

DLV11-F

OFF LINE TEST
CNDVCA0

AH-T434A-MC
FICHE 1 OF 1

MAY 1983
COPYRIGHT © 82-83
MADE IN USA

DISO 20



8200
8201
8202
8203
8204
8205
8206
8207
8208
8209
8210
8211
8212
8213
8214
8215
8216
8217
8218
8219
8220
8221
8222
8223
8224
8225
8226
8227
8228
8229
8230
8231

.REM a

IDENTIFICATION

PRODUCT CODE: AC-T433A-MC
PRODUCT NAME: CNDVCA0 DLV11-F OFFLINE TEST
PRODUCT DATE: DECEMBER, 1982
AUTHOR: ODES CHOATE
MAINTAINER: DIAGNOSTIC SERVICES/ISS

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS DOCUMENT.

NO RESPONSIBILITY IS ASSUMED FOR THE USE OR RELIABILITY OF SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL OR ITS AFFILIATED COMPANIES.

COPYRIGHT (C) 1982,1983 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL	PDP	UNIBUS	MASSBUS
DEC	DECUS	DECTAPE	

8233
8234
8235
8236
8237
8238
8239
8240
8241
8242
8243
8244
8245
8246
8247
8248
8249
8250
8251
8252
8253
8254
8255
8256
8257
8258
8259
8260
8261
8262
8263
8264
8265
8266
8267
8268
8269
8270
8271
8272
8273
8274

NOTE: CVDVCB DIAGNOSTIC WAS MODIFIED TO WORK ON 11/21 PROCESSOR BY LOWERING
PRIORITY 7 TO 6 AND ALSO ADDING A CALL TO CNMAC2.SML TO INIT BRKVEC
AND LKVEL AND WAS RENAMED TO CNDVCA.

TABLE OF CONTENTS

1.0	GENERAL PROGRAM INFORMATION.
1.1	PROGRAM PURPOSE (ABSTRACT).
1.2	SYSTEM REQUIREMENTS.
1.3	RELATED DOCUMENTS AND STANDARDS.
1.4	DIAGNOSTIC HIERARCHY PREREQUISITES.
1.5	ASSUMPTIONS.
2.0	OPERATING INSTRUCTIONS.
2.1	LOADING AND STARTING PROCEDURES.
2.2	SPECIAL ENVIRONMENTS.
2.3	OPERATIONAL SWITCH SETTINGS
2.4	PROGRAM OPTIONS.
2.5	EXECUTION TIMES.
3.0	ERROR INFORMATION.
3.1	ERROR REPORTING PROCEDURE.
3.2	ERROR HALTS.
4.0	PERFORMANCE AND PROGRESS REPORTS.
4.1	PERFORMANCE REPORTS.
5.0	DEVICE INFORMATION TABLES.
6.0	SUMMARY OF TESTS AND SPECIAL SUBROUTINES

8276
8277
8278
8279
8280
8281
8282
8283
8284
8285
8286
8287
8288
8289
8290
8291
8292
8293
8294
8295
8296
8297
8298
8299
8300
8301
8302
8303
8304
8305
8306
8307
8308
8309
8310
8311
8312
8313
8314
8315
8316
8317
8318
8319
8320
8321
8322
8323
8324
8325
8326
8327
8328
8329
8330
8331

1.0 GENERAL PROGRAM INFORMATION.

1.1 PROGRAM PURPOSE (ABSTRACT).

THIS DIAGNOSTIC IS A LOGIC TEST TO VERIFY THE OPERATION OF THE DLV11-F SERIAL LINE INTERFACE. THE USER CAN SELECTIVELY ENABLE AND DISABLE TESTING OF THE OPTIONS BY ALTERING THE CONTENTS OF '\$USER'. THE DIAGNOSTIC IS DESIGNED TO TEST AND DETECT FAULTS TO THE LOGIC LEVEL (NOT TO THE CHIP LEVEL). THIS TEST OPERATES ON UP TO SIXTEEN(16) IDENTICALLY CONFIGURED DLV11-F SERIAL LINE INTERFACES. THE DEFAULT ADDRESSES ARE:

177560 -CONSOLE INTERFACE DEVICE ADDRESS
175610 -FIRST SERIAL LINE ADDRESS OF 15 CONSECUTIVE SERIAL LINE DEVICES.

60 - VECTOR FOR CONSOLE DEVICE INTERFACE.
300 - VECTOR FOR FIRST OF 15 DEVICES.

THIS PROGRAM IS DESIGNED TO RUN ON ANY PDP-11 WITH 4K OF MEMORY AND A DLV11-F (LSI-BUS) MODULE. IT CAN RUN UNDER XXDP, APT, AND ACT MONITORS, AND ON PROCESSORS WITH NO HARDWARE SWITCH REGISTER. A POWER FAILURE WILL CAUSE THE DIAGNOSTIC TO RESTART.

1.2 SYSTEM REQUIREMENTS.

HARDWARE REQUIREMENTS:

ANY PDP-11 FAMILY PROCESSOR
4K MEMORY - MINIMUM
A SPECIAL WRAP CONNECTOR OR EQUIVALENT (OPTIONAL)

SOFTWARE REQUIREMENTS:

THIS DIAGNOSTIC IS DESIGNED TO RUN IN ANY OF THE FOLLOWING WAYS:
STAND ALONE
WITH APT MONITOR
WITH ACT MONITOR
WITH XXDP MONITOR (CHAINABLE)

1.3 RELATED DOCUMENTS AND STANDARDS.

DIAGNOSTIC ENGINEERING STANDARDS AND CONVENTIONS
APT
ACT
SYSMAC

175-003-009-02
MD-11-DZZMA
AUTOCAT-11-QZAUB
MD-11-DZQAC

1.4 DIAGNOSTIC HIERARCHY PREREQUISITES.

8333
8334
8335
8336
8337
8338
8339
8340
8341
8342
8343
8344
8345
8346
8347
8348
8349
8350
8351
8352
8353
8354
8355
8356
8357
8358
8359
8360
8361
8362
8363
8364
8365
8366
8367
8368
8369
8370
8371
8372
8373
8374
8375
8376
8377
8378
8379
8380
8381
8382
8383
8384
8385
8386
8387
8388

NO SPECIAL DIAGNOSTICS ARE REQUIRED TO RUN BEFORE THIS, BUT THE PROCESSOR, MEMORY, AND BUS ARE ASSUMED TO BE FULLY OPERATIONAL.

1.5 ASSUMPTIONS.

THIS DIAGNOSTIC ASSUMES THAT THE OPERATOR HAS INITIALIZED LOCATION '\$USWR' AND '\$DEVN' TO THE PROPER VALUES.

2.0 OPERATING INSTRUCTIONS.

2.1 LOADING AND STARTING PROCEDURES.

USE STANDARD PROCEDURE FOR PDP-11 ABSOLUTE BINARY FORMATTED MEDIA.

THIS DIAGNOSTIC HAS ONLY ONE (1) STARTING ADDRESS. 200 FOR START AND RESTART.

THE USER CAN SELECT A SPECIFIC TEST TO BE EXECUTED BY SETTING SWITCH 8 IN THE SWITCH REGISTER AND THE TEST NUMBER (IN OCTAL) IN THE LOWER BYTE. (NOTE: ALL TESTS PREVIOUS TO THE SELECTED ONE ARE EXECUTED WITHOUT ITERATIONS.)

2.2 SPECIAL ENVIRONMENTS.

THIS DIAGNOSTIC FOLLOWS THE STANDARD PROCEDURE FOR RUNNING UDER APT,ACT,XXDP MONITORS, AS DESCRIBED IN THEIR RESPECTIVE PROCEDURES MANUAL AND SYSMAC PACKAGE.

2.3 OPERATIONAL SWITCH SETTINGS

IF THE DIAGNOSTIC IS RUN ON A CPU WITHOUT A SWITCH REGISTER THEN A SOFTWARE SWITCH REGISTER IS USED WHICH ALLOWS THE USER THE SAME SWITCH OPTIONS AS THE HARDWARE SWITCH REGISTER. IF THE HARDWARE SWITCH REGISTER DOES NOT EXIST OR IF ONE DOES AND IT CONTAINS ALL ONES (177777) THEN THE SOFTWARE SWITCH REGISTER (LOC. 176) IS USED.

CONTROL:

THIS PROGRAM ALSO SUPPORTS THE DYNAMIC LOADING OF THE SOFTWARE SWITCH REGISTER (LOC. 176) FROM THE TTY. THIS CAN BE ACCOMPLISHED BY DOING THE FOLLOWING:

- 1) TYPE CONTROL G <^G>; THIS WILL ALLOW THE TTY TO ENTER DATA INTO LOC. 176 AT SELECTED POINTS WITHIN THE

8390
8391
8392
8393
8394
8395
8396
8397
8398
8399
8400
8401
8402
8403
8404
8405
8406
8407
8408
8409
8410
8411
8412
8413
8414
8415
8416
8417
8418
8419
8420
8421
8422
8423
8424
8425
8426
8427
8428
8429
8430
8431
8432
8433
8434
8435
8436
8437
8438
8439
8440
8441
8442

PROGRAM.

- 2) THE MACHINE WILL THEN TYPE: ' SWR=XXXXXX NEW=' (XXXXXX IS THE OCTAL CONTENTS OF THE SOFTWARE SWITCH REGISTER.)
- 3) AFTER THE 'NEW=' HAS BEEN TYPED THEN THE OPERATOR CAN DO ONE OF THE FOLLOWING AT THE TTY:
 - A) TYPE A NUMBER TO BE LOADED INTO LOC. 176 FOLLOWED BY A <CR>. (ONLY NUMBERS BETWEEN 0-7 WILL BE ACCEPTED). LEADING ZEROS NEED NOT BE TYPED, AND IF MORE THAN 6 DIGITS ARE TYPED THE LAST 6 WILL BE USED. IF A <CR> IS THE FIRST KEY DEPRESSED THE SOFTWARE SWITCH REGISTER CONTENTS WILL NOT BE CHANGED.
 - B) IF A CONTROL U <^U> IS DEPRESSED THEN THE PROGRAM WILL SEND YOU BACK TO STEP 3.
 - C) IF THE INPUT CHARACTER IS NOT ONE OF THE CHARACTERS MENTIONED ABOVE THEN A QUESTION MARK (?) WILL BE TYPED FOLLOWED BY A CARRAGE RETURN AND A LINE FEED SEQUENCE THEN PROCEED FROM STEP 3 (ERASING ALL PREVIOUS INPUT).

DYNAMIC SWITCH REGISTER

- BIT 15 - HALT ON ERROR
- 14 - LOOP ON TEST
- 13 - INHIBIT ERROR TYPEOUTS
- 12 - (UNUSED)
- 11 - INHIBIT ITERATIONS
- 10 - BELL ON ERROR
- 9 - LOOP ON ERROR
- 8 - LOOP ON TEST IN SWR<7:0>
- 7:0 - TEST NUMBER TO LOOP ON (USED WITH BIT 8)

2.4 PROGRAM OPTIONS.

THIS PROGRAM WILL SUPPORT TESTING OF MULTIPLE DLV11-F'S. IT REQUIRES THE ADDRESS OF THE FIRST RCSR (STORED AT '\$BASE') AND ITS INTERRUPT VECTOR (STORED AT '\$VECT1'); AND WILL BE ABLE TO ADDRESS ANY DLV11-F STARTING AT THE SPECIFIED BASE ADDRESS UP TO 16 CONSECUTIVE DEVICES.

EXAMPLES: \$BASE: 175610
\$VECT1: 300

THE PROGRAM WILL BE ABLE TO TEST ANY DLV11-F WITHIN THE ADDRESS RANGE 175610 --> 176000

\$BASE AND \$VECT1 DEFAULT TO 175610 AND 300 RESPECTIVELY.

8444
8445
8446
8447
8448
8449
8450
8451
8452
8453
8454
8455
8456
8457
8458
8459
8460
8461
8462
8463
8464
8465
8466
8467
8468
8469
8470
8471
8472
8473
8474
8475
8476
8477
8478
8479
8480
8481
8482
8483
8484
8485
8486
8487

THE PROGRAM ASSOCIATES UNIT NUMBERS AS FOLLOWS: (NUMBERS IN PARENTHESIS ARE OCTAL)
UNIT#0 -- BASE ADDRESS STORED AT '\$BASE'
ASSOCIATED BASE VECTOR STORED AT '\$VECT1'
UNIT#1 -- BASE ADDRESS + (10)
BASE VECTOR + (10)

UP TO

UNIT#14 -- BASE ADDRESS + (160)
BASE VECTOR + (160)

LOCATION '\$DEVN' IS USED AS A BIT MAP TO INDICATE WHICH UNIT NUMBERS ARE PRESENT AND WILL BE TESTED.

BIT 15	BIT 14	-	-	-	BIT 1	BIT 0
! CON-	! UNIT	!	!	!	! UNIT	! UNIT
! SOLE	! 14	!	!	!	! #1	! #0

A BIT MAP CAN BE ENTERED AT '\$DEVN' PRIOR TO STARTING THE PROGRAM.

EXAMPLE:
\$BASE: 175610
\$VECTOR: 300
\$DEVN: 100013

THE PROGRAM WILL TEST-

UNIT#0	175610	300
UNIT#1	175620	310
UNIT#3	175640	330
CONSOLE	177560	60

OPTIONS

LOCATION \$USWR CONTAINS ALL THE USER SELECTABLE OPTIONS. THE VALUES IN THIS WORD MUST CONFORM TO THE ACTUAL BOARD CONFIGURATION.

8489
8490
8491
8492
8493
8494
8495
8496
8497
8498
8499
8500
8501
8502
8503
8504
8505
8506
8507
8508
8509
8510
8511
8512
8513
8514
8515
8516
8517
8518
8519
8520
8521
8522
8523
8524
8525
8526
8527
8528
8529
8530
8531
8532
8533
8534
8535
8536
8537
8538
8539
8540
8541
8542
8543
8544

THE DEFAULT VALUE OF \$USWR IS AS FOLLOWS:

BIT POSITION	DEFINITION	DEFAULT VALUE
0-3	# OF DATA BITS	10(8) = 8
4	PARITY ENABLED -\ (SEE	0 = NO
5	EVEN ODD PARITY- / NJTE)	0 = ODD
6	COMMON SPEED	1 = YES
7	PROGRAMMABLE BAUD RATE	0 = NO
8-11	BAUD RATE OFFSET (SEE FOLLOWING NOTE)	02(8) = 110 BAUD
12	BREAK GENERATION ENABLED	1 = YES
13	WRAP CONNECTOR INSTALLED	0 = NO
14	MAINT JUMPER (SEE NOTE)	0 = NO
15	ERROR BITS ENABLED	0 = NO

NOTE ON BITS <4:5>
 THIS DIAGNOSTIC DOES NOT TEST THE PARITY LOGIC.

NOTE ON BITS <7:11>
 WHEN THE PROGRAMMABLE BAUD RATE OPTION IS
 ENABLED THE PROGRAMMABLE BAUD RATE TEST
 WILL EXIT WITH THE BAUD RATE SET TO THE
 SELECTED VALUE. TO CHANGE THE DEFAULT
 VALUE OF 110 BAUD REPLACE BITS <11:8>
 WITH THE OFFSET INDICATED IN THE TABLE
 AT THE END OF THE PBR TEST. (TEST #16)

NOTE ON BIT 14
 THIS SWITCH WHEN ON WILL ALLOW THE DIAGNOSTIC
 TO TEST IN MAINTAINCE MODE. IT IS ASSUMED THAT
 THE MAINTAINCE JUMPER IS INSTALLED ON ALL OF
 THE DLV11-F MODULES WHEN THIS BIT IS SET.

DLV11-F INDIVIDUAL TEST REQUIREMENTS TABLE

TEST #	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17	20	21	22	23	24
CONSOLE DEVICE		++					**		++	++	++	++	++			++	++		++	++
APT ENVIRONMENT									++			++			++					
(MAINT) BIT SET			--				**			--	--	--	--	--	--	--	--	--	--	--
(WRAP CON) BIT SET							**													
(ERROR BITS) BIT SET													--							
(COM SPD) BIT SET																				
(BREAK) BIT SET		--																		--
(PROG BAUD RATE) BIT SET														--						

++ TEST WILL NOT RUN IF THIS CONDITION IS TRUE.
 -- TEST WILL NOT RUN IF THIS CONDITION IS FALSE.
 ** TEST WILL NOT RUN IF ALL OF THE CONDITIONS IN THIS COLUMN ARE FALSE.

8546
8547
8548
8549
8550
8551
8552
8553
8554
8555
8556
8557
8558
8559
8560
8561
8562
8563
8564
8565
8566
8567
8568
8569
8570
8571
8572
8573
8574
8575
8576
8577
8578
8579
8580
8581
8582
8583
8584
8585
8586
8587
8588
8589
8590
8591
8592
8593
8594
8595
8596
8597
8598
8599
8600
8601

2.5 EXECUTION TIMES.

EXECUTION TIMES ARE FOR AN LSI-11 PROCESSOR WITH ALL OPTIONS ENABLED ON THE DLV11-F (EXCEPT FOR PROGRAMMABLE BAUD RATE), AT 110 BAUD, AND NOT AT THE CONSOLE ADDRESS.

FIRST PASS- 90 SECONDS
ADDITIONAL PASSES 95 SECONDS
ADDITIONAL DEVICES 95 SECONDS

THE TEST TIME IS BAUD RATE DEPENDANT; HIGHER BAUD GIVES SHORTER PASS TIMES.

IF THE DIAGNOSTIC IS RUN AT THE CONSOLE ADDRESS THE RUNNING TIME IS 5 SECONDS PER PASS.

3.0 ERROR INFORMATION.

3.1 ERROR REPORTING PROCEDURE.

SINCE THIS DIAGNOSTIC WAS DESIGNED TO FIT IN 4-K OF MEMORY THE ERROR TYPEOUT IS VERY BRIEF. THE FORMAT OF THE ERROR TYPEOUT IS AS FOLLOWS:

TEST#____,ERROR#____,PC=____,ADDRESS=____,VECTOR=____

WHERE ALL VALUES TYPED ARE OCTAL.
THE ADDRESS AND VECTOR REFER TO THE FAILING SLU'S.
FOR FURTHER INFORMATION THE LISTING MUST BE CONSULTED.
BITS 15,13,10 AND 9 OF THE SWITCH REGISTER CONTROL THE SEQUENCE OF EVENTS AFTER AN ERROR IS CAUGHT.

BIT 15 - CAUSES THE PROGRAM TO HALT IN THE ERROR ROUTINE. CONTINUEING THE PROGRAM CAUSES IT TO PROCEED.

BIT 13 - DISABLES THE PRINTING OF THE ERROR MESSAGE.

BIT 10 - CAUSES THE BELL TO RING ON ERROR.

BIT 9 - CAUSES THE DIAGNOSTIC TO LOOP FROM BEGINNING OF TEST TO ERROR.

THE ERROR ROUTINE SUPPORTS THE CONTROL G FUNCTION.

3.2 ERROR HALTS.

THE ONLY HALT IN THIS DIAGNOSTIC IS IN THE ERROR ROUTINE, AND IS EXECUTED ONLY IF BIT 15 OF THE SWITCH REGISTER IS A ONE WHEN AN ERROR OCCURS.

8603
8604
8605
8606
8607
8608
8609
8610
8611
8612
8613
8614
8615
8616
8617
8618
8619
8620
8621
8622
8623
8624
8625
8626
8627
8628
8629
8630
8631
8632
8633
8634
8635
8636
8637
8638
8639
8640
8641
8642
8643
8644
8645
8646
8647
8648
8649
8650
8651
8652
8653
8654
8655

4.0 PERFORMANCE AND PROGRESS REPORTS.

4.1 PERFORMANCE REPORTS.

AS EACH DEVICE COMPLETES ONE PASS OF THE DIAGNOSTIC THE FOLLOWING WILL BE TYPED:
 CSR:-----,VECTOR:-----,ERRORS:-----

WHERE. 'CSR:-----' IS THE DEVICE CSR UNDER TEST
 'VECTOR:-----' IS THE ASSOCIATED VECTOR
 AND 'ERRORS:-----' IS THE TOTAL NUMBER OF ERRORS ON THIS DEVICE ON THIS PASS.

NOTE

THIS IS TYPED AFTER THE DEVICE HAS COMPLETED ITS PASS.

AFTER ALL DEVICES HAVE BEEN EXERCISED AN END PASS STATEMENT IS TYPED:

5.0 'ENDPASS#-----'
 DEVICE INFORMATION TABLES.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RCSR:					RCVR				RCVR	RCVR						RDR
					ACT				DONE	IE						ENB
RBUF:	ERRO	OR	FR	P												
	R	ERR	ERR	ERR												
TCSR:	PROGRAMMABLE BAUD				PBR				XMIT	XMIT				MAIN		BREA
	RATE SELECT				ENAB				RDY	IE				T		K
TBUF:																

NOTE

BLANK BOXES INDICATE UNUSED AND RESERVED BIT POSITIONS. SEE THE LISTING FOR AN EXPLANATION OF THE BITS.

8657
8658
8659
8660
8661
8662
8663
8654
8665
8666
8667
8668
8669
8670
8671
8672
8673
8674
8675
8676
8677
8678
8679
8680
8681
8682
8683
8684
8685
8686
8687
8688
8689
8690
8691
8692
8693
8694
8695
8696
8697
8698
8699
8700
8701
8702
8703
8704
8705
8706
8707
8708
8709
8710
8711

6.0 SUMMARY OF TESTS AND SPECIAL SUBROUTINES.

TEST 1 ADDRESSABILITY
---- -

THIS TEST VERIFIES THAT THE ADDRESS AS PLACED IN THE
HARDWARE P-TABLE TO BE CORRECT AND THE DLV11-F
RESPONDS TO THAT ADDRESS SPACE.

TEST 2 BREAK - TCSRO SET, CLEAR, RESET
---- -

TEST 3 MAINT - TCSR2 SET, CLEAR, RESET
---- -

TEST 4 XMITIE - TCSR6 SET, CLEAR, RESET
---- -

TEST 5 RCVRIE - RCSR6 SET, CLEAR, RESET
---- -

THE FOLLOWING 4 TESTS VERIFY THAT RESET (INIT) INITIALIZES
READ ONLY BITS.

TEST 6 RCVRDONE - RCSR 7 - IS CLEARED BY INIT
---- --

TEST 7 RCVRACT - RCSR 11 - 15 CLEARED BY INIT
---- -

TEST 10 XMITRDY - TCSR 7 - IS SET BY INIT
---- --

TEST 11 XMIT RDY - TCSR 7 - CLEARS WHEN TBUF IS LOADED
WITH A CHARACTER AND THAT IT SETS WITHIN A
REASONABLE AMOUNT OF TIME.

TEST 12 OUTPUTTING A CHAR FROM TBUF (WITH MAINT SET)
---- --
RESULTS IN RCVRDONE SETTING WITHIN A
REASONABLE AMOUNT OF TIME AND THAT RESET
CLEARS THE BIT.

8713
8714
8715
8716
8717
8718
8719
8720
8721
8722
8723
8724
8725
8726
8727
8728
8729
8730
8731
8732
8733
8734
8735
8736
8737
8738
8739
8740
8741
8742
8743
8744
8745
8746
8747
8748
8749
8750
8751
8752
8753
8754
8755
8756
8757
8758
8759
8760
8761
8762
8763
8764
8765
8766
8767
8768

TEST 13 RCVRDONE IS CLEARED BY READING RBUF
---- --

TEST 14 RCVRACT - RCSR 11 - SETS WHEN A START BIT IS
---- --
RECEIVED AND CLEARS WHEN RCVRDONE - RCSR 7 -
SETS

TEST 15 OVERRUN BIT - RBUF 14
---- --

TEST 16 PROGRAMMABLE BAUD RATE TEST TEST AT ALL SPEEDS
---- --
AVAILABLE A COMPARISON WILL BE MADE TO SEE IF
NEW TIME IS LESS THAN PREVIOUS.

TEST 17 TRANSMITTER INTERRUPT LOGIC TEST
---- --
LOGICALLY THIS IS 4 SEPARATE TESTS
A) DOES TRANSMITTER INTERRUPT LOGIC WORK
B) AT PRIORITY OF 0
C) AND ONLY ONCE
D) BUT NOT WITH INTERRUPT ENABLE CLEAR

TEST 20 RECEIVER INTERRUPT LOGIC TEST THIS TEST COVERS ALL
---- --
OF THE RECEIVER SIDE OF THE INTERRUPT LOGIC IN
CHARACTER MODE.

TEST 21 TEST ACTUAL DATA TRANSFERED NON-INTERRUPT
---- --
MAINTENANCE BIT SET

TEST 22 TEST DATA THROUGH WRAP
---- --

TEST 23 FULL DATA TRANSFER WITH INTERRUPTS AND MAINTENANCE
---- --
MODE.

TEST 24 TEST BREAK GENERATION LOGIC TRANSMIT KNOWN CHAR
---- --
WITH BREAK SET AND COMPARE RECEIVED WITH 0.

TEST 25 NOT A TEST - SEND BACK TO LOOP
---- --

MAINDEC-11-CNDVC-A MACY11 30(1046) 27-DEC-82 11:37 PAGE 151-1^{M 1}
CNDVCA.P11 27-DEC-82 11:36

SEQ 0012

8769

NOTE

FOR ALL OF THE FOLLOWING ROUTINES THE USE OF (R5) IS PART OF THE LINKAGE MECHANISM BETWEEN THE CALLER AND THE CALLED.

8771
8772
8773
8774
8775
8776
8777
8778
8779
8780
8781
8782
8783
8784
8785
8786
8787
8788
8789
8790
8791
8792
8793
8794
8795
8796
8797
8798
8799
8800
8801
8802
8803
8804
8805
8806
8807
8808
8809
8810
8811
8812
8813
8814
8815
8816
8817
8818
8819
8820
8821
8822
8823
8824
8825

ROUTINE:TIMER

THIS ROUTINE IS USED TO TEST THE STATUS OF ANY BIT IN ANY REGISTER.

INPUTS:

HOWLONG THE MAXIMUM AMOUNT OF TIME TO SPEND IN THIS ROUTINE.
WHICHBIT A MASK WITH THE BIT(S) SET THAT ARE TO BE CHECKED
REG A POINTER TO THE REGISTER TO BE CHECKED
SETCLR THE DESIRED RESULTS -- EITHER SET OR CLEAR

OUTPUT:

THE 'C' BIT IS SET TO INDICATE AN ERROR BUT IT IS TESTED BY THE IF.ERROR STATEMENT.

ROUTINE:DATLNG

THIS ROUTINE SETS UP A MASK FOR DATA, WITH -

INPUT:

NOTHING IS PASSED TO THIS ROUTINE BUT GLOBAL INFORMATION IS ASSUMED TO EXIST:
\$USWR-- THE WORD FOR SOFTWARE PARAMETERS
DATA-- A MASK FOR THE LOCATION OF THE OCTAL NUMBER OF DATA BITS

OUTPUT----

MASK-- A MASK OF BINARY ZEROS RIGHT-JUSTIFIED THE NUMBER OF WHICH IS DEFINED IN \$USWR WORD.

ROUTINE:WAIT

THIS ROUTINE IS USED TO DELAY EXECUTION OF THE MAIN PROGRAM FOR A SPECIFIED AMOUNT OF TIME. THIS IS ACCOMPLISHED BY INCREMENTING A REGISTER UP TO A LIMIT. THE INNER LOOP IS SET TO APPROXIMATE 1 MICRO SEC.

SERVICE ROUTINE: INTSRV

THIS GLOBAL ROUTINE DOES NOTHING BUT INCREMENT

8827
8828
8829
8830
8831
8832
8833
8834
8835
8836
8837
8838
8839
8840
8841

'INTFLAG' EACH TIME IT IS CALLED. IT ASSUMES
THAT THE MAIN CALLING ROUTINE WILL KNOW WHAT
TO LOOK FOR.

