

M8203,LUNT

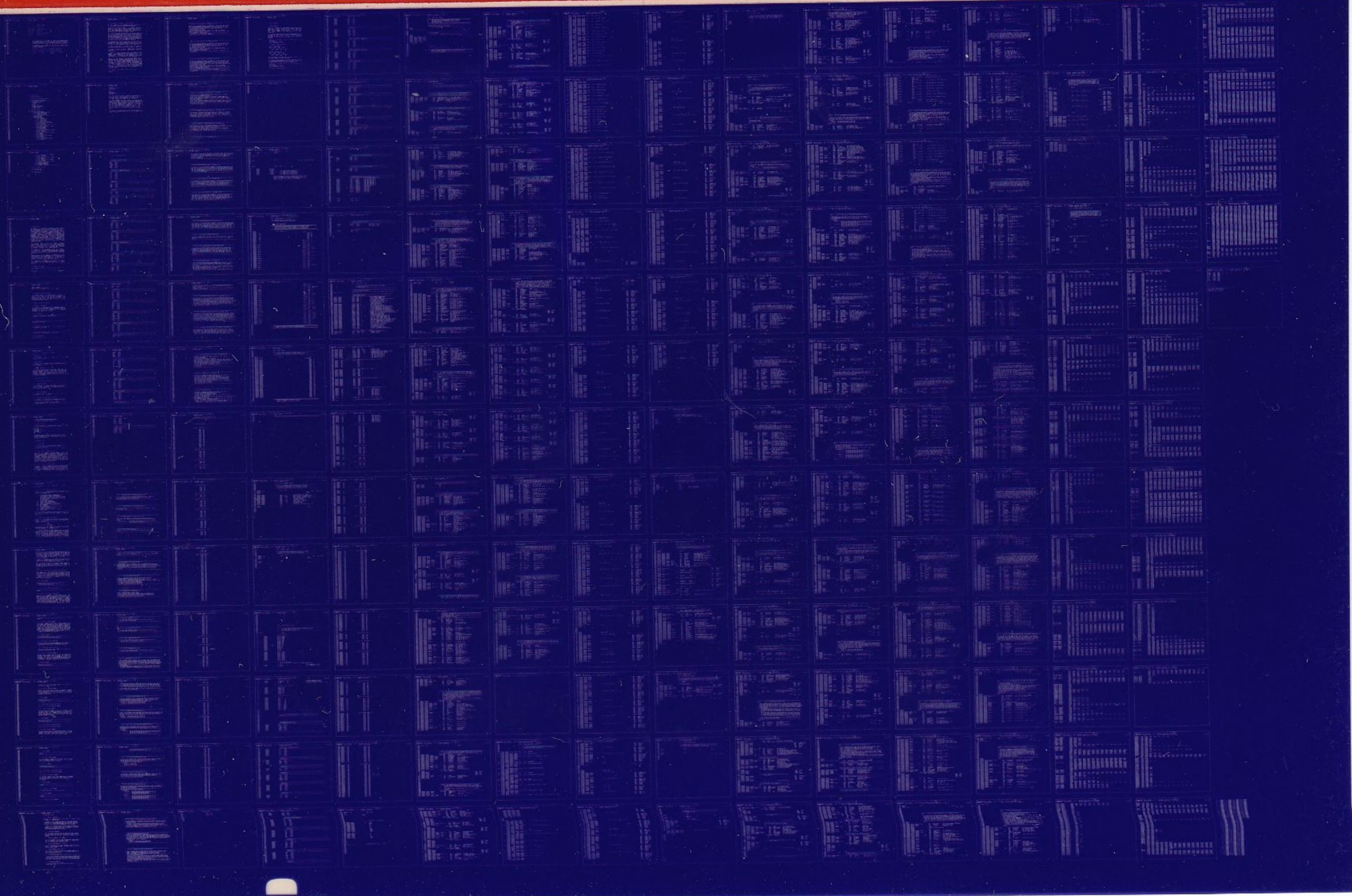
M8203 STATIC DIAG. #1
CZDMRA0

AH-E232A-MC

COPYRIGHT 1979
FICHE 1 OF 1

SEP 1979

digital
MADE IN USA



3298
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3330
3331
3332

.REM @

IDENTIFICATION

PRODUCT CODE: AC-E231A-MC
PRODUCT NAME: CZDMRA0 M8203 STATIC DIAG #1
PRODUCT DATE: JUNE 1979
MAINTAINER: DIAGNOSTIC ENGINEERING
AUTHOR: DAVID HOFFMAN

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) 1979 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

| | | | |
|---------|-------|---------|---------|
| DIGITAL | PDP | UNIBUS | MASSBUS |
| DEC | DECUS | DECTAPE | |

3334
3335
3336
3337
3338
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389

CONTENTS

- 1.0 INTRODUCTION
- 2.0 HARDWARE REQUIREMENTS
- 3.0 PRELIMINARY PROGRAM REQUIREMENTS
- 4.0 GENERAL PROGRAM CONSIDERATIONS
 - 4.1 DIAGNOSTIC SUPERVISOR
 - 4.2 EXECUTION TIME
 - 4.3 XXDP+
 - 4.4 ACT/SLIDE
 - 4.5 APT
 - 4.6 MEMORY MANAGEMENT
 - 4.7 MEMORY PARITY OPTION
 - 4.8 ERROR LOGGING
- 5.0 PROGRAM LOAD MEDIA
- 6.0 OPERATING INSTRUCTIONS
 - 6.1 LOADING AND STARTING PROCEDURES
 - 6.1.1 LOADING PROCEDURES
 - 6.1.2 STARTING PROCEDURES
 - 6.1.3 STEPS FOR QUICK AND SIMPLE EXECUTION
 - 6.2 INITIAL DIALOGUE
 - 6.3 PROGRAM OPTIONS
 - 6.3.1 START COMMAND
 - 6.3.1.1 TESTS SWITCH
 - 6.3.1.2 PASS SWITCH
 - 6.3.1.3 FLAGS SWITCH
 - 6.3.1.4 END OF PASS SWITCH
 - 6.3.1.5 EFFECT OF START COMMAND
 - 6.3.2 RESTART COMMAND
 - 6.3.2.1 TESTS, PASS, AND FLAG SWITCHES
 - 6.3.2.2 UNITS SWITCH
 - 6.3.2.3 EFFECT OF RESTART COMMAND
 - 6.3.3 CONTINUE COMMAND
 - 6.3.3.1 PASS SWITCH
 - 6.3.3.2 FLAGS SWITCH
 - 6.3.3.3 EFFECT OF CONTINUE COMMAND
 - 6.3.4 PROCEED COMMAND
 - 6.3.4.1 FLAGS SWITCH
 - 6.3.4.2 EFFECT OF PROCEED COMMAND
 - 6.3.5 ADD COMMAND
 - 6.3.5.1 UNITS SWITCH
 - 6.3.5.2 EFFECT OF ADD COMMAND
 - 6.3.6 DROP COMMAND
 - 6.3.6.1 UNITS SWITCH
 - 6.3.6.2 EFFECT OF DROP COMMAND
 - 6.3.7 PRINT COMMAND

| | |
|------|--|
| 3390 | 6.3.7.1 EFFECT OF PRINT COMMAND |
| 3391 | 6.3.8 DISPLAY COMMAND |
| 3392 | 6.3.8.1 UNITS SWITCH |
| 3393 | 6.3.8.2 EFFECT OF DISPLAY COMMAND |
| 3394 | 6.3.9 FLAGS COMMAND |
| 3395 | 6.3.9.1 EFFECT OF FLAGS COMMAND |
| 3396 | 6.3.10 ZFLAGS COMMAND |
| 3397 | 6.3.10.1 EFFECT OF ZFLAGS COMMAND |
| 3398 | 6.3.11 CONTROL CHARACTERS |
| 3399 | 6.3.12 HARDWARE PARAMETERS |
| 3400 | 6.3.13 SOFTWARE PARAMETERS |
| 3401 | 6.3.14 EXTENDED DISCUSSION OF P-TABLE DIALOGUE |
| 3402 | |
| 3403 | 7.0 DEVICE INFORMATION TABLES |
| 3404 | |
| 3405 | 8.0 TEST DESCRIPTIONS |
| 3406 | 8.1 DATA PATTERNS USED |
| 3407 | |
| 3408 | 9.0 ERROR INFORMATION |
| 3409 | 9.1 ERROR REPORTING |

3411
3412
3413
3414
3415
3416
3417
3418
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3460
3461
3462
3463
3464
3465
3466

1 0 INTRODUCTION

THE M8203 IS A SINGLE-LINE SYNCHRONOUS LINE UNIT MODULE WHICH SUPPORTS BOTH CHARACTER-ORIENTED (DDCMP, BSC, ETC.) AND BIT-ORIENTED (SDLC, HDLC, ETC.) PROTOCOLS, AND WHICH IS CURRENTLY EMPLOYED IN THE DMP-11 DDCMP MULTIDROP PROJECT. THE PURPOSE OF THIS PROGRAM IS TO PERFORM DIAGNOSTIC TESTING OF ALL M8203 LOGIC IN A RELATIVELY STATIC MANNER. THE FOLLOWING FUNCTIONS WILL BE PERFORMED: LINE UNIT REGISTER ADDRESSING, USYRT ADDRESSING, STATIC BIT INTERACTION AND READ/WRITE LOGIC TESTS, BASIC TRANSMITTER AND RECEIVER SEQUENCING AND DATA BUFFERING AND STATIC OPERATIONS IN CHARACTER AND BIT-STUFFING MODES. IN ADDITION DATA MESSAGES WILL BE SENT AT SPEEDS OF 2400 BAUD TO 1 MEGABAUD, WITH LOOPBACK IN THE USYRT, ON THE LINE UNIT AT TTL LEVEL, OR THROUGH AN EXTERNAL TEST CONNECTOR WITH A SPECIFIC MODEM INTERFACE SELECTED.

