLB    (SP)          ;;MSD?
(1) 026560 103003          BCC    6$           ;;BR IF NO
(1) 026562 116663 000001 177777  MOVB   1(SP),-1(R3)  ;;YES--SET THE SIGN
(1) 026570 052702 000060          6$:    BIS     #'0,R2    ;;MAKE THE BCD DIGIT ASCII
(1) 026574 052702 000040          7$:    BIS     #' ,R2    ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
(1) 026600 110223          MOVB   R2,(R3)+    ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
(1) 026602 005720          TST    (R0)+      ;;JUST INCREMENTING
(1) 026604 020027 000010          CMP    R0,#10     ;;CHECK THE TABLE INDEX
(1) 026610 002746          BLT    2$         ;;GO DO THE NEXT DIGIT
(1) 026612 003002          BGT    8$         ;;GO TO EXIT
(1) 026614 010502          MOV    R5,R2      ;;GET THE LSD
(1) 026616 000764          BR     6$         ;;GO CHANGE TO ASCII
(1) 026620 105726          8$:    TSTB   (SP)+    ;;WAS THE LSD THE FIRST NON-ZERO?
(1) 026622 100003          BPL    9$         ;;BR IF NO
(1) 026624 116663 177777 177776  MOVB   -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
(1) 026632 105013          9$:    CLRB   (R3)    ;;SET THE TERMINATOR
(3) 026634 012605          MOV    (SP)+,R5   ;;POP STACK INTO R5
(3) 026636 012603          MOV    (SP)+,R3   ;;POP STACK INTO R3
(3) 026640 012602          MOV    (SP)+,R2   ;;POP STACK INTO R2
(3) 026642 012601          MOV    (SP)+,R1   ;;POP STACK INTO R1
(3) 026644 012600          MOV    (SP)+,R0   ;;POP STACK INTO R0
(1) 026646 104401 026674          TYPE   $DBLK      ;;NOW TYPE THE NUMBER
(1) 026652 016666 000002 000004  MOV    2(SP),4(SP) ;;ADJUST THE STACK
(1) 026660 012616          MOV    (SP)+,(SP)
(1) 026662 000002          RTI                    ;;RETURN TO USER
(1) 026664 023420          $DTBL: 10000.
(1) 026666 001750          1000.
(1) 026670 000144          100.
(1) 026672 000012          10.
(1) 026674 000004          $DBLK: .BLKW 4

2933
2934
(1)
(2)
(1)
(1) 026704 012737 027050 000024 $PWRDN: MOV    # $ILLUP,@#PWRVEC ;;SET FOR FAST UP
(1) 026712 012737 000340 000026      MOV    #340,@#PWRVEC+2 ;;PRIO:7
(3) 026720 010046          MOV    R0,-(SP)    ;;PUSH R0 ON STACK
(3) 026722 010146          MOV    R1,-(SP)    ;;PUSH R1 ON STACK
(3) 026724 010246          MOV    R2,-(SP)    ;;PUSH R2 ON STACK
(3) 026726 010346          MOV    R3,-(SP)    ;;PUSH R3 ON STACK
(3) 026730 010446          MOV    R4,-(SP)    ;;PUSH R4 ON STACK
(3) 026732 010546          MOV    R5,-(SP)    ;;PUSH R5 ON STACK
(3) 026734 017746 152200          MOV    @SWR,-(SP)  ;;PUSH @SWR ON STACK
(1) 026740 010637 027054          MOV    SP,$SAVR6   ;;SAVE SP
(1) 026744 012737 026756 000024  MOV    # $PWRUP,@#PWRVEC ;;SET UP VECTOR
(1) 026752 000000          HALT
(1) 026754 000776          BR     .-2         ;;HANG UP

(1)
(2)
(1)
(1) 026756 012737 027050 000024 $PWRUP: MOV    # $ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
(1) 026764 013706 027054          MOV    $SAVR6,SP   ;;GET SP
(1) 026770 005037 027054          CLR    $SAVR6      ;;WAIT LOOP FOR THE TTY
    