ROUTINE:CYCLE

THIS ROUTINE CAUSES ADRS TO POINT TO THE
ADDRESS OF DLV11-F UNDER TEST, ADRS +2 TO
POINT TO THE VECTOR OF THE DLV11-F UNDER TEST.
IT KEEPS TRACK OF THE CURRENT DEVICE AND BIT
MASKS.


```

(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) ;***** THE FOLLOWING BREAK VECTOR AND LINE CLOCK VECTOR ARE INCLUDED
(1) 000100 LKVEC= 100 ;:LINE CLOCK VECTOR
(1) 000140 BRKVEC= 140 ;:BREAK VECTOR
(1) 000240 PIRQVEC=240 ;:PROGRAM INTERRUPT REQUEST VECTOR

8849 000004 ILLMEM= 4
8850 000001 ADRS= R1
8851 000002 GOOD= R2
8852 000003 BAD= R3
8853 000001 REGISTER=R1
8854 000002 BIT= R2
8855 000003 FUNCT= R3
8856 000002 LEAD= R2
8857 000004 FOLLOW= R4
8858 175610 DLADDR= 175610
8859
8860 ; THE FOLLOWING DEFINITIONS APPLY TO THE GLOBAL SUBS
8861 177777 SET= -1
8862 000000 CLR= 0
8863
8864
8865 ;:*****
8866 ; RCSR REGISTER BIT NAMES
8867 ;:*****
8868 ; UNUSED BIT15
8869 ; UNUSED BIT14
8870 ; UNUSED BIT13
8671 ; UNUSED BIT12
8872 004000 RCVRACT= BIT11 ; RECEIVER ACTIVE INDICATOR
8873 ; UNUSED BIT10
8874 ; UNUSED BIT09
8875 ; UNUSED BIT08
8876 000200 RCVRDONE= BIT07 ; RECEIVER DONE
8877 000100 RCVRIE= BIT06 ; RECEIVER INTERRUPT ENABLE
8878 ; UNUSED BIT05
  
```

```
8879      : UNUSED      BIT04
8880      : UNUSED      BIT03
8881      : UNUSED      BIT02
8882      : UNUSED      BIT01
8883      000001      RLRRUN=      BIT00      : READER RUN
8884
8885      ;*****
8886      ; RBUF REGISTER BIT NAMES
8887      ;*****
8888      100000      ERROR=      BIT15      : ERROR INDICATOR
8889      040000      ORERR=      BIT14      : OVERRUN ERROR
8890      020000      FRERR=      BIT13      : FRAMING ERROR
8891      010000      PERR=      BIT12      : PARITY ERROR
8892      : UNUSED      BIT11
8893      : UNUSED      BIT10
8894      : UNUSED      BIT09
8895      : UNUSED      BIT08
8896      000200      RDATA7=      BIT07      : \
8897      000100      RDATA6=      BIT06      : |
8898      000040      RDATA5=      BIT05      : |
8899      000020      RDATA4=      BIT04      : | RECEIVED DATA BITS
8900      000010      RDATA3=      BIT03      : |
8901      000004      RDATA2=      BIT02      : |
8902      000002      RDATA1=      BIT01      : |
8903      000001      RDATA0=      BIT00      : /
8904
8905      ;*****
8906      ; TCSR REGISTER BIT NAMES
8907      ;*****
8908      100000      PBAUD3=      BIT15      : \
8909      040000      PBAUD2=      BIT14      : | PROGRAMMABLE BAUD
8910      020000      PBAUD1=      BIT13      : | RATE BITS
8911      010000      PBAUD0=      BIT12      : |
8912      004000      PBAUDSET=    BIT11      : | ENABLE SETTING OF
8913      : UNUSED      BIT10      : | PROGRAMMABLE BAUDE RATE
8914      : UNUSED      BIT09
8915      : UNUSED      BIT08
8916      000200      XMITRDY=    BIT07      : TRANSMITTER READY
8917      000100      XMITIE=      BIT06      : TRANSMITTER INTERRUPT ENABLE
8918      : UNUSED      BIT05
8919      : UNUSED      BIT04
8920      : UNUSED      BIT03
8921      000004      MAINT=      BIT02      : MAINTENANCE SET BIT
8922      : UNUSED      BIT01
8923      000001      BREAK=      BIT00      : SEND BREAK (CONTINUOUS SPACE)
8924
8925      ;*****
8926      ; TBUF REGISTER BIT NAMES
8927      ;*****
8928      : UNUSED      BIT15
8929      : UNUSED      BIT14
8930      : UNUSED      BIT13
8931      : UNUSED      BIT12
8932      : UNUSED      BIT11
```

8935
8936
8937
8938 000200
8939 000100
8940 000040
8941 000020
8942 000010
8943 000004
8944 000002
8945 000001
8946
8947
8948
8949
8950
8951 000017
8952 000020
8953 000040
8954 000100
8955 000200
8956
8957
8958
8959 007400
8960 010000
8961 020000
8962 040000
8963 100000
8964
8965
(1)
(1) 000000
(1)
(1)
(1)
(1)
(1) 000174
(1) 000174 000000
(1) 000176 000000
(1)
(1) 000200 000137 001336

```
: UNUSED BIT10
: UNUSED BIT09
: UNUSED BIT08
TDATA7= BIT07 : \
TLATA6= BIT06 : |
TDATA5= BIT05 : |
TDATA4= BIT04 : | \ TRANSMITTER DATA BUFFER
TDATA3= BIT03 : | /
TDATA2= BIT02 : |
TDATA1= BIT01 : |
TDATA0= BIT00 : /

;*****
; FLAG BITS TO BE USE OR CLEARED IN $USWR.

DATA = 17
PARITY = 20
EVENODD = 40
COMSPD = 100
PBR = 200

: BAUDE MUST BE ON THE UPPER
: BYTE BOUNDRY OF $USWR.--4 BITS
BAUD = 7400
BRK = 10000
WRAP = 20000
MAINTJUMP = 40000
ERRBITS = 100000

;*****
.SBTTL TRAP CATCHER

.=0
;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
.=174
DISPREG: .WORD 0 ;;SOFTWARE DISPLAY REGIS'ER
SWREG: .WORD 0 ;;SOFTWARE SWITCH REGISTER
.SBTTL STARTING ADDRESS(ES)
JMP @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
```

```

8967
8968      .SBTTL  ACT11 HOOKS
(1)
(2)      ;:*****
(1)      ;HOOKS REQUIRED BY ACT11
(1)      $SVPC=.          ;SAVE PC
(1)      .=46
(1) 000046 011404      $ENDAD      ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
(1)      .=52
(1) 000052 000000      .WORD 0      ;;2)SET LOC.52 TO ZERO
(1)      .=$SVPC          ;; RESTORE PC
8969      .=1000
8970      .SBTTL  APT PARAMETER BLOCK
(1)
(2)      ;:*****
(1)      ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
(2)      ;:*****
(1)      . $X=.          ;;SAVE CURRENT LOCATION
(1)      .=24          ;;SET POWER FAIL TO POINT TO START OF PROGRAM
(1) 000024 000200      200          ;;FOR APT START UP
(1)      .=44          ;;POINT TO APT INDIRECT ADDRESS PNTR.
(1) 000044 001000      $APTHDR  ;;POINT TO APT HEADER BLOCK
(1)      .=$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 001174      $MBADR: .WORD $MAIL  ;;ADDRESS OF APT MAILBOX (BITS 0-15)
(1) 001004 000005      $TSTM: .WORD 5      ;;RUN TIM OF LONGEST TEST
(1) 001006 000055      $PASTM: .WORD 45.    ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
(1) 001010 000036      $UNITM: .WORD 30.    ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
(1) 001012 000030      .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)

```

```

8971
(1)
(2)
(1)
(1)
(1)
(1)
(1) 001100          001100          .SBTTL COMMON TAGS
(1) 001100          000000          $CMTAG:          .=1100          ;;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 RE. 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) 001160          000000          $TIMES: 0          ;;MAX. NUMBER OF ITERATIONS
(1) 001162          000000          $ESCAPE: 0          ;;ESCAPE ON ERROR ADDRESS
(1) 001164          177607          $BELL: .ASCIZ <207><377><377>          ;;CODE FOR BELL
(1) 001170          077          $QUES: .ASCII /?/          ;;QUESTION MARK
(1) 001171          015          $CRLF: .ASCII <15>          ;;CARRIAGE RETURN
(1) 001172          000012          $LF: .ASCIZ <12>          ;;LINE FEED
(2)
(2)
(2)
(3)
(2)
(2) 001174          000000          .EVEN
(2) 001174          000000          $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) 001206          000000          $UNIT: .WORD AUNIT          ;;I/O UNIT NUMBER
    
```

000377

```

(2) 001210 000000 $MSGAD: .WORD  AMSGAD  ;;MESSAGE ADDRESS
(2) 001212 000000 $MSGLG: .WORD  AMSGLG  ;;MESSAGE LENGTH
(2) 001214 .      $ETABLE:      ;;APT ENVIRONMENT TABLE
(2) 001214 000    $ENV: .BYTE   AENV     ;;ENVIRONMENT BYTE
(2) 001215 000    $ENVM: .BYTE   AENVM
(2)           ;;ENVIRONMENT MODE BITS
(2) 001216 000000 $$WREG: .WORD  ASWREG  ;;APT SWITCH REGISTER
(2) 001220 011110 $USWR: .WORD  AUSWR   ;;USER SWITCHES
(2) 001222 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) 001224 000    $MAMS1: .BYTE  AMAMS1  ;;HIGH ADDRESS,M.S. BYTE
(2) 001225 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) 001226 000000 $MADR1: .WORD  AMADR1  ;;HIGH ADDRESS,BLK#1
(2)           *      MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF 'TYPE' ABOVE
(2) 001230 000    $MAMS2: .BYTE  AMAMS2  ;;HIGH ADDRESS,M.S. BYTE
(2) 001231 000    $MTYP2: .BYTE  AMTYP2  ;;MEM. TYPE,BLK#2
(2) 001232 000000 $MADR2: .WORD  AMADR2  ;;MEM.LAST ADDRESS,BLK#2
(2) 001234 000    $MAMS3: .BYTE  AMAMS3  ;;HIGH ADDRESS,M.S.BYTE
(2) 001235 000    $MTYP3: .BYTE  AMTYP3  ;;MEM. TYPE,BLK#3
(2) 001236 000000 $MADR3: .WORD  AMADR3  ;;MEM.LAST ADDRESS,BLK#3
(2) 001240 000    $MAMS4: .BYTE  AMAMS4  ;;HIGH ADDRESS,M.S.BYTE
(2) 001241 000    $MTYP4: .BYTE  AMTYP4  ;;MEM. TYPE,BLK#4
(2) 001242 000000 $MADR4: .WORD  AMADR4  ;;MEM.LAST ADDRESS,BLK#4
(2) 001244 000300 $VECT1: .WORD  AVECT1  ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
(2) 001246 000000 $VECT2: .WORD  AVECT2  ;;INTERRUPT VECTOR#2BUS PRIORITY#2
(2) 001250 175610 $BASE: .WORD  ABASE
(2)           ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
(2) 001252 000001 $DEVN: .WORD  ADEVN   ;;DEVICE MAP
(2) 001254 .MEXIT
  
```


8985

8986 001336

(1)
(1)
(1) 001336 012706 001100
(1) 001342 005026
(1) 001344 022706 001140
(1) 001350 001374
(1) 001352 012706 001100
(1)
(1) 001356 012737 013326 000020
(1) 001364 012737 000300 000022
(1) 001372 012737 013126 000030
(1) 001400 012737 000300 000032
(1)
(1) 001406 012737 014260 000034
(1) 001414 012737 000300 000036
(1) 001422 012737 011440 000024
(1) 001430 012737 000300 000026
(1) 001436 016767 007710 007700
(1) 001444 005067 177510
(1) 001450 005067 177506
(1) 001454 112767 000001 177433
(1) 001462 012767 001462 177416
(1) 001470 012767 001470 177412
(2)
(2)
(2) 001476 013746 000004
(2) 001502 012737 001536 000004
(2) 001510 012767 177570 177422
(2) 001516 012767 177570 177416
(2) 001524 022777 177777 177406
(2) 001532 001012
(2)
(2) 001534 000403
(2) 001536 012716 001544
(2) 001542 000002
(2) 001544 012767 000176 177366
(2) 001552 012767 000174 177362
(2) 001560 012637 000004
(1)
(2) 001564 005067 177412
(2) 001570 132767 000200 177417
(2) 001576 001403
(2) 001600 012767 001216 177332
(2) 001606
8987
(1)
(1) 001606 005227 177777
(1) 001612 001037
(1) 001614 022737 011404 000042
(1) 001622 001433
(1) 001624 104401 001672
(2)
(2) 001630 005737 000042
(2) 001634 001012

START:
.SBTTL INITIALIZE THE COMMON TAGS
::CLEAR THE COMMON TAGS (\$CMTAG) AREA
MOV #CMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
CLR (R6)+ ;;CLEAR MEMORY LOCATION
CMP #SWR,R6 ;;DONE?
BNE -6 ;;LOOP BACK IF NO
MOV #STACK,SP ;;SETUP THE STACK POINTER
::INITIALIZE A FEW VECTORS
MOV #SCOPE,@IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
MOV #PR6,@IOTVEC+2 ;;LEVEL 6
MOV #ERROR,@EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
MOV #PR6,@EMTVEC+2 ;;LEVEL 6
::BIT02
MOV #STRAP,@TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
MOV #PR6,@TRAPVEC+2;LEVEL 6
MOV #SPWRDN,@PWRVEC ;;POWER FAILURE VECTOR
MOV #PR6,@PWRVEC+2 ;;LEVEL 6
MOV \$ENDCT,\$EOPCT ;;SETUP END-OF-PROGRAM COUNTER
CLR \$TIMES ;;INITIALIZE NUMBER OF ITERATIONS
CLR \$ESCAPE ;;CLEAR THE ESCAPE ON ERROR ADDRESS
MOVB #1,\$ERMAX ;;ALLOW ONE ERROR PER TEST
MOV #,\$LPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
MOV #,\$LPERR ;;SETUP THE ERROR LOOP ADDRESS
::SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
::EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
MCV @ERRVEC,-(SP) ;;SAVE ERROR VECTOR
MOV #64\$,@ERRVEC ;;SET UP ERROR VECTOR
MOV #DSWR,SWR ;;SETUP FOR A HARDWARE SWICH REGISTER
MOV #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
CMP #-1,@SWR ;;TRY TO REFERENCE HARDWARE SWR
BNE 66\$;;BRANCH IF NO TIMEOUT TRAP OCCURRED
;;AND THE HARDWARE SWR IS NOT = -1
BR 65\$;;BRANCH IF NO TIMEOUT
64\$: MOV #65\$,(SP) ;;SET UP FOR TRAP RETURN
RTI
65\$: MOV #SWREG,SWR ;;POINT TO SOFTWARE SWR
MOV #DISPREG,DISPLAY
66\$: MOV (SP)+,@ERRVEC ;;RESTORE ERROR VECTOR
CLR \$PASS ;;CLEAR PASS COUNT
BITB #APTSIZE,\$ENVM ;;TEST USER SIZE UNDER APT
BEQ 67\$;;YES,USE NON-APT SWITCH
MOV #SWREG,SWR ;;NO,USE APT SWITCH REGISTER
67\$:
.SBTTL TYPE PROGRAM NAME
::TYPE THE NAME OF THE PROGRAM IF FIRST PASS
INC #-1 ;;FIRST TIME?
BNE 68\$;;BRANCH IF NO
CMP #ENDAD,@#42 ;;ACT-11?
BEQ 68\$;;BRANCH IF YES
TYPE ,69\$;;TYPE ASCIZ STRING
.SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
TST @#42 ;;ARE WE RUNNING UNDER XXDP/ACT?
BNE 70\$;;BRANCH IF YES

MAINDEC-11-CNDVC-A
CNDVCA.P11 27-DEC-82

MACY11 30(1046)
11:36

27-DEC-82 11:37 PAGE 156-1
GET VALUE FOR SOFTWARE SWITCH REGISTER

SEQ 0025

(2)	001636	126727	177352	000001		CMPB	\$ENV,#1	::ARE WE RUNNING UNDER APT?
(2)	001644	001406				BEQ	70\$::BRANCH IF YES
(2)	001646	026727	177266	000176		CMP	SWR,#SWREG	::SOFTWARE SWITCH REG SELECTED?
(2)	001654	001005				BNE	71\$::BRANCH IF NO
(2)	001656	104406				GTSWR		::GET SOFT-SWR SETTINGS
(2)	001660	000403				BR	71\$	
(2)	001662	112767	000001	177244	70\$:	MOVB	#1,\$AUTOB	::SET AUTO-MODE INDICATOR
(2)	001670				71\$:			
(1)	001670	000410				BR	68\$::GET OVER THE ASCIZ
(1)					::69\$:	.ASCIZ	<CRLF>*MD-11-CNDVC-A*<CRLF>	
(1)	001712				68\$:			

```

8989
8994 001712                                WHILE $DEVN EQ #0 DO
(4) 001712                                $1:
(6) 001712 005767 177334                    TST    $DEVN
(9) 001716 001101                            BNE    $2
8995 001720                                TYPTXT <<CRLF>! I HAVE NO DEVICE TO TEST.!>
8996 001762                                TYPTXT <<CRLF>! SET UP $DEVN TO INDICATE ACTUAL ONFIGURATION.!>
8997 002050                                TYPTXT <<CRLF>! TYPE PROCEED (P) TO CONTINUE.!>
8998 002116 000000                            HALT
8999 002120                                ENDDO
(4) 002120 000674                            BR     $1
(3) 002122                                $2:
9000 002122                                LET INITFLAG := #1
(4) 002122 012767 000001 006754                MOV    #1,INITFLAG
9001 002130                                LET BITMASK := #BIT15 ; START AT CONSOLE
(4) 002130 012767 100000 006744                MOV    #BIT15,BITMASK
9002 002136                                LOOP:
9003 002136                                CALL   CYCLE ; NO ARGUMENTS--ADDRS -> NEXT ADDRESS
(3) 002136 004767 006520                            JSR    PC,CYCLE
9004
9005
9006 002142                                ; ADDR+2 -> NEXT VECTOR
(4) 002142 012167 177106                            MOV    (ADRS)+,DLADD ; GET UNIT ADDRESS
9007
9008 002146                                ; GET UNIT VECTOR
(4) 002146 011167 177104                            MOV    (ADRS),DLVEC DLVEC := (ADRS)
9009 002152                                LET    ADRS := DLADD
(4) 002152 016701 177076                            MCV    DLADD,ADRS
9010
9011 002156                                ;RCSR = DLADD + 0
(4) 002156 016767 177072 177074                            MOV    DLADD,RCSR LET    RCSR := DLADD
9012 002164                                LET    RBUF := DLADD + #2
(4) 002164 016767 177064 177070                            MOV    DLADD,RBUF
(7) 002172 062767 000002 177062                            ADD    #2,RBUF
9013 002200                                LET    TCSR := DLADD + #4
(4) 002200 016767 177050 177056                            MOV    DLADD,TCSR
(7) 002206 062767 000004 177050                            ADD    #4,TCSR
9014 002214                                LET    TCSRHI := DLADD + #5
(4) 002214 016767 177034 177044                            MOV    DLADD,TCSRHI
(7) 002222 062767 000005 177036                            ADD    #5,TCSRHI
9015 002230                                LET    TBUF := DLADD + #6
(4) 002230 016767 177020 177032                            MOV    DLADD,TBUF
(7) 002236 062767 000006 177024                            ADD    #6,TBUF
9016 002244                                LET    R5 := #R5STACK
(4) 002244 012705 001334                            MOV    #R5STACK,R5
9021
(1) 002250 000005                                RESET ;:ERESET

```

```

9032
9033      ;*****
(3)      ;*TEST 1      ADDRESSABILITY
(4)      ;*          THIS TEST VERIFIES THAT THE ADDRESS AS PLACED IN
(4)      ;*          THE HARDWARE P-TABLE TO BE CORRECT AND THE DLV11-F RESPONDS
(4)      ;*          TO THAT ADDRESS SPACE
(3)      ;*****
(2) 002252 000004      TST1:  SCOPE
(2)
(1) 002254 012767 000002 176676      MOV      #2,$TIMES      ;;DO 2 ITERATIONS
(2) 002262 012767 000001 176710      MOV      #1,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
9038 002270      LET      ADRS := DLADD
(4) 002270 016701 176760      MOV      DLADD,ADRS
9039
9040      SETVEC      ; SET UP INTERRUPT
(5) 002274 010146      MOV      R1,-(SP)      #ILLMEM,#INTSRV,#PR6
(5) 002276 012701 000004      MOV      #ILLMEM,R1
(5) 002302 012721 010652      MOV      #INTSRV,(R1)+
(5) 002306 012711 000300      MOV      #PR6,(R1)
(5) 002312 012601      MOV      (SP)+,R1
9041 002314      LET      I := #0
(4) 002314 005067 176752      CLR      I
9042 002320      REPEAT
(3) 002320      $3:
9043 002320      BGNSUB
(5) 002320 012767 002326 176562      MOV      #64$,$LPERR
9044      ;CLEAR FLAG
9045 002326      LET INTFLAG := #0
(4) 002326 005067 006326      CLR      INTFLAG
9046
9047      ;READ FLAG
9048 002332 005711      TST @ADRS
9049 002334      IF INTFLAG NE #0 THEN
(6) 002334 005767 006320      TST      INTFLAG
(9) 002340 001401      BEQ      $4
9050      ; FATAL ERROR
9051 002342      ERRDF 1,,NODL
(1) 002342 104001      ERROR 1
9052 002344      ENDIF
(4) 002344      $4:
9053 002344      ENDSUB
9054 002344      LET      I := I + #2
(7) 002344 062767 000002 176720      ADD      #2,I
9055 002352      LET      ADRS := DLADD + I
(4) 002352 016701 176676      MOV      DLADD,ADRS
(7) 002356 066701 176710      ADD      I,ADRS
9056 002362      UNTIL I EQ #8.
(3) 002362 026727 176704 000010      CMP      I,#8.
(6) 002370 001353      BNE      $3
9057 002372      CLRVEC ILLMEM
(3) 002372 010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
(3) 002374 010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
(5) 002376 012701 000004      MOV      #ILLMEM,R1
(5) 002402 010102      MOV      R1,R2
(8) 002404 062702 000002      ADD      #2,R2
(5) 002410 010221      MOV      R2,(R1)+

```

(5) 002412 005011
(3) 002414 012602
(3) 002416 012601
9058
9059 002420

CLR (R1)
MOV (SP)+,R2
MOV (SP)+,R1

:::POP STACK INTO R2
:::POP STACK INTO R1
;END OF TEST
ENDTST

```
9061
9066
9067
9068
9069
9070
9078
(3)
(4)
(4)
(4)
(4)
(4)
(3)
(2) 002420 000004
(2)
(1) 002422 012767 000010 176530
(2) 002430 012767 000002 176542
9079
9084 002436
(6) 002436 032767 010000 176554
(8) 002444 001404
(6) 002446 032767 000001 176540
(9) 002454 001404
(6) 002456
9085 002456
(5) 002456 012767 000001 176474
(3) 002464 000452
9086 002466
(4) 002466
9087
9088
9089 002466
(5) 002466 012767 002474 176414
9090
9091 002474
(6) 002474 032777 000001 176562
(9) 002502 001401
9092
9093 002504
(1) 002504 104002
9094 002506
(4) 002506
9095 002506
9096
9097
9098 002506
(5) 002506 012767 002514 176374
9099 002514
(7) 002514 052777 000001 176542
9100
9101 002522
(6) 002522 032777 000001 176534
(9) 002530 001001
9102
9103 002532

*****
* THE FOLLOWING 8 TESTS TEST ALL 'READ WRITE' BITS
*****

*****
*TEST 2 BREAK - TCSRO SET, CLEAR, RESET
* THE BREAK BIT IS USUALLY USED ON THE CONSOLE
* DEVICE. IF ADDITIONAL DLV OPTIONS ARE USED
* IT IS RECOMMENDED TO REMOVE THE 'BG' JUMPER AND
* CLEAR BIT 12 IN $USWR WHICH WILL CAUSE THIS
* TEST TO BE SKIPPED.
*****

TST2: SCOPE
MOV #10,$TIMES ;:DO 10 ITERATIONS
MOV #2,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
IF #BRK NOTSETIN $USWR OR #APTENV SETIN $ENV THE
BIT #BRK,$USWR
BEQ $5
BIT #APTENV,$ENV
BEQ $6
$5: EXIT TEST ; BREAK NOT INSTALLED
MOV #1,$TIMES
BR TST3 ;:EXIT THIS TEST
ENDIF
$6: ; SEE IF IT IS CLEAR
BGNSUB
MOV #64,$$LPERR
IF #BREAK SETIN @TCSR THEN
BIT #BREAK,@TCSR
BEQ $7
; BREAK DID NOT RESET IN TCSR
ERRHRD 2,,DIDNOT
$7: ERROR 2
ENDIF
ENDSUB
; TRY TO SET BREAK BIT
BGNSUB
MOV #64,$$LPERR
LET @TCSR := @TCSR SET.BY #BREAK
BIS #BREAK,@TCSR
; STUCK TO 0
IF #BREAK NOTSETIN @TCSR THEN
; BREAK DID NOT SET IN TCSR
ERRHRD 3,,DIDNOT
```

```

(1) 002532 104003          ERROR 3
9104 002534
(4) 002534          $10:
9105 002534
9106
9107
          ; TRY TO CLEAR A SET BIT
          BGNSUB
9108 002534
(5) 002534 012767 002542 176346      MOV #64$, $LPERR
9109
9110 002542
(7) 002542 042777 000001 176514      BIC #BREAK, @TCSR
          LET @TCSR := @TCSR CLR.BY #BREAK
9111
          ; SHOULD HAVE CLEARED
          IF #BREAK SETIN @TCSR THEN
9112 002550
(6) 002550 032777 000001 176506      BIT #BREAK, @TCSR
(9) 002556 001401
          BEQ $11
          ; BREAK DID NOT CLEAR IN TCSR
          ERRHRD 4,,DIDNOT
9113
9114 002560
(1) 002560 104004          ERROR 4
9115 002562
(4) 002562          $11:
9116 002562
          ENDSUB
9117
          ; NOW SEE IF RESET CLEARS IT
          BGNSUB
9118
9119 002562
(5) 002562 012767 002570 176320      MOV #64$, $LPERR
9120
9121 002570
(7) 002570 052777 000001 176466      BIS #BREAK, @TCSR
          LET @TCSR := @TCSR SET.BY #BREAK
9122
          ; ISSUE BUS RESET
          BRESÉT
9123 002576
(1) 002576 000005          RESET
9124 002600
(6) 002600 032777 000001 176456      BIT #BREAK, @TCSR
(9) 002606 001401
          BEQ $12
          ; BREAK DID NOT RESET IN TCSR
          ERRHRD 5,,DIDNOT
9125
9126 002610
(1) 002610 104005          ERROR 5
9127 002612
(4) 002612          $12:
9128 002612
          ENDSUB
9129 002612
          ENDTST
9130
9131
  
```

```
9133  
9138  
9139  
  (3)  
  (3)  
  (2) 002612 000004  
  (2)  
  (1) 002614 012767 000010 176336  
  (2) 002622 012767 000003 176350  
9144  
9145 002630  
  (6) 002630 032767 040000 176362  
  (8) 002636 001404  
  (6) 002640 126727 006255 000001  
  (9) 002646 001004  
  (6) 002650  
9146 002650  
  (5) 002650 012767 000001 176302  
  (3) 002656 000452  
9147 002660  
  (4) 002660  
9148  
9149  
9150 002660  
  (5) 002660 012767 002666 176222  
9151  
9152 002666  
  (6) 002666 032777 000004 176370  
  (9) 002674 001401  
9153  
9154 002676  
  (1) 002676 104006  
9155 002700  
  (4) 002700  
9156 002700  
9157  
9158  
9159 002700  
  (5) 002700 012767 002706 176202  
9160 002706  
  (7) 002706 052777 000004 176350  
9161  
9162 002714  
  (6) 002714 032777 000004 176342  
  (9) 002722 001001  
9163  
9164 002724  
  (1) 002724 104007  
9165 002726  
  (4) 002726  
9166 002726  
9167  
9168  
9169 002726  
  (5) 002726 012767 002734 176154  
9170
```

```
*****  
*****  
: *TEST 3 MAINT - TCSR2 SET, CLEAR, RESET  
*****  
TST3: SCOPE  
MOV #10,$TIMES ;;DO 10 ITERATIONS  
MOV #3,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX  
IF #MAINTJUMP NOTSETIN $USWR ORB CONSOLE EQ #TRU  
BIT #MAINTJUMP,$USWR  
BEQ $13  
CMPB CONSOLE,#TRUE  
BNE $14  
$13: EXIT TEST  
MOV #1,$TIMES  
BR TST4 ;;EXIT THIS TEST  
$14: ENDF  
; SEE IF IT IS CLEAR  
BGNSUB  
MOV #64,$SLPERR  
IF #MAINT SETIN @TCSR THEN  
BIT #MAINT,@TCSR  
BEQ $15  
; MAINT DID NOT RESET IN TCSR  
ERRHRD 6,,DIDNOT  
$15: ENDF  
ENDSUB  
; TRY TO SET MAINT BIT  
BGNSUB  
MOV #64,$SLPERR  
LET @TCSR := @TCSR SET.BY #MAINT  
BIS #MAINT,@TCSR  
; STUCK TO 0  
IF #MAINT NOTSETIN @TCSR THEN  
BIT #MAINT,@TCSR  
BNE $16  
; MAINT DID NOT SET IN TCSR  
ERRHRD 7,,DIDNOT  
$16: ENDF  
ENDSUB  
; TRY TO CLEAR A SET BIT  
BGNSUB  
MOV #64,$SLPERR
```



```

9171 002734          LET @TCSR := @TCSR CLR.BY #MAINT
(7) 002734 042777 000004 176322      BIC #MAINT,@TCSR
9172                                     ; SHOULD HAVE CLEARED
9173 002742          IF #MAINT SETIN @TCSR THEN
(6) 002742 032777 000004 176314      BIT #MAINT,@TCSR
(9) 002750 001401      BEQ $17
9174                                     ; MAINT DID NOT CLEAR INTCSR
9175 002752          ERRHRD 10,,DIDNOT
(1) 002752 104010      ERROR 10
9176 002754          ENDIF
(4) 002754          $17:
9177 002754          ENDSUB
9178                                     ; NOW SEE IF RESET CLEARS IT
9179                                     BGNSUB
9180 002754          MOV #64$,$LPERR
(5) 002754 012767 002762 176126
9181
9182 002762          LET @TCSR := @TCSR SET.BY #MAINT
(7) 002762 052777 000004 176274      BIS #MAINT,@TCSR
9183                                     ; ISSUE BUS RESET
9184 002770          BRESÉT
(1) 002770 000005      RESET
9185 002772          IF #MAINT SETIN @TCSR THEN
(6) 002772 032777 000004 176264      BIT #MAINT,@TCSR
(9) 003000 001401      BEQ $20
9186                                     ; MAINT DID NOT RESET IN TCSR
9187 003002          ERRHRD 11,,DIDNOT
(1) 003002 104011      ERROR 11
9188 003004          ENDIF
(4) 003004          $20:
9189 003004          ENDSUB
9190 003004          ENDTST
9191
9192
9193
  
```

```

9195
9200
9201
(3)
(3)
(2) 003004 000004
(2)
(1) 003006 012767 000010 176144 MOV #10,$TIMES ;;DO 10 ITERATIONS
(2) 003014 012767 000004 176156 MOV #4,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
;; USE PRIORITY OF 6
9202
9207 003022 012746 000300 MOV #PR6,-(SP) ;;PUT NEW PS ON STACK
(1) 003026 012746 003034 MOV #64$,-(SP) ;;PUT NEW PC ON STACK
(1) 003032 000002 RTI ;;POP NEW PC AND PS
(1) 003034
64$:
; SEE IF IT IS CLEAR
; BGNSUB
9212
9217
9218 003034
(5) 003034 012767 003042 176046 MOV #65$,$LPERR
9219
9220 003042 IF #XMITIE SETIN @TCSR THEN
(6) 003042 032777 000100 176214 BIT #XMITIE,@TCSR
(9) 003050 001401 BEQ $21
; XMITIE DID NOT RESET IN TCSR
ERRHRD 12,,DIDNOT
9221
9222 003052
(1) 003052 104012 ERROR 12
9223 003054
(4) 003054 $21: ENDF
9224 003054 ENDSUB
9225
9226 ; TRY TO SET XMITIE BIT
; BGNSUB
9227 003054
(5) 003054 012767 003062 176026 MOV #64$,$LPERR
9228 003062 LET @TCSR := @TCSR SET.BY #XMITIE
(7) 003062 052777 000100 176174 BIS #XMITIE,@TCSR
9229 ; STUCK TO 0
9230 003070 IF #XMITIE NOTSETIN @TCSR THEN
(6) 003070 032777 000100 176166 BIT #XMITIE,@TCSR
(9) 003076 001001 BNE $22
; XMIT DID NOT RESET IN TCSR
ERRHRD 13,,DIDNOT
9231
9232 003100
(1) 003100 104013 ERROR 13
9233 003102
(4) 003102 $22: ENDF
9234 003102 ENDSUB
9235
9236 ; TRY TO CLEAR A SET BIT
; BGNSUB
9237 003102
(5) 003102 012767 003110 176000 MOV #64$,$LPERR
9238
9239 003110 LET @TCSR := @TCSR CLR.BY #XMITIE
(7) 003110 042777 000100 176146 BIC #XMITIE,@TCSR
9240 ; SHOULD HAVE CLEARED
9241 003116 IF #XMITIE SETIN @TCSR THEN
(6) 003116 032777 000100 176140 BIT #XMITIE,@TCSR
(9) 003124 001401 BEQ $23
9242 ; XMIT DID NOT CLEAR IN TCSR