THE STATIC LOGIC TESTS WILL PROVIDE EXTENSIVE TROUBLESHOOTING CAPABILITIES, SUCH AS TIGHT SCOPE LOOPS, SWITCH OPTIONS, AND ABILITY TO 'LOCK' ONTO INTERMITTENT ERRORS. IN ADDITION TESTS WILL BE DESIGNED AND STRUCTURED TO ACHIEVE MAXIMUM FAULT RESOLUTION AND FACILITATE REPLACEMENT OF THE SMALLEST FIELD REPLACEABLE UNIT.

THIS PROGRAM WILL BE IMPLEMENTED USING THE DIAGNOSTIC SUPERVISOR AND A STRUCTURED PROGRAMMING APPROACH. BECAUSE THE DESIGN WILL CONFORM TO THE SUPERVISOR (STANDALONE VERSION) THE PROGRAM WILL BE COMPATIBLE WITH ACT, APT, XXDP+, AND SLIDE.

THROUGH DIALOGUE WITH THE OPERATOR, THE PROGRAM WILL ALLOW MODIFICATION OF DEVICE PARAMETERS, SUCH AS UNIBUS ADDRESS, VECTOR ADDRESSES AND DEVICE PRIORITY. IN ADDITION, THE OPERATOR CAN SPECIFY PARTICULAR TESTS TO BE RUN AND A VARIETY OF LOOPING, RUNNING, AND REPORTING MODES.

DEVICE ERRORS WILL BE REPORTED AS THEY OCCUR. THE REPORT WILL INCLUDE A TEST NUMBER AND DESCRIPTION OF THE ERROR, GOOD AND BAD TEST DATA, AND APPLICABLE DEVICE REGISTER CONTENTS.

2.0 HARDWARE REQUIREMENTS

THE FOLLOWING HARDWARE IS REQUIRED TO RUN THE M8203 STATIC LOGIC TESTS:

PDP-11/04,05,10,20,30,34,35,40,45,50,60, OR 70
16K MEMORY

3467
3468
3469
3470
3471
3472
3473
3474
3475
3476
3477
3478
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3520
3521
3522

CONSOLE TERMINAL
DMC-11 OR KMC-11 MICROPROCESSOR
M8203 LINE UNIT AND BC08S-1 CABLE AND BERG CONNECTORS

3.0 PRELIMINARY PROGRAM REQUIREMENTS

THIS PROGRAM OPERATES THE MICROPROCESSOR EXTENSIVELY IN ORDER TO TEST THE LINE INIT. FOR THIS REASON, THE MICROPROCESSOR DIAGNOSTIC AND SUBSYSTEM FUNCTIONAL TESTS SHOULD BE RUN FIRST, AND ANY FAULTS FOUND IN THE MICROPROCESSOR MODULE SHOULD BE REPAIRED, PRIOR TO RUNNING THE M8203 STATIC LOGIC TESTS.

4.0 GENERAL PROGRAM CONSIDERATIONS

4.1 DIAGNOSTIC SUPERVISOR

THIS PROGRAM IS COMPATIBLE WITH THE STANDALONE DIAGNOSTIC SUPERVISOR, AND MUST BE LOADED TO BE CO-RESIDENT WITH THE SUPERVISOR, OR BE PREVIOUSLY COMBINED WITH THE SUPERVISOR AND LOADED AS A SINGLE FILE. IN EITHER CASE, THE COMBINED PROGRAM WILL NOT EXCEED 16K OF MEMORY.

4.2 EXECUTION TIME

THE MAXIMUM TIME REQUIRED TO RUN THE M8203 STATIC LOGIC TESTS IS ABOUT 45 SECONDS PER PASS FOR EACH UNIT.

4.3 XXDP+

THIS PROGRAM MAY BE LOADED UNDER XXDP+, AND MAY BE RUN IN DUMP MODE OR CHAIN MODE.

4.4 ACT/SLIDE

THIS PROGRAM MAY BE LOADED UNDER ACT OR SLIDE AND MAY BE RUN IN DUMP MODE OR CHAIN MODE.

4.5 APT

THIS PROGRAM MAY BE LOADED BY THE APT SYSTEM (INCLUDING APT-RD) AND RUN IN PROGRAM MODE OR SCRIPT MODE.

4.6 MEMORY MANAGEMENT

MEMORY MANAGEMENT IS NOT UTILIZED IN THIS PROGRAM. IF IT IS

3523
3524
3525
3526
3527
3528
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3560
3561
3562
3563
3564
3565
3566
3567
3568
3569
3570
3571
3572
3573
3574
3575
3576
3577
3578

INSTALLED, IT IS DISABLED BY THE PROGRAM.

4.7 MEMORY PARITY OPTION

IF PARITY MEMORY IS INSTALLED, MEMORY PARITY TRAPS ARE DISABLED BY THE PROGRAM.

4.8 ERROR LOGGING

AT THE END OF EACH PASS ON ALL UNITS, THE PROGRAM PRINTS OUT THE CUMULATIVE TOTAL NUMBER OF ERRORS SINCE THE LAST START OR RESTART COMMAND.

5.0 PROGRAM LOAD MEDIA

THIS PROGRAM CAN BE LOADED FROM PAPER TAPE USING THE ABSOLUTE LOADER OR FROM ACT, SLIDE, OR APT SYSTEMS, OR FROM ANY MEDIA SUPPORTED BY XXDP+. WHEN USING THE PAPER TAPE ABSOLUTE LOADER, THE PROGRAM SHOULD BE LOADED FIRST, FOLLOWED BY THE DIAGNOSTIC SUPERVISOR. WHEN USING XXDP+, THE DIAGNOSTIC SUPERVISOR SHOULD BE LOADED FIRST, FOLLOWED BY THE DIAGNOSTIC PROGRAM.

6.0 OPERATING INSTRUCTIONS

6.1 LOADING AND STARTING PROCEDURES

6.1.1 LOADING PROCEDURES

THIS PROGRAM MAY BE LOADED FROM PAPER TAPE USING THE ABSOLUTE LOADER. IT MAY ALSO BE LOADED FROM ANY XXDP+ LOAD MEDIA. WHEN LOADED UNDER XXDP+, THE DIAGNOSTIC SUPERVISOR WILL BE LOADED AUTOMATICALLY.

6.1.2 STARTING PROCEDURES

THE PROGRAM STARTS AT LOCATION 200. USE STANDARD DEC PROCEDURES TO START THE PROGRAM.

6.1.3 STEPS FOR QUICK AND SIMPLE EXECUTION

THE DIAGNOSTIC CAN BE EXECUTED STANDALONE UNDER XXDP+, WITHOUT READING THE REMAINDER OF THIS DOCUMENT, AS FOLLOWS:

- A) LOAD AND START DIAGNOSTIC USING RUN COMMAND
- B) RECEIVE DIAGNOSTIC SUPERVISOR IDENTIFICATION AND PROMPT (DRS-(>))
- C) ENTER STA<CR>

3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3599
3600
3601
3602
3603
3604
3605
3606
3607
3608
3609
3610
3611
3612
3613
3614
3615
3616
3617
3618
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629
3630
3631
3632
3633
3634

D) ANSWER HARDWARE AND SOFTWARE QUESTIONS
E) GET END OF PASS MESSAGES OR ERROR MESSAGES
F) TO END EXECUTION, ENTER CONTROL/C

6.2 INITIAL DIALOGUE

AFTER THE PROGRAM AND THE SUPERVISOR ARE LOADED AND THE PROGRAM IS STARTED, THE FOLLOWING IDENTIFICATION IS TYPED :

DRS LOADED
DIAG. RUN-TIME SERVICES
CZDMR-A-0
M8203 STATIC LOGIC TESTS - PART 1 OF 2
UNIT IS M8203
DR>

THE OPERATOR THEN PROCEEDS BY TYPING ONE OR MORE OF THE COMMANDS DESCRIBED IN THE FOLLOWING SECTION 6.3. (FOR MORE DETAILED INFORMATION, REFER TO THE DIAGNOSTIC SUPERVISOR FUNCTIONAL SPECIFICATION).