```



```

(1) ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
(1) ;*NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
(1) ;*NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
(1) ;*NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
(1) ;*
(1) ;*CALL:
(1) ;*1) USING A TRAP INSTRUCTION
(1) ;* TYPE ,MESADR ;:MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
(1) ;*OR
(1) ;* TYPE
(1) ;* MESADR
(1) ;*
(1) 027222 105737 001157 $TYPE: TSTB $TPFLG ;:IS THERE A TERMINAL?
(1) 027226 100002 BPL 1$ ;:BR IF YES
(1) 027230 000000 HALT ;:HALT HERE IF NO TERMINAL
(1) 027232 000430 BR 3$ ;:LEAVE
(1) 027234 010046 1$: MOV RO,-(SP) ;:SAVE RO
(1) 027236 017600 000002 MOV @2(SP),RO ;:GET ADDRESS OF ASCIZ STRING
(1) 027242 122737 000001 001216 CMPB #APTENV,$ENV ;:RUNNING IN APT MODE
(1) 027250 001011 BNE 62$ ;:NO,GO CHECK FOR APT CONSOLE
(1) 027252 132737 000100 001217 BITB #APTPOOL,$ENVM ;:SPOOL MESSAGE TO APT
(1) 027260 001405 BEQ 62$ ;:NO,GO CHECK FOR CONSOLE
(1) 027262 010037 027272 MOV RO,61$ ;:SETUP MESSAGE ADDRESS FOR APT
(1) 027266 004737 025450 JSR PC,$ATY3 ;:SPOOL MESSAGE TO APT
(1) 027272 000000 61$: .WORD 0 ;:MESSAGE ADDRESS
(1) 027274 132737 000040 001217 62$: BITB #APTC SUP,$ENVM ;:APT CONSOLE SUPPRESSED
(1) 027302 001003 BNE 60$ ;:YES,SKIP TYPE OUT
(1) 027304 112046 2$: MOVB (RO)+,-(SP) ;:PUSH CHARACTER TO BE TYPED ONTO STACK
(1) 027306 001005 BNE 4$ ;:BR IF IT ISN'T THE TERMINATOR
(1) 027310 005726 TST (SP)+ ;:IF TERMINATOR POP IT OFF THE STACK
(1) 027312 012600 60$: MOV (SP)+,RO ;:RESTORE RO
(1) 027314 062716 000002 3$: ADD #2,(SP) ;:ADJUST RETURN PC
(1) 027320 000002 RTI ;:RETURN
(1) 027322 122716 000011 4$: CMPB #HT,(SP) ;:BRANCH IF <HT>
(1) 027326 001430 BEQ 8$ ;:BRANCH IF NOT <CRLF>
(1) 027330 122716 000200 CMPB #CRLF,(SP) ;:BRANCH IF NOT <CRLF>
(1) 027334 001006 BNE 5$ ;:BRANCH IF NOT <CRLF>
(1) 027336 005726 TST (SP)+ ;:POP <CR><LF> EQUIV
(1) 027340 104401 TYPE ;:TYPE A CR AND LF
(1) 027342 001173 $CRLF
(1) 027344 105037 027500 CLRB $CHARCNT ;:CLEAR CHARACTER COUNT
(1) 027350 000755 BR 2$ ;:GET NEXT CHARACTER
(1) 027352 004737 027434 5$: JSR PC,$TYPEC ;:GO TYPE THIS CHARACTER
(1) 027356 123726 001156 6$: CMPB $FILLC,(SP)+ ;:IS IT TIME FOR FILLER CHARS.?
(1) 027362 001350 BNE 2$ ;:IF NO GO GET NEXT CHAR.
(1) 027364 013746 001154 MOV $NULL,-(SP) ;:GET # OF FILLER CHARS. NEEDED
(1) ;:AND THE NULL CHAR.
(1) 027370 105366 000001 7$: DECB 1(SP) ;:DOES A NULL NEED TO BE TYPED?
(1) 027374 002770 BLT 6$ ;:BR IF NO--GO POP THE NULL OFF OF STACK
(1) 027376 004737 027434 JSR PC,$TYPEC ;:GO TYPE A NULL
(1) 027402 105337 027500 DECB $CHARCNT ;:DO NOT COUNT AS A COUNT
(1) 027406 000770 BR 7$ ;:LOOP
(1)
(1) ;HORIZONTAL TAB PROCESSOR
(1)
    
```

```

(1) 027410 112716 000040      8$:  MOV#  (SP)      ;;REPLACE TAB WITH SPACE
(1) 027414 004737 027434      9$:  JSR  PC,$TYPEC  ;;TYPE A SPACE
(1) 027420 132737 000007 027500 BITB  #7,$CHARCNT  ;;BRANCH IF NOT AT
(1) 027426 001372              BNE   9$          ;;TAB STOP
(1) 027430 005726              TST   (SP)+      ;;POP SPACE OFF STACK
(1) 027432 000724              BR    2$          ;;GET NEXT CHARACTER
(1) 027434 105777 151510      $TYPEC: TSTB @ $STPS  ;;WAIT UNTIL PRINTER IS READY
(1) 027440 100375              BPL   $TYPEC
(1) 027442 116677 000002 151502 MOV#  2(SP),@ $STPB ;;LOAD CHAR TO BE TYPED INTO DATA REG.
(1) 027450 122766 000015 000002 CMP#  #CR,2(SP)    ;;IS CHARACTER A CARRIAGE RETURN?
(1) 027456 001003              BNE   1$          ;;BRANCH IF NO
(1) 027460 105037 027500      CLRB  $CHARCNT    ;;YES--CLEAR CHARACTER COUNT
(1) 027464 000406              BR    $TYPEX     ;;EXIT
(1) 027466 122766 000012 000002 1$:  CMP#  #LF,2(SP)   ;;IS CHARACTER A LINE FEED?
(1) 027474 001402              BEQ   $TYPEX     ;;BRANCH IF YES
(1) 027476 105227              INCB  (PC)+      ;;COUNT THE CHARACTER
(1) 027500 000000      $CHARCNT: .WORD  0 ;;CHARACTER COUNT STORAGE
(1) 027502 000207      $TYPEX: RTS      PC