```

```
9243 003126 ERRHRD 14,,DIDNOT
(1) 003126 104014 ERROR 14
9244 003130 ENDIF
(4) 003130 $23:
9245 003130 ENDSUB
9246
9247 ; NOW SEE IF RESET CLEARS IT
9248 003130 BGNSUB
(5) 003130 012767 003136 175752 MOV #64$, $LPERR
9249
9250 003136 LET @TCSR := @TCSR SET.BY #XMITIE
(7) 003136 052777 000100 176120 BIS #XMITIE, @TCSR
9251 ; ISSUE BUS RESET
9252 003144 BRESÉT
(1) 003144 000005 RESET
9253 003146 IF #XMITIE SET IN @TCSR THEN
(6) 003146 032777 000100 176110 BIT #XMITIE, @TCSR
(9) 003154 001401 BEQ $24
9254 ; XMIT DID NOT RESET IN TCSR
9255 003156 ERRHRD 15,,DIDNOT
(1) 003156 104015 ERROR 15
9256 003160 ENDIF
(4) 003160 $24:
9257 003160 ENDSUB
9258 003160 ENDTST
9259
9260
9261
```

```

9263
9268
9271
(3)
(3)
(2) 003160 000004
(2)
(1) 003162 012767 000010 175770
(2) 003170 012767 000005 176002
9276
9277 003176
(5) 003176 012767 003204 175704
9278
9279 003204
(6) 003204 032777 000100 176046
(9) 003212 001401
9280
9281 003214
(1) 003214 104035
9282 003216
(4) 003216
9283 003216
9284
9285
9286 003216
(5) 003216 012767 003224 175664
9287 003224
(7) 003224 052777 000100 176026
9288
9289 003232
(6) 003232 032777 000100 176020
(9) 003240 001001
9290
9291 003242
(1) 003242 104036
9292 003244
(4) 003244
9293 003244
9294
9295
9296 003244
(5) 003244 012767 003252 175636
9297
9298 003252
(7) 03252 042777 000100 176000
9299
9300 003260
(6) 003260 032777 000100 175772
(9) 003266 001401
9301
9302 003270
(1) 003270 104037
9303 003272
(4) 003272
9304 003272
9305
  
```

```

*****
*****
*TEST 5          RCVRIE - RCSR6 SET, CLEAR, RESET
*****
TSTS:  SCOPE
      MOV #10,$TIMES      ;;DO 10 ITERATIONS
      MOV #5,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
                          ; SEE IF IT IS CLEAR
                          BGNSUB
      MOV #64,$LPERR
      IF #RCVRIE SETIN @RCSR THEN
      BIT #RCVRIE,@RCSR
      BEQ $25
                          ; RCVRIE DID NOT RESET IN RCSR
                          ERRHRD 35,,DIDNOT
      ERROR 35
      ENDIF
      $25:
      ENDSUB
                          ; TRY TO SET RCVRIE BIT
                          BGNSUB
      MOV #64,$LPERR
      LET @RCSR := @RCSR SET.BY #RCVRIE
      BIS #RCVRIE,@RCSR
                          ; STUCK TO 0
      IF #RCVRIE NOTSETIN @RCSR THEN
      BIT #RCVRIE,@RCSR
      BNE $26
                          ; RCVRIE DID NOT SET IN RCSR
                          ERRHRD 36,,DIDNOT
      ERROR 36
      ENDIF
      $26:
      ENDSUB
                          ; TRY TO CLEAR A SET BIT
                          BGNSUB
      MOV #64,$LPERR
      LET @RCSR := @RCSR CLR.BY #RCVRIE
      BIC #RCVRIE,@RCSR
                          ; SHOULD HAVE CLEARED
      IF #RCVRIE SETIN @RCSR THEN
      BIT #RCVRIE,@RCSR
      BEQ $27
                          ; RCVRIE DID NOT CLEAR IN RCSR
                          ERRHRD 37,,DIDNOT
      ERROR 37
      ENDIF
      $27:
      ENDSUB
  
```

K 3

; NOW SEE IF RESET CLEARS IT
BGNSUB

```
9306  
9307 003272  
  (5) 003272 012767 003300 175610      MOV    #64$, $LPERR  
9308  
9309 003300  
  (7) 003300 052777 000100 175752      BIS    #RCVRIE, @RCSR      LET    @RCSR := @RCSR SET.BY #RCVRIE  
9310  
9311 003306  
  (1) 003306 000005      RESET      ; ISSUE BUS RESET  
9312 003310  
  (6) 003310 032777 000100 175742      BIT    #RCVRIE, @RCSR      BRESÉT  
  (9) 003316 001401      BEQ    $30      IF    #RCVRIE SETIN @RCSR THEN  
9313  
9314 003320  
  (1) 003320 104040      ERROR  40      ; RCVRIE DID NOT RESET IN RCSR  
9315 003322  
  (4) 003322      $30:      ERRHRD 40,,DIDNOT  
9316 003322  
9317 003322      ENDIF  
9318 003322      CKLOOP  
9319  
9320  
9321  
9322      ENDSUB  
      ENDTST
```

9324
9329
9330
9331
9332
9333
(3)
(3)
(2)
(2)
(1)
(2)
9334
9339
9340
9341
9342
(5)
9343
(6)
(9)
9344
9345
9346
9347
(1)
9348
9349
(1)
9350
(4)
9351
9352
9353
9354
9355
9356
9357

003322 000004

003324 012767 000010 175626
003332 012767 000006 175640

003340 012767 003346 175542
003346 032777 000200 175704
003354 001402

003356 104041

003360 000005

003362

003362
003362
003362

* THE FOLLOWING 4 TESTS VERIFY
* THAT RESET (INIT) INITIALIZES READ ONLY BITS.

*TEST 6 TEST THAT RCVRDONE - RCSR 7 - IS CLEARED BY INIT

TST6: SCOPE

MOV #10,\$TIMES ;;DO 10 ITERATIONS
MOV #6,\$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

BGNSUB
MOV #6,\$, \$LPERR IF #RCVRDONE SET IN @RCSR THEN
BIT #RCVRDONE,@RCSR
BEQ \$31

;RCVRDONE SHOULD HAVE CLEARED BY INIT
; RCVRDONE DID NOT CLEAR IN RCSR
ERRHRD 41,HRESET, DIDNOT

ERROR 41

;REISSUE RESET
BRESET

RESET

ENDIF

\$31:

;ALLOW LOOPING AFTER ERROR

CKLOOP

ENDSUB

ENDTST


```
9395
9396 ;TESTING EFFECT OF RESET ON BIT
9397
9398 ;RCVRACT DID NOT CLEAR IN RCSR
9399
9400 ;ALLOW ANOTHER TRY
9401 003474 BRESET
(1) 003474 000005 RESET
9402 003476 ENDIF
(4) 003476 $40:
9403 ;ALLOW LOOPING ON ERROR
9404 003476 CKLOOP
9405 003476 ENDSUB
9406 003476 ENDTST
9407
```



```

9442
9447
9452
(3)
(4)
(4)
(3)
(2) 003536 000004
(2)
(1) 003540 012767 000001 175412      MOV #1,$TIMES      ;;DO 1 ITERATION
(2) 003546 012767 000011 175424      MOV #11,$TESTN    ;;SET TEST NUMBER IN APT MAIL BOX
9457
9458 003554                                IFB CONSOLE EQ #TRUE OR #APTENV SETIN $ENV THEN
(6) 003554 126727 005341 000001      CMPB CONSOLE,#TRUE
(8) 003562 001404                        BEQ $42
(6) 003564 032767 000001 175422      BIT #APTENV,$ENV
(9) 003572 001404                        BEQ $43
(6) 003574                                $42:
9459 003574                                EXIT TEST
(5) 003574 012767 000001 175356      MOV #1,$TIMES
(3) 003602 000513                        BR TST12          ;;EXIT THIS TEST
9460 003604                                $43:
(4) 003604                                ENDIF
9461
9462 003604                                LET PASS := #1 ;INIT COUNT OF TIMES THRU
(4) 003604 012767 000001 000212      MOV #1,PASS
9463 003612                                LOOP              ; START OF LOOP
(3) 003612                                $44:
9464
9465 003612                                ; MAX OF 2 TIMES THRU
(4) 003612 012767 000000 000206      MOV #CLR,ERRORFLAG LET ERRORFLAG := #CLR
9466 003620                                LET EXITFLAG := #CLR
(4) 003620 012767 000000 000202      MOV #CLR,EXITFLAG
9467
9468
9469
9470
9471
9472
9473 003626                                ;SEND A CHARACTER
(4) 003626 105077 175436      CLRB @TBUF        LET @TBUF :B= #0
9474
9475
9476
9477 003632                                ;WAIT A MAXIMUM
(3) 003632 010546                                ;OF 500 MSEC FOR
(7) 003634 012745 177777      MOV R5,-(SP)      CALL TIMER IN <#500,#XMITRDY,TCSR,#SET>
(6) 003640 016745 175420      MOV #SET,-(R5)
(5) 003644 012745 000200      MOV TCSR,-(R5)
(4) 003650 012745 000500      MOV #XMITRDY,-(R5)
(3) 003654 004767 004434      MOV #500,-(P5)
(3) 003660 012605      JSR PC,TIMER
9478
9479
9480 003662                                ;TIMER RETURNS AN ERROR IF BIT DID
(6) 003662 103001      MOV (SP)+,R5     ;NOT MEET CONDITION WITHIN TIME LIMIT
                                IF.ERROR THEN

```

```
9481                                     ;XMIT RDY DID NOT SET IN TCSR
9482 003664                               ERRHRD 66,,DIDNOT
  (1) 003664 104066                       ERROR 66
9483 003666                               ENDIF
  (4) 003666                               $46:
9484
9485                                     ; LOAD TBUF WITH A SECOND CHARACTER
9486                                     ; CHECK IMMEDIATELY THAT XMITRDY IS CLEAR
9487                                     ; AND THEN WAIT FOR IT TO SET
9488
9489                                     ;SEND SECOND CHARACTER
9490 003666                               LET @TBUF :B= #0
  (4) 003666 105077 175376                 CLRB @TBUF
9491 003672 000240                         NOP
9492
9493                                     ; GIVE IT TIME TO CLEAR
9494 003674                                     ; XMITRDY SHOULD HAVE CLEARED UPON
  (6) 003674 032777 000200 175362         BIT #XMITRDY,@TCSR
  (9) 003702 001404                         BEQ $47
9495                                     ; XMITRDY DID NOT CLEAR IN TCSR
9496 003704                               LET ERRORFLAG := #SET
  (4) 003704 012767 177777 000114         MOV #SET,ERRORFLAG
9497
9498                                     ; DEFER ERROR TYPEOUT
9499 003712                                     ELSE
  (4) 003712 000416                         BR $50
  (3) 003714                               $47:
9500
9501                                     ;WAIT A MAXIMUM
9502                                     ;OF 500 MSEC FOR
9503 003714                                     ;XMIT RDY TO SET IN TCSR
  (3) 003714 010546                         MOV R5, -(SP)
  (7) 003716 012745 177777                 MOV #SET, -(R5)
  (6) 003722 016745 175336                 MOV TCSR, -(R5)
  (5) 003726 012745 000200                 MOV #XMITRDY, -(R5)
  (4) 003732 012745 000500                 MOV #500, -(R5)
  (3) 003736 004767 004352                 JSR PC,TIMER
  (3) 003742 012605                         MOV (SP)+,R5
9504 003744                                     IF.ERROR THEN
  (6) 003744 103001                         BCC $51
9505
9506 003746                                     ;XMIT RDY DID NOT SET IN TCSR
  (1) 003746 104070                         ERRHRD 70,,DIDNOT
9507 003750                               ENDIF
  (4) 003750                               $51:
9508 003750                               ENDIF ; OF DEFERED ERROR CALL
  (4) 003750                               $50:
9509 003750                                     IF ERRORFLAG EQ #SET THEN
  (6) 003750 026727 000052 177777         CMP ERRORFLAG,#SET
  (9) 003756 001011                         BNE $52
9510 003760                                     IF PASS GT #1 THEN
  (6) 003760 026727 000040 000001         CMP PASS,#1
  (9) 003766 003404                         BLE $53
9511
9512 003770                                     ; CALL ERROR IF 2ND TRY
  (1) 003770 104067                         ERROR 67
                                     ERRHRD 67,,DIDNOT
```

```

9513 00377-          LET EXITFLAG := #SET
(4) 00377 012767 177777 000030      MOV  #SET,EXITFLAG
9514 00400u          ENDIF
(4) 004000          $53:
9515 004000          ELSE          ; NO ERROR
(4) 004000 000403      BR  $54
(3) 004002          $52:
9516 004002          LET EXITFLAG := #SET
(4) 004002 012767 177777 000020      MOV  #SET,EXITFLAG
9517 004010          ENDIF
(4) 004010          $54:
9518 004010          EXIF          EXITFLAG EQ #SET
(3) 004010 026727 000014 177777      CMP  EXITFLAG,#SET
(5) 004016 001401      BEQ  $45
9519 004020          ENDLOOP
(4) 004020 000674      BR  $44
(3) 004022          $45:
9520 004022          EXIT ; SKIP AROUND FLAG WORDS
(3) 004022 000403      BR  TST12          :::EXIT THIS TEST
9521 004024 000000      PASS:          0
9522 004026 000000      ERRORFLAG:      0
9523 004030 000000      EXITFLAG:          C
9524 004032          ENDTST
  
```

```

9526
9531
9536
(3)
(4)
(4)
(3)
(2) 004032 000004
(2)
(1) 004034 012767 000001 175116      MOV #1,$TIMES      ;;DO 1 ITERATION
(2) 004042 012767 000012 175130      MOV #12,$TESTN    ;;SET TEST NUMBER IN APT MAIL BOX
9541
9542 004050
(6) 004050 032767 040000 175142      BIT #MAINTJUMP,$USWR      IF #MAINTJUMP NOTSETIN $USWR ORB CONSOLE EQ #TRU
(8) 004056 001404
(6) 004060 126727 005035 000001      BEQ $S5
(9) 004066 001004
(6) 004070
9543 004070
(5) 004070 012767 000001 175062      MOV #1,$TIMES
(3) 004076 000442
9544 004100
(4) 004100
9545
9546
9547 004100
(7) 004100 052777 000004 175156      BIS #MAINT,@TCSR      ; SET THE MAINTENANCE BIT
9548
9549 004106
(5) 004106 012767 004114 174774      MOV #64,$SLPERR      LET @TCSR := @TCSR SET.BY #MAINT
9550
9551
9552 004114
(4) 004114 105077 175150      CLRB @TBUF      BGNSUB
9553
9554
9555
9556
9557 004120
(3) 004120 010546
(7) 004122 012745 177777
(6) 004126 016745 175126
(5) 004132 012745 000200
(4) 004136 012745 000500
(3) 004142 004767 004146
(3) 004146 012605
9558
9559
9560 004150
(6) 004150 103004
9561
9562
9563 004152
(7) 004152 042777 000004 175104      BIC #MAINT,@TCSR      ; DIDN'T SET IN TIME
9564 004160
(1) 004160 104071
ERROR 71      IF.ERROR THEN
; RCVRDONE DID NOT SET IN RCSR
; CAN NOT LEAVE WITH MAINT SET
LET @TCSR := @TCSR CLR.BY #MAINT
ERRHRD 71,,DIDNOT

```

```

9565 004162
(4) 004162
9566
9567 004162
9568
9569 004162
(5) 004162 012767 004170 174720
9570 MOV #64$, $LPERR
9571 ; NOW THAT IT IS SET SEE IF IT CAN BE RESET
9572 ; THIS ALSO WILL CLEAR THE MAINT. BIT
(1) 004170 000005 RESET BRESET
9573
9574 004172
(6) 004172 032777 000200 175060 BIT #RCVRDONE, @RCSR THEN
(9) 004200 001401 BEQ $60
9575 ; RCVRDONE DID NOT RESET IN RCSR.
9576 004202 104072 ERROR 72 ERRHRD 72,,DIDNOT
(1) 004202
9577 004204
(4) 004204
9578 004204
9579 004204
  
```

\$57:

\$60:

ENDSUB
 ENDTST

```
9581
9586
9587
(3)
(3)
(2) 004204 000004
(2)
(1) 004206 012767 000010 174744
(2) 004214 012767 000013 174756
9588
9593 004222
(6) 004222 032767 040000 174770
(8) 004230 001404
(6) 004232 126727 004663 000001
(9) 004240 001004
(6) 004242
9594 004242
(5) 004242 012767 000001 174710
(3) 004250 000440
9595 004252
(4) 004252
9596
9597
9598 004252
(7) 004252 052777 000004 175004
9599 004260
(5) 004260 012767 004266 174622
9600
9601
9602
9603
9604 004266
(4) 004266 105077 174776
9605
9606
9607
9608 004272
(3) 004272 010546
(7) 004274 012745 177777
(6) 004300 016745 174754
(5) 004304 012745 000200
(4) 004310 012745 000500
(3) 004314 004767 003774
(3) 004320 012605
9609 004322
(7) 004322 042777 000004 174734
9610
9611 004330
(6) 004330 103001
9612
9613 004332
(1) 004332 104073
9614 004334
(4) 004334
9615 004334
9616

*****
*****
*TEST 13 TEST THAT RCVRDONE IS CLEARED BY READING RBUF
*****
TST13: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #13,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
IF #MAINTJUMP NOTSETIN $USWR ORB CONSOLE EG #TRU
BIT #MAINTJUMP,$USWR
BEQ $61
CMPB CONSOLE,#TRUE
BNE $62
$61:
EXIT TEST
MOV #1,$TIMES
BR TST14 ;;EXIT THIS TEST
$62:
ENDIF
; SET MAINT. BIT
LET @TCSR := @TCSR SET.BY #MAINT
BGNSUB
MOV #64,$SLPERR
; OUTPUT A CHARACTER WITH MAINTENANCE
; SET, AND WAIT FOR XMITRDY TO SET.
; OUTPUT A CHARACTER
LET @TBUF :B= #0
; WAIT MAXIMUM OF 500 MSEC
; FOR RCVRDONE TO SET IN
; RCSR
CALL TIMER IN <#500,#RCVRDONE,RCSR,#SET>
LET @TCSR := @TCSR CLR.BY #MAINT
; DID IT BECAME READY?
IF.ERROR THEN
;RCVRDONE DID NOT SET IN RCSR
ERRHRD 73,, DIDNOT
ENDIF
$63:
ENDSUB
```

```

9617                                     : NOW THAT IT IS SET LETS SEE IF READING THE
9618                                     ; BUFFER CLEARS RCVRDONE.
9619
9620                                     ;READ BUFFER
9621 004334                               LET R0 :B= @RBUF
  (4) 004334 117700 174722             MOVB @RBUF,R0
9622
9623 004340                               IF #RCVRDONE SETIN @RCSR THEN
  (6) 004340 032777 000200 174712     BIT #RCVRDONE,@RCSR
  (9) 004346 001401                     BEQ $64
9624                                     ;RCVRDONE DID NOT CLEAR IN RCSR
9625 004350                               ERRHRD 74,,DIDNOT
  (1) 004350 104074                     ERROR 74
9626 004352                               ENDIF
  (4) 004352                               $64:
9627 004352                               ENDTST
  
```



```

9529
9634
9639
(3)
(4)
(4)
(3)
(2) 004352 000004
(2)
(1) 004354 012767 000010 174576
(2) 004362 012767 000014 174610
9644 004370
(6) 004370 126727 004525 000001
(8) 004376 001404
(6) 004400 032767 040000 174612
(9) 004406 001404
(6) 004410
9645 004410
(5) 004410 012767 000001 174542
(3) 004416 000526
9646 004420
(4) 004420
9647 004420
(6) 004420 032767 000001 174566
(9) 004426 001404
9648 004430
(5) 004430 012767 000001 174522
(3) 004436 000516
9649 004440
(4) 004440
9650
9651
9652
9653 004440
(7) 004440 052777 000004 174616
9654 004446
(4) 004446 012700 000000
9655 004452
(4) 004452 005001
9656
9657
9658
9659
9660 004454
(4) 004454 105077 174610
9661 004460
(3) 004460
9662 004460
(6) 004460 032777 004000 174572
(9) 004466 001403
9663 004470
(4) 004470 012700 177777
9664 004474
(4) 004474 000401
(3) 004476
9665 004476

```

```

*****
*****
TEST 14 TEST THAT RCVRACT - RCSR 11 - SETS
WHEN A START BIT IS RECEIVED AND
CLEARS WHEN RCVRDONE - RCSR 7 - SETS
*****
TST14: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #14,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
IFB CONSOLE EQ #TRUE OR #MAINTJUMP NOTSETIN $USW
CMPB CONSOLE,#TRUE
BEQ $65
BIT #MAINTJUMP,$USWR
BNE $66
$65:
EXIT TEST
MOV #1,$TIMES
BR TST15 ;;EXIT THIS TEST
ENDIF
$66:
IF #APTENV SETIN $ENV THEN
EXIT TEST
MOV #1,$TIMES
BR TST15 ;;EXIT THIS TEST
ENDIF
$67:
LET @TCSR := @TCSR SET.BY #MAINT
LET R0 := #CLR
LET R1 := #0
;LOAD A CHARACTER INTO TBUF
;WAIT FOR RCVRACT TO SET
;SEND A CHARACTER
LET @TBUF :B= #0
REPEAT
IF #RCVRACT SETIN @RCSR THEN
LET R0 := #SET
ELSE
LET R1 := R1 + #1

```

```
(7) 004476 005201          INC      R1
9666 004500
(4) 004500          $72:
9667 004500
(4) 004500 020027 177777    CMP      R0,#SET
(6) 004504 001403          BEQ      $73
(4) 004506 020167 000160    CMP      R1,MAX
(7) 004512 101762          BLOS     $70
(4) 004514          $73:
9668 004514
(6) 004514 020167 000152    CMP      R1,MAX
(9) 004520 101410          BLOS     $74
9669
9670
9671
9672 004522
(7) 004522 042777 000004 174534  BIC      #MAINT,@TCSR
9673 004530
(1) 004530 104075          ERROR    75
9674 004532
(5) 004532 012767 000001 174420  MOV      #1,$TIMES
(3) 004540 000455          BR       TST15          :::EXIT THIS TEST
9675 004542          $74:
(4) 004542
9676
9677
9678
9679
9680
9681
9682 004542          $75:
(4) 004542
(6) 004542 032777 004000 174510  BIT      #RCVRACT,@RCSR
(9) 004550 001421          BEQ      $76
9683
9684 004552
(6) 004552 032777 000200 174500  BIT      #RCVRDONE,@RCSR
(9) 004560 001414          BEQ      $77
9685 004562
(6) 004562 032777 004000 174470  BIT      #RCVRACT,@RCSR
(9) 004570 001410          BEQ      $100
9686
9687
9688
9689 004572
(7) 004572 042777 000004 174464  BIC      #MAINT,@TCSR
9690 004600
(1) 004600 104076          ERROR    76
9691
9692 004602
(5) 004602 012767 000001 174350  MOV      #1,$TIMES
(3) 004610 000431          BR       TST15          :::EXIT THIS TEST
9693 004612          $100:
(4) 004612          $77:
9694 004612
(4) 004612

;IT NEVER SET
;RCVRACT DID NOT SET IN RCSR.
; CAN NOT LEAVE WITH MAINT SET
LET @TCSR := @TCSR CLR.BY #MAINT
ERRHRD 75,, DIDNOT
EXIT TEST
:::EXIT THIS TEST
ENDIF
;CHECK FOR TIMING OF RCVRACT. CLEARING
;VS RCVRDONE SETTING
WHILE #RCVRACT SETIN @RCSR DO
IF #RCVRDONE SETIN @RCSR THEN
IF #RCVRACT SETIN @RCSR THEN
;RCVRDONE AND RCVRACT
;BOTH SET
; CAN NOT LEAVE WITH MAINT SET
LET @TCSR := @TCSR CLR.BY #MAINT
ERRHRD 76, DONEACT
;NO USE CONTINUING
EXIT TST
ENDIF
ENDIF
```

```

9695 004612          ENDDO
(4) 004612 000753
(3) 004614          $76: BR      $75
9696
9697
9698 004614          ;RCVRCT = 0 NOW.
(6) 004614 032777 000200 174436 BIT    #RCVRDONE,@RCSR
(9) 004622 001010 BNE    $101
9699
9700
9701 004624          ;RCVRDONE DID NOT SET IN RCSR
(7) 004624 042777 000004 174432 BIC    #MAINT,@TCSR
9702 004632          ; CAN NOT LEAVE WITH MAINT SET
(1) 004632 104077 ERROR  77      LET    @TCSR := @TCSR CLR.BY #MAINT
9703 004634          ERRHRD 77,,DIDNOT
(5) 004634 012767 000001 174316 MOV    #1,$TIMES
(3) 004642 000414 BR      TST15      ;::EXIT THIS TEST
9704 004644          ENDIF
(4) 004644          $101:
9705
9706          ;TEST THAT READING THE RECEIVER
9707          ;BUFFER CLEARS RCVRDONE
9708
9709
9710 004644          ;READ CHAR.
(4) 004644 017700 174412 MOV    @RBUF,RO    LET RO := @RBUF
9711
9712 004650          IF #RCVRDONE SETIN @RCSR THEN
(6) 004650 032777 000200 174402 BIT    #RCVRDONE,@RCSR
(9) 004656 001404 BEQ    $102
9713
9714          ;RCVRDONE DID NOT CLEAR IN RCSR
9715          ; CAN NOT LEAVE WITH MAINT SET
9716 004660          LET    @TCSR := @TCSR CLR.BY #MAINT
(7) 004660 042777 000004 174376 BIC    #MAINT,@TCSR
9717 004666          ERRHRD 100,,DIDNOT
(1) 004666 104100 ERROR  100
9718
9719          ENDIF
(3) 004670          $102:
9720 004670 000401 BR      TST15      EXIT
9721 004672 070000 MAX:70000 ;::EXIT THIS TEST
9722 004674          ENDTST
9723
  
```

```

9725
9730
9731
9732
(3)
(3)
(2) 004674 000004
(2)
(1) 004676 012767 000010 174254
(2) 004704 012767 000015 174266
9737
9738 004712
(6) 004712 032767 100000 174300
(8) 004720 001404
(6) 004722 126727 004173 000001
(9) 004730 001004
(6) 004732
9739 004732
(5) 004732 012767 000001 174220
(3) 004740 000547
9740 004742
(4) 004742
9741 004742
(6) 004742 032767 040000 174250
(9) 004750 001004
9742 004752
(5) 004752 012767 000001 174200
(3) 004760 000537
9743 004762
(4) 004762
9744
9745 004762
(7) 004762 052777 000004 174274
9746
9747
9748
9749 004770
(5) 004770 012767 004776 174112
9750
9751
9752
9753
9754
9755 004776
(4) 004776 105077 174266
9756
9757 005002
(4) 005002 010546
(5) 005004 012745 000310
(4) 005010 004767 003556
(4) 005014 012605
9758
9759
9760 005016
(4) 005016 105077 174246
9761

```

```

*****
*****
*TEST 15 TEST THE OVERRUN BIT - RBUF 14
*****
TST15: SCOPE
MOV #10,$TIMES ::DO 10 ITERATIONS
MOV #15,$TESTN ::SET TEST NUMBER IN APT MAIL BOX
IF #ERRBITS NOTSETIN $USWR ORB CONSOLE EQ #TRUE
BIT #ERRBITS,$USWR
BEQ $103
CMPB CONSOLE,#TRUE
BNE $104
$103:
EXIT TEST
MOV #1,$TIMES
BR TST16 :::EXIT THIS TEST
ENDIF
$104:
IF #MAINTJUMP NOTSETIN $USWR THEN
EXIT TEST
MOV #1,$TIMES
BR TST16 :::EXIT THIS TEST
ENDIF
$105:
LET @TCSR := @TCSR SET BY #MAINT
BGNSUB
MOV #64,$SLPERR
:OUTPUT 2 CHARACTERS WITH
:AMPLE DELAYS BETWEEN FOR RECEPTION.
:THIS SHOULD AN CAUSE OVERRUN ERROR.
:OUTPUT 1 CHARACTER
LET @TBUF :B= #0
:GO AWAY FOR 200 M SEC
WAITMS 200.
MOV R5,-(SP)
MOV #200,-(R5)
JSR PC,WAIT
MOV (SP)+,R5
:OUTPUT 2ND CHARACTER
LET @TBUF :B= #0
:LET OVERRUN HAPPEN

```

```

9762 005022                                WAITMS 200.
(4) 005022 010546                          MOV R5, -(SP)
(5) 005024 012745 000310                   MOV #200, -(R5)
(4) 005030 004767 003536                   JSR PC, WAIT
(4) 005034 012605                          MOV (SP)+, R5
9763
9764                                        ;READ BUFFER AND ERROR BITS
9765 005036                                LET R4 := @RBUF
(4) 005036 017704 174220                   MOV @RBUF, R4
9766
9767                                        ;IT DIDN'T SET
9768 005042                                IF #ORERR NOTSET IN R4 THEN
(6) 005042 032704 040000                   BIT #ORERR, R4
(9) 005046 001010                          BNE $106
9769                                        ;ORERR DID NOT SET IN RBUF
9770                                        ; CAN NOT LEAVE WITH MAINT SET
9771 005050                                LET @TCSR := @TCSR CLR.BY #MAINT
(7) 005050 042777 000004 174206           BIC #MAINT, @TCSR
9772 005056                                ERRHRD 101,,DIDNOT
(1) 005056 104101                          ERROR 101
9773
9774                                        ;NO USE COMPOUNDING ERRORS
9775 005060                                EXIT TST
(5) 005060 012767 000001 174072           MOV #1, $TIMES
(3) 005066 000474                          BR TST16                                :::EXIT THIS TEST
9776 005070                                ENDIF
(4) 005070                                $106:
9777 005070                                ENDSUB
9778
9779                                        ;NOW SEE IF ERROR BIT SET WITH OVERRUN ERROR:
9780 005070                                BGNSUB
(5) 005070 012767 005076 174012           MOV #64$, $LPERR
9781 005076                                IF #ERROR NOTSET IN R4 THEN
(6) 005076 032704 100000                   BIT #ERROR, R4
(9) 005102 001010                          BNE $107
9782
9783                                        ;ERROR DID NOT SET IN RBUF
9784                                        ; CAN NOT LEAVE WITH MAINT SET
9785 005104                                LET @TCSR := @TCSR CLR.BY #MAINT
(7) 005104 042777 000004 174152           BIC #MAINT, @TCSR
9786 005112                                ERRHRD 102,,DIDNOT
(1) 005112 104102                          ERROR 102
9787
9788                                        ;-WHEN ORERR SET.
9789                                        ;GET OUT NOW.
9790                                        EXIT TST
9791 005114                                ;
(5) 005114 012767 000001 174036           MOV #1, $TIMES
(3) 005122 000456                          BR TST16                                :::EXIT THIS TEST
9792 005124                                ENDIF
(4) 005124                                $107:
9793                                        ENDSUB
9794                                        BGNSUB
(5) 005124 012767 005132 173756           MOV #64$, $LPERR
9795                                        ;CHECK REAL RBUF TO SEE IF ORERR IS STILL SET.
9796

```


(7) 005250 042777 000004 174006
9831 005256
(1) 005256 104105
9832
9833 005260
(4) 005260
9834 005260
9835 005260
9836

BIC #MAINT,@TCSR
ERROR 105

ERRHRD 105,,DIDNOT

\$112:

ENDIF
ENDSUB
ENDTST

9838
 9839
 9844
 9850
 (3)
 (4)
 (4)
 (4)
 (3)
 (2) 005260 000004
 (2)
 (1) 005262 012767 000010 173670
 (2) 005270 012767 000016 173702
 9855
 9856
 9857
 9858 005276
 (6) 005276 032767 000200 173714
 (8) 005304 001404
 (6) 005306 032767 040000 173704
 (9) 005314 001004
 (6) 005316
 9859 005316
 (5) 005316 012767 000001 173634
 (3) 005324 000553
 9860 005326
 (4) 005326
 9861
 9862 005326
 (6) 005326 132767 000001 173660
 (9) 005334 001404
 9863 005336
 (5) 005336 012767 000001 173614
 (3) 005344 000543
 9864 005346
 (4) 005346
 9865
 9866 005346
 (4) 005346 005067 002614
 9867 005352
 (4) 005352 012767 177777 000270
 9868 005360
 (4) 005360 012767 177777 000264
 9869 005366
 (7) 005366 052777 000004 173670
 9870
 9871 005374
 (4) 005374 005003
 (5) 005376 000401
 (4) 005400
 (8) 005400 005203
 (5) 005402
 (5) 005402 020327 000017
 (7) 005406 003062
 9872 005410
 (4) 005410 017700 173646

```

*****
*****
*TEST 16      PROGRAMMABLE BAUD RATE TEST
*             TEST AT ALL SPEEDS AVAILABLE
*             A COMPARISON WILL BE MADE TO SEE
*             IF NEW TIME IS LESS THAN PREVIOUS.
*****
  
```

```

TST16: SCOPE
      MOV      #10,$TIMES      ;;DO 10 ITERATIONS
      MOV      #16,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX

      IF #PBR NOTSETIN $USWR OR #MAINTJUMP NOTSETIN $U
      BIT      #PBR,$USWR
      BEQ      $113
      BIT      #MAINTJUMP,$USWR
      BNE      $114
$113:      EXIT TEST
      MOV      #1,$TIMES
      BR       TST17          ;;:EXIT THIS TEST
                                ENDIF
$114:
      IFB #APTENV SETIN $ENV THEN
      BITB     #APTENV,$ENV
      BEQ      $115
      EXIT TST
      MOV      #1,$TIMES
      BR       TST17          ;;:EXIT THIS TEST
                                ENDIF
$115:
      LET ERRCHK := #0      ; CLEAR ERROR WORD
      LET OLD := #-1
      LET OLD+2 := #-1
      LET @TCSR := @TCSR SET.BY #MAINT
      ;EACH BAUD RATE
      INCR R3 FROM #0 TO #15. BY #1

      CLR      R3
      BR       $116
$117:
      INC      R3
$116:
      CMP      R3,#15.
      BGT      $120
      LET R0 := @RBUF
      MOV      @RBUF,R0
  
```



```

9873
9874 005414
(4) 005414 116377 005624 173644      MOVB   RATES(R3),@TCSRHI
9875
9876 005422
(4) 005422 005002                      CLR    BIT
9877
9878 005424
(4) 005424 005077 173640              CLR    @TBUF
9879
9880 005430
(4) 005430 005067 000210              CLR    NEW
9881 005434
(4) 005434 005067 000206              CLR    NEW+2
9882 005440
(4) 005440                               $121:
(6) 005440 005702                      TST    BIT
(9) 005442 001014                      BNE    $122
9883 005444
(6) 005444 032777 000200 173606      BIT    #RCVRDONE,@RCSR
(9) 005452 001403                      BEQ    $123
9884
9885 005454
(4) 005454 012702 000001              MOV    #1,BIT
9886 005460
(4) 005460 000404                      BR     $124
(3) 005462                               $123:
9887
9888 005462
(7) 005462 005267 000156              INC    NEW
9889 005466
(7) 005466 005567 000154              ADC    NEW+2
9890 005472
(4) 005472                               $124:
9891
9892 005472
(4) 005472 000762                      BR     $121
(3) 005474                               $122:
9893
9894 005474
(6) 005474 026767 000146 000150      CMP    NEW+2,OLD+2
(9) 005502 103001                      BHIS   $125
9895
9896 005504
(4) 005504 000414                      BR     $126
(3) 005506                               $125:
9897
9898 005506
(6) 005506 026767 000134 000136      CMP    NEW+2,OLD+2
(9) 005514 001005                      BNE    $127
(6) 005516 026767 000122 000124      CMP    NEW,OLD
(9) 005524 103001                      BHIS   $127
9899
9900 005526
(4) 005526 000403                      BR     $130
(3) 005530                               $127:
  
```

```

;CHANGE BAUDE RATE
LET @TCSRHI :=RATES(R3)
;FLAG
LET BIT := #0
;OUTPUT THE CHARACTER
LET @TBUF := #0
;INITIALIZE COUNTER
LET NEW := #0
LET NEW+2 := #0
WHILE BIT EQ #0 DO
  IF #RCVRDONE SETIN @RCSR THEN
    ;DONE - ITS READY
    LET BIT := #1
  ELSE
    ;OTHERWISE-INCREMENT TIME
    LET NEW := NEW + #1
    LET NEW+2 := NEW+2 + CARRY
  ENDIF
;SIGNALS DONE
ENDDO
IF NEW+2 LO OLD+2 THEN
  ; OK
ELSE
  ; NEW+2 >= OLD+2
  IF NEW+2 EQ OLD+2 AND NEW LO OLD THEN
    ;OK
  ELSE
  
```

```

9901                                     ;NEW+2 > OLD+2 OR
9902                                     ;(NEW+2 = OLD+2 AND
9903                                     ; NEW >= OLD)
9904                                     ;BAUD RATE DIDN'T CHANGE
9905 005530                                LET ERRCHK := #BIT2 ; SET ERROR INDICATOR
  (4) 005530 012767 000004 002430      MOV    #BIT2,ERRCHK
9906 005536                                ENDIF
  (4) 005536                                $130:
9907 005536                                ENDIF
  (4) 005536                                $126:
9908                                     ;UPDATE OLD TIME
9909 005536                                LET OLD := NEW
  (4) 005536 016767 000102 000104      MOV    NEW,OLD
9910 005544                                LET OLD+2 := NEW+2
  (4) 005544 016767 000076 000100      MOV    NEW+2,OLD+2
9911
9912 005552                                ENDINC ;BAUD RATE
  (4) 005552 000712                      BR    $117
  (3) 005554                                $120:
9913 005554                                LET R3 :B= $USWR+1 AND #17      ; PUT BAUD BACK
  (4) 005554 116703 173441              MOVB  $USWR+1,R3
  (7) 005560 110346                      MOVB  R3,-(SP)
  (7) 005562 142716 000017              BICB  #17,(SP)
  (7) 005566 142603                      BICB  (SP)+,R3
9914 005570                                LET R3 := R3 CLR.BY #177400
  (7) 005570 042703 177400              BIC  #177400,P3
9915 005574                                LET @TCSRHI :B= RATES(R3)      ; LIKE HE WANTED IT
  (4) 005574 116377 005624 173464      MCVB  RATES(R3),@TCSRHI
9916
9917                                     ; CAN NOT LEAVE WITH MAINT SET
9918 005602                                LET @TCSR := @TCSR CLR.BY #MAINT
  (7) 005602 042777 000004 173454      BIC  #MAINT,@TCSR
9919 005610                                I #BIT2 SETIN ERRCHK THEN
  (6) 005610 032767 000004 002350      BIT  #BIT2,ERRCHK
  (9) 005616 001401                      BEQ  $131
9920                                     ; REPORT DEFERED ERROR
9921 005620                                ERRHRD 126
  (1) 005620 104126                      ERROR 126
9922 005622                                ENDIF
  (4) 005622                                $131:
9923 005622                                EXIT ;SKIP TABLE
  (3) 005622 000414                      BR    TST17      ;:EXIT THIS TEST
9924
9925 005624                                RATES: ;A TABLE OF THE ACTUAL BYTES TO MOVE INTO THE
9926                                     ;UPPER BYTE OF XCSR FOR EACH BAUD RATE
9927                                     ;** NOTE::
9928                                     ;** THE VALUE INDICATED IN THE COLUMN 'OFFSET
9929                                     ;** INTO TABLE' CAN BE PLACED INTO BITS<11:8>
9930                                     ;** OF LOCATION '$USWR' TO CAUSE THE CORROSPONDING
9931                                     ;** BAUD TO BE SELECTED IN THE DLV11-F UPON
9932                                     ;** COMPLETION OF THIS TEST.
9933
9934 005624 010                                R0050: .BYTE 010      ; BAUD 50      OFFSET INTO TABLE
9935 005625 030                                R0070: .BYTE 030      ; BAUD 70      OFFSET INTO TABLE
9936 005626 050                                R0110: .BYTE 050      ; BAUD 110     OFFSET INTO TABLE
9937 005627 070                                R0135: .BYTE 070      ; BAUD 135     OFFSET INTO TABLE

```



```

9958
9963
9971
(3)
(4)
(4)
(4)
(4)
(4)
(3)
(2) 005654 000004
(2)
(1) 005656 012767 000010 173274 MOV #10,$TIMES ;;DO 10 ITERATIONS
(2) 005664 012767 000017 173306 MOV #17,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
9976
9977
9978
9979 005672 IF #APTENV SET IN $ENV THEN
(6) 005672 032767 000001 173314 BIT #APTENV,$ENV
(9) 005700 001404 BEQ $132
9980 005702 EXIT TEST
(5) 005702 012767 000001 173250 MOV #1,$TIMES
(3) 005710 000532 BR TST20 ;;EXIT THIS TEST
9981 005712 ENDF
(4) 005712 $132:
9982
9983
9984 ;CLEAR 'INTERRUPT OCCURED' FLAG
9985 005712 LET INTFLAG := #0
(4) 005712 005067 002742 CLR INTFLAG
9986
9987 ;GET VECTOR ADDRESS
9988 005716 LET R3 := DLVEC
(4) 005716 016703 173334 MOV DLVEC,R3
9989
9990 ;FOR THE TRANSMITTER
005722 LET R3 := R3 + #4
(7) 005722 062703 000004 ADD #4,R3
9991
9992 ;SET VECTOR TO POINT TO TRANS.SRV AT PRI
005726 SETVEC R3, #INTSRV, #PR6
(5) 005726 010146 MOV R1,-(SP)
(5) 005730 010301 MOV R3,R1
(5) 005732 012721 010652 MOV #INTSRV,(R1)+
(5) 005736 012711 000300 MOV #PR6,(R1)
(5) 005742 012601 MOV (SP)+,R1
9993 005744 BGNSUB
(5) 005744 012767 005752 173136 MOV #64,$LPERR
9994
9995 ;; MAKE SURE THAT TRANSMITTER READY IS SET
005752 CALL TIMER IN <#500,#XMITRDY,TCSR,#SET>
(3) 005752 010546 MOV R5,-(SP)
(7) 005754 012745 177777 MOV #SET,-(R5)
(6) 005760 016745 173300 MOV TCSR,-(R5)
(5) 005764 012745 000200 MOV #XMITRDY,-(R5)
(4) 005770 012745 000500 MOV #500,-(R5)
(3) 005774 004767 002314 JSR PC,TIMER
(3) 006000 012605 MOV (SP)+,R5
9996
  
```

```

9997                                     ;CLEAR INTERRUPT ENABLE
9998 006002                                LET @TCSR := @TCSR CLR.BY #XMITIE
(7) 006002 042777 000100 173254          BIC #XMITIE,@TCSR
9999
10004                                     ;SET IT TO 0
10005 006010 012746 000000                MOV #PRO,-(SP)      ;;PUT NEW PS ON STACK
(1) 006014 012746 006022                  MOV #65$,-(SP)     ;;PUT NEW PC ON STACK
(1) 006020 000002                            RTI                ;;POP NEW PC AND PS
(1) 006022                                65$:
10010
10011
10012 006022                                ;NOW SET I E. BIT
(7) 006022 052777 000100 173234          BIS #XMITIE,@TCSR
10013
10014                                     ;LET INTERRUPT HAVE TIME TO OCCUR
10015 006030                                WAITMS 200.
(4) 006030 010546                            MOV R5,-(SP)
(5) 006032 012745 000310                    MOV #200,-(R5)
(4) 006036 004767 002530                    JSR PC,WAIT
(4) 006042 012605                            MOV (SP)+,R5
10016
10017
10018 006044                                ;DID EXACTLY 1 INTERRUPT OCCUR
(6) 006044 026727 002610 000001          CMP INTFLAG,#1
(9) 006052 001406                            BEQ $133
10019
10020                                     ;NO - WAS IT 0 OR MORE THAN ONCE
(6) 006054 005767 002600                TST INTFLAG
(9) 006060 001002                            BNE $134
10021
10022                                     ;TRANSMITTER DID NOT INTERRUPT IN TIME
(1) 006062 104106                            ERROR 106          ERRHRD 106,,DIDNOT
10023                                     ELSE
(4) 006064 000401                            BR $135
(3) 006066                                $134:
10024
10025                                     ;TWICE
10026 006066                                ;TRANSMITTER INTERRUPTED TWICE
(1) 006066 104107                            ERROR 107          ERRHRD 107,,TWICE
10027                                     ENDIF
(4) 006070                                $135:
10028 006070                                $133:
(4) 006070
10029 006070
10030                                     ENDSUB
10031                                     ;INTERRUPT WITHOUT INTERRUPT ENABLE SET
(5) 006070 012767 006076 173012          MOV #64$,$LPERR   BGNSUB
10032
10033                                     ;CLEAR 'INTERRUPT OCCURED' FLAG
(4) 006076 005067 002556                  CLR INTFLAG       LET INTFLAG := #0
10034
10035                                     ;CLEAR INTERRUPT ENABLE
(7) 006102 042777 000100 173154          BIC #XMITIE,@TCSR LET @TCSR := @TCSR CLR.BY #XMITIE
10036
10041 006110 012746 000000                MOV #PRO,-(SP)    ;NO INTERRUPTS SHOULD OCCUR.
(1) 006114 012746 006122                  MOV #65$,-(SP)   ;;PUT NEW PS ON STACK
;;PUT NEW PC ON STACK

```

```
(1) 006120 000002          RTI          ;;POP NEW PC AND PS
(1) 006122          65$:          ;;DARE IT TO HAPPEN
10046          WAITMS 2
10047 006122          MOV      R5,-(SP)
(4) 006122 010546          MOV      #2,-(R5)
(5) 006124 012745 000002          JSR     PC,WAIT
(4) 006130 004767 002436          MOV     (SP)+,R5
(4) 006134 012605
10048 006136          IF INTFLAG NE #0 THEN
(6) 006136 005767 002516          TST     INTFLAG
(9) 006142 001401          BEQ     $136
10049          ;;INTERRUPT OCCURED WITH I E CLEARED
10050 006144          ERRHRD 110,NOTENAB
(1) 006144 104110          ERROR   110
10051 006146          ENDIF
(4) 006146          $136:          BRESET
10052 006146          BRESET
(1) 006146 000005          RESET
10053 006150          ENDSUB
10054          ;;RESTORE VECTOR AREA
10055 006150          CLRVEC R3
(3) 006150 010146          MOV     R1,-(SP)          ;;PUSH R1 ON STACK
(3) 006152 010246          MOV     R2,-(SP)          ;;PUSH R2 ON STACK
(5) 006154 012701 000003          MOV     #R3,R1
(5) 006160 010102          MOV     R1,R2
(8) 006162 062702 000002          ADD     #2,R2
(5) 006166 010221          MOV     R2,(R1)+
(5) 006170 005011          CLR     (R1)
(3) 006172 012602          MOV     (SP)+,R2          ;;POP STACK INTO R2
(3) 006174 012601          MOV     (SP)+,R1          ;;POP STACK INTO R1
10056
10057 006176          ENDTST
10058
10059
10060
10061
10062
10063
```

```
10065
10070
10076
(3)
(4)
(4)
(4)
(3)
(2) 006176 000004
(2)
(1) 006200 012767 000010 172752
(2) 006206 012767 000020 172764
10081 006214
(6) 006214 032767 040000 172776
(8) 006222 001404
(6) 006224 126727 002671 000001
(9) 006232 001002
(6) 006234
10082 006234 000167 000242
10083 006240
(4) 006240
10084
10085
10086
10087 006240
(5) 006240 010146
(5) 006242 016701 173010
(5) 006246 012721 010652
(5) 006252 012711 000300
(5) 006256 012601
10088
10089 006260
(5) 006260 012767 006266 172622
10090 006266
(4) 006266 005067 002366
10091
10092 006272
(7) 006272 052777 000004 172764
10093
10094 006300
(7) 006300 042777 000100 172752
10095
10096
10101 006306 012746 000000
(1) 006312 012746 006320
(1) 006316 000002
(1) 006320
10106
10107
10108 006320
(4) 006320 105077 172744
10109
10110
10111
10112 006324
(3) 006324 010546

*****
*****
*TEST 20 RECEIVER INTERRUPT LOGIC TEST
* THIS TEST COVERS ALL OF THE RECEIVER
* SIDE OF THE INTERRUPT LOGIC IN
* CHARACTER MODE.
*****
TST20: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #20,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
IF #MAINTJUMP NOTSET IN $USWR ORB CONSOLE EQ #TRU
BIT #MAINTJUMP,$USWR
BEQ $137
CMPB CONSOLE,#TRUE
BNE $140
$137: JMP TST21 ; EXIT TEST
ENDIF
$140:
;CLEAR INTERRUPT OCCURED FLAG
;SET UP RECEIVER INTER.VECTOR
SETVEC DLVEC,#INTSRV,#PR6
MOV R1,-(SP)
MOV DLVEC,R1
MVC #INTSRV,(R1)+
MOV #PR6,(R1)
MOV (SP)+,R1
;PRIORITY 0 AND MULTIPLE INTERRUPT TEST.-RCVRIE
BGNSUB
MOV #64,$LPERR
LET INTFLAG := #0
CLR INTFLAG
;SET MAINT. BIT
LET @TCSR := @TCSR SET.BY #MAINT
;CLEAR INTERRUPTS
LET @RCSR := @RCSR CLR.BY #RCVRIE
;CHANGE PRIORITY
;...TO 0
MOV #PRO,-(SP) ;;PUT NEW PS ON STACK
MOV #65,-(SP) ;;PUT NEW PC ON STACK
RTI ;;POP NEW PC AND PS
65$:
;SEND A CHARACTER
LET @TBUF :B= #0
;WAIT A MAXIMUM
;OF 500 MSEC FOR
;RCVR DONE TO SET IN RCSR
(CALL TIMER IN <#500,#RCVRDONE,RCSR,#SET>
MOV R5,-(SP)
```

```

(7) 006326 012745 177777 MOV #SET,-(R5)
(6) 006332 016745 172722 MOV RCSR,-(R5)
(5) 006336 012745 000200 MOV #RCVRDONE,-(R5)
(4) 006342 012745 000500 MOV #500,-(R5)
(3) 006346 004767 001742 JSR PC,TIMER
(3) 006352 012605 MOV (SP)+,R5
10113
10114 006354
(7) 006354 052177 000100 172676 BIS #RCVRIE,@RCSR
10115
10116 006362
(4) 006362 010546 MOV R5,-(SP)
(5) 006364 012745 000001 MOV #1,-(R5)
(4) 006370 004767 002176 JSR PC,WAIT
(4) 006374 012605 MOV (SP)+,R5
10117
10118 006376
(4) 006376 017700 172660 MOV @RBUF,R0
10119
10120 006402
(6) 006402 026727 002252 000001 CMP INTFLAG,#1
(9) 006410 001411 BEQ $141
10121
10122
10123 006412
(7) 006412 042777 000004 172644 BIC #MAINT,@TCSR
10124 006420
(6) 006420 005767 002234 TST INTFLAG
(9) 006424 001002 BNE $142
10125
10126 006426
(1) 006426 104111 ERROR 111
10127
10128 006430
(4) 006430 000401 BR $143
(3) 006432 $142:
10129
10130 006432
(1) 006432 104112 ERROR 112
10131 006434
(4) 006434 $143:
10132 006434
(4) 006434 $141:
10133 006434
10134
10135
10136
10137 006434
(7) 006434 042777 000100 172616 BIC #RCVRIE,@RCSR
10138
10139
10140
10141 006442
(7) 006442 042777 000004 172614 BIC #MAINT,@TCSR
10142
10143 006450

```

```

;SET INTERRUPT ENABLE
LET @RCSR := @RCSR SET.BY #RCVRIE
;LET IT COME IN.
WAITMS 1
LET R0 := @RBUF ; CLEAR RCVRDONE
;DID HE DO IT RIGHT?
IF INTFLAG NE #1 THEN
;NONE OCCURED
; CAN NOT LEAVE WITH MAINT SET
LET @TCSR := @TCSR CLR.BY #MAINT
IF INTFLAG EQ #0 THEN
;RECEIVER DID NOT INTERRUPT IN TIME
ERRHRD 111,,DIDNOT
;TWICE OR MORE
ELSE
;RECEIVER INTERRUPTED TWICE
ERRHRD 112,,TWICE
ENDIF
ENDIF
ENDSUB
; CLEAR THE WORLD
LET @RCSR := @RCSR CLR.BY #RCVRIE
;RESET MAINT. BIT.
LET @TCSR := @TCSR CLR.BY #MAINT
LET R4 := @DLVEC

```


(4)	006450	017704	172602	MOV	@DLVEC,R4		
10144	006454					CLRVEC R4	
(3)	006454	010146		MOV	R1,-(SP)	::PUSH R1 ON STACK	
(3)	006456	010246		MOV	R2,-(SP)	::PUSH R2 ON STACK	
(5)	006460	012701	000004	MOV	#R4,R1		
(5)	006464	010102		MOV	R1,R2		
(8)	006466	062702	000002	ADD	#2,R2		
(5)	006472	010221		MOV	R2,(R1)+		
(5)	006474	005011		CLR	(R1)		
(3)	006476	012602		MOV	(SP)+,R2	::POP STACK INTO R2	
(3)	006500	012601		MOV	(SP)+,R1	::POP STACK INTO R1	
10145	006502					ENDTST	

```

10147
10152
10156
(3)
(4)
(3)
(2) 006502 000004
(2)
(1) 006504 012767 000001 172446      MOV #1,$TIMES      ;;DO 1 ITERATION
(2) 006512 012767 000021 172460      MOV #21,$TESTN    ;;SET TEST NUMBER IN APT MAIL BOX
10161 006520                                IF #MAINTJUMP NOTSETIN $USWR ORB CONSOLE EQ #TRU
(6) 006520 032767 040000 172472      BIT #MAINTJUMP,$USWR
(8) 006526 001404                                BEQ $144
(6) 006530 126727 002365 000001      CMPB CONSOLE,#TRUE
(9) 006536 001004                                BNE $145
(6) 006540                                $144:
10162 006540                                EXIT TEST
(5) 006540 012767 000001 172412      MOV #1,$TIMES
(3) 006546 000526                                BR TST22          ;;EXIT THIS TEST
10163 006550                                ENDIF
(4) 006550                                $145:
10164
10165 006550                                LET ERRCHK := #0
(4) 006550 005067 001412              CLR ERRCHK
10166                                ;SET MAINT. BIT
10167 006554                                LET @TCSR := @TCSR SET.BY #MAINT
(7) 006554 052777 000004 172502      BIS #MAINT,@TCSR
10168
10169                                ;CHANGE PRIORITY
10170                                ;... TO 0
10175 006562 012746 000000              MOV #PRO,-(SP)    ;;PUT NEW PS ON STACK
(1) 006566 012746 006574              MOV #64$,-(SP)   ;;PUT NEW PC ON STACK
(1) 006572 000002              RTI              ;;POP NEW PC AND PS
(1) 006574                                64$:
10180                                ;GET DATA MASK.
10181 006574                                CALL DATLNG OUT <R1>
(4) 006574 162705 000002              SUB #1*2,R5
(3) 006600 004767 001666              JSR PC,DATLNG
(4) 006604 012501              MOV (R5)+,R1
10182 006606                                LET R0 := @RBUF ; START CLEAR
(4) 006606 017700 172450              MOV @RBUF,R0
10183
10184                                ;ALL BINARY CHAR.
10185 006612                                INCR R2 FROM #0 TO #377 BY #1
(4) 006612 005002              CLR R2
(5) 006614 000401              BR $146
(4) 006616                                $147:
(8) 006616 005202              INC R2
(5) 006620                                $146:
(5) 006620 020227 000377              CMP R2,#377
(7) 006624 003062              BGT $150
10186
10187                                ;TRANSMIT CHAR IN R2
10188
10189                                CALL TIMER IN <#500,#XMITRDY,TCSR,#SET>
10190 006626

```

(3)	006626	010546			MOV	R5,-(SP)		
(7)	006630	012745	177777		MOV	#SET,-(R5)		
(6)	006634	016745	172424		MOV	TCSR,-(R5)		
(5)	006640	012745	000200		MOV	#XMITRDY,-(R5)		
(4)	006644	012745	000500		MOV	#500,-(R5)		
(3)	006650	004767	001440		JSR	PC,TIMER		
(3)	006654	012605			MOV	(SP)+,R5		
10191	006656						IF.ERROR THEN	
(6)	006656	103003			BCC	\$151		LET ERRCHK := ERRCHK SET.BY #BIT3
10192	006660							ENDIF
(7)	006660	052767	000010	001300	BIS	#BIT3,ERRCHK		
10193	006666							
(4)	006666					\$151:		
10194								
10195								
10196	006666							
(4)	006666	110277	172376		MOVB	R2,@TBUF		
10197								
10198	006672							CALL TIMER IN <#500,#RCVRDONE,RCSR,#SET>
(3)	006672	010546			MOV	R5,-(SP)		
(7)	006674	012745	177777		MOV	#SET,-(R5)		
(6)	006700	016745	172354		MOV	RCSR,-(R5)		
(5)	006704	012745	000200		MOV	#RCVRDONE,-(R5)		
(4)	006710	012745	000500		MOV	#500,-(R5)		
(3)	006714	004767	001374		JSR	PC,TIMER		
(3)	006720	012605			MOV	(SP)+,R5		
10199	006722						IF.ERROR THEN	
(6)	006722	103003			BCC	\$152		LET ERRCHK := ERRCHK SET.BY #BIT4
10200	006724							ENDIF
(7)	006724	052767	000020	001234	BIS	#BIT4,ERRCHK		
10201	006732							
(4)	006732					\$152:		
10202								
10203	006732							
(4)	006732	017703	172324		MOV	@RBUF,R3		
10204								
10205								
10206								
10207								
10208								
10209								
10210	006736							
(4)	006736	010204			MOV	R2,R4		
(7)	006740	040104			BIC	R1,R4		
10211	006742							
(7)	006742	040103			BIC	R1,R3		
10212								
10213								
10214	006744							
(6)	006744	020403			CMP	R4,R3		
(9)	006746	001410			BEQ	\$153		
10215								
10216								
10217	006750							
(7)	006750	042777	000004	172306	BIC	#MAINT,@TCSR		
10218	006756							

```

IF.ERROR THEN
  LET ERRCHK := ERRCHK SET.BY #BIT3
ENDIF

;TRANSMIT IT
LET @TBUF :B= R2

CALL TIMER IN <#500,#RCVRDONE,RCSR,#SET>

IF.ERROR THEN
  LET ERRCHK := ERRCHK SET.BY #BIT4
ENDIF

;AND SAVE IT
LET R3 := @RBUF

;COMPARE TO SEE IF WE RECEIVED IT ALL

;CLEAN OFF NON-DATA BITS
;ON BOTH TRANSMITTED AND
LET R4 := R2 CLR.BY R1

LET R3 := R3 CLR.BY R1

;RECEIVED DATA
IF R4 NE R3 THEN

;DATA COMPARE ERROR
; CAN NOT LEAVE WITH MAINT SET
LET @TCSR := @TCSR CLR.BY #MAINT

ERRHRD 116,COMP,SBWAS
  
```

```
(1) 006756 104116 ERROR 116  
10219 006760  
(5) 006760 012767 000001 172172 MOV #1,$TIMES EXIT TEST ; ON ERROR  
(3) 006766 000416 BR TST2 ;;;EXIT THIS TEST  
10220 006770 ENDIF  
(4) 006770 $153: ENDINC ; R2  
10221 006770 BR $147  
(4) 006770 000712 $150:  
(3) 006772  
10222  
10223 ;RESET MAINT. BIT.  
10224 006772 LET @TCSR := @TCSR CLR.BY #MAINT  
(7) 006772 042777 000004 172264 BIC #MAINT,@TCSR  
10225 007000 IF #BIT3 SETIN ERRCHK THEN  
(6) 007000 032767 000010 001160 BIT #BIT3,ERRCHK  
(9) 007006 001401 BEQ $154  
10226 007010 ERRHRD 130  
(1) 007010 104130 ERROR 130  
10227 007012 ENDIF  
(4) 007012 $154: IF #BIT4 SETIN ERRCHK THEN  
10228 007012 BIT #BIT4,ERRCHK  
(6) 007012 032767 000020 001146 BEQ $155  
(9) 007020 001401  
10229 007022 ERRHRD 131  
(1) 007022 104131 ERROR 131  
10230 007024 ENDIF  
(4) 007024 $155: ENDTST  
10231 007024  
10232  
10233  
10234
```

```

10236
10241
10242
(3)
(3)
(2) 007024 000004
(2)
(1) 007026 012767 000001 172124
(2) 007034 012767 000022 172136
10247 007042
(6) 007042 032767 020000 172150
(8) 007050 001404
(6) 007052 032767 000100 172140
(9) 007060 001004
(6) 007062
10248
10249 007062
(5) 007062 012767 000001 172070
(3) 007070 000516
10250 007072
(4) 007072
10251
10252 007072
(7) 007072 042777 000004 172164
10253
10254
10255
10256
10257
10258
10259
10260
10261
10262 007100
(7) 007100 052777 000011 172152
10263
10264
10269 007106 012746 000000
(1) 007112 012746 007120
(1) 007116 000002
(1) 007120
10274
10275 007120
(4) 007120 162705 000002
(3) 007124 004767 001342
(4) 007130 012501
10276 007132
(4) 007132 017700 172124
10277
10278 007136
(4) 007136 005002
(5) 007140 000401
(4) 007142
(8) 007142 005202
(5) 007144
(5) 007144 020227 000377

```

```

*****
*****
*TEST 22 TEST DATA THROUGH WRAP
*****
TST22: SCOPE
MOV #1,$TIMES ;;DO 1 ITERATION
MOV #22,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
IF #WRAP NOTSETIN $USWR OR #COMSPD NOTSETIN $USW
BIT #WRAP,$USWR
BEQ $156
BIT #COMSPD,$USWR
BNE $157
$156:
;;CAN'T TEST WITHOUT A WRAP
EXIT TST
MOV #1,$TIMES
BR TST23 ;;EXIT THIS TEST
ENDIF
$157:
;;DON'T USE MAINT.
LET @TCSR := @TCSR CLR.BY #MAINT
; IF A SPECIAL TURN AROUND CARD IS
; CONNECTED IN PLACE OF THE WRAP
; SETTING READER RUN WILL ENABLE IT.
; THIS MODULE IS ONLY USED IN MANUFACTUR
; AND ONLY ON THE CONSOLE DLV11-F.
; IF NO SPECIAL MODULE IS AVAILABLE,
; AND THE WRAP BIT IS SET IN $USWR
; THEN THIS TEST WILL ERROR ON THE CONSO
LET @RCSR := @RCSR SET.BY #11
;CHANGE PRIORITY
;..TO 0
MOV #PRO,-(SP) ;;PUT NEW PS ON STACK
MOV #64$,-(SP) ;;PUT NEW PC ON STACK
RTI ;;POP NEW PC AND PS
;GET DATA MASK
CALL DATLNG OUT <R1>
LET R0 := @RBUF ; START CLEAN
;BINARY COUNT PATTERN
INCR R2 FROM #0 TO #377 BY #1
$161:
INC R2
$160:
CMP R2,#377