6.3 PROGRAM OPTIONS

6.3.1 START COMMAND

STA(RT)/TESTS:<TEST-LIST>/PASS:<PASS-CNT>/FLAGS:
<FLAG-LIST>/EOP:<INCR>

6.3.1.1 TESTS SWITCH (/TESTS:<TEST-LIST>)

<TEST-LIST> IS A SEQUENCE OF DECIMAL NUMBERS (1:2 ETC.) OR RANGES OF DECIMAL NUMBERS (1-5:8-10 ETC.) THAT SPECIFY THE TESTS TO BE EXECUTED. THE NUMBERS ARE SEPARATED BY COLONS. THE NUMBERS RANGE FROM 1 TO THE LARGEST TEST NUMBER IN THE DIAGNOSTIC. THEY MAY BE SPECIFIED IN ANY ORDER. TESTS WILL BE EXECUTED IN NUMERICAL ORDER REGARDLESS OF THE ORDER OF SPECIFICATION. THE DEFAULT IS TO EXECUTE ALL TESTS. ON THIS AND ALL SWITCHES, THE ANGLE BRACKETS <> ARE PUNCTUATION USED IN THE DEFINITION ONLY, AND ARE NOT TO BE TYPED BY THE OPERATOR. SEE EXAMPLE AT END OF 6.3.1.5.

6.3.1.2 PASS SWITCH (/PASS:<PASS-CNT>)

<PASS-CNT> IS A DECIMAL NUMBER INDICATING THE DESIRED NUMBER OF PASSES. A PASS IS DEFINED AS THE EXECUTION OF THE FULL DIAGNOSTIC (ALL SELECTED TESTS) AGAINST ALL UNITS SUBMITTED. THE DEFAULT IS NON-ENDING EXECUTION. IN THIS CASE EXIT FROM THE PROGRAM IS ACCOMPLISHED EITHER BY TYPING A CONTROL/C OR BY OCCURANCE OF AN ERROR WITH THE HALT ON ERROR FLAG BEING

3635
3636
3637
3638
3639
3640
3641
3642
3643
3644
3645
3646
3647
3648
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658
3659
3660
3661
3662
3663
3664
3665
3666
3667
3668
3669
3670
3671
3672
3673
3674
3675
3676
3677
3678
3679
3680
3681
3682
3683
3684
3685
3686
3687
3688
3689
3690

SET. THE EXIT IS A RETURN TO COMMAND MODE. SEE EXAMPLE AT
END OF 6.3.1.5.

6.3.1.3 FLAGS SWITCH (/FLAGS:<FLAG-LIST>)

<FLAG-LIST> IS A SEQUENCE OF ELEMENTS OF THE FORM <FLAG>, <FLAG=1>, OR <FLAG=0>, SEPARATED BY COLONS, WHERE <FLAG> HAS ONE OF THE FOLLOWING VALUES:

- HOE HALT ON ERROR, CAUSING COMMAND MODE TO BE ENTERED WHEN AN ERROR IS ENCOUNTERED
- LOE LOOP ON ERROR, CAUSING THE DIAGNOSTIC TO LOOP CONTINUOUSLY WITHIN THE SMALLEST DEFINED BLOCK OF CODING (SEGMENT, SUBTEST, OR TEST) CONTAINING THE ERROR
- IER INHIBIT ERROR REPORTING
- IBE INHIBIT BASIC ERROR REPORTS
- IXE INHIBIT EXTENDED ERROR REPORTS
- PRI DIRECT ALL MESSAGES TO A LINE PRINTER
- PNT PRINT NUMBER OF TEST BEING EXECUTED
- BOE BELL ON ERROR
- UAM RUN IN UNATTENDED MODE, BYPASSING MANUAL INTERVENTION TESTS
- ISR INHIBIT STATISTICAL REPORTS
- IDU INHIBIT DROPPING OF UNITS BY DIAGNOSTIC
- LOT LOOP ON TEST

THE FLAGS NAMED OR EQUATED TO 1 ARE SET, THOSE EQUATED TO 0 ARE CLEARED. A FLAG NOT SPECIFIED IS CLEARED. IF THE FLAGS SWITCH IS NOT GIVEN ALL FLAGS ARE CLEARED. SEE EXAMPLE AT END OF 6.3.1.5.

6.3.1.4 END OF PASS SWITCH (/EOP:<INCR>)

<INCR> IS A DECIMAL NUMBER INDICATING HOW OFTEN (IN TERMS OF PASSES) IT IS DESIRED THAT THE END OF PASS MESSAGE BE PRINTED. THE DEFAULT IS AT THE END OF EVERY PASS. SEE EXAMPLE AT END OF 6.3.1.5.

6.3.1.5 EFFECT OF START COMMAND

THE EFFECT OF THE START COMMAND IS TO INITIATE THE HARDWARE PARAMETER DIALOGUE, THE SOFTWARE PARAMETER DIALOGUE, AND THEN THE DIAGNOSTIC TESTS THEMSELVES.

THE HARDWARE PARAMETER DIALOGUE COMMENCES WITH THE QUESTION '# UNITS?' TO WHICH THE OPERATOR REPLIES WITH A DECIMAL NUMBER N FROM 1 TO 16. THE TERM 'UNIT' REFERS TO THE DEVICE TO WHICH THIS SERIES OF DIAGNOSTICS IS DEDICATED. FOLLOWING THIS ARE THE QUESTIONS WHEREBY THE P-TABLES THEMSELVES WILL BE BUILT. EACH P-TABLE IS A CORE-RESIDENT TABLE CONTAINING ALL THE HARDWARE INFORMATION FOR ONE UNIT. THE OPERATOR MUST SUPPLY N (NUMBER OF UNITS) VALUES FOR EACH QUESTION.

3691
3692
3693
3694
3695
3696
3697
3698
3699
3700
3701
3702
3703
3704
3705
3706
3707
3708
3709
3710
3711
3712
3713
3714
3715
3716
3717
3718
3719
3720
3721
3722
3723
3724
3725
3726
3727
3728
3729
3730
3731
3732
3733
3734
3735
3736
3737
3738
3739
3740
3741
3742
3743
3744
3745
3746

HE MAY DO THIS BY GIVING ONE ANSWER TO EACH QUESTION (IN WHICH CASE THE SERIES OF QUESTIONS WILL BE POSED N TIMES) OR BY GIVING N VALUES, SEPARATED BY COMMAS, TO EACH QUESTION (SERIES WILL BE POSED ONCE). EACH QUESTION IS FOLLOWED BY THE RESPONSE RADIX (D FOR DECIMAL, B FOR BINARY, O FOR OCTAL, L FOR YES/NO) IN PARENTHESES AND THE DEFAULT VALUE AFTER THE PARENTHESES.

FOLLOWING THE HARDWARE QUESTIONS ARE THE SOFTWARE QUESTIONS TO BUILD THE SOFTWARE TABLES, WHICH DEFINE THE MODE (QUICK VERIFY ETC.) THAT THE DIAGNOSTIC WILL EXECUTE IN.

WHEN THE QUESTION '# UNITS?' IS ANSWERED, MEMORY STORAGE IS ALLOCATED FOR THE P-TABLES, AND IF THERE IS NOT ENOUGH TO ACCOMMODATE THEM THE MESSAGE 'TOO MANY UNITS' IS ISSUED. IN THIS CASE THE DIAGNOSTIC MUST BE EXECUTED MORE THAN ONCE TO TEST ALL UNITS.

EXAMPLE:

STA/TESTS:1:2-4:6:8-10/PASS:3/FLAGS:IER:HOE=1:UAM:LOE

THIS COMMAND WILL CAUSE THREE PASSES TO BE MADE, EACH PASS CONSISTING OF TESTS 1,2,3,4,6,8,9, AND 10 EXECUTED AGAINST ALL UNITS. THERE IS NO DIFFERENCE BETWEEN SAYING <FLAG> AND SAYING <FLAG=1>. THE NOTATION <FLAG=0> IS MEANINGFUL ONLY ON A COMMAND OTHER THAN START TO CLEAR A FLAG THAT WAS PREVIOUSLY SET. NOTE THAT ON ALL COMMANDS ONLY THE FIRST THREE LETTERS ARE SCANNED.

6.3.2 RESTART COMMAND

```
*****  
RES(TART)/TESTS:<TEST-LIST>/PASS:<PASS-CNT>/FLAGS:  
<FLAG-LIST>/UNITS:<UNIT-LIST>  
*****
```

6.3.2.1 TESTS, PASS, AND FLAGS SWITCHES

<TEST-LIST>, <PASS-CNT>, AND <FLAG-LIST> ARE AS IN THE START COMMAND.