(1)
2940
2941      .SBTTL  INTEGER MULTIPLY ROUTINE
(1)
(2)      ;;*****
(1)      ;;*CALL
(1)      ;;*
(1)      ;;*   MOV    MULTIPLER,-(SP)
(1)      ;;*   MOV    MULTIPLICAND,-(SP)
(1)      ;;*   JSR    PC,@ $MULT
(1)      ;;*   RETURN  ;;PRODUCT IS ON THE STACK
(1)      ;;*
(1)      ;;*   STACK  PRODUCT
(1)      ;;*   -----
(1)      ;;*   TOP    LSB'S
(1)      ;;*   +2     MSB'S
(1)
(1) 027504      $MULT:
(3) 027504 010046      MOV    R0,-(SP)    ;;PUSH R0 ON STACK
(3) 027506 010146      MOV    R1,-(SP)    ;;PUSH R1 ON STACK
(3) 027510 010246      MOV    R2,-(SP)    ;;PUSH R2 ON STACK
(1) 027512 005046      CLR    -(SP)      ;;CLEAR THE SIGN KEY
(1) 027514 016601 000012  MOV    12(SP),R1   ;;GET THE MULTIPLICAND
(1) 027520 100002      BPL   1$          ;;BR IF PLUS
(1) 027522 005216      INC   (SP)       ;;SET THE SIGN KEY
(1) 027524 005401      NEG   R1         ;;MAKE THE MULTIPLICAND POSTIVE
(1) 027526 016602 000014  1$:  MOV    14(SP),R2   ;;GET THE MULTIPLIER
(1) 027532 100002      BPL   2$          ;;BR IF PLUS
(1) 027534 005316      DEC   (SP)       ;;UPDATE THE SIGN KEY
(1) 027536 005402      NEG   R2         ;;MAKE THE MULTIPLIER POSTIVE
(1) 027540 012746 000021  2$:  MOV    #17,-(SP)  ;;SET THE LOOP COUNT
(1) 027544 005000      CLR   R0         ;;SETUP FOR THE MULTIPLY LOOP
(1) 027546 103001      3$:  BCC   4$          ;;DON'T ADD IF MULTIPLICAND = 0
(1) 027550 060200      ADD   R2,R0
(1) 027552 006000      4$:  ROR   R0         ;;POSITION THE PARITIAL PRODUCT AND
(1) 027554 006001      ROR   R1         ;;THE MULTIPLICAND
(1) 027556 005316      DEC   (SP)       ;;HAS ALL BITS OF THE MULTIPLICAND BEEN DONE?
(1) 027560 001372      BNE   3$          ;;BR IF NO
(1) 027562 022616      CMP   (SP)+,(SP) ;;SHOULD PRODUCT BE NEGATIVE?
    
```

```
(1) 027564 001403      BEQ      5$          ;;GO TO EXIT IF NO  
(1) 027566 005400      NEG      R0          ;;YES--SO MAKE IT SO  
(1) 027570 005401      NEG      R1  
(1) 027572 005600      SBC      R0  
(1) 027574 005726      5$: TST      (SP)+      ;;CLEAR SIGN INFO. OFF OF STACK  
(1) 027576 010066 000012 MOV      R0,12(SP)   ;;PUT THE PRODUCT ON THE STACK (MSB'S)  
(1) 027602 010166 000010 MOV      R1,10(SP)   ;;LSB'S  
(3) 027606 012602      MOV      (SP)+,R2    ;;POP STACK INTO R2  
(3) 027610 012601      MOV      (SP)+,R1    ;;POP STACK INTO R1  
(3) 027612 012600      MOV      (SP)+,R0    ;;POP STACK INTO R0  
(1) 027614 000207      RTS      PC
```

```
2942  
2943  
(1) .SBTTL TRAP DECODER  
(2) ;*****  
(1) ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE 'TRAP' INSTRUCTION  
(1) ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS  
(1) ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL  
(1) ;*GO TO THAT ROUTINE.
```

```
(1) 027616 010046      $TRAP: MOV      R0,-(SP)    ;;SAVE R0  
(1) 027620 016600 000002 MOV      2(SP),R0    ;;GET TRAP ADDRESS  
(1) 027624 005740      TST      -(R0)      ;;BACKUP BY 2  
(1) 027626 111000      MOV      (R0),R0    ;;GET RIGHT BYTE OF TRAP  
(1) 027630 006300      ASL      R0          ;;POSITION FOR INDEXING  
(1) 027632 016000 027652 MOV      $TRPAD(R0),R0 ;;INDEX TO TABLE  
(1) 027636 000200      RTS      R0          ;;GO TO ROUTINE
```

(1) ;:THIS IS USE TO HANDLE THE 'GETPRI' MACRO

```
(1) 027640 011646      $TRAP2: MOV      (SP),-(SP) ;;MOVE THE PC DOWN  
(1) 027642 016666 000004 000002 MOV      4(SP),2(SP) ;;MOVE THE PSW DOWN  
(1) 027650 000002      RTI          ;;RESTORE THE PSW
```

```
(3) .SBTTL TRAP TABLE  
(3) ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED  
(3) ;*BY THE 'TRAP' INSTRUCTION.
```

```
(3) ; ROUTINE  
(3) ;-----  
(3) $TRPAD: .WORD $TRAP2  
(3) $TYPE      ;;CALL=TYPE      TRAP+1(104401) TTY TYPEOUT ROUTINE  
(3) $TYPOC     ;;CALL=TYPOC     TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)  
(3) $TYPOS     ;;CALL=TYPOS     TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)  
(3) $TYPON     ;;CALL=TYPON     TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)  
(3) $TYPDS     ;;CALL=TYPDS     TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
```