```

(7)	007150	003063		BGT	\$162		
10279							
10280							
10281							; TRANSMIT THE CHAR. IN R2.
10282							
10283							; MAKE SURE IT'S READY
10284	007152						CALL TIMER IN <#500,#XMITRDY,TCSR,#SET>
(3)	007152	010546		MOV	R5,-(SP)		
(7)	007154	012745	177777	MOV	#SET,-(R5)		
(6)	007160	016745	172100	MOV	TCSR,-(R5)		
(5)	007164	012745	000200	MOV	#XMITRDY,-(R5)		
(4)	007170	012745	000500	MOV	#500,-(R5)		
(3)	007174	004767	001114	JSR	PC,TIMER		
(3)	007200	012605		MOV	(SP)+,R5		
10285	007202						IF.ERROR THEN
(6)	007202	103005		BCC	\$163		
10286	007204	104123		ERROR	123		; TRANSMITTER NEVER BECAME READY
10287	007206						EXIT TEST
(5)	007206	012767	000001 171744	MOV	#1,\$TIMES		
(3)	007214	000444		BR	TST23		:::EXIT THIS TEST
10288	007216						ENDIF
(4)	007216		\$163:				
10289							
10290							; START IT ON ITS WAY
10291	007216						LET @TBUF :B= R2
(4)	007216	110277	172046	MOVB	R2,@TBUF		
10292							; NOW WAIT FOR RECIEVER DONE
10293	007222						CALL TIMER IN <#500,#RCVRDONE,RCSR,#SET>
(3)	007222	010546		MOV	R5,-(SP)		
(7)	007224	012745	177777	MOV	#SET,-(R5)		
(6)	007230	016745	172024	MOV	RCSR,-(R5)		
(5)	007234	012745	000200	MOV	#RCVRDONE,-(R5)		
(4)	007240	012745	000500	MOV	#500,-(R5)		
(3)	007244	004767	001044	JSR	PC,TIMER		
(3)	007250	012605		MOV	(SP)+,R5		
10294	007252						IF.ERROR THEN
(6)	007252	103005		BCC	\$164		
10295	007254						ERRHRD 124
(1)	007254	104124		ERROR	124		
10296							; RECIEVER NEVER BECAME READY
10297	007256						EXIT TEST
(5)	007256	012767	000001 171674	MOV	#1,\$TIMES		
(3)	007264	000420		BR	TST23		:::EXIT THIS TEST
10298	007266						ENDIF
(4)	007266		\$164:				
10299							
10300							; RETRIEVE
10301	007266						LET R3 := @RBUF
(4)	007266	017703	171770	MOV	@RBUF,R3		
10302							
10303							; STRIP OFF JUNK ON BOTH
10304	007272						LET R4 := R2 CLR.BY R1
(4)	007272	010204		MOV	R2,R4		
(7)	007274	040104		BIC	R1,R4		
10305	007276						LET R3 := R3 CLR.BY R1
(7)	007276	040103		BIC	R1,R3		

```

10306
10307
10308 007300
      (6) 007300 020403      CMP    R4,R3
      (9) 007302 001405      BEQ    $165
10309
10310 007304
      (1) 007304 104117      ERROR  117
10311 007306
      (5) 007306 012767 000001 171644  MOV    #1,$TIMES
      (3) 007314 000404      BR     TST23
10312 007316
      (4) 007316
10313
10314 007316
      (4) 007316 000711      BR     $161
      (3) 007320
10315
10316
10317
10318 007320
      (7) 007320 052777 000011 171732  BIS    #11,@RCSR
10319
10320
10321
10322 007326
10323
10324
10325
10326
  
```

```

      ;WE HAVE TROUBLE
      IF R4 NE R3 THEN

      ;DATA COMPARE ERROR
      ERRHRD 117,COMP,SBWAS

      EXIT TEST ; ON ERROR

      ;;EXIT THIS TEST
      ENDIF

      ENDINC ; R2

      ; NOW THAT THE TEST IS DONE
      ; WE WILL TOGGLE READER RUN
      ; TO TURN OFF THE SPECIAL MODULE.
      LET @RCSR := @RCSR SET.BY #11

      ENDTST
  
```

```

10328
10333
10337
(3)
(4)
(3)
(2) 007326 000004
(2)
(1) 007330 012767 000001 171622
(2) 007336 012767 000023 171634
10342
10343 007344
(6) 007344 032767 040000 171646
(8) 007352 001404
(6) 007354 126727 001541 000001
(9) 007362 001004
(6) 007364
10344 007364
(5) 007364 012767 000001 171566
(3) 007372 000553
10345 007374
(4) 007374
10346
10347
10352 007374
(4) 007374 162705 000002
(3) 007400 004767 001066
(4) 007404 012503
10353
10354
10355
10356
10357
10358
10359
10360
10361
10362
10363
10364
10365
10366
10371 007406 012746 000000
(1) 007412 012746 007420
(1) 007416 000002
(1) 007420
10376
10377 007420
(4) 007420 016701 171632
10378
10379 007424
(4) 007424 012721 007626
10380 007430
(4) 007430 012721 000300
10381
10382

```

```

*****
*****
*TEST 23 FULL DATA TRANSFER WITH INTERRUPTS
* AND MAINTENANCE MODE.
*****
TST23: SCOPE
MOV #1,$TIMES ;;DO 1 ITERATION
MOV #25,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
IF #MAINTJUMP NOTSETIN $USWR ORB CONSOLE EQ #TRU
BIT #MAINTJUMP,$USWR
BEQ $166
CMPB CONSOLE,#TRUE
BNE $167
$166:
EXIT TEST
MOV #1,$TIMES
BR TST24 ;;EXIT THIS TEST
ENDIF
$167:
;GET DATA MASK
CALL DATLNG OUT <R3>
SUB #1*2,R5
JSR PC,DATLNG
MCV (R5)+,R3
; THIS TEST WILL RUN BOTH TRANSMITTER AND
; RECIEVER AT FULL SPEED TESTING
; THE ABILITY OF THE MODULE
; TO HANDLE INTERRUPTS FROM BOTH SIDES
; AT ONCE. ALSO, THE DOUBLE BUFFERING LOGIC
; OF THE UART WILL BE FULLY TESTED.
; THIS TEST WILL TRANSFER A MAXIMUM OF 400(8)
; CHARACTERS THROUGH THE MODULE, BUT IF AN ERROR
; IS DETECTED BY THE TEST A PREMATURE SHUTDOWN OCCURS.
;CHANGE PRIORITY
;..TO 0
MOV #PRO,-(SP) ;;PUT NEW PS ON STACK
MOV #64$,-(SP) ;;PUT NEW PC ON STACK
RTI ;;POP NEW PC AND PS
64$:
;GET VECTOR ADDRESS
LET R1 := DLVEC
;RCVR VECTOR
LET (R1)+ := #REC
LET (R1)+ := #PR6
;POINT TO TRANSMITTER VECTOR
;AND SET IT UP ALSO

```


10383	007434					LET (R1)+ := #TRAN
(4)	007434	012721	007564	MOV	#TRAN,(R1)+	LET (R1) := #PR6
10384	007440					
(4)	007440	012711	000300	MOV	#PR6,(R1)	
10385						
10386						; CLEAR ERROR COUNTER
10387	007444					LET ERRCNT := #0
(4)	007444	005067	000106	CLR	ERRCNT	
10388						
10389	007450					; INITIALIZE COUNTERS
(4)	007450	012701	177777	MOV	#-1,R1	LET R1 := #-1
10390						
10391	007454					;RECEIVER STORAGE
(4)	007454	005002		CLR	R2	LET R2 := #0
10392						
10393	007456					;# OF RECEIVED CHAR. COUNT.
(4)	007456	012704	177777	MOV	#-1,R4	LET R4 := #-1
10394						
10395	007462					BRESET ;SET UP ALL REGISTERS
(1)	007462	000005		RESET		
10396						
10397	007464					;SET UP MAINTENANCE
(7)	007464	052777	000004 171572	BIS	#MAINT,@TCSR	LET @TCSR := @TCSR SET.BY #MAINT
10398						
10399						
10400	007472					;SET I.E. IN TRANSMITTER
(7)	007472	052777	000100 171564	BIS	#XMITIE,@TCSR	LET @TCSR := @TCSR SET.BY #XMITIE
10401						
10402	007500					;AND RECEIVER
(7)	007500	052777	000100 171552	BIS	#RCVRIE,@RCSR	LET @RCSR := @RCSR SET.BY #RCVRIE
10403						
10404						
10405						
10406	007506					;NOW WE WAIT UNTIL R4 COUNT (RECEIVED) IS EQUAL
(3)	007506					REPEAT
10407	007506					
(4)	007506	020467	000046	CMP	R4,NUMBER	UNTIL R4 EQ NUMBER OR ERRCNT GT #0
(6)	007512	001403		BEQ	\$171	
(4)	007514	005767	000036	TST	ERRCNT	
(7)	007520	003772		BLE	\$170	
(4)	007522					
10408						
10409	007522					
(7)	007522	042777	000004 171534	BIC	#MAINT,@TCSR	LET @TCSR := @TCSR CLR.BY #MAINT
10410						
10411	007530					; CHECK FOR DATA COMPARE ERRORS.
(6)	007530	005767	000022	TST	ERRCNT	IF ERRCNT NE #0 THEN
(9)	007534	001401		BEQ	\$172	
10412						
10413	007536					;DATA COMPARE ERROR
(1)	007536	104120		ERROR	120	ERRHRD 120,COMP,FIRST
10414	007540					
(4)	007540					ENDIF
10415						
10416	007540					
(7)	007540	042777	000100 171516	BIC	#XMITIE,@TCSR	LET @TCSR := @TCSR CLR.BY #XMITIE

```

10417 007546          LET @RCSR := @RCSR CLR.BY #RCVRIE
(7) 007546 042777 000100 171504      BIC  #RCVRIE,@RCSR
10418 007554          EXIT      ;SKIP OVER SUPPORT ROUTINES & STORAGE
(3) 007554 000462          BR      TST24          ;::EXIT THIS TEST
10419
10420 007556 000000      ERRCNT: 0
10421 007560 001000      NUMBER: 1000
10422 007562          000      SB: .BYTE 0
10423 007563          000      WAS: .BYTE 0
10424
10425
10430          ;:*****
10435          ;RECEIVER INTERRUPT HANDLER
10436 007564          BGNSRV REC
(1) 007564
10441
10446          ;:*****
10447          ;INCREMENT CHAR COUNT
10448 007564 005201          INC  R1          LET R1 := R1 + #1
(7) 007564
10449          ;SET UP FOR TRANSFER
10450 007566          LET HOLD := R1 CLR.BY R3
(4) 007566 010167 000030      MOV  R1,HOLD
(7) 007572 040367 000024      BIC  R3,HOLD
10451          ;AND SEND.
10452 007576          LET @TBUF := HOLD
(4) 007576 016777 000020 171464      MOV  HOLD,@TBUF
10453          ;ALL DONE
10454 007604          IF R1 EQ NUMBER THEN
(6) 007604 020167 177750      CMP  R1,NUMBER
(9) 007610 001003          BNE  $173
10455          ;STOP INTERRUPT PROCESSING
10456 007612          LET @TCSR := @TCSR CLR.BY #XMITIE
(7) 007612 042777 000100 171444      BIC  #XMITIE,@TCSR
10457 007620          ENDIF
(4) 007620
10458          $173:
10459 007620 000401          BR   ZZZ          ; EXIT SRV
10460          HOLD:0
10461 007622 000000
10462
10463 007624          ZZZ:          ENDSRV
(1) 007624 000002          RTI
10464
10465
10466          ;:*****
10471          ;RECEIVER INTERRUPT HANDLER
10476          BGNSRV REC
10477 007626          REC:
(1) 007626
10482          ;:*****
10487
10488          ;COUNT THIS CHAR.
10489 007626          LET R4 := R4 + #1
(7) 007626 005204          INC  R4
10490          ;GET CHAR IN + MASK IT
  
```

10491	007630					LET R2 := @RBUF CLR.BY R3
(4)	007630	017702	171426	MOV	@RBUF,R2	
(7)	007634	040302		BIC	R3,R2	
10492						;RHLD WILL CONTAIN EXPECTED INPUT
10493	007636					LET RHLD := R4 CLR.BY R3
(4)	007636	010467	000054	MOV	R4,RHLD	
(7)	007642	040367	000050	BIC	R3,RHLD	
10494						
10495						;DO THEY COMPARE
10496	007646					IF R2 NE RHLD THEN
(6)	007646	020267	000044	CMP	R2,RHLD	
(9)	007652	001412		BEQ	\$174	
10497						;FIRST ERROR
10498	007654					IF ERRCNT EQ #0 THEN
(6)	007654	005767	177676	TST	ERRCNT	
(9)	007660	001005		BNE	\$175	
10499						;SAVE RECORD OF FIRST MISS
10500	007662					LET SB :B= RHLD
(4)	007662	116767	000030 177672	MOVB	RHLD,SB	
10501	007670					LET WAS :B= R2
(4)	007670	110267	177667	MOVB	R2,WAS	
10502	007674					ENDIF
(4)	007674					
10503						;COUNT IT.
10504	007674					LET ERRCNT := ERRCNT + #1
(7)	007674	005267	177656	INC	ERRCNT	
10505	007700					ENDIF
(4)	007700					
10506						
10507						;ALL DONE?
10508	007700					IF R4 EQ NUMBER THEN
(6)	007700	020467	177654	CMP	R4,NUMBER	
(9)	007704	001003		BNE	\$176	
10509						;STOP RECEIVER INTERRUPTS
10510	007706					LET @RCSR := @RCSR CLR.BY #RCVRIE
(7)	007706	042777	000100 171344	BIC	#RCVRIE,@RCSR	
10511						; INDICATE ALL DONE TO TIMER
10512						;MAIN REPEAT LOOP IS CHECKING
10513						;FOR 'R4 = NUMBER' ALSO
10514	007714					ENDIF
(4)	007714					
10515						
10516	007714	000401		BR	ZZZZ	; EXIT SRV
10517						
10518	007716	000000				RHLD:0
10519	007720					
10520	007720					ENDSRV
(1)	007720	000002		RTI		
10521						
10522	007722					ENDTST
10523						
10524						
10525						

```

10527
10532
10539
(3)
(4)
(4)
(4)
(4)
(3)
(2) 007722 000004
(2)
(1) 007724 012767 000010 171226
(2) 007732 012767 000024 171240
10544 007740
(6) 007740 032767 040000 171252
(8) 007746 001404
(6) 007750 032767 010000 171242
(9) 007756 001004
(6) 007760
10545 007760
(5) 007760 012767 000001 171172
(3) 007766 000500
10546 007770
(4) 007770
10547 007770
(6) 007770 126727 001125 000001
(9) 007776 001004
10548 010000
(5) 010000 012767 000001 171152
(3) 010006 000470
10549 010010
(4) 010010
10550
10551 010010
(4) 010010 005067 000152
10552
10553 010014
(7) 010014 052777 000004 171242
10554
10555 010022
(7) 010022 052777 000001 171234
10556
10557 010030
(4) 010030 012777 000252 171232
10558
10559 010036
(3) 010036 010546
(7) 010040 012745 177777
(6) 010044 016745 171210
(5) 010050 012745 000200
(4) 010054 012745 000500
(3) 010060 004767 000230
(3) 010064 012605
10560 010066
(6) 010066 103001
10561

```

```

*****
*****
*****
*TEST 24      TEST BREAK GENERATION LOGIC
*              TRANSMIT KNOWN CHAR WITH BREAK SET
*              AND COMPARE RECEIVED WITH 0.
*              FRAMING ERROR WILL ALSO BE CHECKED
*              IF ERROR BITS ARE ENABLED.
*****

```

```

TST24: SCOPE
MOV #10,$TIMES      ;;DO 10 ITERATIONS
MOV #24,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
                    IF #MAINTJUMP NOTSETIN $USWR OR #BRK NOTSETIN $U
BIT #MAINTJUMP,$USWR
BEQ $177
BIT #BRK,$USWR
BNE $200
$177:
                    EXIT TEST
MOV #1,$TIMES
BR TST25           ;;:EXIT THIS TEST
                    ENDIF
$200:
                    IFB CONSOLE EQ #TRUE THEN
CMPB CONSOLE,#TRUE
BNE $201
                    EXIT TEST
MOV #1,$TIMES
BR TST25           ;;:EXIT THIS TEST
                    ENDIF
$201:
LET ERRCHK := #0      ; CLEAR ERROR WORD
                    ;SET MAINTENANCE BIT
LET @TCSR := @TCSR SET.BY #MAINT
                    ;SET BREAK BIT
LET @TCSR := @TCSR SET.BY #BREAK
                    ;NON-ZERO CHAR. '*'
LET @TBUF := #252
                    ;WAIT FOR DONE
CALL TIMER IN <#500,#RCVRDONE,RCSR,#SET>
MOV R5,-(SP)
MOV #SET,-(R5)
MOV RCSR,-(R5)
MOV #RCVRDONE,-(R5)
MOV #500,-(R5)
JSR PC,TIMER
MOV (SP)+,R5
IF.ERROR THEN
                    ; RECIEVER DONE DID NOT SET

```



```
10589
10594
10595      ;:*****
(3)      ;:*TEST 25      NOT A TEST - SEND BACK TO LOOP
(3)      ;:*****
(2) 010170 000004      TST25: SCOPE
(2)
(1) 010172 012767 000001 170760      MOV      #1,$TIMES      ;;DO 1 ITERATION
10596 010200 104401 010206      TYPE      ,65$      ;;TYPE ASCIZ STRING
(1) 010204 000404      BR      64$      ;;GET OVER THE ASCIZ
(1)      ;;65$: .ASCIZ <CRLF>*CSR: *
(1) 010216      64$:
10597 010216 016746 171032      MOV      DLADD,-(SP)      ;;SAVE DLADD FOR TYPEOUT
(1) 010222 104402      TYPOC      ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
10598 010224 104401 010232      TYPE      ,67$      ;;TYPE ASCIZ STRING
(1) 010230 000405      BR      66$      ;;GET OVER THE ASCIZ
(1)      ;;67$: .ASCIZ *,VECTOR: *
(1) 010244      66$:
10599 010244 016746 171006      MOV      DLVEC,-(SP)      ;;SAVE DLVEC FOR TYPEOUT
(1) 010250 104402      TYPOC      ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
10600 010252 104401 010260      TYPE      ,69$      ;;TYPE ASCIZ STRING
(1) 010256 000405      BR      68$      ;;GET OVER THE ASCIZ
(1)      ;;69$: .ASCIZ *,ERRORS: *
(1) 010272      68$:
10601 010272 016746 170614      MOV      $ERTTL,-(SP)      ;;SAVE $ERTTL FOR TYPEOUT
(1) 010276 104405      TYPDS      ;;GO TYPE--DECIMAL ASCII WITH SIGN
10602 010300 005067 170606      CLR      $ERTTL      ; RESET FOR NEXT DEVICE/PASS
10603 010304 104401 001171      TYPE      ,$CRLF
10604 010310 000167 171622      JMP      LOOP      ; BACK UP TO THE BEGINNING
```

10610
10615
10616
10617
10622
(2)
10623
10624
10625
10626
10627
10628
10629
10630
10631
10632
10633
10634
10635
10636
10637
10638
10639
10640
10645
10650
10651
10652
10653
10654
(4)
10655
(4)
10656
(4)
10657
10658
10659
10660
10661
(3)
10662
10663
(6)
(9)
10664
(4)
10665
(4)
(3)
10666
(4)
10667
(4)
10668
10669

010314
010314

000001
000000

010314
010314
010322
010322
010330
010330

016567 000004 000136
016567 000000 000132
112767 000000 000126

```

;:BGNMOD          SUBS
;:*****
ROUTINE TIMER <HOWLONG,WHICHBIT,REG,SETCLR>
TIMER:
* ROUTINE:TIMER
* THIS ROUTINE IS USED TO TEST THE STATUS OF ANY BIT
* IN ANY REGISTER.
* INPUTS:
*   HOWLONG   THE MAXIMUM AMOUNT OF TIME TO SPEND IN
*             THIS ROUTINE.
*   WHICHBIT  A MASK WITH THE BIT(S) SET THAT ARE
*             TO BE CHECKED.
*   REG       A POINTER TO THE REGISTER TO BE CHECKED
*   SETCLR    THE DESIRED RESULTS
*             EITHER #SET OR #CLEAR
* OUTPUT:
*   THE 'C' BIT IS SET TO INDICATE AN ERROR
*   BUT IT IS TESTED BY THE IF.ERROR STATEMENT
*
* NOTE:: THE USE OF (R5) IS PART OF THE LINKAGE
*        MECHANISM BETWEEN THE CALLER AND THE CALLED
;:*****

```

```

TRUE= 1
FALSE= 0

LET REGSAV := REG(R5) ; GET POINTER TO REGIST
LET TIMSAV := HOWLONG(R5) ; SAVE HOWLONG FOR
LET FLAG :B= #FALSE ; INITIALIZE THE EXIT FLA

; START OF AN INFINITE LOOP
LOOP
$213:
; TEST TO SEE IF WHICHBIT IS SET
IF WHICHBIT(R5) NOTSETIN @REGSAV THEN
LET HOLDSC :B= #CLR
ELSE
BR $216
$215:
LET HOLDSC :B= #SET ; REMEMBER THIS
ENDIF
$216:
; NOW SEE IF THAT WAS WHAT WE WANTED

```

\$213:
\$215:
\$216:

```

10670 010364          IFB    HOLDSC EQ SETCLR(R5) THEN
(6) 010364 126765 000075 000006      CMPB  HOLDSC,SETCLR(R5)
(9) 010372 001003          BNE   $217
10671          ; JUST THE THING WE NEEDED
10672 010374          LET    FLAG :B= #TRUE
(4) 010374 112767 000001 000062      MOVB  #TRUE,FLAG
10673 010402          ENDIF
(4) 010402          $217:
10674
10675 010402          EXIFB  FLAG EQ #TRUE OR TIMSAV LE #0
(4) 010402 126727 000056 000001      CMPB  FLAG,#TRUE
(6) 010410 001414          BEQ   $214
(4) 010412 005767 000044          TST  TIMSAV
(6) 010416 003411          BLE   $214
10676          ; ONE WAY OR THE OTHER, WE ARE DONE
10677          ; IF WE ARE STILL HERE THEN HANG AROUND A WHILE
10678
10679 010420          WAITMS 1          ;WAIT FOR 1 MILLI-SECONDS
(4) 010420 010546          MOV   R5,-(SP)
(5) 010422 012745 000001      MOV   #1,-(R5)
(4) 010426 004767 000140      JSR  PC,WAIT
(4) 010432 012605          MOV   (SP)+,R5
10680 010434          LET    TIMSAV := TIMSAV - #1 ; COUNTING DOWN
(7) 010434 005367 000022      DEC  TIMSAV
10681 010440          ENDLOOP          ; CONTINUED AT THE TOP
(4) 010440 000736          BR    $213
(3) 010442          $214:
10682
10683          ; ONLY 2 WAYS TO GET HERE
10684          ; 1). WE RAN OUT OF TIME---ERROR !!
10685          ; 2). THE BIT IS IN THE CORRECT CONDITION--GOOD !!
10686
10687 010442          IFB    FLAG EQ #TRUE THEN
(6) 010442 126727 000016 000001      CMPB  FLAG,#TRUE
(9) 010450 001001          BNE   $220
10688          RETURN NO.ERROR          ; GOOD
(4) 010452 000405          BR    $211
10689 010454          ENDIF
(4) 010454          $220:
10690 010454          RETURN ERROR          ; BAD
(2) 010454 000261          SEC
(4) 010456 000404          BR    $212
10691
10692 010460 000000          REGSAV: .WORD 0
10693 010462 000000          TIMSAV: .WORD 0
10694 010464 000          FLAG: .BYTE 0
10695 010465 000          HOLDSC: .BYTE 0
10696          ; WE ARE DONE GO BACK HOME
10697 010466          ENDRTN
(3) 010466          $211:
(2) 010466 000241          CLC
(3) 010470          $212:
(2) 010470 000207          RTS  PC

```



```

10699
10700
10705
10710 010472
      (2) 010472
10711
10712
10713
10714
10715
10716
10717
10718
10719
10720
10721
10726
10731
10732 010472
      (4) 010472 005065 000000
10733 010476
      (4) 010476 016767 170516 000062
      (7) 010504 016746 000056
      (7) 010510 042716 000017
      (7) 010514 042667 000046
10734
10735 010520
      (4) 010520 012767 000001 170544
      (5) 010526 000402
      (4) 010530
      (8) 010530 005267 170536
      (5) 010534
      (5) 010534 026767 170532 000024
      (7) 010542 003006
10736 010544
      (8) 010544 006365 000000
10737 010550
      (7) 010550 052765 000001 000000
10738 010556
      (4) 010556 000764
      (3) 010560
10739 010560
      (7) 010560 005165 000000
10740 010564
      (4) 010564 000401
10741 010566 000000
10742 010570
      (3) 010570
      (3) 010570
      (2) 010570 000207

```

```

*****
ROUTINE DATLNG <MASK>
DATLNG:
* ROUTINE:DATLNG
* THIS ROUTINE SETS UP A MASK FOR DATA, WITH
* INPUT - NOTHING IS PASSED TO THIS ROUTINE
* BUT GLOBAL INFORMATION IS ASSUMED TO EXIST:
* $USWR-- THE WORD FOR SOFTWARE PARAMETERS
* DATA-- A MASK FOR THE LOCATION OF THE OCTAL
*          NUMBER OF DATA BITS
*
* OUTPUT----
* MASK-- A MASK OF BINARY ONES RIGHT-JUSTIFIED
*        THE NUMBER OF WHICH IS DEFINED IN $USWR WORD.
*****

```

```

          LET MASK(R5) := #0 ; START
          LET NUMBR := $USWR AND #DATA
          MOV $USWR,NUMBR
          MOV NUMBR,-(SP)
          BIC #DATA,(SP)
          BIC (SP)+,NUMBR
          INCR I FROM #1 TO NUMBR BY #1
          MCV #1,I
          BR $223
          $224: INC I
          $223: CMP I,NUMBR
          BGT $225
          ASL MASK(R5)
          LET MASK(R5) := MASK(R5) SHIFT 1
          LET MASK(R5) := MASK(R5) SET.BY #1
          BIS #1,MASK(R5)
          BR $224
          ENDINC
          COM MASK(R5)
          LET MASK(R5) := COMP MASK(R5)
          BR $221
          NUMBR:0
          RETURN
          $221:
          $222:
          ENDRTN
          RTS PC

```

10744
 10745
 10750
 10755 010572
 (2) 010572
 10756
 10757
 10758
 10759
 10760
 10761
 10766
 10767 010572 010146
 (2) 010574 010246
 (2) 010576 010346
 10772 010600
 (4) 010600 016501 000000
 10773 010604
 (4) 010604 012702 000001
 (5) 010610 000402
 (4) 010612
 (8) 010612 062702 000001
 (5) 010616
 (5) 010616 020201
 (7) 010620 101010
 10774 010622
 (4) 010622 005003
 (5) 010624 000401
 (4) 010626
 (8) 010626 005203
 (5) 010630
 (5) 010630 020327 000100
 (7) 010634 003001
 10775 010636
 (4) 010636 000773
 (3) 010640
 10776 010640
 (4) 010640 000764
 (3) 010642
 10781 010642 012603
 (2) 010644 012602
 (2) 010646 012601
 10786 010650
 (3) 010650
 (3) 010650
 (2) 010650 000207

```

*****
ROUTINE WAIT <TIME>
WAIT:
* ROUTINE:WAIT
* THIS ROUTINE IS USED TO DELAY EXECUTION OF THE
* MAIN PROGRAM FOR A SPECIFIED AMOUNT OF TIME.
* THIS IS ACCOMPLISHED BY INCREMENTING A
* REGISTER UP TO A LIMIT. THE INNER LOOP IS SET
* TO APPROXIMATE 1 MILLI SEC.
*****
MOV R1,-(SP) ;;PUSH R1 ON STACK
MOV R2,-(SP) ;;PUSH R2 ON STACK
MOV R3,-(SP) ;;PUSH R3 ON STACK
LET R1 := TIME(R5)
MOV TIME(R5),R1
INCRU R2 FROM #1 TO R1 BY #1
MOV #1,R2
BR $230
$231: ADD #01,R2
$230: CMP R2,R1
BHI $232
INCR R3 FROM #0 TO #100 BY #1
CLR R3
BR $233
$234: INC R3
$233: CMP R3,#100
BGT $235
ENDINC
BR $234
$235: ENDINC
BR $231
$232: MOV (SP)+,R3 ;;POP STACK INTO R3
MOV (SP)+,R2 ;;POP STACK INTO R2
MOV (SP)+,R1 ;;POP STACK INTO R1
ENDRTN
$226:
$227: RTS PC
  
```

10788
10793
10794
10795
10796 010652
10797
10798
10799
10800
10801
10806
10811
10812
10813 010652
(7) 010652 005267 000002
10814 010656
(1) 010656 000002
10815 010660 000000

```
.SBTTL INTSRV INTERRUPT SERVICE ROUTINE
*****
INTSRV:
* SERVICE ROUTINE: INTSRV
* THIS GLOBAL ROUTINE DOES NOTHING BUT INCREMENT
* 'INTFLAG' EACH TIME IT IS CALLED. IT ASSUMES
* THAT THE MAIN CALLING ROUTINE WILL KNOW WHAT
* TO LOOK FOR.
*****
;ADD 1 TO 'INTERRUPT OCCURED' FLAG
LET INTFLAG := INTFLAG + #1
INC INTFLAG
ENDSRV
RTI
;THAT'S ALL
INTFLAG: 0
```

MAINDEC-11-CNDVC-A MACY11 30(1046) 27-DEC-82 11:37 PAGE 183 F 7
CNDVCA.F11 27-DEC-82 11:36 INTSRV INTERRUPT SERVICE ROUTINE

SEQ 0083

10817

10819
10820 010662
(2) 010662
10825
10826
10827
10828
10829
10830
10831
10832
10833
10834
10839 010662
(4) 010662 112767 000000 000230
10840 010670
(4) 010670 112767 000000 000223
10841 010676
(3) 010676
10842 010676
(6) 010676 005767 000200
(9) 010702 001027
10843 010704
(6) 010704 026727 000174 000001
(9) 010712 001003
10844 010714
(4) 010714 005067 000164
10845 010720
(4) 010720 000403
(3) 010722
10846 010722
(3) 010722 004767 000370
10847
10848 010726
10849 010726
(4) 010726 012600
10850 010730
(4) 010730
10851 010730
(4) 010730 012767 000001 000144
10852 010736
(4) 010736 012767 000001 170240
10853 010744
(4) 010744 016767 170300 000134
10854 010752
(4) 010752 016767 170266 000130
10855 010760
(4) 010760 000410
(3) 010762
10856 010762
(4) 010762 012704 000010
10857 010766
(8) 010766 006167 000110
10858 010772
(7) 010772 060467 000110
10859 010776

ROUTINE CYCLE
CYCLE:

* ROUTINE: CYCLE
* THIS ROUTINE CAUSES ADRS TO POINT TO THE
* ADDRESS OF DLV11-F UNDER TEST, ADRS +2 TO
* POINT TO THE VECTOR OF THE DLV11-F UNDER TEST.
* IT KEEPS TRACK OF THE CURRENT DEVICE AND BIT
* MASKS. THE CONSOLE IS TREATED SPECIAL BY THIS ROUTINE.
* IT IS ONLY TESTED ONCE IF UNDER APT. IF NOT UNDER APT
* ALL TESTS THAT REQUIRE THE MAINT BIT ARE NOT RUN.

```

LET APTCON :B= #FALSE ; SET DEFAULT VALUE
MOV B #FALSE, APTCON
LET CONSOLE :B= #FALSE
MOV B #FALSE, CONSOLE
REPEAT ; UNTIL BITMASK SET IN $DEV M
IF BITMASK EQ #0 THEN
  TST BITMASK
  BNE $241
  IF INITFLAG EQ #1 THEN
    CMP INITFLAG, #1
    BNE $242
    LET INITFLAG := #0
  ELSE
    BR $243
  CALL $EOP ; AS A SUBROUTINE
SPECIALADDRESS: ; BECAUSE $EOP RETURNS AS A JUMP
  LET R0 := POP
  MOV (SP)+, R0
  ENDIF
  LET BITMASK := #1
  LET $DEVCT := #1
  LET ADDRESS := $BASE
  LET VECTOR := $VECT1
  ELSE
    BR $244
  LET R4 := #10
  LET BITMASK := BITMASK ROTATE 1
  LET ADDRESS := ADDRESS + R4
  LET VECTOR := VECTOR + R4

```

```

(7) 010776 060467 000106          ADD    R4,VECTOR
10860 011002                      ENDIF
(4) 011002          $244:
10861 011002                      UNTIL  BITMASK SETIN $DEV
(3) 011002 036767 000074 170242    BIT    BITMASK,$DEV
(6) 011010 001732                      BEQ    $240
10862 011012                      IF BITMASK EQ #BIT15 THEN
(6) 011012 026727 000064 100000    CMP    BITMASK,#BIT15
(9) 011020 001023                      BNE    $245
10863 011022                      LET CONSOLE :B= #TRUE
(4) 011022 112767 000001 000071    MOVB   #TRUE,CONSOLE
10864 011030                      LET ADDRESS := CONADR
(4) 011030 016767 000060 000050    MOV    CONADR,ADDRESS
10865 011036                      LET VECTOR := CONVECT
(4) 011036 016767 000054 000044    MOV    CONVECT,VECTOR
10866          .....
10867          .....
10868          .....
10869 011044                      IF #CONMAINT NOTSETIN $USWR THEN
(6) 011044 032767 000001 170142    BIT    #APTENV,$ENV
(9) 011052 001406                      BEQ    $246
10870 011054                      IF $PASS NE #0 THEN ; NOT FIRST PASS
(6) 011054 005767 170122          TST    $PASS
(9) 011060 001403                      BEQ    $247
10871          ; DEFINE DEVICE AS APT CONSOLE
10872 011062                      LET APTCON :B= #TRUE
(4) 011062 112767 000001 000030    MOVB   #TRUE,APTCON
10873 011070                      ENDIF          ; FIRST PASS
(4) 011070          $247:
10874 011070                      ENDIF          ; APT
(4) 011070          $246:
10875 011070                      ENDIF          ; BITMASK
(4) 011070          $245:
10876
10877 011070                      LET ADRS := #ADDRESS
(4) 011070 012701 011106          MOV    #ADDRESS,ADRS
10878 011074                      LET $DEVCT := $DEVCT + #1
(7) 011074 005267 170104          INC    $DEVCT
10879 011100                      RETURN
(4) 011100 000411          BR     $236
10880 011102 100000          BITMASK: 100000          ; CONSOLE FIRST
10881 011104 000001          INITFLAG: 1
10882 011106 000000          ADDRESS: 0
10883 011110 000000          VECTOR: 0
10884 011112 000000          OK: 0
10885 011114 177560          CONADR: 177560          ; CONSOLE ADDRESS
10886 011116 000060          CONVECT: 60          ; CONSOLE VECTOR
10887 011120 000          APTCON: .BYTE 0
10888 011121 000          CONSOLE: .BYTE 0
10889 011122 000          NOCONMANT: .BYTE 0
10890          011124          .EVEN
10891
10892 011124                      ENDRTN
(3) 011124          $236:
(3) 011124          $237:
(2) 011124 000207          RTS    PC
  
```

MAINDEC-11-CNDVC-A MACY11 30(1046) 27-DEC-82 11:37 PAGE 184-2¹ 7
CNDVCA.F11 27-DEC-82 11:36 RGUTINE - CYCLE

SEQ 0086

10893
10894

```

10896
10897
10898 011126          ROUTINE MYTYPE
      (2) 011126          MYTYPE:
10903 011126 104401 011134  ::*****
10904 011126 104401 011134  ::          TYPE      ,65$          ::TYPE ASCII STRING
      (1) 011132 000405          BR          64$          ::GET OVER THE ASCII
      (1)          ::65$: .ASCII <CRLF>*TEST # *
      (1) 011146          64$:
10905 011146 016746 170026          MOV      $TESTN,-(SP)      ::SAVE $TESTN FOR TYPEOUT
      (1) 011152 104402          TYPOC          ::GO TYPE--OCTAL ASCII(ALL DIGITS)
10906 011154 104401 011162          TYPE      ,67$          ::TYPE ASCII STRING
      (1) 011160 000405          BR          66$          ::GET OVER THE ASCII
      (1)          ::67$: .ASCII *,ERROR # *
      (1) 011174          66$:
10907 011174 116767 167714 167774          MOV      $ITEMB,$FATAL      :: APT FATAL ERROR NUMBER
10908 011202 016746 167770          MOV      $FATAL,-(SP)      ::SAVE $FATAL FOR TYPEOUT
      (1) 011206 104403          TYPOS          ::GO TYPE--OCTAL ASCII
      (1) 011210 006          .BYTE      6          ::TYPE 6 DIGITS
      (1) 011211 000          .BYTE      0          ::SUPPRESS LEADING ZEROS
10909 011212 104401 011220          TYPE      ,69$          ::TYPE ASCII STRING
      (1) 011216 000404          BR          68$          ::GET OVER THE ASCII
      (1)          ::69$: .ASCII *,PC = *
      (1) 011230          68$:
10910 011230 016746 167662          MOV      $ERRPC,-(SP)      ::SAVE $ERRPC FOR TYPEOUT
      (1) 011234 104402          TYPOC          ::GO TYPE--OCTAL ASCII(ALL DIGITS)
10911 011236 104401 011244          TYPE      ,71$          ::TYPE ASCII STRING
      (1) 011242 000404          BR          70$          ::GET OVER THE ASCII
      (1)          ::71$: .ASCII *,CSR: *
      (1) 011254          70$:
10912 011254 016746 167774          MOV      DLADD,-(SP)      ::SAVE DLADD FOR TYPEOUT
      (1) 011260 104402          TYPOC          ::GO TYPE--OCTAL ASCII(ALL DIGITS)
10913 011262 104401 011270          TYPE      ,73$          ::TYPE ASCII STRING
      (1) 011266 000405          BR          72$          ::GET OVER THE ASCII
      (1)          ::73$: .ASCII *,VECTOR: *
      (1) 011302          72$:
10914 011302 016746 167750          MOV      DLVEC,-(SP)      ::SAVE DLVEC FOR TYPEOUT
      (1) 011306 104402          TYPOC          ::GO TYPE--OCTAL ASCII(ALL DIGITS)
10915 011310 104401 001171          TYPE      ,$CRLF
10920 011314          ENDRTN
      (3) 011314          $250:
      (3) 011314          $251:
      (2) 011314 000207          RTS      PC
  
```


10922
 10927

.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 SPECIALADDRESS
  
```

\$EOP:

```

SCOPE
CLR $TSTNM ;;ZERC 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

```

$EOPCT
TYPE $SENDMG ;;TYPE 'END PASS #'
MOV $PASS,-(SP) ;;SAVE $PASS FOR TYPEOUT
TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN
TYPE $SENULL ;;TYPE A NULL CHARACTER
  
```

```

$GET42: MOV @#42,R0 ;;GET MONITOR ADDRESS
BEQ $DOAGN ;;BRANCH IF NO MONITOR
RESET ;;CLEAR THE WORLD
  
```

\$ENDAD: JSR PC,(R0) ;;GO TO MONITOR

```

NOP ;;SAVE ROOM
NOP ;;FOR
NOP ;;ACT11
  
```

\$DOAGN: JMP @(PC)+ ;;RETURN

\$RTNAD: .WORD SPECIALADDRESS

\$ENULL: .BYTE -1,-1,0 ;;NULL CHARACTER STRING

\$ENDMG: .ASCIZ <15><12>/END PASS #/

```

(1) 011316
(1) 011316 000004
(1) 011320 005067 167556
(1) 011324 005067 167630
(1) 011330 005267 167646
(1) 011334 042767 100000 167640
(1) 011342 005327
(1) 011344 000001
(1) 011346 003022
(1) 011350 012737
(1) 011352 000001
(1) 011354 011344
(1) 011356 104401 011423
(2) 011362 016746 167614
(2) 011366 104405
(1) 011370 104401 011420
(1) 011374 013700 000042
(1) 011400 001405
(1) 011402 000005
(1) 011404 004710
(1) 011406 000240
(1) 011410 000240
(1) 011412 000240
(1) 011414
(1) 011414 000137
(1) 011416 010726
(1)
(1)
(1) 011420 377 377 000
(1) 011423 015 042412 042116
(1) 011430 050040 051501 020123
(1) 011436 000043
  
```

10929
10930

.SBTTL POWER DOWN AND UP ROUTINES

```
(1) (2) (1)
(1) 011440 012737 011604 000024 $PWRDN: MOV #SILLUP,@PWRVEC ;;SET FOR FAST UP
(1) 011446 012737 000300 000026 MOV #PR6,@PWRVEC+2 ;;PRIO:6
(3) 011454 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
(3) 011456 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
(3) 011460 010246 MOV R2,-(SP) ;;PUSH R2 ON STACK
(3) 011462 010346 MOV R3,-(SP) ;;PUSH R3 ON STACK
(3) 011464 010446 MOV R4,-(SP) ;;PUSH R4 ON STACK
(3) 011466 010546 MOV R5,-(SP) ;;PUSH R5 ON STACK
(3) 011470 017746 167444 MOV @SWR,-(SP) ;;PUSH @SWR ON STACK
(1) 011474 010667 000110 MOV SP,$SAVR6 ;;SAVE SP
(1) 011500 012737 011512 000024 MOV $PWRUP,@PWRVEC ;;SET UP VECTOR
(1) 011506 000000 HALT
(1) 011510 000776 BR .-2 ;;HANG UP
```

```
(1) (2) (1)
(1) 011512 012737 011604 000024 $PWRUP: MOV #SILLUP,@PWRVEC ;;SET FOR FAST DOWN
(1) 011520 016706 000064 MOV $SAVR6,SP ;;GET SP
(1) 011524 005067 000060 CLR $SAVR6 ;;WAIT LOOP FOR THE TTY
(1) 011530 005267 000054 1$: INC $SAVR6 ;;WAIT FOR THE INC
(1) 011534 001375 BNE 1$ ;;OF WORD
(3) 011536 012677 167376 MOV (SP)+,@SWR ;;POP STACK INTO @SWR
(3) 011542 012605 MCV (SP)+,R5 ;;POP STACK INTO R5
(3) 011544 012604 MOV (SP)+,R4 ;;POP STACK INTO R4
(3) 011546 012603 MOV (SP)+,R3 ;;POP STACK INTO R3
(3) 011550 012602 MOV (SP)+,R2 ;;POP STACK INTO R2
(3) 011552 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
(3) 011554 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
(1) 011556 012737 011440 000024 MOV $PWRDN,@PWRVEC ;;SET UP THE POWER DOWN VECTOR
(1) 011564 012737 000300 000026 MOV #PR6,@PWRVEC+2 ;;PRIO:6
(1) 011572 104401 TYPE $PWRMG: .WORD $POWER ;;REPORT THE POWER FAILURE
(1) 011574 011612 $PWRMG: MOV (PC)+,(SP) ;;POWER FAIL MESSAGE POINTER
(1) 011576 012716 $PWRAD: .WORD START ;;RESTART AT START
(1) 011600 001336 RTI ;;RESTART ADDRESS
(1) 011604 000000 $ILLUP: HALT ;;THE POWER UP SEQUENCE WAS STARTED
(1) 011606 000776 BR .-2 ;;BEFORE THE POWER DOWN WAS COMPLETE
(1) 011610 000000 $SAVR6: 0 ;;PUT THE SP HERE
(1) 011612 005015 047520 042527 $POWER: .ASCIZ <15><12>'POWER'
(1) 011620 000122 .EVEN
```



```

(1) 012002 105367 000072          DECB   $CHARCNT      ;;DO NOT COUNT AS A COUNT
(1) 012006 000770                BR     7$           ;;LOOP
(1)
(1)                                ;HORIZONTAL TAB PROCESSOR
(1)
(1) 012010 112716 000040          8$:   MOVB   #' (SP)      ;;REPLACE TAB WITH SPACE
(1) 012014 004767 000014          9$:   JSR    PC,$TYPEC    ;;TYPE A SPACE
(1) 012020 132767 000007 000052  BITB   #7,$CHARCNT     ;;BRANCH IF NOT AT
(1) 012026 001372                BNE    9$           ;;TAB STOP
(1) 012030 005726                TST   (SP)+         ;;POP SPACE OFF STACK
(1) 012032 000724                BR     2$           ;;GET NEXT CHARACTER
(1) 012034 105777 167110          $TYPEC: TSTB  @2$PS     ;;WAIT UNTIL PRINTER IS READY
(1) 012040 100375                BPL   $TYPEC
(1) 012042 116677 000002 167102  MOVB   2(SP),@2$PB    ;;LOAD CHAR TO BE TYPED INTO DATA REG.
(1)
(1) 012050 122766 000015 000002  CMPB   #CR,2(SP)      ;;IS CHARACTER A CARRIAGE RETURN?
(1) 012056 001003                BNE   1$           ;;BRANCH IF NO
(1) 012060 105067 000014          CLRB   $CHARCNT     ;;YES--CLEAR CHARACTER COUNT
(1) 012064 000406                BR     $TYPEX       ;;EXIT
(1) 012066 122766 000012 000002  1$:   CMPB   #LF,2(SP)  ;;IS CHARACTER A LINE FEED?
(1) 012074 001402                BEQ   $TYPEX       ;;BRANCH IF YES
(1) 012076 105227                INCB  (PC)+         ;;COUNT THE CHARACTER
(1) 012100 000000          $CHARCNT: .WORD 0    ;;CHARACTER COUNT STORAGE
(1) 012102 000207          $TYPEX: RTS      PC
(1)
(1)
(1)

```

10935
10936

.SBTTL TTY INPUT ROUTINE

.ENABL LSB

*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
*WHEN OPERATING IN TTY FLAG MODE.

(1)	012104	022767	000176	167026	\$CKSWR: CMP	#SWREG,SWR	:: IS THE SOFT-SWR SELECTED?
(1)	012112	001074			BNE	15\$:: BRANCH IF NO
(1)	012114	105777	167024		TSTB	@\$TKS	:: CHAR THERE?
(1)	012120	100071			BPL	15\$:: IF NO, DON'T WAIT AROUND
(1)	012122	117746	167020		MOVB	@\$TKB,-(SP)	:: SAVE THE CHAR
(1)	012126	042716	177600		BIC	#^C177,(SP)	:: STRIP-OFF THE ASCII
(1)	012132	022726	000007		CMP	#7,(SP)+	:: IS IT A CONTROL G?
(1)	012136	001062			BNE	15\$:: NO, RETURN TO USER
(1)	012140	126727	166770	000001	CMPB	\$AUTOB,#1	:: ARE WE RUNNING IN AUTO-MODE?
(1)	012146	001456			BEQ	15\$:: BRANCH IF YES
(1)	012150	104401	012631		TYPE	,\$CNTLG	:: ECHO THE CONTROL-G (^G)
(1)	012154	104401	012636		\$GTSWR: TYPE	,\$MSWR	:: TYPE CURRENT CONTENTS
(2)	012160	016746	166012		MOV	SWREG,-(SP)	:: SAVE SWREG FOR TYPEOUT
(2)	012164	104402			TYPOC		:: GO TYPE--OCTAL ASCII(ALL DIGITS)
(1)	012166	104401	012647		TYPE	,\$MNEW	:: PROMPT FOR NEW SWR
(1)	012172	005046			19\$: CLR	-(SP)	:: CLEAR COUNTER
(1)	012174	005046			CLR	-(SP)	
(1)	012176	105777	166742		7\$: TSTB	NEW SWR @\$TKS	:: CHAR THERE?
(1)	012202	100375			BPL	7\$:: IF NOT TRY AGAIN
(1)	012204	117746	166736		MOVB	@\$TKB,-(SP)	:: PICK UP CHAR
(1)	012210	042716	177600		BIC	#^C177,(SP)	:: MAKE IT 7-BIT ASCII
(1)							
(1)	012214	021627	000025		9\$: CMP	(SP),#25	:: IS IT A CONTROL-U?
(1)	012220	001005			BNE	10\$:: BRANCH IF NOT
(1)	012222	104401	012624		TYPE	,\$CNTLU	:: YES, ECHO CONTROL-U (^U)
(1)	012226	062706	000006		20\$: ADD	#6,SP	:: IGNORE PREVIOUS INPUT
(1)	012232	000757			BR	19\$:: LET'S TRY IT AGAIN
(1)							
(1)	012234	021627	000015		10\$: CMP	(SP),#15	:: IS IT A <CR>?
(1)	012240	001022			BNE	16\$:: BRANCH IF NO
(1)	012242	005766	000004		BT	4(SP)	:: YES, IS IT THE FIRST CHAR?
(1)	012246	001403			BEQ	11\$:: BRANCH IF YES
(1)	012250	016677	000002	166662	MOV	2(SP),@SWR	:: SAVE NEW SWR
(1)	012256	062706	000006		11\$: ADD	#6,SP	:: CLEAR UP STACK
(1)	012262	104401	001171		14\$: TYPE	,\$CRLF	:: ECHO <CR> AND <LF>
(1)	012266	126727	166643	000001	CMPB	\$INTAG,#1	:: RE-ENABLE TTY KBD INTERRUPTS?
(1)	012274	001003			BNE	15\$:: BRANCH IF NOT

```

(1) 012276 012777 000100 166640      MOV      #100,@$TKS      ;;RE-ENABLE TTY KBD INTFRRUPTS
(1) 012304 000002      15$: RTI                ;;RETURN
(1) 012306 004767 177522      16$: JSR      PC,$TYPEC  ;;ECHO CHAR
(1) 012312 021627 000060      CMP      (SP),#60      ;;CHAR < 0?
(1) 012316 002420      BLT      18$           ;;BRANCH IF YES
(1) 012320 021627 000067      CMP      (SP),#67      ;;CHAR > 7?
(1) 012324 003015      BGT      18$           ;;BRANCH IF YES
(1) 012326 042726 000060      BIC      #60,(SP)+     ;;STRIP-CrF ASCII
(1) 012332 005766 000002      TST      2(SP)         ;;IS THIS THE FIRST CHAR
(1) 012336 001403      BEQ      17$           ;;BRANCH IF YES
(1) 012340 006316      ASL      (SP)          ;;NO, SHIFT PRESENT
(1) 012342 006316      ASL      (SP)          ;;CHAR OVER TO MAKE
(1) 012344 006316      ASL      (SP)          ;;ROOM FOR NEW ONE.
(1) 012346 005266 000002      17$: INC      2(SP)     ;;KEEP COUNT OF CHAR
(1) 012352 056616 177776      BIS      -2(SP),(SP)  ;;SET IN NEW CHAR
(1) 012356 000707      BR       7$           ;;GET THE NEXT ONE
(1) 012360 104401 001170      18$: TYPE     ,$QUES   ;;TYPE ?<CR><LF>
(1) 012364 000720      BR       20$          ;;SIMULATE CONTROL-U
(1) .DSABL  LSB
  
```

```

(1) *****
(1) *THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
(1) *CALL:
(1) * RDCHR                ;;INPUT A SINGLE CHARACTER FROM THE TTY
(1) * RETURN HERE         ;;CHARACTER IS ON THE STACK
(1) *                     ;;WITH PARITY BIT STRIPPED OFF
(1) *
  
```

```

(1) 012366 011646      $RDCHR: MOV      (SP),-(SP)  ;;PUSH DOWN THE PC
(1) 012370 016666 000004 000002  MOV      4(SP),2(SP)  ;;SAVE THE PS
(1) 012376 105777 166542 1$: TSTB     @$TKS     ;;WAIT FOR
(1) 012402 100375      BPL      1$           ;;A CHARACTER
(1) 012404 117766 166536 000004  MOVB     @$TKB,4(SP)  ;;READ THE TTY
(1) 012412 042766 177600 000004  BIC      #^C<177>,4(SP) ;;GET RID OF JUNK IF ANY
(1) 012420 026627 000004 000023  CMP      4(SP),#23   ;;IS IT A CONTROL-S?
(1) 012426 001013      BNE      3$           ;;BRANCH IF NO
(1) 012430 105777 166510 2$: TSTB     @$TKS     ;;WAIT FOR A CHARACTER
(1) 012434 100375      BPL      2$           ;;LOOP UNTIL ITS THERE
(1) 012436 117746 166504      MOVB     @$TKB,-(SP)  ;;GET CHARACTER
(1) 012442 042716 177600      BIC      #^C<177>,(SP) ;;MAKE IT 7-BIT ASCII
(1) 012446 022627 000021      CMP      (SP)+,#21   ;;IS IT A CONTROL-Q?
(1) 012452 001366      BNE      2$           ;;IF NOT DISCARD IT
(1) 012454 000750      BR       1$           ;;YES, RESUME
(1) 012456 026627 000004 000140 3$: CMP      4(SP),#140  ;;IS IT UPPER CASE?
(1) 012464 002407      BLT      4$           ;;BRANCH IF YES
(1) 012466 026627 000004 000175  CMP      4(SP),#175  ;;IS IT A SPECIAL CHAR?
(1) 012474 003003      BGT      4$           ;;BRANCH IF YES
(1) 012476 042766 000040 000004  BIC      #40,4(SP)   ;;MAKE IT UPPER CASE
(1) 012504 000002      4$: RTI                ;;GO BACK TO USER
  
```

```

(1) *****
(1) *THIS ROUTINE WILL INPUT A STRING FROM THE TTY
(1) *CALL:
(1) * RDLIN                ;;INPUT A STRING FROM THE TTY
(1) * RETURN HERE         ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
(1) *                     ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
  
```

```

(1)
(1) 012506 010346 $RDLIN: MOV R3,-(SP) ;;SAVE R3
(1) 012510 012703 012614 1$: MOV #$TTYIN,R3 ;;GET ADDRESS
(1) 012514 022703 012624 2$: CMP #$TTYIN+8.,R3 ;;BUFFER FULL?
(1) 012520 101405 BLOS 4$ ;;BR IF YES
(1) 012522 104410 RDCHR ;;GO READ ONE CHARACTER FROM THE TTY
(1) 012524 112613 MOVB (SP)+,(R3) ;;GET CHARACTER
(1) 012526 122713 000177 10$: CMPB #177,(R3) ;;IS IT A PUBOUT
(1) 012532 001003 BNE 3$ ;;SKIP IF NOT
(1) 012534 104401 001170 4$: TYPE ,SQUES ;;TYPE A '?'
(1) 012540 000763 BR 1$ ;;CLEAR THE BUFFER AND LOOP
(1) 012542 111367 000044 3$: MOVB (R3),9$ ;;ECHO THE CHARACTER
(1) 012546 104401 012612 TYPE ,9$
(1) 012552 122723 000015 CMPB #15,(R3)+ ;;CHECK FOR RETURN
(1) 012556 001356 BNE 2$ ;;LOOP IF NOT RETURN
(1) 012560 105063 177777 CLR B -1(R3) ;;CLEAR RETURN (THE 15)
(1) 012564 104401 001172 TYPE ,SLF ;;TYPE A LINE FEED
(1) 012570 012603 MOV (SP)+,R3 ;;RESTORE R3
(1) 012572 011646 MOV (SP),-(SP) ;;ADJUST THE STACK AND PUT ADDRESS OF THE
(1) 012574 016666 000004 000002 MOV 4(SP),2(SP) ;; FIRST ASCII CHARACTER ON IT
(1) 012602 012766 012614 000004 MOV #$TTYIN,4(SP)
(1) 012610 000002 RTI ;;RETURN
(1) 012612 000 9$: .BYTE 0 ;;STORAGE FOR ASCII CHAR. TO TYPE
(1) 012613 000 .BYTE 0 ;;TERMINATOR
(1) 012614 000010 $TTYIN: .BLKB 8. ;;RESERVE 8 BYTES FOR TTY INPUT
(1) 012624 052536 005015 000 $CNTLU: .ASCIZ /^U/<15><12> ;;CONTROL 'U'
(1) 012631 136 006507 000012 $CNTLG: .ASCIZ /^G/<15><12> ;;CONTROL 'G'
(1) 012636 005015 053523 020122 $MSWR: .ASCIZ <15><12>/SWR = /
(1) 012644 020075 000
(1) 012647 040 047040 053505 $MNEW: .ASCIZ / NEW = /
(1) 012654 036440 000040
  
```

10938
10939

.SBTTL APT COMMUNICATIONS ROUTINE

```
*****  
(1) 012660 112767 000001 000236 $ATY1: MOVB #1,$FFLG ;;TO REPORT FATAL ERROR  
(1) 012666 112767 000001 000226 $ATY3: MOVB #1,$MFLG ;;TO TYPE A MESSAGE  
(1) 012674 000403 BR $ATYC  
(1) 012676 112767 000001 000220 $ATY4: MOVB #1,$FFLG ;;TO ONLY REPORT FATAL ERROR  
(1) 012704 $ATYC:  
(3) 012704 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK  
(3) 012706 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK  
(1) 012710 105767 000206 TSTB $MFLG ;;SHOULD TYPE A MESSAGE?  
(1) 012714 001450 BEQ 5$ ;;IF NOT: BR  
(1) 012716 122767 000001 166270 CMPB #APTENV,$ENV ;;OPERATING UNDER APT?  
(1) 012724 001031 BNE 3$ ;;IF NOT: BR  
(1) 012726 132767 000100 166261 BITB #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?  
(1) 012734 001425 BEQ 3$ ;;IF NOT: BR  
(1) 012736 017600 000004 MOV @4(SP),R0 ;;GET MESSAGE ADDR.  
(1) 012742 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.  
(1) 012750 005767 166220 1$: TST $MSGTYPE ;;SEE IF DONE W/ LAST XMISSION?  
(1) 012754 001375 BNE 1$ ;;IF NOT: WAIT  
(1) 012756 010067 166226 MOV R0,$MSGAD  
(1) ;;PUT ADDR IN MAILBOX  
(1) 012762 105720 2$: TSTB (R0)+ ;;FIND END OF MESSAGE  
(1) 012764 001376 BNE 2$  
(1) 012766 166700 166216 SUB $MSGAD,R0 ;;SUB START OF MESSAGE  
(1) 012772 006200 ASR R0 ;;GET MESSAGE LGTH IN WORDS  
(1) 012774 010067 166212 MCV R0,$MSGLGT ;;PUT LENGTH IN MAILBOX  
(1) 013000 012767 000004 166166 MOV #4,$MSGTYPE ;;TELL APT TO TAKE MSG.  
(1) 013006 000413 BR 5$  
(1) 013010 017667 000004 000016 3$: MOV @4(SP),4$ ;;PUT MSG ADDR IN JSR LINKAGE  
(1) 013016 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDRESS  
(3) 013024 016746 164746 MOV 177776,-(SP) ;;PUSH 177776 ON STACK  
(1) 013030 004767 176566 JSR PC,$TYPE ;;CALL TYPE MACRO  
(1) 013034 000000 4$: .WORD 0  
(1) 013036 5$:  
(1) 013036 105767 000062 10$: TSTB $FFLG ;;SHOULD REPORT FATAL ERROR?  
(1) 013042 001416 BEQ 12$ ;;IF NOT: BR  
(1) 013044 005767 166144 TST $ENV ;;RUNNING UNDER APT?  
(1) 013050 001413 BEQ 12$ ;;IF NOT: BR  
(1) 013052 005767 166116 11$: TST $MSGTYPE ;;FINISHED LAST MESSAGE?  
(1) 013056 001375 BNE 11$ ;;IF NOT: WAIT  
(1) 013060 017667 000004 166110 MOV @4(SP),$FATAL ;;GET ERROR #  
(1) 013066 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.  
(1) 013074 005267 166074 INC $MSGTYPE ;;TELL APT TO TAKE ERROR  
(1) 013100 105067 000020 12$: CLRB $FFLG ;;CLEAR FATAL FLAG  
(1) 013104 105067 000013 CLRB $LFLG ;;CLEAR LOG FLAG  
(1) 013110 105067 000006 CLRB $MFLG ;;CLEAR MESSAGE FLAG  
(3) 013114 012601 MOV (SP)+,R1 ;;POP STACK INTO R1  
(3) 013116 012600 MOV (SP)+,R0 ;;POP STACK INTO R0  
(1) 013120 000207 RTS PC ;;RETURN  
(1) 013122 000 $MFLG: .BYTE 0 ;;MESSG. FLAG  
(1) 013123 000 $LFLG: .BYTE 0  
(1) ;;LOG FLAG  
(1) 013124 000 $FFLG: .BYTE 0 ;;FATAL FLAG  
(1) 013126 .EVEN
```