6.3.2.2 UNITS SWITCH (/UNITS:<UNIT-LIST>)

<UNIT-LIST> IS A SEQUENCE OF DECIMAL NUMBERS (0,1 ETC.) OR RANGES OF DECIMAL NUMBERS (0-5, 8-10 ETC.) THAT SPECIFY THE UNITS TO BE TESTED. THE NUMBERS ARE SEPARATED BY COLONS. THE NUMBERS MAY RANGE FROM 0 THRU N-1 (N IS THE NUMBER OF UNITS SPECIFIED IN THE PREVIOUS START COMMAND). THE NUMBER INDICATES THE POSITION OF THE P-TABLE AS THE DATA WAS ENTERED DURING THE HARDWARE DIALOGUE. THE UNITS WHICH ARE SELECTED MUST NOT HAVE BEEN DROPPED BY THE DROP COMMAND. SEE THE DISCUSSION OF ADD AND DROP COMMANDS BELOW. DEFAULT

3747
3748
3749
3750
3751
3752
3753
3754
3755
3756
3757
3758
3759
3760
3761
3762
3763
3764
3765
3766
3767
3768
3769
3770
3771
3772
3773
3774
3775
3776
3777
3778
3779
3780
3781
3782
3783
3784
3785
3786
3787
3788
3789
3790
3791
3792
3793
3794
3795
3796
3797
3798
3799
3800
3801
3802

IS TO TEST ALL UNITS WHICH HAVE NOT BEEN DROPPED BY A DROP
COMMAND.

6.3.2.3 EFFECT OF RESTART COMMAND

THE RESTART COMMAND DIFFERS FROM THE START COMMAND IN THAT
THE P-TABLES FROM THE PREVIOUS START COMMAND (THERE MUST
HAVE BEEN ONE) ARE USED, INSTEAD OF NEW ONES BEING BUILT.
THE UNITS SWITCH GIVES THE ABILITY TO SELECT A SUBSET OF
THESE. THE SOFTWARE DIALOGUE MAY OPTIONALLY BE REEXECUTED
(OPERATOR WILL BE ASKED). THE COMMAND CAN BE USED AFTER
COMMAND MODE HAS BEEN REENTERED IN ANY OF THE THREE NORMAL
WAYS: A) THE REQUESTED NUMBER OF PASSES HAVE BEEN MADE B)
AN ERROR WAS ENCOUNTERED WITH THE HALT ON ERROR FLAG SET C)
A CONTROL/C WAS ENTERED BY THE OPERATOR.

6.3.3 CONTINUE COMMAND

CON(TINUE)/PASS:<PASS-CNT/FLAGS:<FLAG-LIST>

6.3.3.1 PASS SWITCH (/PASS:<PASS-CNT>)

<PASS-CNT> IS SAME AS IN START COMMAND, BUT THE DEFAULT IS
THE UNSATISFIED PASS-CNT FROM THE PREVIOUS START OR RESTART.
IF NONE REMAINS, THE DEFAULT IS NON-ENDING EXECUTION.

6.3.3.2 FLAG SWITCH (/FLAGS:<FLAG-LIST>)

<FLAG-LIST> IS SAME AS IN START COMMAND, BUT UNSPECIFIED
FLAGS RETAIN THEIR CURRENT VALUE.

6.3.3.3 EFFECT OF CONTINUE COMMAND

CONTINUE MUST FOLLOW A START OR RESTART, AND COMMAND MODE
MUST HAVE BEEN ENTERED DUE TO A HALT ON ERROR OR A
CONTROL/C. THE EFFECT OF THE COMMAND IS TO GO TO THE
BEGINNING OF THE TEST THAT WAS BEING EXECUTED WHEN THE HALT
OR CONTROL/C TOOK PLACE. SOFTWARE DIALOGUE MAY OPTIONALLY
BE REEXECUTED. HARDWARE PARAMETERS MAY NOT BE CHANGED.

6.3.4 PROCEED COMMAND

PRO(CEED)/FLAGS:<FLAG-LIST>

6.3.4.1 FLAGS SWITCH (/FLAGS:<FLAG-LIST>)

3803
3804
3805
3806
3807
3808
3809
3810
3811
3812
3813
3814
3815
3816
3817
3818
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829
3830
3831
3832
3833
3834
3835
3836
3837
3838
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858

<FLAG-LIST> IS AS IN THE START COMMAND, BUT UNSPECIFIED
FLAGS RETAIN THEIR CURRENT VALUE.

6.3.4.2 EFFECT OF PROCEED COMMAND

PROCEED MUST FOLLOW A START, RESTART, OR CONTINUE. COMMAND
MODE MUST HAVE BEEN ENTERED VIA A HALT ON ERROR. THE EFFECT
OF THE COMMAND IS TO BEGIN EXECUTION AT THE LOCATION
FOLLOWING THE ERROR CALL. NEITHER HARDWARE NOR SOFTWARE
PARAMETERS MAY BE ALTERED.

6.3.5 ADD COMMAND

ADD/UNITS:<UNIT-LIST>

6.3.5.1 UNITS SWITCH (/UNITS:<UNIT-LIST>)

<UNIT-LIST> IS AS IN THE RESTART COMMAND.

6.3.5.2 EFFECT OF ADD COMMAND

THE UNITS SPECIFIED ARE ADDED TO THE TEST SEQUENCE. EACH
UNIT MUST HAVE A P-TABLE IN MEMORY DUE TO AN EARLIER
HARDWARE DIALOGUE. THIS COMMAND MUST BE FOLLOWED BY A
RESTART OR CONTINUE. THE UNITS SWITCH MUST BE SPECIFIED.
THE ADD COMMAND IS MEANINGFUL ONLY FOR UNITS THAT WERE
PREVIOUSLY DROPPED.

6.3.6 DROP COMMAND

DRO(P)/UNITS:<UNIT-LIST>

6.3.6.1 UNITS SWITCH (/UNITS:<UNIT-LIST>)

<UNIT-LIST> IS AS IN THE RESTART COMMAND.

6.3.6.2 EFFECT OF DROP COMMAND

THE UNITS SPECIFIED WILL BE DROPPED FROM TESTING. THE UNITS
WILL BE RESELECTED ONLY BY THE EXECUTION OF AN ADD OR START
COMMAND. THE UNITS SWITCH MUST BE ENTERED. THIS COMMAND
MUST BE FOLLOWED BY A RESTART OR A CONTINUE COMMAND.

3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3910
3911
3912
3913
3914

6.3.7 PRINT COMMAND

PRI(NT)

6.3.7.1 EFFECT OF PRINT COMMAND

THE TOTAL NUMBER OF ERRORS FOR EACH UNIT SINCE THE LAST
START OR RESTART COMMAND ARE PRINTED. THE ISR (INHIBIT
STATISTICAL REPORTING) FLAG IS CLEARED.

6.3.8 DISPLAY COMMAND

DIS(PLAY)/UNITS:<UNIT-LIST>

6.3.8.1 UNITS SWITCH (/UNITS:<UNIT-LIST>)

<UNIT-LIST> IS AS IN THE RESTART COMMAND.

6.3.8.2 EFFECT OF DISPLAY COMMAND

THE HARDWARE P-TABLES FOR ALL UNITS UNDER TEST ARE PRINTED
OUT IN THE FORMAT IN WHICH THEY WERE ENTERED. ANY UNITS
THAT WERE DROPPED BY THE OPERATOR 'DROP' COMMAND ARE SO
DESIGNATED.

6.3.9 FLAGS COMMAND

FLA(GS)

6.3.9.1 EFFECT OF FLAGS COMMAND

THE CURRENT SETTINGS OF ALL FLAGS ARE PRINTED.

6.3.10 ZFLAGS COMMAND

ZFL(AGS)

6.3.10.1 EFFECT OF ZFLAGS COMMAND

3915
3916
3917
3918
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928
3929
3930
3931
3932
3933
3934
3935
3936
3937
3938
3939
3940
3941
3942
3943
3944
3945
3946
3947
3948
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958
3959
3960
3961
3962
3963
3964
3965
3966
3967
3968
3969
3970

ALL FLAGS ARE CLEARED.

6.3.11 CONTROL CHARACTERS

A CONTROL C (C) ENTERED DURING THE EXECUTION OF A DIAGNOSTIC CAUSES A RETURN TO COMMAND MODE.

A CONTROL Z (Z) ENTERED DURING ONE OF THE THREE OPERATOR DIALOGUES- HARD CORE QUESTIONS (SEE 6.2), HARDWARE DIALOGUE (SEE 6.3.1.5), OR SOFTWARE DIALOGUE (SEE 6.3.1.5) CAUSES THE DEFAULTS TO BE TAKEN FOR THE REMAINDER OF THAT DIALOGUE.

A CONTROL O (O) ENTERED DURING THE EXECUTION OF A DIAGNOSTIC CAUSES ALL TELETYPE OUTPUT TO BE SURPRESSED FOR THE REMAINDER OF THE DIAGNOSTIC OR UNTIL ANOTHER O IS TYPED, WHICH RESTORES NORMAL TELETYPE OUTPUT.