```
2946 .SBTTL ASCII MESSAGES
2947 027666 005015 051412 040524 HEADER: .ASCIZ <15><12><12>/START OF CZVSBB VS60 INSTRUCTION TEST PART II/<15><12>
2948 027752 005015 042523 020124 ECOMSG: .ASCIZ <15><12>/SET SWITCH REGISTER BIT 10 (002000) IF ECO VT48-0012 IS INSTALL
2949 030061 115 044501 052116 EM1: .ASCIZ /MAINT. SWITCH REG/
2950 030103 114 040517 044504 EM2: .ASCIZ /LOADING D.P.C. REG <STATIC>/
2951 030137 114 040517 044504 EM3: .ASCIZ /LOADING RELOCATE REG/
2952 030164 042522 042101 047111 EM4: .ASCIZ /READING D.P.C. WITH RELOCATE REG SET/
2953 030231 117 043106 042523 EM5: .ASCIZ /OFFSET INDICATOR FAILED TO SET/
2954 030270 043117 051506 052105 EM6: .ASCIZ /OFFSET POLARITY BIT FAILED TO SET/
2955 030332 043117 051506 052105 EM7: .ASCIZ /OFFSET INSTR FAILED TO LOAD DYNAMIC REG/
2956 030402 043117 051506 052105 EM10: .ASCIZ /OFFSET INSTR FAILED TO LOAD EXTENDED POS BITS/
2957 030460 041523 046101 020105 EM11: .ASCIZ /SCALE ER/
2958 030471 101 051502 053040 EM12: .ASCIZ /ABS VECTOR FAILED TO SET UP X OR Y POS REG/
2959 030544 051107 050101 050110 EM13: .ASCIZ /GRAPHPLOT INC REG ER/
2960 030571 114 040517 044504 EM14: .ASCIZ /LOADING X POS REG/
2961 030613 114 040517 044504 EM15: .ASCIZ /LOADING Y POS REG/
2962 030635 111 041516 042522 EM16: .ASCIZ /INCREMENTING DELTA X OR DELTA Y REG/
2963 030701 104 041505 042522 EM17: .ASCIZ /DECREMENTING DELTA X OR DELTA Y REG/
2964 030745 105 043504 020105 EM20: .ASCIZ /EDGE INDICATOR UPON EXCEEDING AN EDGE/
2965 031013 114 040517 044504 EM21: .ASCIZ /LOADING CHAR REG/
2966 031034 047111 047503 051122 EM22: .ASCIZ /INCORRECT OR UNEXPECTED CHANGE IN X OR Y POSITION REGS/
2967 031123 127 047522 043516 EM23: .ASCIZ /WRONG DELTA X - SCALED CHAR MODE/
2968 031164 051127 047117 020107 EM24: .ASCIZ /WRONG DELTA Y - SCALED CHAR MODE/
2969 031225 127 047522 043516 EM25: .ASCIZ /WRONG DELTA Y - SCALED CHAR MODE ROTATED/
2970 031276 051127 047117 020107 EM26: .ASCIZ /WRONG DELTA X - SCALED CHAR MODE ROTATED/
2971 031347 107 040522 044120 EM27: .ASCIZ /GRAPHPLOT INC FAILED TO SET UP X OR Y POS REG/
2972 031425 102 051501 041511 EM30: .ASCIZ /BASIC VECTOR FAILED TO LOAD X OR Y POS REG/
2973 031500 040502 044523 020103 EM31: .ASCIZ /BASIC SHORT VECTOR FAILED TO LOAD X OR Y POS REG/
2974 031561 122 051505 052105 EM32: .ASCIZ /RESET FAILED TO CLEAR A REG/
2975 031615 105 043504 020105 EM33: .ASCIZ /EDGE FLAG INTR ER/
2976 031637 111 052116 020122 EM34: .ASCIZ /INTR PRIORITY FAILURE USING INTERNAL STOP/
2977 031711 123 044510 052106 EM35: .ASCIZ /SHIFT-OUT BIT SET IN ERROR/
2978 031744 044123 043111 026524 EM36: .ASCIZ /SHIFT-OUT BIT FAILED TO SET/
2979 032000 047111 042524 047122 EM37: .ASCIZ /INTERNAL STOP FLAG FAILED TO SET/
2980 032041 047 051106 046501 EM40: .ASCIZ /'FRAMES PER SECOND' SYNC ER/
2981
2982 032075 114 047117 020107 EM41: .ASCIZ /LONG VECTOR FAILED TO SET UP X OR Y POS REG/
2983 032151 120 044517 052116 EM42: .ASCIZ /POINT OR ABS VEC WITH AN OFFSET FAILED TO SET UP X OR Y POS/
2984 032245 123 050125 051105 EM43: .ASCIZ /'SUPER/SUBSCRIPT CHAR SCALE OR DELTA X OR Y ER'
2985
2986 032323 105 051122 041520 DH1: .ASCIZ /ERRPC TSTNUM BUSADR EXPCT RCVD/
2987 032370 051105 050122 020103 DH2: .ASCIZ /ERRPC TSTPC/
2988 032406 051105 050122 020103 DH11: .ASCIZ /ERRPC TSTNUM EXPCT RCVD SCALE STARTVAL/
2989 032470 001116 001734 001122 .EVEN
2990 032504 001116 001106 000000 DT1: $ERRPC,$TSTNUM,$BDADR,$GDDAT,$BDDAT,0
2991 032512 001116 001734 001124 DT2: $ERRPC,$LPADR,0
2992 032512 001116 001734 001124 DT11: $ERRPC,$TSTNUM,$GDDAT,$BDDAT,$TMP0,$REG0,0
2993
2994
2995 ;:*****
2996 ; THIS IS THE WORKING AREA FOR ALL VS60 NPR'S (INSTR'S AND DATA)
2997 ; FROM HERE TO THE END OF 8K
2998 ;:*****
2999 BUFFER: 0
3000 .END
```

ABASE = 172000	21#	36		
ACDW1 = 000000	36			
ACDW2 = 000000	36			
ACPUOP= 000000	36			
ADDW0 = 000000	36			
ADDW1 = 000000	36			
ADDW10= 000000	36			
ADDW11= 000000	36			