(1)	000200	APTSIZE=200
(1)	000001	APTENV=001
(1)	000100	APTPOOL=100
(1)	000040	APTCSUP=040


```

10944
10945
(1) .SBTTL SCOPE HANDLER ROUTINE STARTS
(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) $SCOPE:
(1) 013326 104407 CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
(1) 013330 032777 040000 165602 1$: BIT #BIT14,@SWR ;;LOOP ON PRESENT TEST?
(1) 013336 001114 BNE $OVER ;;YES IF SW14=1
(1) ;*****START OF CODE FOR THE XOR TESTER*****
(1) $XTSTR: BR 6$ ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
(1) 013340 000416 MOV @#ERRVEC,-(SP) ;;THIS INSTRUCTION TO A 'NOP' (NOP=240)
(1) 013342 013746 000004 MOV #5$,@#ERRVEC ;;SAVE THE CONTENTS OF THE ERROR VECTOR
(1) 013346 012737 013366 000004 TST @#177060 ;;SET FOR TIMEOUT
(1) 013354 005737 177060 (SP)+,@#ERRVEC ;;TIME OUT ON XOR?
(1) 013360 012637 000004 BR $SVLAD ;;RESTORE THE ERROR VECTOR
(1) 013364 000463 $SVLAD ;;GO TO THE NEXT TEST
(1) 013366 022626 5$: CMP (SP)+,(SP)+ ;;CLEAR THE STACK AFTER A TIME OUT
(1) 013370 012637 000004 MOV (SP)+,@#ERRVEC ;;RESTORE THE ERROR VECTOR
(1) 013374 000423 BR 7$ ;;LOOP ON THE PRESENT TEST
(1) 013376 6$:;*****END OF CODE FOR THE XOR TESTER*****
(1) 013376 032777 000400 165534 BIT #BIT08,@SWR ;;LOOP ON SPEC. TEST?
(1) 013404 001404 BEQ 2$ ;;BR IF NO
(1) 013406 127767 165526 165466 CMPB @SWR,$TSTNM ;;ON THE RIGHT TEST? SWR<7:0>
(1) 013414 001465 BEQ $OVER ;;BR IF YES
(1) 013416 105767 165461 2$: TSTB $ERFLG ;;HAS AN ERROR OCCURRED?
(1) 013422 001421 BEQ 3$ ;;BR IF NO
(1) 013424 126767 165465 165451 CMPB $ERMAX,$ERFLG ;;MAX. ERRORS FOR THIS TEST OCCURRED?
(1) 013432 101015 BHI 3$ ;;BR IF NO
(1) 013434 032777 001000 165476 BIT #BIT09,@SWR ;;LOOP ON ERROR?
(1) 013442 001404 BEQ 4$ ;;BR IF NO
(1) 013444 016767 165440 165434 7$: MOV $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
(1) 013452 000446 BR $OVER
(1) 013454 105067 165423 4$: CLRB $ERFLG ;;ZERO THE ERROR FLAG
(1) 013460 005067 165474 CLR $TIMES ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
(1) 013464 000415 BR 1$ ;;ESCAPE TO THE NEXT TEST
(1) 013466 032777 004000 165444 3$: BIT #BIT11,@SWR ;;INHIBIT ITERATIONS?
(1) 013474 001011 BNE 1$ ;;BR IF YES
(1) 013476 005767 165500 TST $PASS ;;IF FIRST PASS OF PROGRAM
(1) 013502 001406 BEQ 1$ ;; INHIBIT ITERATIONS
(1) 013504 005267 165374 INC $ICNT ;;INCREMENT ITERATION COUNT
(1) 013510 026767 165444 165366 CMP $TIMES,$ICNT ;;CHECK THE NUMBER OF ITERATIONS MADE
(1) 013516 002024 BGE $OVER ;;BR IF MORE ITERATION REQUIRED
(1) 013520 012767 000001 165356 1$: MOV #1,$ICNT ;;REINITIALIZE THE ITERATION COUNTER
(1) 013526 016767 000052 165424 $SVLAD: MOV $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
(1) 013534 105267 165342 MOV $TSTNM ;;COUNT TEST NUMBERS
(1) 013540 116767 165336 165432 MOVB $TSTNM,$TESTN ;;SET TEST NUMBER IN APT MAILBOX
(1) 013546 011667 165334 MOV (SP),$LPADR ;;SAVE SCOPE LOOP ADDRESS

```

(1)	013552	011667	165332		MOV	(SP), \$LPERR	:: SAVE ERROR LOOP ADDRESS
(1)	013556	005067	165400		CLR	\$ESCAPE	:: CLEAR THE ESCAPE FROM ERROR ADDRESS
(1)	013562	112767	000001	165325	MOVB	#1, \$ERMAX	:: ONLY ALLOW ONE(1) ERROR ON NEXT TEST
(1)	013570	016777	165306	165344	\$OVER: MOV	\$STNM, @DISPLAY	:: DISPLAY TEST NUMBER
(1)	013576	016716	165304		MOV	\$LPADR, (SP)	:: FUDGE RETURN ADDRESS
(1)	013602	000002			RTI		:: FIXES PS
(1)	013604	003720			\$MXCNT: 2000.		:: MAX. NUMBER OF ITERATIONS

(3)	013766	012602			MOV	(SP)+,R2	::POP STACK INTO R2
(3)	013770	012601			MOV	(SP)+,R1	::POP STACK INTO R1
(3)	013772	012600			MOV	(SP)+,R0	::POP STACK INTO R0
(1)	013774	104401	014022		TYPE	\$DBLK	::NOW TYPE THE NUMBER
(1)	014000	016666	000002	000004	MOV	2(SP),4(SP)	::ADJUST THE STACK
(1)	014006	012616			MOV	(SP)+,(SP)	
(1)	014010	000002			RTI		::RETURN TO USER
(1)	014012	023420			\$DTBL:	10000.	
(1)	014014	001750				1000.	
(1)	014016	000144				100.	
(1)	014020	000012				10.	
(1)	014022	000004			\$DBLK:	.BLKW 4	

(1)	014172	001403			BEQ	5\$::BR IF YES
(1)	014174	005204			4\$: INC	R4		::DON'T SUPPRESS ANYMORE 0'S
(1)	014176	052703	000060		BIS	#'0,P3		::MAKE THIS DIGIT ASCII
(1)	014202	052703	000040		5\$: BIS	#',R3		::MAK: ASCII IF NOT ALREADY
(1)	014206	110367	000040		MOVB	R3,8\$::SAVE FOR TYPING
(1)	014212	104401	014252		TYPE	,8\$::GO TYPE THIS DIGIT
(1)	014216	105367	000032		7\$: DECB	\$OCNT		::COUNT BY 1
(1)	014222	003347			BGT	2\$::BR IF MORE TO DO
(1)	014224	002402			BLT	6\$::BR IF DONE
(1)	014226	005204			INC	R4		::INSURE LAST DIGIT ISN'T A BLANK
(1)	014230	000744			BR	2\$::GO DO THE LAST DIGIT
(1)	014232	012605			6\$: MOV	(SP)+,R5		::RESTORE R5
(1)	014234	012604			MOV	(SP)+,R4		::RESTORE R4
(1)	014236	012603			MOV	(SP)+,R3		::RESTORE R3
(1)	014240	016666	000002	000004	MOV	2(SP),4(SP)		::SET THE STACK FOR RETURNING
(1)	014246	012616			MOV	(SP)+,(SP)		
(1)	014250	000002			RTI			::RETURN
(1)	014252	000			8\$: .BYTE	0		::STORAGE FOR ASCII DIGIT
(1)	014253	000			.BYTE	0		::TERMINATOR FOR TYPE ROUTINE
(1)	014254	000			\$OCNT: .BYTE	0		::OCTAL DIGIT COUNTER
(1)	014255	000			\$OFI.L: .BYTE	0		::ZERO FILL SWITCH
(1)	014256	000000			\$OMOJE: .WORD	0		::NUMBER OF DIGITS TO TYPE

(1)	014362	051122	050125	020124	
(1)	014370	020055	044504	041523	
(1)	014376	047117	042516	052103	
(1)	014404	046040	041524	000040	
10957		000001			.END

\$LOCTA= 177777

7962#	8994	8999	9042	9049	9052	9056	9084	9086	9091	9094	9101	9104
9112	9115	9124	9127	9145	9147	9152	9155	9162	9165	9173	9176	9185
9188	9220	9223	9230	9233	9241	9244	9253	9256	9279	9282	9289	9292
9300	9303	9312	9315	9343	9350	9374	9376	9377	9379	9380	9382	9384
9385	9386	9390	9402	9426	9433	9458	9460	9463	9480	9483	9494	9499
9504	9507	9508	9509	9510	9514	9515	9517	9518	9519	9542	9544	9560
9565	9574	9577	9593	9595	9611	9614	9623	9626	9644	9646	9647	9649
9661	9662	9664	9666	9667	9668	9675	9682	9684	9685	9693	9694	9695
9698	9704	9712	9717	9738	9740	9741	9743	9768	9776	9781	9791	9797
9805	9816	9825	9827	9833	9858	9860	9862	9864	9871	9882	9883	9886
9890	9892	9894	9896	9898	9900	9906	9907	9912	9919	9922	9979	9981
10018	10020	10023	10027	10028	10048	10051	10081	10083	10120	10124	10128	10131
10132	10161	10163	10185	10191	10193	10199	10201	10214	10220	10221	10225	10227
10228	10230	10247	10250	10278	10285	10288	10294	10298	10308	10312	10314	10343
10345	10406	10407	10411	10414	10454	10457	10496	10498	10502	10505	10508	10514
10544	10546	10547	10549	10560	10563	10565	10568	10569	10570	10572	10573	10574
10578	10580	10581	10583	10622	10661	10663	10665	10667	10670	10673	10675	10681
10687	10688	10689	10690	10697	10710	10735	10738	10740	10742	10755	10773	10774
10775	10776	10786	10820	10841	10842	10843	10845	10850	10855	10860	10861	10862
10869	10870	10873	10874	10875	10879	10892	10898	10920				

\$LPADR 001106
\$LPERR 001110

8971#	8986*	10945*										
8971#	8986*	9043*	9089*	9098*	9108*	9119*	9150*	9159*	9169*	9180*	9218*	9227*
9237*	9248*	9277*	9286*	9296*	9307*	9342*	9388*	9424*	9549*	9569*	9599*	9749*
9780*	9794*	9808*	9993*	10031*	10089*	10942	10945*					

\$LSTCN= 177777

7962#	7965#	8994	8999	9042	9049	9052	9056	9084	9086	9091	9094	9101
9104	9112	9115	9124	9127	9145	9147	9152	9155	9162	9165	9173	9176
9185	9188	9220	9223	9230	9233	9241	9244	9253	9256	9279	9282	9289
9292	9300	9303	9312	9315	9343	9350	9374	9376	9377	9379	9380	9382
9384	9385	9386	9390	9402	9426	9433	9458	9460	9463	9480	9483	9494
9499	9504	9507	9508	9509	9510	9514	9515	9517	9519	9542	9544	9560
9565	9574	9577	9593	9595	9611	9614	9623	9626	9644	9646	9647	9649
9661	9662	9664	9666	9667	9668	9675	9682	9684	9685	9693	9694	9695
9698	9704	9712	9717	9738	9740	9741	9743	9768	9776	9781	9791	9797
9805	9816	9825	9827	9833	9858	9860	9862	9864	9871	9882	9883	9886
9890	9892	9894	9896	9898	9900	9906	9907	9912	9919	9922	9979	9981
10018	10020	10023	10027	10028	10048	10051	10081	10083	10120	10124	10128	10131
10132	10161	10163	10185	10191	10193	10199	10201	10214	10220	10221	10225	10227
10228	10230	10247	10250	10278	10285	10288	10294	10298	10308	10312	10314	10343
10345	10406	10407	10411	10414	10454	10457	10496	10498	10502	10505	10508	10514
10544	10546	10547	10549	10560	10563	10565	10568	10569	10570	10572	10573	10574
10578	10580	10581	10583	10622	10661	10663	10665	10667	10670	10673	10681	10687
10689	10697	10710	10735	10738	10742	10755	10773	10774	10775	10776	10786	10820
10841	10842	10843	10845	10850	10855	10860	10861	10862	10869	10870	10873	10874
10875	10892	10898	10920									

\$LSTIN= 000000

7962#	7963#	8994	8999	9000	9001	9003	9006	9008	9009	9011	9012	9013
9014	9015	9016	9038	9040	9041	9043	9045	9049	9054	9055	9056	9057
9084	9085	9089	9091	9098	9099	9101	9108	9110	9112	9119	9121	9124
9145	9146	9150	9152	9159	9160	9162	9169	9171	9173	9180	9182	9185
9218	9220	9227	9228	9230	9237	9239	9241	9248	9250	9253	9277	9279
9286	9287	9289	9296	9298	9300	9307	9309	9312	9342	9343	9374	9376
9377	9379	9380	9382	9383	9388	9390	9393	9424	9426	9458	9459	9462
9465	9466	9473	9477	9480	9490	9494	9496	9499	9503	9504	9509	9510
9513	9515	9516	9518	9519	9542	9543	9547	9549	9552	9557	9560	9563
9569	9574	9593	9594	9598	9599	9604	9608	9609	9611	9621	9623	9644
9645	9647	9648	9653	9654	9655	9660	9662	9663	9664	9665	9667	9668
9672	9674	9682	9684	9685	9689	9692	9695	9698	9701	9703	9710	9712

9715	9738	9739	9741	9742	9745	9749	9755	9757	9760	9762	9765	9768
9771	9775	9780	9781	9785	9790	9794	9797	9801	9804	9808	9812	9814
9816	9819	9824	9827	9830	9858	9859	9862	9863	9866	9867	9868	9869
9871	9872	9874	9876	9878	9880	9881	9882	9883	9885	9886	9888	9889
9892	9894	9896	9892	9900	9905	9909	9910	9912	9913	9914	9915	9918
9919	9979	9980	9985	9988	9990	9992	9993	9995	9998	10012	10015	10018
10020	10023	10031	10033	10035	10047	10048	10055	10081	10087	10089	10090	10092
10094	10108	10112	10114	10116	10118	10120	10123	10124	10128	10137	10141	10143
10144	10161	10162	10165	10167	10181	10182	10185	10190	10191	10192	10196	10198
10199	10200	10203	10210	10211	10214	10217	10219	10221	10224	10225	10228	10247
10249	10252	10262	10275	10276	10278	10284	10285	10287	10291	10293	10294	10297
10301	10304	10305	10308	10311	10314	10318	10343	10344	10352	10377	10379	10380
10383	10384	10387	10389	10391	10393	10397	10400	10402	10407	10409	10411	10416
10417	10448	10450	10452	10454	10456	10489	10491	10493	10496	10498	10500	10501
10504	10508	10510	10544	10545	10547	10548	10551	10553	10555	10557	10559	10560
10565	10567	10568	10569	10570	10571	10578	10581	10654	10655	10656	10663	10664
10665	10666	10670	10672	10675	10679	10679	10681	10687	10688	10690	10697	10732
10733	10735	10736	10737	10738	10739	10739	10742	10772	10773	10774	10775	10776
10786	10813	10839	10840	10842	10843	10844	10845	10846	10849	10851	10852	10853
10854	10855	10856	10857	10858	10859	10861	10862	10863	10864	10865	10869	10870
10872	10877	10878	10879	10892	10920							
7962#	7966#	8994	8999	9042	9049	9052	9056	9084	9086	9091	9094	9101
9104	9112	9115	9124	9127	9145	9147	9152	9155	9162	9165	9173	9176
9185	9188	9220	9223	9230	9233	9241	9244	9253	9256	9279	9282	9289
9292	9300	9303	9312	9315	9343	9350	9374	9376	9377	9379	9380	9382
9384	9385	9386	9390	9402	9426	9433	9458	9460	9463	9480	9483	9494
9499	9504	9507	9508	9509	9510	9514	9515	9517	9518	9519	9542	9544
9560	9565	9574	9577	9593	9595	9611	9614	9623	9626	9644	9646	9647
9649	9661	9662	9664	9666	9667	9668	9675	9682	9684	9685	9693	9694
9695	9698	9704	9712	9717	9738	9740	9741	9743	9768	9776	9781	9791
9797	9805	9816	9825	9827	9833	9858	9860	9862	9864	9871	9882	9883
9886	9890	9892	9894	9896	9898	9900	9906	9907	9912	9919	9922	9979
9981	10018	10020	10023	10027	10028	10048	10051	10081	10083	10120	10124	10128
10131	10132	10161	10163	10185	10191	10193	10199	10201	10214	10220	10221	10225
10227	10228	10230	10247	10250	10278	10285	10288	10294	10298	10308	10312	10314
10343	10345	10406	10407	10411	10414	10454	10457	10496	10498	10502	10505	10508
10514	10544	10546	10547	10549	10560	10563	10565	10568	10569	10570	10572	10573
10574	10578	10580	10581	10583	10622	10661	10663	10665	10667	10670	10673	10675
10681	10687	10689	10697	10710	10735	10738	10742	10755	10773	10774	10775	10776
10786	10820	10841	10842	10843	10845	10850	10855	10860	10861	10862	10869	10870
10873	10874	10875	10892	10898	10920							
7962#	7964#	8994	8999	9042	9052	9084	9086	9094	9104	9115	9127	9145
9147	9155	9165	9176	9188	9223	9233	9244	9256	9282	9292	9303	9315
9350	9376	9379	9382	9384	9385	9386	9402	9433	9458	9460	9463	9483
9499	9507	9508	9514	9515	9517	9519	9542	9544	9565	9577	9593	9595
9614	9626	9644	9646	9649	9661	9664	9666	9667	9675	9682	9693	9694
9695	9704	9717	9738	9740	9743	9776	9791	9805	9825	9833	9858	9860
9864	9871	9882	9886	9890	9892	9896	9900	9906	9907	9912	9922	9981
10023	10027	10028	10051	10081	10083	10128	10131	10132	10161	10163	10185	10193
10201	10220	10221	10227	10230	10247	10250	10278	10288	10298	10312	10314	10343
10345	10406	10407	10414	10457	10502	10505	10514	10544	10546	10549	10563	10568
10572	10573	10574	10580	10583	10622	10661	10665	10667	10673	10681	10689	10697
10710	10735	10738	10742	10755	10773	10774	10775	10776	10786	10820	10841	10845
10850	10855	10860	10873	10874	10875	10892	10898	10920				

\$LSTST= 177777

\$LSTTA= 00000C

\$MADR1 001226
\$MADR2 001232

8971#
8971#

		9662#	9664	9666	9684#	9694	9882#	9892	9894#	9896	9907	10020#	10023	10027
		10124#	10128	10131	10191#	10193	10199#	10201	10214#	10220	10285#	10288	10294#	10298
		10308#	10312	10498#	10502	10569#	10575	10661#	10675	10681	10687#	10689	10735#	10738
		10773#	10776	10841#	10861	10862#	10875							
\$NSK2 = 000110		9380#	9382	9384	9504#	9507	9510#	9514	9685#	9693	9883#	9886	9890	9898#
		9900	9906	10570#	10572	10663#	10665	10667	10670#	10673	10774#	10775	10842#	10855
		10860	10869#	10874										
\$NSK3 = 000110		10843#	10845	10850	10870#	10873								
\$NULL 001154		8971#	10933											
\$NWTST= 000001		9033#	9078#	9139#	9201#	9271#	9333#	9365#	9415#	9452#	9536#	9587#	9639#	9732#
		9850#	9971#	10076#	10156#	10242#	10337#	10539#	10595#					
\$OCNT 014254		10951#*												
\$OMODE 014256		10951#*												
\$OVER 013570		10945#												
\$PASS 001202		8971#	8986*	10870	10927*	10945								
\$PASTM 001006		8970#												
\$POWER 011612		10930#												
\$PWAD 011600		10930#												
\$PWADN 011440		8986	10930#											
\$PWRMG 011574		10930#												
\$PWRUP 011512		10930#												
\$QUES 001170		8971#	10933	10936	10942									
\$RDCHR 012366		10936#	10954											
\$RDDEC= ***** U		10954												
\$RDLIN 012506		10936#	10954											
\$RDOCT= ***** U		10954												
\$RDSZ = 000010		10936#												
\$RTNAD 011416		10927#												
\$R2A = ***** U		10954												
\$SAVLE= 177777		7962#	8999#	9519#	9695#	9871#	9892#	10185#	10278#	10681#	10735#	10773#	10774#	
\$SAVRE= ***** U		10954												
\$SAVR6 011610		10930#*												
\$SCOPE 013326		8986	10945#											
\$SETUP= 000137		8983#	8986	8987	10927	10936	10942	10945						
\$SSKO = 000234		8999#	9519#	9695#	9871#	9892#	10185#	10278#	10681#	10735#	10773#	10774#		
\$STUP = 177777		8983#												
\$SVLAD 013534		10945#												
\$SVPC = 000204		8968#												
\$SWR = 167400		7967#	8845	8846	8971	8986	9033	9078	9139	9201	9271	9333	9365	9415
		9452	9536	9587	9639	9732	9850	9971	10076	10156	10242	10337	10539	10595
		10927	10930	10942	10945									
\$SWREG 001216		8971#	8986											
\$SWRMK= 000000		8846	10945											
\$TAGLE= 177777		7962#	8994#	8999#	9042#	9049#	9052#	9056#	9084#	9086#	9091#	9094#	9101#	9104#
		9112#	9115#	9124#	9127#	9145#	9147#	9152#	9155#	9162#	9165#	9173#	9176#	9185#
		9188#	9220#	9223#	9230#	9233#	9241#	9244#	9253#	9256#	9279#	9282#	9289#	9292#
		9300#	9303#	9312#	9315#	9343#	9350#	9374#	9376#	9377#	9379#	9380#	9382#	9384#
		9385#	9386#	9390#	9402#	9426#	9433#	9458#	9460#	9463#	9480#	9483#	9494#	9499#
		9504#	9507#	9508#	9509#	9510#	9514#	9515#	9517#	9518	9519#	9542#	9544#	9560#
		9565#	9574#	9577#	9593#	9595#	9611#	9614#	9623#	9626#	9644#	9646#	9647#	9649#
		9661#	9662#	9664#	9666#	9667#	9668#	9675#	9682#	9684#	9685#	9693#	9694#	9695#
		9698#	9704#	9712#	9717#	9738#	9740#	9741#	9743#	9768#	9776#	9781#	9791#	9797#
		9805#	9816#	9825#	9827#	9833#	9858#	9860#	9862#	9864#	9871#	9882#	9883#	9886#
		9890#	9892#	9894#	9896#	9898#	9900#	9906#	9907#	9912#	9919#	9922#	9979#	9981#
		10018#	10020#	10023#	10027#	10028#	10048#	10051#	10081#	10083#	10120#	10124#	10128#	10131#
		10132#	10161#	10163#	10185#	10191#	10193#	10199#	10201#	10214#	10220#	10221#	10225#	10227#

\$TN = 000026	7968#	8845	9033#	9078#	9085	9139#	9146	9201#	9271#	9333#	9365#	9383	9415#
	9452#	9459	9520	9536#	9543	9587#	9594	9639#	9645	9648	9674	9692	9703
	9719	9732#	9739	9742	9775	9790	9804	9824	9850#	9859	9863	9923	9971#
	9980	10076#	10156#	10162	10219	10242#	10249	10287	10297	10311	10337#	10344	10418
	10539#	10545	10548	10584	10595#								
\$TPB 001152	8971#	10933*											
\$TPFLG 001157	8971#	10933											
\$TPS 001150	8971#	10933											
\$TRAP 014260	8986	10954#											
\$TRAP2 014302	10954#												
\$TRP = 000012	10954#												
\$TRPAD 014314	10954#												
\$TSKO = 000245	8994#	8999	9042#	9056	9084#	9086	9091#	9094	9101#	9104	9112#	9115	9124#
	9127	9145#	9147	9152#	9155	9162#	9165	9173#	9176	9185#	9188	9220#	9223
	9230#	9233	9241#	9244	9253#	9256	9279#	9282	9289#	9292	9300#	9303	9312#
	9315	9343#	9350	9374#	9376#	9386	9390#	9402	9426#	9433	9458#	9460	9463#
	9519	9542#	9544	9560#	9565	9574#	9577	9593#	9595	9611#	9614	9623#	9626
	9644#	9646	9647#	9649	9661#	9667	9668#	9675	9682#	9695	9698#	9704	9712#
	9717	9738#	9740	9741#	9743	9768#	9776	9781#	9791	9797#	9805	9816#	9825
	9827#	9833	9858#	9860	9862#	9864	9871#	9912	9919#	9922	9979#	9981	10018#
	10028	10048#	10051	10081#	10083	10120#	10132	10161#	10163	10185#	10221	10225#	10227
	10228#	10230	10247#	10250	10278#	10314	10343#	10345	10406#	10407	10411#	10414	10454#
	10457	10496#	10505	10508#	10514	10544#	10546	10547#	10549	10560#	10563	10565#	10568#
	10574	10578#	10580	10581#	10583	10661#	10681	10687#	10689	10735#	10738	10773#	10776
	10841#	10861	10862#	10875									
\$TSK1 = 000246	8994#	8999	9049#	9052	9377#	9379#	9385	9463#	9518	9519	9662#	9664#	9666
	9682#	9695	9871#	9912	10020#	10023#	10027	10124#	10128#	10131	10185#	10221	10278#
	10314	10498#	10502	10569#	10573	10661#	10675	10681	10735#	10738	10773#	10776	10842#
	10855#	10860	10869#	10874									
\$TSK2 = 000247	9380#	9382#	9384	9480#	9483	9494#	9499#	9508	9509#	9515#	9517	9684#	9694
	9882#	9892	9894#	9896#	9907	10191#	10193	10199#	10201	10214#	10220	10285#	10288
	10294#	10298	10308#	10312	10570#	10572	10663#	10665#	10667	10670#	10673	10774#	10775
	10843#	10845#	10850	10870#	10873								
\$TSK3 = 000234	9504#	9507	9510#	9514	9685#	9693	9882#	9892	9898#	9900#	9906	10774#	10775
\$TSK4 = 000124	9883#	9886#	9890										
\$TSTM 001004	8970#												
\$TSTM 001102	8971#	10927*	10942	10945*									
\$TTYIN 012614	10936#												
\$TYPBN= ***** U	10954												
\$TYPDS 013606	10948#	10954											
\$TYPE 011622	10933#	10939	10954										
\$TYPEC 012034	10933#	10936											
\$TYPEX 012102	10933#												
\$TYPOC 014056	10951#	10954											
\$TYPCN 014072	10951#	10954											
\$TYPOS 014032	10951#	10954											
\$UNIT 001206	8971#												
\$UNITM 001010	8970#												
\$USWR 001220	8971#	9084	9145	9377	9380	9542	9593	9644	9738	9741	9858	9913	10081
	10161	10247	10343	10544	10569	10733							
	8971#	10854											
\$VECT1 001244	8971#												
\$VECT2 001246	8971#												
\$XTSTR 013340	10945#												
\$YESNO= 000001	9012#	9013#	9014#	9015#	9054#	9055#	9057#	9099#	9110#	9121#	9160#	9171#	9182#
	9228#	9239#	9250#	9287#	9298#	9309#	9393#	9547#	9563#	9598#	9609#	9653#	9665#
	9672#	9689#	9701#	9715#	9745#	9771#	9785#	9801#	9819#	9830#	9869#	9871#	9888#

\$105	004762	9741	9743#
\$106	005070	9768	9776#
\$107	005124	9781	9791#
\$11	002562	9112	9115#
\$110	005162	9797	9805#
\$111	005240	9816	9825#
\$112	005260	9827	9833#
\$113	005316	9858#	
\$114	005326	9858	9860#
\$115	005346	9862	9864#
\$116	005402	9871#	
\$117	005400	9871#	9912
\$12	002612	9124	9127#
\$120	005554	9871	9912#
\$121	005440	9882#	9892
\$122	005474	9882	9892#
\$123	005462	9883	9886#
\$124	005472	9886	9890#
\$125	005506	9894	9896#
\$126	005536	9896	9907#
\$127	005530	9898	9900#
\$13	002650	9145#	
\$130	005536	9900	9906#
\$131	005622	9919	9922#
\$132	005712	9979	9981#
\$133	006070	10018	10028#
\$134	006066	10020	10023#
\$135	006070	10023	10027#
\$136	006146	10048	10051#
\$137	006234	10081#	
\$14	002660	9145	9147#
\$140	006240	10081	10083#
\$141	006434	10120	10132#
\$142	006432	10124	10128#
\$143	006434	10128	10131#
\$144	006540	10161#	
\$145	006550	10161	10163#
\$146	006620	10185#	
\$147	006616	10185#	10221
\$15	002700	9152	9155#
\$150	006772	10185	10221#
\$151	006666	10191	10193#
\$152	006732	10199	10201#
\$153	006770	10214	10220#
\$154	007012	10225	10227#
\$155	007024	10228	10230#
\$156	007062	10247#	
\$157	007072	10247	10250#
\$16	002726	9162	9165#
\$160	007144	10278#	
\$161	007142	10278#	10314
\$162	007320	10278	10314#
\$163	007216	10285	10288#
\$164	007266	10294	10298#
\$165	007316	10308	10312#
\$166	007364	10343#	

\$167	007374	10343	10345#
\$17	002754	9173	9176#
\$170	007506	10406#	10407
\$171	007522	10407#	
\$172	007540	10411	10414#
\$173	007620	10454	10457#
\$174	007700	10496	10505#
\$175	007674	10498	10502#
\$176	007714	10508	10514#
\$177	007760	10544#	
\$2	002122	8994	8999#
\$20	003004	9185	9188#
\$200	007770	10544	10546#
\$201	010010	10547	10549#
\$202	010072	10560	10563#
\$203	010110	10565	10568#
\$204	010136	10568	10574#
\$205	010136	10569	10573#
\$206	010136	10570	10572#
\$207	010152	10578	10580#
\$21	003054	9220	9223#
\$210	010164	10581	10583#
\$211	010466	10688	10697#
\$212	010470	10690	10697#
\$213	010336	10661#	10681
\$214	010442	10675	10681#
\$215	010356	10663	10665#
\$216	010364	10665	10667#
\$217	010402	10670	10673#
\$22	003102	9230	9233#
\$220	010454	10687	10689#
\$221	010570	10740	10742#
\$222	010570	10742#	
\$223	010534	10735#	
\$224	010530	10735#	10738
\$225	010560	10735	10738#
\$226	010650	10786#	
\$227	010650	10786#	
\$23	003130	9241	9244#
\$230	010616	10773#	
\$231	010612	10773#	10776
\$232	010642	10773	10776#
\$233	010630	10774#	
\$234	010626	10774#	10775
\$235	010640	10774	10775#
\$236	011124	10879	10892#
\$237	011124	10892#	
\$24	003160	9253	9256#
\$240	010676	10841#	10861
\$241	010762	10842	10855#
\$242	010722	10843	10845#
\$243	010730	10845	10850#
\$244	011002	10855	10860#
\$245	011070	10862	10875#
\$246	011070	10869	10874#
\$247	011070	10870	10873#

MAINDEC-11-CNDVC-A
CNDVCA.F11

27-DEC-82

MACY11 30(1046)
11:36

27-DEC-82 11:37 PAGE 196-19
CROSS REFERENCE TABLE -- USER SYMBOLS

I 10

SEQ 0125

9858	9862	9882	9883	9894	9898	9919	9979	10018	10020	10048	10081	10120
10124	10161	10191	10199	10214	10225	10228	10247	10285	10294	10308	10343	10407
10411	10454	10496	10498	10508	10544	10547	10560	10565	10569	10570	10578	10581
10596#	10663	10670	10687	10842	10843	10861	10862	10869	10870	10890#	10904#	10909#
10911#	10927	10930	10933	10936#	10939#	10942	10945	10948#	10956#			
10939												
8970#												

.SASTA= ***** U
.SX = 001^00

MAINDEC-11-CNDVC-A
CNDVCA.P11 27-DEC-82

MACY11 30(1046)
11:36

27-DEC-82 11:37 PAGE 197-2
CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0128