6.3.12 HARDWARE PARAMETERS

THE FOLLOWING 4 QUESTIONS WILL BE ASKED ON A START COMMAND. THE VALUE LOCATED TO THE LEFT OF THE QUESTION MARK IS THE DEFAULT VALUE THAT WILL BE TAKEN ON A CARRIAGE RETURN RESPONSE.

1. DEVICE CSR ADDRESS : (O) 160170?

THIS IS THE ADDRESS AT WHICH THE CSR REGISTERS (SELO) RESIDE ON THE UNIBUS. THE ALLOWABLE RANGE IS 160000-177776 (OCTAL), AND THE DEFAULT VALUE IS 160170.

2. DEVICE VECTOR ADDRESS : (O) 300 ?

THIS IS THE ADDRESS OF THE INPUT INTERRUPT VECTOR FOR THIS DEVICE. THE ALLOWABLE RANGE IS 000-674 (OCTAL), AND THE DEFAULT VALUE IS 300.

3. DEVICE PRIORITY LEVEL : (O) 5 ?

THIS IS THE CPU PRIORITY AT WHICH THE INTERRUPT HANDLERS OF THIS DEVICE WILL BE EXECUTED. THE ALLOWABLE RANGE IS 0-7, AND THE DEFAULT VALUE IS 5.

4. MICROPROCESSOR RUN SWITCH - TYPE 0 IF OFF, 1 IF ON : (O) 1 ?

THIS TELLS THE PROGRAM IF THE RUN SWITCH ON THE MICROPROCESSOR IS SET OR NOT. IF IT IS SET, RUN IS NOT INHIBITED, AND TESTS REQUIRING THE RUN STATE CAN BE EXECUTED. THE ALLOWABLE VALUES ARE 0 AND 1, AND THE DEFAULT VALUE IS 1 (RUN IS NOT INHIBITED).

6.3.13 SOFTWARE PARAMETERS

NO SOFTWARE PARAMETER QUESTIONS ARE ASKED BY PART 1 OF THE STATIC LOGIC TESTS.

3971
3972
3973
3974
3975
3976
3977
3978
3979
3980
3981
3982
3983
3984
3985
3986
3987
3988
3989
3990
3991
3992
3993
3994
3995
3996
3997
3998
3999
4000
4001
4002
4003
4004
4005
4006
4007
4008
4009
4010
4011
4012
4013
4014
4015
4016
4017
4018
4019
4020
4021
4022
4023
4024
4025
4026

6.3.14 EXTENDED DISCUSSION OF P-TABLE DIALOGUE

THE FULL CAPABILITY OF THE HARDWARE DIALOGUE IS REVEALED BY THE FOLLOWING DISCUSSION OF WHAT HAPPENS INTERNALLY.

AS SOON AS THE QUESTION '# UNITS?' IS ANSWERED (WITH THE NUMBER N, SAY) SPACE IN CORE IS ALLOCATED FOR N P-TABLES. ALL OF THE P-TABLES ARE OF THE SAME FORMAT, AND THERE IS A ONE-TO ONE CORRESPONDENCE BETWEEN THE HARDWARE PARAMETER QUESTIONS AND THE SLOTS IN THE P-TABLE FORMAT.

ON THE FIRST TRIP THRU THE QUESTIONS, ALL OF THE SLOTS IN ALL OF THE P-TABLES ARE FILLED. IF THE OPERATOR TYPES IN LESS THAN N EXPLICIT VALUES IN RESPONSE TO A PARTICULAR QUESTION, THESE VALUES ARE PLACED IN THE P-TABLES (ONE VALUE GOING INTO THE PROPER SLOT OF EACH P-TABLE BEGINNING WITH THE FIRST P-TABLE) UNTIL THE STRING OF VALUES IS EXHAUSTED. THE LAST VALUE IN THE STRING BECOMES THE NEW DEFAULT AND IS USED TO FILL THAT SLOT IN THE REMAINING P-TABLES.

ON SUBSEQUENT TRIPS THRU THE QUESTIONS, THE SAME PROCESS IS CARRIED OUT, EXCEPT THAT THE EARLIEST P-TABLE NOT TO HAVE RECEIVED AN EXPLICIT VALUE IN ANY OF ITS SLOTS NOW ASSUMES THE ROLE THAT TABLE NUMBER ONE PLAYED IN THE FIRST TRIP.

THE SERIES OF QUESTIONS IS REISSUED UNTIL AT LEAST ONE QUESTION HAS RECEIVED N EXPLICIT VALUES FROM THE OPERATOR.

IN GIVING A STRING OF VALUES, COMMAS WITHOUT INTERVENING VALUES MAY BE USED TO INDICATE A REPETITION OF THE LAST NAMED VALUE.

A STRING OF VALUES MAY BE GIVEN AS A RANGE (6-10 FOR EXAMPLE). IF THE VALUES REPRESENT PURE NUMERICAL DATA, THIS SAMPLE RANGE TRANSLATES TO THE STRING 6,7,8,9,10 (AN INCREMENT OF 1). IF THE VALUES ARE ADDRESSES, THE SAMPLE RANGE TRANSLATES TO THE STRING 6,8,10 (AN INCREMENT OF 2).

NOW LET US SEE HOW WE COULD USE THESE CAPABILITIES TO CONSTRUCT A SET OF P-TABLES. ASSUME THAT WE HAVE 16 UNITS, AND THAT THERE ARE THREE HARDWARE PARAMETERS FOR EACH (THREE SLOTS IN THE P-TABLE, THREE HARDWARE QUESTIONS IN THE DIALOGUE). LET THE DESIRED VALUE FOR THE FIRST PARAMETER BE THE NUMBER 75 FOR ALL 16 TABLES. LET THE DESIRED VALUE FOR THE SECOND PARAMETER BE EQUAL TO THE UNIT NUMBER (0,1,2,...,15) EXCEPT FOR UNIT 12, WHICH SHOULD RECEIVE THE VALUE 11. LET THE DESIRED VALUE FOR THE THIRD PARAMETER BE THE NUMBER 76 FOR THE FIRST 7 UNITS AND THE NUMBER 77 FOR THE LAST 9 UNITS.

THE FOLLOWING DIALOGUE WOULD ACCOMPLISH THIS GOAL:

UNITS (D) ? 16
UNIT 0

4027
4028
4029
4030
4031
4032
4033
4034
4035
4036
4037
4038
4039
4040
4041
4042
4043
4044
4045
4046
4047
4048
4049
4050
4051
4052
4053
4054
4055

<QUESTION 1> ? 75
<QUESTION 2> ? 0-6
<QUESTION 3> ? 76

UNIT 7
<QUESTION 1> ?
<QUESTION 2> ? 7-11,,13-15
<QUESTION 3> ? 77

THE FIRST TIME THE SERIES IS ASKED, SLOT ONE RECEIVES A 75 IN ALL 16 TABLES. SLOT TWO RECEIVES THE VALUES 0,1,2,...,6 IN TABLES 0 THRU 6 AND A CONSTANT 6 IN TABLES 7 THRU 15. SLOT THREE RECEIVES A CONSTANT 76 IN ALL 16 TABLES.

THE SECOND TIME THRU THE SERIES, TABLES 7 THRU THE END ARE GOING TO BE AFFECTED (NOTE THAT THIS PIECE OF INFORMATION IS PRINTED OUT FOR THE THE OPERATOR IN THE FORM 'UNIT XX' AT THE BEGINNING OF EACH SERIES). QUESTION 1 IS RESPONDED TO BY A <CR>, SO SLOT ONE STAYS AT CONSTANT 75 IN TABLES 7 THRU 15, SINCE NO NEW EXPLICIT VALUES ARE TYPED IN. SLOT TWO GETS THE VALUES 7,8,9,10,11 IN TABLES 7 THRU 11, AND GETS AN 11 IN SLOT 12, AND GETS THE VALUES 13,14,15 IN TABLES 13 THRU 15. SLOT THREE GETS THE VALUE 77 IN TABLES 7 THRU 15.

THE DIALOGUE IS TERMINATED WHEN THE SOFTWARE RECOGNIZES THAT 16 EXPLICIT VALUES HAVE BEEN GIVEN FOR AT LEAST ONE QUESTION (NAMELY QUESTION 2).