ADDW12= 000000	36			
ADDW13= 000000	36			
ADDW14= 000000	36			
ADDW15= 000000	36			
ADDW2 = 000000	36			
ADDW3 = 000000	36			
ADDW4 = 000000	36			
ADDW5 = 000000	36			
ADDW6 = 000000	36			
ADDW7 = 000000	36			
ADDW8 = 000000	36			
ADDW9 = 000000	36			
ADEVCT= 000000	36			
ADEVN = 000000	36			
AENV = 000000	36			
AENVN = 000000	36			
AFATAL= 000000	36			
AMADR1= 000000	36			
AMADR2= 000000	36			
AMADR3= 000000	36			
AMADR4= 000000	36			
AMAMS1= 000000	36			
AMAMS2= 000000	36			
AMAMS3= 000000	36			
AMAMS4= 000000	36			
AMSGAD= 000000	36			
AMSGLG= 000000	36			
AMSGTY= 000000	36			
AMTYP1= 000000	36			
AMTYP2= 000000	36			
AMTYP3= 000000	36			
AMTYP4= 000000	36			
ANAME 002000	334#			
APASS = 000000	36			
APRIOR= 000000	36			
APTCSU= 000040	2924#	2939		
APTENV= 000001	2924#	2925	2939	
APTSIZ= 000200	361	2924#		
APTSP0= 000100	2924#	2939		
ASWREG= 000000	36			
ATESTN= 000000	36			
AUNIT = 000000	36			
AUSWR = 000000	36			
AVECT1= 100320	22#	36		
AVECT2= 000000	36			
BEGIN 002050	27	360#		
BEGINA 002040	29	358#	377	
BEGIN1 002054	359	361#		









TST10	003562	522	526#		
TST100	022450	2490	2499	2505	2509#
TST101	023020	2531	2539	2547	2566#
TST102	023156	2583	2591#		
TST103	023336	2609	2615	2622#	
TST104	023466	2635	2639#		
TST105	023610	2653	2659#		
TST106	023776	2677	2685	2692#	
TST107	024102	2704	2707#		
TST11	004004	554	560#		
TST110	024360	2748	2751	2753	2762#
TST12	004242	592	598#		
TST13	004512	635	641#		
TST14	004654	664#			
TST15	005016	687#			
TST16	005252	722#			
TST17	005626	777	782#		
TST2	002572	412#			
TST20	005756	799	803#		
TST21	006140	833#			
TST22	006324	865#			
TST23	006562	897	903#		
TST24	006752	935#			
TST25	007136	966#			
TST26	007372	997	1003#		
TST27	007562	1035#			
TST3	002670	425#			
TST30	007746	1066#			
TST31	010202	1097	1103#		
TST32	010456	1148#			
TST33	010714	1180	1186#		
TST34	011060	1206	1212#		
TST35	011246	1232	1244#		
TST36	011506	1276	1282#		
TST37	011746	1314	1320#		
TST4	002764	439#			
TST40	012246	1352	1374#		
TST41	012524	1411	1415#		
TST42	013024	1452	1472#		
TST43	013254	1502	1508#		
TST44	013562	1561	1568#		
TST45	013762	1587	1596	1602	1605#
TST46	014126	1624	1630	1633#	
TST47	014232	1648	1652#		
TST5	003106	457#			
TST50	014370	1673	1677#		
TST51	015356	1784	1836#		
TST52	015530	1851	1860#		
TST53	015700	1875	1880	1884#	
TST54	016066	1907	1911#		
TST55	016212	1927	1931#		
TST56	016336	1947	1951#		
TST57	016462	1968	1971#		
TST6	003254	475	483#		
TST60	016606	1988	1991#		
TST61	016732	2008	2011#		

TST62	017056	2028	2031#															
TST63	017202	2048	2051#															
TST64	017362	2074	2077#															
TST65	017506	2094	2097#															
TST66	017632	2114	2117#															
TST67	017756	2134	2137#															
TST7	003420	501	505#															
TST70	020102	2154	2157#															
TST71	020226	2174	2177#															
TST72	020352	2194	2197#															
TST73	020476	2214	2217#															
TST74	021174	2268	2320#															
TST75	021644	2371	2409#															
TST76	022016	2426	2432	2436#														
TST77	022230	2453	2462	2468	2472#													
TYPDS =	104405	2767	2943#															
TYPE =	104401	385	387	2767	2925	2928	2931	2932	2934	2939	2943#							
TYPOC =	104402	2928	2943#															
TYPON =	104404	2943#																
TYPOS =	104403	2943#																
XDOFF	001774	332#	538	541	561*	576	599*	619	643	653*	655	700	702	741				
		794	796	1116*	1121	1748												
XPORR1	021614	2375	2395#															
XPOS	001764	328#	573	575	616	618	736	740	817	820	848	851	881	884				
		918	921	949	952	981	984	1018	1021	1049	1052	1081	1084	1119				
		1120	1164	1167	1213	1227	1259	1262	1304	1307	1333	1335	1390	1392				
		1428	1430	1488	1490	1540	1551	1746	1747	1869	1873	1895	1901	1922				
		1928	1942	1952	1963	2070	2138	2149	2158	2169	2178	2189	2198	2209				
		2241	2243	2344	2346	2410	2421	2446	2460	2492	2497	2541	2545					