LOOP	6691#	9463	10661												
MSG	9026#	9033	9071#	9078	9269#	9271	9448#	9452	9532#	9536	9635#	9639	9845#	9850	9964#
	9971	10071#	10076	10153#	10156	10334#	10337	10533#	10539						
MULT	4484#	8848#													
NEWTST	1626#	8848#	9033	9078	9139	9201	9271	9333	9365	9415	9452	9536	9587	9639	9732
	9850	9971	10076	10156	10242	10337	10539	10595							
NOLOCA	7183#														
POINTE	8171#														
POP	2153#	8848#	9057	10055	10144	10781	10930	10939	10948						
PRINTB	8174#														
PUSH	2145#	8848#	9057	10055	10144	10767	10930	10939	10948						
REPEAT	6516#	9042	9661	10406	10841										
REPORT	5463#	7960#	8848#												
RETURN	7196#	10688	10690	10740	10879										
ROUTIN	7111#	10622	10710	10755	10820	10898									
SAVR14	7138#														
SCOPE	8848#	9033	9078	9139	9201	9271	9333	9365	9415	9452	9536	9587	9639	9732	9850
	9971	10076	10156	10242	10337	10539	10595	10927							
SELECT	7250#														
SETPRI	1279#	8848#	9207	10005	10041	10101	10175	10269	10371						
SETTRA	10954#														
SETUP	1337#	8848#	8986												
SETVEC	8177#	9040	9992	10087											
SKIP	1733#	8848#	9085	9146	9383	9459	9520	9543	9594	9645	9648	9674	9692	9703	9719
	9739	9742	9775	9790	9804	9824	9859	9863	9923	9980	10162	10219	10249	10287	10297
	10311	10344	10418	10545	10548	10584									
SLASH	1517#	8848#													
SPACE	8848#														
STARS	1485#	8848#	8865	8867	8885	8887	8905	8907	8927	8929	8948	8964	8968	8970	8971
	9033	9066	9068	9078	9138	9139	9200	9201	9268	9271	9329	9332	9333	9364	9365
	9414	9415	9447	9452	9531	9536	9586	9587	9634	9639	9731	9732	9844	9850	9963
	9971	10070	10076	10152	10156	10241	10242	10333	10337	10430	10441	10471	10482	10532	10539
	10595	10617	10645	10705	10726	10750	10766	10795	10806	10825	10834	10903	10927	10930	10933
	10936	10939	10942	10948	10951	10954									
STRUCT	5720#	7961#	7962												
SWRSU	1453#	8848#	8986#												
TRMTRP	10954#														
TYPBIN	2088#	8848#													
TYPDEC	2058#	8848#	10601	10927											
TYPNAM	1826#	8848#	8987												
TYPNUM	2025#	8848#													
TYPOCS	1978#	8848#	10908												
TYPOCT	1941#	8848#	10597	10599	10905	10910	10912	10914	10936						
TYPTXT	1894#	8848#	8995	8996	8997	10596	10598	10600	10904	10906	10909	10911	10913		
UNTIL	6650#	9056	9667	10407	10861										
UNTILB	6671#														
WAITMS	8188#	9757	9762	9814	10015	10047	10116	10679							
WHILE	6487#	8994	9682	9882											
WHILEB	6492#														
\$ADDON	5805#	8994	8999	9042	9049	9084	9091	9101	9112	9124	9145	9152	9162	9173	9185
	9220	9230	9241	9253	9279	9289	9300	9312	9343	9374	9376	9377	9379	9380	9382
	9390	9426	9458	9463	9480	9494	9499	9504	9509	9510	9515	9519	9542	9560	9574
	9593	9611	9623	9644	9647	9661	9662	9664	9667	9668	9682	9684	9685	9695	9698
	9712	9738	9741	9768	9781	9797	9816	9827	9858	9862	9871	9882	9883	9886	9892
	9894	9896	9898	9900	9919	9979	10018	10020	10023	10048	10081	10120	10124	10128	10161
	10185	10191	10199	10214	10225	10228	10247	10278	10285	10294	10308	10343	10406	10407	10411

	10454	10496	10498	10508	10544	10547	10560	10565	10568	10569	10570	10578	10581	10622	10661
	10663	10665	10670	10681	10687	10710	10735	10755	10773	10774	10820	10841	10842	10843	10845
	10855	10862	10869	10870	10898										
\$AND	6092#	9898													
\$BRANC	5931#	8994	8999	9049	9056	9084	9091	9101	9112	9124	9145	9152	9162	9173	9185
	9220	9230	9241	9253	9279	9289	9300	9312	9343	9374	9376	9377	9379	9380	9382
	9390	9426	9458	9480	9494	9499	9504	9509	9510	9515	9518	9519	9542	9560	9574
	9593	9611	9623	9644	9647	9662	9664	9667	9668	9682	9684	9685	9695	9698	9712
	9738	9741	9768	9781	9797	9816	9827	9858	9862	9871	9882	9883	9886	9892	9894
	9896	9898	9900	9912	9919	9979	10018	10020	10023	10048	10081	10120	10124	10128	10161
	10185	10191	10199	10214	10221	10225	10228	10247	10278	10285	10294	10308	10314	10343	10407
	10411	10454	10496	10498	10508	10544	10547	10560	10565	10568	10569	10570	10578	10581	10663
	10665	10670	10675	10681	10687	10688	10690	10735	10738	10740	10773	10774	10775	10776	10842
	10843	10845	10855	10861	10862	10869	10870	10879							
\$BRCOD	6230#	9084	9145	9458	9518	9542	9593	9644	9667	9738	9858	9871	10081	10161	10185
	10247	10278	10343	10407	10544	10675	10735	10773	10774						
\$CALL	7039#	9003	9477	9503	9557	9608	9757	9762	9814	9995	10015	10047	10112	10116	10181
	10190	10198	10275	10284	10293	10352	10559	10679	10846						
\$CHECK	6288#	8994	9049	9084	9091	9101	9112	9124	9145	9152	9162	9173	9185	9220	9230
	9241	9253	9279	9289	9300	9312	9343	9374	9377	9380	9390	9426	9458	9494	9509
	9510	9542	9574	9593	9623	9644	9647	9662	9668	9682	9684	9685	9698	9712	9738
	9741	9768	9781	9797	9816	9827	9858	9862	9882	9883	9894	9898	9919	9979	10018
	10020	10048	10081	10120	10124	10161	10214	10225	10228	10247	10308	10343	10411	10454	10496
	10498	10508	10544	10547	10565	10569	10570	10578	10581	10663	10670	10687	10842	10843	10862
	10869	10870													
\$CHK1	7725#	9000	9001	9006	9008	9009	9011	9016	9038	9040	9041	9043	9045	9057	9085
	9089	9098	9108	9119	9146	9150	9159	9169	9180	9218	9227	9237	9248	9277	9286
	9296	9307	9342	9383	9388	9424	9459	9462	9465	9466	9473	9490	9496	9513	9516
	9543	9549	9552	9569	9594	9599	9604	9621	9645	9648	9654	9655	9660	9663	9674
	9692	9703	9710	9739	9742	9749	9755	9760	9765	9775	9780	9790	9794	9804	9808
	9812	9824	9859	9863	9866	9867	9868	9871	9872	9874	9876	9878	9880	9881	9885
	9905	9909	9910	9915	9980	9985	9988	9992	9993	10031	10033	10055	10087	10089	10090
	10108	10118	10143	10144	10162	10165	10182	10185	10196	10203	10219	10249	10276	10278	10287
	10291	10297	10301	10311	10344	10377	10379	10380	10383	10384	10387	10389	10391	10393	10452
	10500	10501	10545	10548	10551	10557	10654	10655	10656	10664	10666	10672	10732	10735	10739
	10772	10773	10774	10839	10840	10844	10849	10851	10852	10853	10854	10856	10863	10864	10865
	10872	10877													
\$CKIP2	7826#	9012	9013	9014	9015	9054	9055	9057	9099	9110	9121	9160	9171	9182	9228
	9239	9250	9287	9298	9309	9393	9547	9563	9598	9609	9653	9665	9672	9689	9701
	9715	9745	9771	9785	9801	9819	9830	9869	9888	9889	9913	9914	9918	9990	9998
	10012	10035	10055	10092	10094	10114	10123	10137	10141	10144	10167	10192	10200	10210	10211
	10217	10224	10252	10262	10304	10305	10318	10397	10400	10402	10409	10416	10417	10448	10450
	10456	10489	10491	10493	10504	10510	10553	10555	10567	10571	10680	10733	10736	10737	10813
	10857	10858	10859	10878											
\$CKR6	7554#	9913	10733												
\$CMND	6119#	8994	9049	9084	9091	9101	9112	9124	9145	9152	9162	9173	9185	9220	9230
	9241	9253	9279	9289	9300	9312	9343	9374	9377	9380	9390	9426	9458	9494	9509
	9510	9542	9574	9593	9623	9644	9647	9662	9668	9682	9684	9685	9698	9712	9738
	9741	9768	9781	9797	9816	9827	9858	9862	9882	9883	9894	9898	9919	9979	10018
	10020	10048	10081	10120	10124	10161	10214	10225	10228	10247	10308	10343	10411	10454	10496
	10498	10508	10544	10547	10565	10569	10570	10578	10581	10663	10670	10687	10842	10843	10862
	10869	10870													
\$COMPA	6280#	8994	9049	9084	9091	9101	9112	9124	9145	9152	9162	9173	9185	9220	9230
	9241	9253	9279	9289	9300	9312	9343	9374	9377	9380	9390	9426	9458	9480	9494
	9504	9509	9510	9542	9560	9574	9593	9611	9623	9644	9647	9662	9668	9682	9684
	9685	9698	9712	9738	9741	9768	9781	9797	9816	9827	9858	9862	9871	9882	9883

	9894	9898	9919	9979	10018	10020	10048	10081	10120	10124	10161	10185	10191	10199	10214
	10225	10228	10247	10278	10285	10294	10308	10343	10411	10454	10496	10498	10508	10544	10547
	10560	10565	10569	10570	10578	10581	10663	10670	10687	10735	10773	10774	10842	10843	10862
	10869	10870													
\$SCOUNT	7032#	9003	9477	9503	9557	9608	9757	9762	9914	9995	10015	10047	10112	10116	10181
	10190	10198	10275	10284	10293	10352	10559	10679	10846						
\$SDO	6046#	8994	9682	9882											
\$ELSE	6387#														
\$ERRMS	6002#														
\$EXIFA	6705#														
\$EXIFO	6742#														
\$EXIF2	6800#	10675													
\$EXIF3	6779#														
\$GENBR	5924#	8994	8999	9049	9056	9084	9091	9101	9112	9124	9145	9152	9162	9173	9185
	9220	9230	9241	9253	9279	9289	9300	9312	9343	9374	9376	9377	9379	9380	9382
	9390	9426	9458	9480	9494	9499	9504	9509	9510	9515	9518	9519	9542	9560	9574
	9593	9611	9623	9644	9647	9662	9664	9667	9668	9682	9684	9685	9695	9698	9712
	9738	9741	9768	9781	9797	9816	9827	9858	9862	9871	9882	9883	9886	9892	9894
	9896	9898	9900	9912	9919	9979	10018	10020	10023	10048	10081	10120	10124	10128	10161
	10185	10191	10199	10214	10221	10225	10228	10247	10278	10285	10294	10308	10314	10343	10407
	10411	10454	10496	10498	10508	10544	10547	10560	10565	10568	10569	10570	10578	10581	10663
	10665	10670	10675	10681	10687	10688	10690	10735	10738	10740	10773	10774	10775	10776	10842
	10843	10845	10855	10861	10862	10869	10870	10879							
\$GENTA	5900#	8994	8999	9042	9052	9084	9086	9094	9104	9115	9127	9145	9147	9155	9165
	9176	9188	9223	9233	9244	9256	9282	9292	9303	9315	9350	9376	9379	9382	9384
	9385	9386	9402	9433	9458	9460	9463	9483	9499	9507	9508	9514	9515	9517	9519
	9542	9544	9565	9577	9593	9595	9614	9626	9644	9646	9649	9661	9664	9666	9667
	9675	9682	9693	9694	9695	9704	9717	9738	9740	9743	9776	9791	9805	9825	9833
	9858	9860	9864	9871	9882	9886	9890	9892	9896	9900	9906	9907	9912	9922	9981
	10023	10027	10028	10051	10081	10083	10128	10131	10132	10161	10163	10185	10193	10201	10220
	10221	10227	10230	10247	10250	10278	10288	10298	10312	10314	10343	10345	10406	10407	10414
	10457	10502	10505	10514	10544	10546	10549	10563	10568	10572	10573	10574	10580	10583	10661
	10665	10667	10673	10681	10689	10697	10735	10738	10742	10773	10774	10775	10776	10786	10841
	10845	10850	10855	10860	10873	10874	10875	10892	10920						
\$IF	6306#	9049	9084	9091	9101	9112	9124	9145	9152	9162	9173	9185	9220	9230	9241
	9253	9279	9289	9300	9312	9343	9374	9377	9380	9390	9426	9458	9494	9509	9510
	9542	9574	9593	9623	9644	9647	9662	9668	9684	9685	9698	9712	9738	9741	9768
	9781	9797	9816	9827	9858	9862	9883	9894	9898	9919	9979	10018	10020	10048	10081
	10120	10124	10161	10214	10225	10228	10247	10308	10343	10411	10454	10496	10498	10508	10544
	10547	10565	10569	10570	10578	10581	10663	10670	10687	10842	10843	10862	10869	10870	
\$IFCOD	6178#	8994	9049	9056	9084	9091	9101	9112	9124	9145	9152	9162	9173	9185	9220
	9230	9241	9253	9279	9289	9300	9312	9343	9374	9377	9380	9390	9426	9458	9494
	9509	9510	9542	9574	9593	9623	9644	9647	9662	9667	9668	9682	9684	9685	9698
	9712	9738	9741	9768	9781	9797	9816	9827	9858	9862	9882	9883	9894	9898	9919
	9979	10018	10020	10048	10081	10120	10124	10161	10214	10225	10228	10247	10308	10343	10407
	10411	10454	10496	10498	10508	10544	10547	10565	10569	10570	10578	10581	10663	10670	10687
	10842	10843	10861	10862	10869	10870									
\$IFCON	6345#	9480	9504	9560	9611	10191	10199	10285	10294	10560					
\$IFOPR	5944#	8994	9049	9056	9084	9091	9101	9112	9124	9145	9152	9162	9173	9185	9220
	9230	9241	9253	9279	9289	9300	9312	9343	9374	9377	9380	9390	9426	9458	9480
	9494	9504	9509	9510	9542	9560	9574	9593	9611	9623	9644	9647	9662	9667	9668
	9682	9684	9685	9698	9712	9738	9741	9768	9781	9797	9816	9827	9858	9862	9882
	9883	9894	9898	9919	9979	10018	10020	10048	10081	10120	10124	10161	10191	10199	10214
	10225	10228	10247	10285	10294	10308	10343	10407	10411	10454	10496	10498	10508	10544	10547
	10560	10565	10569	10570	10578	10581	10663	10670	10687	10842	10843	10861	10862	10869	10870
\$LET	7909#	9000	9001	9006	9008	9009	9011	9012	9013	9014	9015	9016	9038	9040	9041

	9043	9045	9054	9055	9057	9085	9089	9098	9099	9108	9110	9119	9121	9146	9150
	9159	9160	9169	9171	9180	9182	9218	9227	9228	9237	9239	9248	9250	9277	9286
	9287	9296	9298	9307	9309	9342	9383	9388	9393	9424	9459	9462	9465	9466	9473
	9490	9496	9513	9516	9543	9547	9549	9552	9563	9569	9594	9598	9599	9604	9609
	9621	9645	9648	9653	9654	9655	9660	9663	9665	9672	9674	9689	9692	9701	9703
	9710	9715	9739	9742	9745	9749	9755	9760	9765	9771	9775	9780	9785	9790	9794
	9801	9804	9808	9812	9819	9824	9830	9859	9863	9866	9867	9868	9869	9872	9874
	9876	9878	9880	9881	9885	9888	9889	9905	9909	9910	9913	9914	9915	9918	9980
	9985	9988	9990	9992	9993	9998	10012	10031	10033	10035	10055	10087	10089	10090	10092
	10094	10108	10114	10118	10123	10137	10141	10143	10144	10162	10165	10167	10182	10192	10196
	10200	10203	10210	10211	10217	10219	10224	10249	10252	10262	10276	10287	10291	10297	10301
	10304	10305	10311	10318	10344	10377	10379	10380	10383	10384	10387	10389	10391	10393	10397
	10400	10402	10409	10416	10417	10448	10450	10452	10456	10489	10491	10493	10500	10501	10504
	10510	10545	10548	10551	10553	10555	10557	10567	10571	10654	10655	10656	10664	10666	10672
	10680	10732	10733	10736	10737	10739	10772	10813	10839	10840	10844	10849	10851	10852	10853
	10854	10856	10857	10858	10859	10863	10864	10865	10872	10877	10878				
SLPCNT	6927#	9871	10185	10278	10735	10773	10774								
SOPADD	7566#	9012	9013	9014	9015	9054	9055	9057	9665	9871	9888	9889	9990	10055	10144
	10185	10278	10448	10489	10504	10735	10773	10774	10813	10858	10859	10878			
SOPAND	7608#	9913	10733												
SOPCD1	7649#	9012	9013	9014	9015	9054	9055	9057	9099	9110	9121	9160	9171	9182	9228
	9239	9250	9287	9298	9309	9393	9547	9563	9598	9609	9653	9665	9672	9689	9701
	9715	9745	9771	9785	9801	9819	9830	9869	9871	9888	9889	9913	9914	9918	9990
	9998	10012	10035	10055	10092	10094	10114	10123	10137	10141	10144	10167	10185	10192	10200
	10210	10211	10217	10224	10252	10262	10278	10304	10305	10318	10397	10400	10402	10409	10416
	10417	10448	10450	10456	10489	10491	10493	10504	10510	10553	10555	10567	10571	10680	10733
	10735	10736	10737	10739	10773	10774	10813	10857	10858	10859	10878				
SOPCD2	7524#	10736	10739	10857											
SOPCOD	7710#	9012	9013	9014	9015	9054	9055	9057	9099	9110	9121	9160	9171	9182	9228
	9239	9250	9287	9298	9309	9393	9547	9563	9598	9609	9653	9665	9672	9689	9701
	9715	9745	9771	9785	9801	9819	9830	9869	9871	9888	9889	9913	9914	9918	9990
	9998	10012	10035	10055	10092	10094	10114	10123	10137	10141	10144	10167	10185	10192	10200
	10210	10211	10217	10224	10252	10262	10278	10304	10305	10318	10397	10400	10402	10409	10416
	10417	10448	10450	10456	10489	10491	10493	10504	10510	10553	10555	10567	10571	10680	10733
	10735	10736	10737	10739	10773	10774	10813	10857	10858	10859	10878				
SOPCOM	7461#	10739													
SOPDEF	5909#	8994	8999	9000	9001	9003	9006	9008	9009	9011	9012	9013	9014	9015	9016
	9038	9040	9041	9043	9045	9049	9054	9055	9056	9057	9084	9085	9089	9091	9098
	9099	9101	9108	9110	9112	9119	9121	9124	9145	9146	9150	9152	9159	9160	9162
	9169	9171	9173	9180	9182	9185	9218	9220	9227	9228	9230	9237	9239	9241	9248
	9250	9253	9277	9279	9286	9287	9289	9296	9298	9300	9307	9309	9312	9342	9343
	9374	9376	9377	9379	9380	9382	9383	9388	9390	9393	9424	9426	9458	9459	9462
	9465	9466	9473	9477	9480	9490	9494	9496	9499	9503	9504	9509	9510	9513	9515
	9516	9518	9519	9542	9543	9547	9549	9552	9557	9560	9563	9569	9574	9593	9594
	9598	9599	9604	9608	9609	9611	9621	9623	9644	9645	9647	9648	9653	9654	9655
	9660	9662	9663	9664	9665	9667	9668	9672	9674	9682	9684	9685	9689	9692	9695
	9698	9701	9703	9710	9712	9715	9738	9739	9741	9742	9745	9749	9755	9757	9760
	9762	9765	9768	9771	9775	9780	9781	9785	9790	9794	9797	9801	9804	9808	9812
	9814	9816	9819	9824	9827	9830	9858	9859	9862	9863	9866	9867	9868	9869	9871
	9872	9874	9876	9878	9880	9881	9882	9883	9885	9886	9888	9889	9892	9894	9896
	9898	9900	9905	9909	9910	9912	9913	9914	9915	9918	9919	9979	9980	9985	9988
	9990	9992	9993	9995	9998	10012	10015	10018	10020	10023	10031	10033	10035	10047	10048
	10055	10081	10087	10089	10090	10092	10094	10108	10112	10114	10116	10118	10120	10123	10124
	10128	10137	10141	10143	10144	10161	10162	10165	10167	10181	10182	10185	10190	10191	10192
	10196	10198	10199	10200	10203	10210	10211	10214	10217	10219	10221	10224	10225	10228	10247
	10249	10252	10262	10275	10276	10278	10284	10285	10287	10291	10293	10294	10297	10301	10304

	10305	10308	10311	10314	10318	10343	10344	10352	10377	10379	10380	10383	10384	10387	10389
	10391	10393	10397	10400	10402	10407	10409	10411	10416	10417	10448	10450	10452	10454	10456
	10489	10491	10493	10496	10498	10500	10501	10504	10508	10510	10544	10545	10547	10548	10551
	10553	10555	10557	10559	10560	10565	10567	10568	10569	10570	10571	10578	10581	10654	10655
	10656	10663	10664	10665	10666	10670	10672	10675	10679	10680	10681	10687	10688	10690	10697
	10732	10733	10735	10736	10737	10738	10739	10740	10742	10772	10773	10774	10775	10776	10786
	10813	10839	10840	10842	10843	10844	10845	10846	10849	10851	10852	10853	10854	10855	10856
	10857	10858	10859	10861	10862	10863	10864	10865	10869	10870	10872	10877	10878	10879	10892
	10920														
\$OPEQU	7642#														
\$OPNAN	7627#														
\$OPNEG	7473#														
\$OPNOR	7621#														
\$OPNOT	7616#	9110	9171	9239	9298	9393	9563	9609	9672	9689	9701	9715	9771	9785	9801
	9819	9830	9914	9918	9998	10035	10094	10123	10137	10141	10210	10211	10217	10224	10252
\$OPOR	10304	10305	10409	10416	10417	10450	10456	10491	10493	10510					
	7603#	9099	9121	9160	9182	9228	9250	9287	9309	9547	9598	9653	9745	9869	10012
	10092	10114	10167	10192	10200	10262	10318	10397	10400	10402	10553	10555	10567	10571	10737
\$OPROT	7493#	10857													
\$OPRO	7752#	9000	9001	9006	9008	9009	9011	9016	9038	9040	9041	9043	9045	9057	9085
	9089	9098	9108	9119	9146	9150	9159	9169	9180	9218	9227	9237	9248	9277	9286
	9296	9307	9342	9383	9388	9424	9459	9462	9465	9466	9473	9490	9496	9513	9516
	9543	9549	9552	9569	9594	9599	9604	9621	9645	9648	9654	9655	9660	9663	9674
	9692	9703	9710	9739	9742	9749	9755	9760	9765	9775	9780	9790	9794	9804	9808
	9812	9824	9859	9863	9866	9867	9868	9871	9872	9874	9876	9878	9880	9881	9885
	9905	9909	9910	9915	9980	9985	9988	9992	9993	10031	10033	10055	10087	10089	10090
	10108	10118	10143	10144	10162	10165	10182	10185	10196	10203	10219	10249	10276	10278	10287
	10291	10297	10301	10311	10344	10377	10379	10380	10383	10384	10387	10389	10391	10393	10452
	10500	10501	10545	10548	10551	10557	10654	10655	10656	10664	10666	10672	10732	10735	10772
	10773	10774	10839	10840	10844	10849	10851	10852	10853	10854	10856	10863	10864	10865	10872
	10877														
\$OPR1	7799#	9871	10185	10278	10735	10739	10773	10774							
\$OPR2	7868#	9012	9013	9014	9015	9054	9055	9057	9099	9110	9121	9160	9171	9182	9228
	9239	9250	9287	9298	9309	9393	9547	9563	9598	9609	9653	9665	9672	9689	9701
	9715	9745	9771	9785	9801	9819	9830	9869	9888	9889	9913	9914	9918	9990	9998
	10012	10035	10055	10092	10094	10114	10123	10137	10141	10144	10167	10192	10200	10210	10211
	10217	10224	10252	10262	10304	10305	10318	10397	10400	10402	10409	10416	10417	10448	10450
	10456	10489	10491	10493	10504	10510	10553	10555	10567	10571	10680	10733	10736	10737	10813
	10857	10858	10859	10878											
\$OPSHF	7510#	10736													
\$OPSUB	7586#	10680													
\$OPSWB	7485#														
\$OPXOR	7633#														
\$OR	6065#	9084	9145	9458	9542	9593	9644	9738	9858	10081	10161	10247	10343	10544	
\$PUT	7022#	9477	9503	9557	9608	9757	9762	9814	9995	10015	10047	10112	10116	10190	10198
	10284	10293	10559	10679											
\$STRUC	5765#														
\$SUBON	5811#	8999	9052	9056	9086	9094	9104	9115	9127	9147	9155	9165	9176	9188	9223
	9233	9244	9256	9282	9292	9303	9315	9350	9376	9379	9382	9384	9385	9386	9402
	9433	9460	9483	9499	9507	9508	9514	9515	9517	9519	9544	9565	9577	9595	9614
	9626	9646	9649	9664	9666	9667	9675	9693	9694	9695	9704	9717	9740	9743	9776
	9791	9805	9825	9833	9860	9864	9871	9886	9890	9892	9896	9900	9906	9907	9912
	9922	9981	10023	10027	10028	10051	10083	10128	10131	10132	10163	10185	10193	10201	10220
	10221	10227	10230	10250	10278	10288	10298	10312	10314	10345	10407	10414	10457	10502	10505
	10514	10546	10549	10563	10568	10572	10573	10574	10580	10583	10665	10667	10673	10681	10689
	10697	10735	10738	10742	10773	10774	10775	10776	10786	10845	10850	10855	10860	10861	10873

	10874	10875	10892	10920														
\$THEN	6027#	9049	9084	9091	9101	9112	9124	9145	9152	9162	9173	9185	9220	9230	9241			
	9253	9279	9289	9300	9312	9343	9374	9377	9380	9390	9426	9458	9494	9509	9510			
	9542	9574	9593	9623	9644	9647	9662	9668	9684	9685	9698	9712	9738	9741	9768			
	9781	9797	9816	9827	9858	9862	9883	9894	9898	9919	9979	10018	10020	10048	10081			
	10120	10124	10161	10214	10225	10228	10247	10308	10343	10411	10454	10496	10498	10508	10544			
	10547	10565	10569	10570	10578	10581	10663	10670	10687	10842	10843	10862	10869	10870				
\$STILA	6526#																	
\$STILO	6563#																	
\$SUNTL2	6621#	9667	10407															
\$SUNTL3	6600#																	
\$SWHILE	6460#	8994	9682	9882														
\$SCMRE	8971#																	
\$SCMTM	8971#																	
\$SDEFA	7354#																	
\$SENDS	7305#																	
\$SERRO	6012#																	
\$SESCA	1711#	8848#																
\$SGEN	5888#	8994	8999	9042	9052	9084	9086	9094	9104	9115	9127	9145	9147	9155	9165			
	9176	9188	9223	9233	9244	9256	9282	9292	9303	9315	9350	9376	9379	9382	9384			
	9385	9386	9402	9433	9458	9460	9463	9483	9499	9507	9508	9514	9515	9517	9519			
	9542	9544	9565	9577	9593	9595	9614	9626	9644	9646	9649	9661	9664	9666	9667			
	9675	9682	9693	9694	9695	9704	9717	9738	9740	9743	9776	9791	9805	9825	9833			
	9858	9860	9864	9871	9882	9886	9890	9892	9896	9900	9906	9907	9912	9922	9981			
	10023	10027	10028	10051	10081	10083	10128	10131	10132	10161	10163	10185	10193	10201	10220			
	10221	10227	10230	10247	10250	10278	10288	10298	10312	10314	10313	10345	10406	10407	10414			
	10457	10502	10505	10514	10544	10546	10549	10563	10568	10572	10573	10574	10580	10583	10622			
	10661	10665	10667	10673	10681	10689	10697	10710	10735	10738	10742	10755	10773	10774	10775			
	10776	10786	10820	10841	10845	10850	10855	10860	10873	10874	10875	10892	10898	10920				
\$SGETS	5851#	8999	9052	9056	9086	9094	9104	9115	9127	9147	9155	9165	9176	9188	9223			
	9233	9244	9256	9282	9292	9303	9315	9350	9376	9379	9382	9384	9385	9386	9402			
	9433	9460	9483	9499	9507	9508	9514	9515	9517	9518	9519	9544	9565	9577	9595			
	9614	9626	9646	9649	9664	9666	9667	9675	9693	9694	9695	9704	9717	9740	9743			
	9776	9791	9805	9825	9833	9860	9864	9871	9886	9890	9892	9896	9900	9906	9907			
	9912	9922	9981	10023	10027	10028	10051	10083	10128	10131	10132	10163	10185	10193	10201			
	10220	10221	10227	10230	10250	10278	10288	10298	10312	10314	10345	10407	10414	10457	10502			
	10505	10514	10546	10549	10563	10568	10572	10573	10574	10580	10583	10665	10667	10673	10675			
	10681	10689	10697	10735	10738	10742	10773	10774	10775	10776	10786	10845	10850	10855	10860			
	10861	10873	10874	10875	10892	10920												
\$SGETT	5861#	9376	9379	9382	9499	9515	9518	9664	9886	9896	9900	10023	10128	10568	10665			
	10675	10845	10855															
\$SSLPCN	6887#	9871	10185	10278	10735	10773	10774											
\$SNEWT	1662#	8848#	9033	9078	9139	9201	9271	9333	9365	9415	9452	9536	9587	9639	9732			
	9850	9971	10076	10156	10242	10337	10539	10595										
\$SPOP	5875#	8999	9052	9056	9086	9094	9104	9115	9127	9147	9155	9165	9176	9188	9223			
	9233	9244	9256	9282	9292	9303	9315	9350	9376	9379	9382	9384	9385	9386	9402			
	9433	9460	9483	9499	9507	9508	9514	9515	9517	9519	9544	9565	9577	9595	9614			
	9626	9646	9649	9664	9666	9667	9675	9693	9694	9695	9704	9717	9740	9743	9776			
	9791	9805	9825	9833	9860	9864	9871	9886	9890	9892	9896	9900	9906	9907	9912			
	9922	9981	10023	10027	10028	10051	10083	10128	10131	10132	10163	10185	10193	10201	10220			
	10221	10227	10230	10250	10278	10288	10298	10312	10314	10345	10407	10414	10457	10502	10505			
	10514	10546	10549	10563	10568	10572	10573	10574	10580	10583	10665	10667	10673	10681	10689			
	10697	10735	10738	10742	10773	10774	10775	10776	10786	10845	10850	10855	10860	10861	10873			
	10874	10875	10892	10920														
\$SPUSH	5839#	8994	8999	9042	9049	9084	9091	9101	9112	9124	9145	9152	9162	9173	9185			
	9220	9230	9241	9253	9279	9289	9300	9312	9343	9374	9376	9377	9379	9380	9382			

.\$SUPR	4913#		
.\$STRAP	4073#	7959#	10954
.\$STYPB	3287#		
.\$STYPD	3209#	7959#	10948
.\$STYPE	2985#	7958#	10933
.\$STYPO	3112#	7960#	10951
.\$40CA	972#		

. ABS. 014412 000

ERRORS DETECTED: 0

CNDVCA,CNDVCA/CRF/NL:TOC=CNMAC2.SML,CNMAC.MAC,CNDVCA.P11
RUN-TIME: 75 78 4 SECONDS
RUN-TIME RATIO: 235/158=1.4
CORE USED: 43K (86 PAGES)