4057
4058
4059
4060
4061
4062
4063
4064
4065
4066
4067
4068
4069
4070
4071
4072
4073
4074
4075
4076
4077
4078
4079
4080
4081
4082
4083
4084
4085
4086
4087
4088
4089
4090
4091
4092
4093
4094
4095
4096
4097
4098
4099
4100
4101
4102
4103
4104
4105
4106
4107
4108
4109
4110
4111
4112

7.0 DEVICE INFORMATION TABLES

```
:*****  
* MAINTENANCE REGISTER - BSEL1  
:*****  
RUN      = BIT7  
MCLR     = BIT6  
STEPLU   = BIT4  
LULOOK   = BIT3  
ROMO     = BIT2  
ROMI     = BIT1  
STEPMP   = BIT0  
  
:*****  
* OBUS REG 10 - TRANSMITTER BUFFER  
:*****  
TX7      = BIT7  
TX6      = BIT6  
TX5      = BIT5  
TX4      = BIT4  
TX3      = BIT3  
TX2      = BIT2  
TX1      = BIT1  
TX0      = BIT0  
  
:*****  
* OBUS REG 11  
:*****  
OC       = BIT7  
GOAH     = BIT3  
ABORT    = BIT2  
EOM      = BIT1  
SOM      = BIT0  
  
:*****  
* OBUS REG 12  
:*****  
IC       = BIT7  
BPOLL    = BIT6  
LULP     = BIT5  
  
:*****  
* OBUS REG 13  
:*****  
POLL     = BIT7  
DTR      = BIT6  
SELFR    = BIT5  
HDX      = BIT4  
MAINT1   = BIT3  
MAINT2   = BIT2  
SELSBY   = BIT1
```

4113
4114
4115
4116
4117
4118
4119
4120
4121
4122
4123
4124
4125
4126
4127
4128
4129
4130
4131
4132
4133
4134
4135
4136
4137
4138
4139
4140
4141
4142
4143
4144
4145
4146
4147
4148
4149
4150
4151
4152
4153
4154
4155
4156
4157
4158
4159
4160
4161
4162
4163
4164
4165
4166
4167
4168

```
*****  
* OBUS REG 14  
*****  
TXEN      = BIT6  
DISSI     = BIT5  
RDAX      = BIT4  
WAX       = BIT3  
ENAX      = BIT2  
AX2       = BIT1  
AX1       = BIT0  
  
*****  
* OBUS REG 17  
*****  
CRC2      = BIT7  
CRC1      = BIT6  
IDLE      = BIT5  
SECA      = BIT4  
STRIP     = BIT3  
RDALL     = BIT2  
IERR      = BIT1  
DDCMP     = BIT0  
  
*****  
* IBUS REG 10 - RECEIVER BUFFER  
*****  
RX7       = BIT7  
RX6       = BIT6  
RX5       = BIT5  
RX4       = BIT4  
RX3       = BIT3  
RX2       = BIT2  
RX1       = BIT1  
RX0       = BIT0  
  
*****  
* IBUS REG 11  
*****  
OC        = BIT7  
OACT      = BIT6  
SW3       = BIT5  
ORDY      = BIT4  
SW2       = BIT3  
SW1       = BIT2  
SW0       = BIT1  
UNRR      = BIT0  
  
*****  
* IBUS REG 12  
*****  
IC        = BIT7  
IACT      = BIT6  
LULP     = BIT5  
IRDY     = BIT4  
OVRT     = BIT3  
RAB      = BIT2
```

4169 EBLK = BIT1
4170 BCC = BIT0

4171
4172 ;*****
4173 * IBUS REG 13

4174 ;*****
4175 RING = BIT7
4176 DTR = BIT6
4177 RTS = BIT5
4178 HDX = BIT4
4179 MODR = BIT3
4180 CS = BIT2
4181 STBY = BIT1
4182 CARR = BIT0
4183

4184 ;*****
4185 * IBUS REG 14

4186 ;*****
4187 READY = BIT7
4188 TXEN = BIT6
4189 DISSI = BIT5
4190 RDAX = BIT4
4191 WAX = BIT3
4192 ENAX = BIT2
4193 AX2 = BIT1
4194 AX1 = BIT0
4195

4196 ;*****
4197 * IBUS REG 17

4198 ;*****
4199 SIGR = BIT7
4200 SIGQ = BIT6
4201 TXDATA = BIT5
4202 OCOR = BIT4
4203 ICIR = BIT3
4204 TESTMD = BIT2
4205 MCLK = BIT1
4206 DDCMP = BIT0
4207

4208 ;*****
4209 * AX0-15 - USYRT REG 0 (READ ONLY)

4210 ;*****
4211 RX7 = BIT7
4212 RX6 = BIT6
4213 RX5 = BIT5
4214 RX4 = BIT4
4215 RX3 = BIT3
4216 RX2 = BIT2
4217 RX1 = BIT1
4218 RX0 = BIT0
4219

4220 ;*****
4221 * AX0-16 - USYRT REG 1 (READ ONLY)

4222 ;*****
4223 RERR = BIT7
4224 ASBC2 = BIT6

4225 ASBC1 = BIT5
4226 ASBC0 = BIT4
4227 ROR = BIT3
4228 RABT = BIT2
4229 REOM = BIT1
4230 RSOM = BIT0

* AX1-15 - USYRT REG 2

4231 TX7 = BIT7
4232 TX6 = BIT6
4233 TX5 = BIT5
4234 TX4 = BIT4
4235 TX3 = BIT3
4236 TX2 = BIT2
4237 TX1 = BIT1
4238 TX0 = BIT0

* AX1-16 - USYRT REG 3

4239 TERR = BIT7
4240 TXGA = BIT3
4241 TXAB = BIT2
4242 TEOM = BIT1
4243 TSOM = BIT0

* AX2-15 - USYRT REG 4

4244 SYN7 = BIT7
4245 SYN6 = BIT6
4246 SYN5 = BIT5
4247 SYN4 = BIT4
4248 SYN3 = BIT3
4249 SYN2 = BIT2
4250 SYN1 = BIT1
4251 SYN0 = BIT0
4252 SYNCH = 226

* AX2-16 - USYRT REG 5

4253 APA = BIT7
4254 DDC = BIT6
4255 STR = BIT5
4256 SEC = BIT4
4257 IDL = BIT3
4258 CRCTY2 = BIT2
4259 CRCTY1 = BIT1
4260 CRCTY0 = BIT0

* AX3-15 - USYRT REG 6

4261

4281 I422 = BIT7
4282 XYZ = BIT6
4283 V35 = BIT4
4284 INTGRL = BIT3
4285 OP = BIT1
4286 TEST = BIT0
4287 AX315U = I422!XYZ!V35!INTGRL.OP
4288
4289

:*****
* AX3-16 - USYRT REG 7
:*****

4291
4292 TXLEN2 = BIT7
4293 TXLEN1 = BIT6
4294 TXLENO = BIT5
4295 RXLEN2 = BIT2
4296 RXLEN1 = BIT1
4297 RXLENO = BIT0
4298
4299
4300
4301
4302

4304
4305
4306
4307
4308
4309
4310
4311
4312
4313
4314
4315
4316
4317
4318
4319
4320
4321
4322
4323
4324
4325
4326
4327
4328
4329
4330
4331
4332
4333
4334
4335
4336
4337
4338
4339
4340
4341
4342
4343
4344
4345
4346
4347
4348
4349
4350
4351
4352
4353
4354
4355
4356
4357
4358
4359

8.0 TEST DESCRIPTIONS

TEST 1 - MICROPROCESSOR CSR ADDRESSING TEST (SELO)
*
* THIS TEST ADDRESSES THE FIRST MICROPROCESSOR CSR (SELO), TO MAKE SURE
* THAT A NON-EXISTENT MEMORY TIME-OUT TRAP DOES NOT OCCUR WHILE
* ATTEMPTING TO ADDRESS THE MICROPROCESSOR.

TEST 2 - INBUS/OUTBUS REG 14 INITIALIZATION TEST
*
* MASTER CLEAR (MCLR) IS SET IN THE MICROPROCESSOR, IBUS REG 14 IS READ
* AND COMPARED TO 200.

TEST 3 - INBUS/OUTBUS REG 14 READ/WRITE BIT TEST
*
* WRITE, READ, AND COMPARE ALL WORDS OF DATA PATTERN A INTO REG 14,
* A BYTE AT A TIME. NON-R/W BITS ARE MASKED OFF TO 0 BEFORE WRITING AND
* READING.
* DATA PATTERN A = 125,252,000,377,001,002,004,010,020,040,100,200,376,
* 375,373,367,357,337,277,177.

TEST 4 - REG 14 MASTER CLEAR TEST
* WRITE 377 INTO REG 14, ISSUE MASTER CLEAR, READ REG 14 AND COMPARE
* TO 200.