XPOSR	021114	2227	2291#															
XPOSRR	021144	2272	2306#															
XPOSR1	021564	2330	2381#															
YDOFF	001776	333#	545	547	562*	584	600*	627	666	676*	678	707	709	751				
		798	800	1117*	1128	1757												
YPORR1	021630	2376	2402#															
YPOS	001766	329#	581	583	624	626	746	750	823	826	854	857	888	891				
		924	925	955	958	988	991	1024	1027	1055	1058	1088	1091	1126				
		1127	1171	1174	1187	1202	1267	1270	1297	1300	1340	1342	1396	1398				
		1435	1437	1494	1496	1544	1555	1755	1756	1837	1846	1877	1881	1903				
		1908	1948	1972	1983	1992	2003	2012	2023	2032	2043	2052	2063	2078				
		2089	2098	2109	2118	2129	2248	2250	2351	2353	2428	2433	2451	2455				
		2482	2488	2525	2529	2549	2554	2611	2616	2693	2702							
YPOSR	021130	2228	2299#															
YPOSRR	021160	2273	2313#															
YPOSR1	021600	2331	2388#															
ZDOFF	002016	341#	485	493	500	506	514	521										
ZPOS	002014	340#																
\$APTHD	001000	34#																
\$ASTAT=	***** U	2924																
\$ATYC	025466	2924#																
\$ATY1	025442	2924#																
\$ATY3	025450	2924#	2939															
\$ATY4	025460	2924#	2925															
\$AUTOB	001134	36#																
\$BASE	001252	36#	367															
\$BDADR	001122	36#	401*	413*	426*	441*	467*	485*	506*	538*	545*	573*	581*	616*				

624*	643*	666*	700*	707*	736*	746*	796*	800*	820*	826*	851*	857*
881*	888*	921*	924*	952*	958*	981*	988*	1021*	1027*	1052*	1058*	1081*
1088*	1119*	1126*	1164*	1171*	1187*	1213*	1259*	1267*	1297*	1304*	1333*	1340*
1390*	1396*	1428*	1435*	1488*	1494*	1569*	1606*	1634*	1653*	1725*	1732*	1746*
1755*	1770*	1837*	1873*	1881*	1901*	1908*	1928*	1948*	1952*	1972*	1992*	2012*
2032*	2052*	2078*	2098*	2118*	2138*	2158*	2178*	2198*	2234*	2241*	2248*	2337*
2344*	2351*	2410*	2433*	2451*	2460*	2469*	2488*	2497*	2506*	2529*	2537*	2545*
2554*	2562*	2581*	2607*	2616*	2624*	2636*	2644*	2660*	2693*	2709*	299)	
36#	404*	405	417*	418*	419	431*	432	447*	448	470*	476*	495*
502*	516*	523*	541*	542	547*	548	575*	578	583*	586	618*	621
626*	629	655*	656	678*	679	702*	703*	704	709*	710*	711	740*
743	750*	753	794*	798*	817*	818	823*	824	848*	849	854*	855
884*	885	891*	892	918*	919	925*	926	949*	950	955*	956	984*
985	991*	992	1018*	1019	1024*	1025	1049*	1050	1055*	1056	1084*	1085
1091*	1092	1120*	1123	1127*	1130	1167*	1168	1174*	1175	1202*	1203	1227*
1228*	1229	1262*	1263*	1264	1270*	1271	1300*	1301	1307*	1308*	1309	1335*
1336*	1337	1342*	1343*	1344	1392*	1393	1398*	1399	1430*	1431*	1432	1437*
1438*	1439	1490*	1491	1496*	1497	1540*	1541	1544*	1545	1551*	1552	1555*
1556	1583*	1584	1592*	1593	1600*	1601	1620*	1621	1628*	1629	1646*	1647
1665*	1666	1671*	1672*	1727*	1728*	1739*	1740*	1741	1747*	1750	1756*	1759
1778*	1779*	1780	1846*	1847*	1848	1869*	1871	1877*	1878*	1879	1895*	1896*
1899	1903*	1904*	1906	1922*	1923*	1926	1942*	1943*	1946	1963*	1964*	1967
1983*	1984*	1987	2003*	2004*	2007	2023*	2024*	2027	2043*	2044*	2047	2063*
2064*	2067	2070*	2071*	2073	2089*	2090*	2093	2109*	2110*	2113	2129*	2130*
2133	2149*	2150*	2153	2169*	2170*	2173	2189*	2190*	2193	2209*	2210*	2213
2236*	2237*	2238	2243*	2244*	2245	2250*	2251*	2252	2339*	2340*	2341	2346*
2347*	2348	2353*	2354*	2355	2421*	2423	2428*	2429*	2431	2446*	2449	2455*
2456*	2458	2466*	2467	2482*	2483*	2486	2492*	2493*	2495	2503*	2504	2525*
2526*	2527	2533*	2535	2541*	2543	2549*	2550*	2552	2558*	2560	2578*	2579
2604*	2605	2611*	2612*	2614	2633*	2634*	2651*	2675*	2683*	2702*	2703*	2711*
2915*	2920*	2990	2992									

\$BDDAT 001126

\$CDW1 001256  
 \$CDW2 001260  
 \$CHARC 027500  
 \$CKSWR= \*\*\*\*\* U  
 \$CMTAG 001100  
 \$CM1 = 000001  
 \$CM2 = 000002  
 \$CM3 = 000001  
 \$CM4 = 000001  
 \$CPUOP 001224  
 \$CRLF 001173  
 \$DBLK 026674  
 \$DEVCT 001206  
 \$DEVN 001254  
 \$DOAGN 024500  
 \$DTBL 026664  
 \$ENDAD 024470  
 \$ENDCT 024436  
 \$ENDMG 024507  
 \$ENULL 024504  
 \$ENV 001216  
 \$ENVN 001217  