4360
4361
4362
4363
4364
4365
4366
4367
4368
4369
4370
4371
4372
4373
4374
4375
4376
4377
4378
4379
4380
4381
4382
4383
4384
4385
4386
4387
4388
4389
4390
4391
4392
4393
4394
4395
4396
4397
4398
4399
4400
4401
4402
4403
4404
4405
4406
4407
4408
4409
4410
4411
4412
4413
4414
4415

```
*****  
TEST 5 - REG 14 UNIBUS RESET (INIT) TEST  
* WRITE 377 INTO REG 14, ISSUE UNIBUS RESET (INIT), READ REG 14 AND COMPARE  
* TO 200.  
*****
```

```
*****  
TEST 6 - LINE UNIT FALSE SELECTION TEST  
* FIRST, A MASTER CLEAR IS PERFORMED. THEN, THE PROGRAM SINGLE-STEPS THE  
* MICROPROCESSOR THROUGH AN INSTRUCTION WHICH LOADS 041 (OCT) INTO THE MAR  
* REGISTER (OBUS* ADRS 14). THEN, THE LINE UNIT REGISTER 14 IS READ AND CHECKED  
* TO BE UNAFFECTED (STILL = 0). THIS TEST IS INTENDED TO DETECT A FALSE  
* SELECTION OF THE LINE UNIT REGISTERS, WHEN THE LINE UNIT IS NOT BEING  
* ACCESSED.  
*****
```

```
*****  
TEST 7 - INBUS REG MASTER CLEAR TEST  
* FIRST, ALL READ/WRITE BITS OF REGS 10-17 ARE SET BY LOADING A  
* DIFFERENT WORD OF PATTERN G INTO EACH REG. THEN,  
* A MASTER CLEAR IS ISSUED AND EACH REG IS READ AND COMPARED TO A WORD OF  
* PATTERN M, WHICH CONTAINS THE INITIALIZED STATES OF THE REGS. (UNPREDICTABLE  
* BITS ARE MASKED OFF TO 0 BEFORE COMPARISON).  
* PATTERN G = 000,000,240,120,177,000,000,001  
* PATTERN M = 000,020,000,000,200,000,000,051  
*****
```

```
*****  
TEST 8 - REGISTER 10-17 ADDRESSING TEST  
* FIRST, A MASTER CLEAR IS ISSUED. THEN,  
* WRITE A DIFFERENT WORD OF DATA PATTERN B INTO EACH OF REGS 10-17,  
* AND AFTER EACH WRITE, READ AND COMPARE ALL REGS TO EXPECTED VALUES.  
* UNPREDICTABLE BITS ARE MASKED OFF TO 0 WHEN READ FOR COMPARISON.  
* PATTERN B = 000,000,040,100,220,000,000,051  
*****
```

4416
4417
4418
4419
4420
4421
4422
4423
4424
4425
4426
4427
4428
4429
4430
4431
4432
4433
4434
4435
4436
4437
4438
4439
4440
4441
4442
4443
4444
4445
4446
4447
4448
4449
4450
4451
4452
4453
4454
4455
4456
4457
4458
4459
4460
4461
4462
4463
4464
4465
4466
4467
4468
4469
4470
4471

; TEST 9 - REG 11 READ/WRITE BIT TEST

*
* WRITE, READ, AND COMPARE EACH WORD OF DATA PATTERN C INTO REG 11 :
* DATA PATTERN C = 020,020,020.

; TEST 10 - REG 12 READ/WRITE BIT TEST

*
* WRITE, READ, AND COMPARE EACH WORD OF DATA PATTERN D INTO REG 12 :
* DATA PATTERN D = 000,040,000.

; TEST 11 - REG 13 READ/WRITE BIT TEST

*
* WRITE, READ, AND COMPARE EACH WORD OF DATA PATTERN E INTO REG 13 :
* DATA PATTERN E = 000,120,020,100,120,000.

; TEST 12 - REG 17 READ/WRITE BIT TEST

*
* WRITE, READ, AND COMPARE EACH WORD OF DATA PATTERN F INTO REG 17 :
* DATA PATTERN F = 050,051,050.

; TEST 13 - MAINTENANCE CLOCK BIT TEST

*
* FIRST, A MASTER CLEAR IS ISSUED TO INIT ALL REGS. THEN, THE MICROPROCESSOR
* IS PLACED IN A LOOP ON AN INSTRUCTION, BY SETTING THE INSTRUCTION IN SEL6
* AND SETTING ROMI AND RUN IN BSEL1. THE INSTRUCTION IS ONE WHICH REPETITIVELY
* READS LINE UNIT REG 17 INTO BSEL2. THE PDP-11 CAN THEN SCAN BSEL2 TO MONITOR
* THE MAINTENANCE CLOCK BIT, MCLK. THE FOLLOWING SEQUENCE IS THEN PERFORMED
* TO MONITOR MCLK :
* - THE PROGRAM REPEATEDLY CHECKS THE MCLK BIT FOR THE 1 STATE, AND IF IT IS
* NOT FOUND WITHIN SEVERAL HUNDRED MILLI-SEC (DEPENDING ON THE PROCESSOR)
* AN ERROR IS REPORTED. (THE MAINTENANCE CLOCK HAS A PERIOD OF 41.6 MICRO-

4472
4473
4474
4475
4476
4477
4478
4479
4480
4481
4482
4483
4484
4485
4486
4487
4488
4489
4490
4491
4492
4493
4494
4495
4496
4497
4498
4499
4500
4501
4502
4503
4504
4505
4506
4507
4508
4509
4510
4511
4512
4513
4514
4515
4516
4517
4518
4519
4520
4521
4522
4523
4524
4525
4526
4527

* SEC).
* - THE PROGRAM NEXT REPEATEDLY CHECKS THE MCLK BIT FOR THE 0 STATE, AND IF
* IT IS NOT FOUND WITHIN SEVERAL HUNDRED MILLI-SEC AN ERROR IS REPORTED.
* - THE PROGRAM NEXT REPEATEDLY CHECKS MCLK BIT FOR THE 1 STATE AGAIN, AND
* IF IT IS NOT FOUND WITHIN SEVERAL HUNDRED MILLI-SEC, AN ERROR IS REPORTED.
*
* IF THE P-TABLE FOR THIS UNIT INDICATES THAT THE MICROPROCESSOR RUN SWITCH
* IS OFF, THE TEST WILL BE SKIPPED.
;*****

;*****
TEST 14 - EXTENDED REGISTER MASTER CLEAR TEST
*
* FIRST, ALL READ/WRITE BITS OF EXTENDED REGS AX0-AX3 ARE SET BY LOADING
* A DIFFERENT WORD OF PATTERN H INTO EACH REG. THEN, A MASTER CLEAR IS
* ISSUED AND EACH REG IS READ AND COMPARED TO A WORD OF PATTERN I, WHICH
* CONTAINS THE INITIALIZED STATES OF ALL THE EXTENDED REGS.
* PATTERN H = 000,000,377,017,377,377,375,377
* PATTERN I = 000,000,000,000,000,103,000,000
;*****

;*****
TEST 15 - EXTENDED REGISTER ADDRESSING TEST
*
* FIRST, ISSUE A MASTER CLEAR TO PUT REGS INTO INITIALIZED STATES SHOWN IN
* PATTERN I. THEN, WRITE A DIFFERENT WORD OF PATTERN J INTO EACH EXTENDED (AX)
* REG, AND AFTER EACH WRITE, READ AND COMPARE ALL EXTENDED REGS TO EXPECTED
* VALUES.
* PATTERN I = 000,000,000,000,000,103,000,000
* PATTERN J = 000,000,001,002,004,103,040,100
;*****

;*****
TEST 16 - REGS 15,16 / AX2-15,AX2-16 READ/WRITE BIT TEST
*
* USING REGS 15,16, THE INDIRECT REGS AX2-15,AX2-16 (USYRT REGS 4,5) ARE
* WRITTEN AND READ USING EACH WORD OF PATTERN K. AX2-15 IS COMPARED
* TO THE WORD WRITTEN, AND AX2-16 IS ALWAYS COMPARED TO 103. (AX2-16 IS NOT
* WRITEABLE).
* PATTERN K =
* FOR REG 15: 000,377,125,252,001,002,004,010,020,040,100,200,000,000,
* 000,000,000,000,000,000,000,376,375,373,367,357,337,277,177,
* 377,377,377,377,377,377,377,377
* FOR REG 16: 000,377,125,252,000,000,000,000,000,000,000,000,001,002,