 \$EOP 024402  
 \$EOPCT 024430  
 \$ERFLG 001103

36#  
 36#  
 2939#\*  
 2943  
 36# 361  
 36#  
 36#  
 36#  
 36#  
 36#  
 36# 2925 2928 2939  
 2932#  
 36#  
 36#  
 2767#  
 2932#  
 31 2767# 2925  
 361 2767#  
 2767#  
 2767#  
 2767#  
 36# 2924 2925 2939  
 36# 361 2924 2939  
 2764 2767#  
 361\* 2767#  
 36# 2923\* 2925\*

\$ERMAX	001115	36#	361*	2923*										
\$ERROR	025710	361	2925#											
\$ERRPC	001116	36#	2925*	2928	2990	2991	2992							
\$ERRTB	001262	36#	2928											
\$ERRTY	026076	2925	2928#											
\$ERTTL	001112	36#	2925*											
\$ESCAP	001170	36#	361*	2923*	2925									
\$ETABL	001216	36#												
\$ETEND	001262	34	36#											
\$FATAL	001200	36#	2924*											
\$FFLG	025706	2924#*												
\$FILLC	001156	36#	2939											
\$FILLS	001155	36#	2939											
\$GDADR	001120	36#												
\$GDDAT	001124	36#	402*	403	405	408*	415*	416	419	422*	428*	430	432	436*
		443*	448	451*	459*	473*	486*	497*	507*	518*	539*	540*	542	546*
		548	574*	578	582*	586	617*	621	625*	629	644*	656	659*	667*
		679	682*	701*	704	708*	711	743	753	791*	806*	812	813	818
		824	829*	836*	849	855	861*	882*	885	889*	892	906*	919	926
		931*	938*	950	956	962*	982*	985	989*	992	1006*	1019	1025	1031*
		1038*	1050	1056	1062*	1082*	1085	1089*	1092	1123	1130	1165*	1168	1172*
		1175	1189*	1203	1208*	1215*	1229	1235*	1260*	1264	1268*	1271	1298*	1301
		1305*	1309	1334*	1337	1341*	1344	1391*	1393	1397*	1399	1429*	1432	1436*
		1439	1489*	1491	1495*	1497	1538*	1541	1545	1549*	1552	1556	1582*	1591*
		1599*	1619*	1627*	1645*	1647	1664*	1670*	1726*	1733*	1738*	1741	1745*	1750
		1754*	1759	1777*	1780	1840*	1848	1853*	1870*	1871	1879	1897*	1898*	1899
		1905*	1906	1924*	1925*	1926	1944*	1945*	1946	1965*	1966*	1967	1985*	1986*
		1987	2005*	2006*	2007	2025*	2026*	2027	2045*	2046*	2047	2065*	2066*	2067
		2072*	2073	2091*	2092*	2093	2111*	2112*	2113	2131*	2132*	2133	2151*	2152*
		2153	2171*	2172*	2173	2191*	2192*	2193	2211*	2212*	2213	2235*	2238	2242*
		2245	2249*	2252	2338*	2341	2345*	2348	2352*	2355	2422*	2423	2430*	2431
		2447*	2448*	2449	2457*	2458	2484*	2485*	2486	2494*	2495	2524*	2527	2534*
		2542*	2543	2551*	2552	2559*	2577*	2603*	2613*	2614	2632*	2645*	2661*	2701*
		2710*	2885*	2909*	2990	2992								
\$GET42	024460	2767#												
\$GTSWR=	***** U	2943												
\$HD =	000000	13												
\$HIBTS	001000	34#												
\$ICNT	001104	36#	2923*											
\$ILLUP	027050	2934#												
\$INTAG	001135	36#												
\$ITEMB	001114	36#	2925*	2928										
\$IF	001174	36#	2925	2939										
\$LFLG	025705	2924#*												
\$LPADR	001106	36#	361*	2923*	2991									
\$LPERR	001110	36#	361*	400*	414*	427*	446*	527*	563*	601*	642*	665*	698*	733*
		804*	834*	879*	904*	936*	979*	1004*	1036*	1079*	1114*	1162*	1257*	1295*
		1321*	1387*	1426*	1486*	1534*	1838*	2232*	2335*	2923*	2925			
\$LSTAD	027220	381	2957#*											
\$MADR1	001230	36#												
\$MADR2	001234	36#												
\$MADR3	001240	36#												
\$MADR4	001244	36#												
\$MAIL	001176	34	36#	361	399	412	425	439	457	483	505	526	560	598
		641	664	687	722	782	803	833	865	903	935	966	1003	1035
		1066	1103	1148	1186	1212	1244	1282	1320	1374	1415	1472	1508	1568

		1605	1633	1652	1677	1836	1860	1884	1911	1931	1951	1971	1991	2011
		2031	2051	2077	2097	2117	2137	2157	2177	2197	2217	2320	2409	2436
		2472	2509	2566	2591	2622	2639	2659	2692	2707	2762	2923	2925	2939
\$MAMS1	001226	36#												
\$MAMS2	001232	36#												
\$MAMS3	001236	36#												
\$MAMS4	001242	36#												
\$MBADR	001002	34#												
\$MFLG	025704	2924#*												
\$MSGAD	001212	36#	2924*											
\$MSGLG	001214	36#	2924*											
\$MSGTY	001176	36#	2924*											
\$MTYP1	001227	36#												
\$MTYP2	001233	36#												
\$MTYP3	001237	36#												
\$MTYP4	001243	36#												
\$MULT	027504	2839	2941#											
\$MXCNT	025440	2923#												
\$NULL	001154	36#	2939											
\$NWTST=	000001	399#	412#	425#	439#	457#	483#	505#	526#	560#	598#	641#	664#	687#
		722#	782#	803#	833#	865#	903#	935#	966#	1003#	1035#	1066#	1103#	1148#
		1186#	1212#	1244#	1282#	1320#	1374#	1415#	1472#	1508#	1568#	1605#	1633#	1652#
		1677#	1836#	1860#	1884#	1911#	1931#	1951#	1971#	1991#	2011#	2031#	2051#	2077#
		2097#	2117#	2137#	2157#	2177#	2197#	2217#	2320#	2409#	2436#	2472#	2509#	2566#
		2591#	2622#	2639#	2659#	2692#	2707#	2762#						
\$OCNT	026454	2931#*												
\$OMODE	026456	2931#*												
\$OVER	025424	2923#												
\$PASS	001204	36#	361*	2767*	2773	2923								
\$PASTM	001006	34#												
\$PWRAD	027044	2934#												
\$PWRDN	026704	361	2934#											
\$PWRMG	027040	2934#												
\$PWRUP	026756	2934#												
\$QUES	001172	36#	2925	2939										
\$RDCHR=	***** U	2943												
\$RDDEC=	***** U	2943												
\$RDLIN=	***** U	2943												
\$RDOCT=	***** U	2943												
\$REGAD	001160	36#												
\$REGO	001162	36#	883*	890*	983*	990*	1083*	1090*	1112*	1133*	1140*	1166*	1173*	1261*
		1269*	1299*	1306*	1388*	1485*	1536*	2992						
\$RTNAD	024502	2767#												
\$RZA =	***** U	2943												
\$SAVRE=	***** U	2943												
\$SAVR6	027054	2934#*												
\$SCOPE	025164	361	2923#											
\$SETUP=	000037	357#	361	2767	2923	2925								
\$SIZE	027124	380	2937#											
\$STUP =	177777	357#												
\$SVLAD	025370	2923#												
\$SVPC	000214	31#												
\$SWR	165400	5#	13	23#	25	36	361	399	412	425	439	457	483	505
		526	560	598	641	664	687	722	782	803	833	865	903	935
		966	1003	1035	1066	1103	1148	1186	1212	1244	1282	1320	1374	1415
		1472	1508	1568	1605	1633	1652	1677	1836	1860	1884	1911	1931	1951







TYPBIN	19#														
TYPDEC	19#	2767													
TYPNAM	19#														
TYPNUM	19#														
TYPOCS	19#														
TYPOCT	19#	2928													
TYPTXT	19#														
\$\$CMRE	36#														
\$\$CMTM	36#														
\$\$ESCA	19#														
\$\$NEWT	19#	399	412	425	439	457	483	505	526	560	598	641	664	687	722
	782	803	833	865	903	935	966	1003	1035	1066	1103	1148	1186	1212	1244
	1282	1320	1374	1415	1472	1508	1568	1605	1633	1652	1677	1836	1860	1884	1911
	1931	1951	1971	1991	2011	2031	2051	2077	2097	2117	2137	2157	2177	2197	2217
	2320	2409	2436	2472	2509	2566	2591	2622	2639	2659	2692	2707	2762		
\$\$SET	2943#														
\$\$SETM	361#														
\$\$SKIP	19#	475	501	522	554	592	635	777	799	897	997	1097	1180	1206	1232
	1276	1314	1352	1411	1452	1502	1561	1587	1596	1602	1624	1630	1648	1673	1784
	1851	1875	1880	1907	1927	1947	1968	1988	2008	2028	2048	2074	2094	2114	2134
	2154	2174	2194	2214	2268	2371	2426	2432	2453	2462	2468	2490	2499	2505	2531
	2539	2547	2583	2609	2615	2635	2653	2677	2685	2704	2748	2751	2753		
.EQUAT	7#	19													
.HEADE	7#	13													
.KT11	9#														
.SETUP	8#	357													
.SWRHI	9#	25													
.SWRLO	25#														
.\$ACT1	10#	31													
.\$APT8	10#	36#													
.\$APTH	10#	34													
.\$APTY	10#	2924													
.\$CATC	7#	27													
.\$CMTA	7#	36													
.\$EOP	7#	2767													
.\$ERRO	7#	2925													
.\$ERRT	9#	2928													
.\$MULT	9#	2941													
.\$PARM	8#														
.\$POWE	8#	2934													
.\$READ	8#														
.\$SAVE	8#														
.\$SCOP	8#	2923													
.\$SIZE	9#	2937													
.\$SPAC	8#														
.\$SWDO	8#														
.\$TRAP	8#	2943													
.\$TYPD	9#	2932													
.\$TYPE	7#	8#	2939												
.\$TYPO	7#	2931													

. ABS. 032532 000 CON RO ABS GBL I

ERRORS DETECTED: 0

CZVSBB -- VS60 INSTRUCTION TEST PART II MACY11 30G(1063) 17-SEP-79 08:46 PAGE 17-2  
CZVSBB.P11 10-SEP-79 11:32 CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0103

CZVSBB.CZVSBB/CRF=CZVSBB  
RUN-TIME: 33 20 1 SECONDS  
RUN-TIME RATIO: 132/55-2.3  
CORE USED: 26K (51 PAGES)