4584
4585
4586
4587
4588
4589
4590
4591
4592
4593
4594
4595
4596
4597
4598
4599
4600
4601
4602
4603
4604
4605
4606
4607
4608
4609
4610
4611
4612
4613
4614
4615
4616
4617
4618
4619
4620
4621
4622
4623
4624
4625
4626
4627
4628
4629
4630
4631
4632
4633
4634
4635
4636
4637
4638
4639

```
*****  
TEST 20 - REG 17 - AX2-16 READ/WRITE, MASTER CLEAR TEST  
*  
* THIS TEST CONSISTS OF 2 SUBTESTS. IN THE FIRST SUBTEST, EACH BYTE OF PAT O  
* IS WRITTEN INTO REG 17 AND AFTER EACH WRITE, AX2-16 IS READ AND COMPARED  
* TO A BYTE OF PAT P.  
* PATTERN O = 000,041,004,010,020,040,100,101,200,201,300,111,301,375  
* PATTERN P = 000,113,200,040,020,010,001,104,007,105,007,144,107,157  
* IN THE SECOND SUBTEST, REG 17 IS LOADED WITH 375, A MASTER CLEAR IS ISSUED,  
* AND AX2-16 IS COMPARED TO ITS INITIALIZED STATE (103).  
*****
```

```
*****  
TEST 21 - TRANSMITTER BUFFER DATA TEST  
* A MASTER CLEAR IS DONE FIRST, AND THEN A BYTE OF PATTERN N IS LOADED INTO  
* REG 11 AND THE NEXT BYTE IS LOADED TWICE INTO REG 10. THE PROGRAM THEN WAITS  
* AT LEAST 50 MICRO-SEC, AND THEN IT READS AND COMPARES AX1-15 TO THE BYTE WHICH  
* WAS LOADED INTO REG 10, AND IT READS AND COMPARES AX1-16 TO THE BYTE WHICH WAS  
* LOADED INTO REG 11. THIS PROCESS IS REPEATED (INCLUDING THE MASTER CLEAR)  
* FOR EACH PAIR OF BYTES IN PATTERN N.  
* PATTERN N =  
* FOR REG 10: 000,125,252,377,000,000,000  
* FOR REG 11: 000,000,000,000,005,012,017  
*****
```

```
*****  
TEST 22 - TRANSMITTER BUFFER SEQUENCING TEST  
*  
* FIRST, A MASTER CLEAR IS DONE, AND THE PROGRAM CHECKS FOR ORDY=1, OCOR=0.  
* THEN, 2 TSOM CHARS ARE LOADED INTO THE TX SILO, AND ALLOWED TO RIPPLE  
* DOWN TO THE OUTPUT. THE PROGRAM CHECKS FOR ORDY=1, OCOR=1.  
* NEXT, THE PROGRAM CYCLES THE STEPLU BIT UNTIL OCOR=0 AGAIN, AND CHECKS FOR  
* THIS TO OCCUR WITHIN 3 CYCLES.  
* THE SILO IS THEN FILLED WITH 64 BYTES OF A 256-BYTE BINARY COUNT PATTERN  
* (000-377) AND THE PROGRAM CHECKS FOR ORDY=0 AFTER THE 64TH CHAR IS LOADED.  
* THE PROGRAM CYCLES STEPLU FOR 8 CYCLES AND CHECKS THAT AFTER THE 8TH, ORDY=1.  
* AX1-15 IS READ AND COMPARED TO EXPECTED DATA.  
* THE REST OF THE BINARY COUNT DATA BYTES ARE LOADED, CYCLED 8 CLOCKS, READ AND  
* COMPARED, A BYTE AT A TIME. UPON COMPLETION, THE SILO IS CHECKED TO BE EMPTY  
* WITH ORDY=1, OCOR=0.  
*****
```

4640
4641
4642
4643
4644
4645
4646
4647
4648
4649
4650
4651
4652
4653
4654
4655
4656
4657
4658
4659
4660
4661
4662
4663
4664
4665
4666
4667
4668
4669
4670
4671
4672
4673
4674
4675
4676
4677
4678
4679
4680
4681
4682
4683
4684
4685
4686
4687
4688
4689
4690
4691
4692
4693
4694
4695

```
*****  
; TEST 23 - TX MSG TIMING TEST, CHAR MODE, WITH CRC  
*  
* IN THIS TEST, AN ENTIRE MESSAGE IS TRANSMITTED (USING STEPLU AND LULOOP)  
* AND THE PROGRAM MONITORS THE OCCURRENCE OF USYRT TX BUFFER EMPTY FLAGS  
* (BY SCANNING ORDY AND OCOR) AND OACT, AT EACH STEP. THE TEST IS DONE IN  
* CHARACTER ORIENTED PROTOCOL MODE, USING 8-BIT CHARS AND CRC-16.  
* THE FOLLOWING STEPS ARE DONE:  
* A MASTER CLEAR IS DONE, AND THE LINE UNIT IS PLACED IN CHAR MODE.  
* SOM IS SET TWICE TO SEND 2 SYNCH CHARS. THEN, 2 000 CHARS ARE SENT, AND  
* THEN 2 TERMINATING SYNCHS ARE SENT.  
* THE TRANSMITTER IS THEN DISABLED, USING OC, AND OACT IS MONITORED FOR THE  
* CLEARED STATE AFTER THE 3RD SYNCH COMPLETES.  
*****
```

```
*****  
; TEST 24 - TX MSG TIMING TEST, BIT MODE, WITH CRC  
*  
* IN THIS TEST, AN ENTIRE MESSAGE IS TRANSMITTED (USING STEPLU AND LULOOP)  
* AND THE PROGRAM MONITORS THE OCCURRENCE OF USYRT TX BUFFER EMPTY FLAGS  
* (BY SCANNING ORDY AND OCOR) AND OACT, AT EACH STEP. THE TEST IS DONE IN  
* BIT ORIENTED PROTOCOL MODE, USING 8-BIT CHARS AND CRC-CCITT-1.  
* THE FOLLOWING STEPS ARE DONE:  
* A MASTER CLEAR IS DONE, AND THE LINE UNIT IS PLACED IN BIT MODE.  
* SOM IS SET TWICE TO SEND 2 FLAG CHARS. THEN, 2 000 CHARS ARE SENT, AND  
* THEN 2 TERMINATING FLAGS ARE SENT.  
* THE TRANSMITTER IS THEN DISABLED, USING OC, AND OACT IS MONITORED FOR THE  
* CLEARED STATE.  
*****
```

```
*****  
; TEST 25 - TX MSG TIMING TEST, CHAR MODE, WITH NO CRC  
*  
* IN THIS TEST, AN ENTIRE MESSAGE IS TRANSMITTED (USING STEPLU AND LULOOP)  
* AND THE PROGRAM MONITORS THE OCCURRENCE OF USYRT TX BUFFER EMPTY FLAGS  
* (BY SCANNING ORDY AND OCOR) AND OACT, AT EACH STEP. THE TEST IS DONE IN  
* CHARACTER ORIENTED PROTOCOL MODE, USING 8-BIT CHARS AND NO ERROR CHECKING.  
* THE FOLLOWING STEPS ARE DONE:  
* A MASTER CLEAR IS DONE, AND THE LINE UNIT IS PLACED IN CHAR MODE.  
* SOM IS SET TWICE TO SEND 2 SYNCH CHARS. THEN, 2 000 CHARS ARE SENT, AND  
* THEN 2 TERMINATING SYNCHS ARE SENT.  
* THE TEST IS PERFORMED WITH TXEN (REG 14, BIT6) SET, AND THE PROGRAM CHECKS  
* THAT THIS HOLDS RTS HIGH PAST THE END OF THE MESSAGE.  
* THE TRANSMITTER IS THEN DISABLED, USING OC, AND OACT IS MONITORED FOR THE  
* CLEARED STATE.
```

4696
4697
4698
4699
4700
4701
4702
4703
4704
4705
4706
4707
4708
4709
4710
4711
4712
4713
4714
4715
4716
4717
4718
4719
4720
4721
4722
4723
4724
4725
4726
4727
4728
4729
4730
4731
4732
4733
4734
4735
4736
4737
4738
4739
4740
4741
4742
4743
4744
4745
4746
4747
4748
4749
4750
4751

TEST 26 - TX UNDERRUN SET AND CLEAR TEST - CHAR MODE

*
* IN THIS TEST, A TX UNDERRUN ERROR IS FORCED IN EACH OF 2 SITUATIONS,
* AND THEN CLEARED DIFFERENTLY IN EACH.
* IN THE FIRST, A MESSAGE IS INITIATED, A 000 CHAR IS SENT, AND THE TX
* BUFFER IS NOT SERVICED IN RESPONSE TO THE USYRT TX BUFFER EMPTY FLAG,
* WHICH CAUSES UNRR TO SET IN REG 11. THEN, SOM IS SET TO CLEAR THE ERROR,
* AND THIS IS VERIFIED.
* IN THE SECOND SITUATION, A MSG IS INITIATED, A 000 CHAR IS SENT, AND THE TX
* BUFFER IS NOT SERVICED IN RESPONSE TO THE USYRT TX BUFFER EMPTY FLAG, WHICH
* AGAIN CAUSES UNRR TO SET. THEN, A MASTER CLEAR IS DONE, AND THE UNRR BIT
* IS CHECKED TO BE CLEARED.

TEST 27 - TRANSMIT CHAR LENGTH TIMING TEST - CHAR MODE, CRC

*
* THE LINE UNIT IS PLACED IN CHAR MODE (DDCMP) AND A MESSAGE IS INITIATED
* WITH AN 8-BIT SYNCH AND A 5-BIT SYNCH CHAR. NEXT, A 000 CHAR IS SENT WITH
* EACH OF THE FOLLOWING TX CHAR LENGTHS : 5 BITS, 6 BITS, 7 BITS, 8 BITS.
* (FOR EXAMPLE, A 5-BIT CHAR REQUIRES 5 CLOCK CYCLES TO BE TRANSMITTED). TWO
* TERMINATING SYNCHS ARE SENT AFTER THE DATA.

TEST 28 - TRANSMIT CHAR LENGTH TIMING TEST - BIT MODE, CRC

*
* THE LINE UNIT IS PLACED IN BIT MODE AND A MESSAGE IS INITIATED
* WITH 2 FLAG CHARS. NEXT, 2 8-BIT 000 CHARS ARE SENT, FOLLOWED BY 000 CHARS
* WITH EACH OF THE FOLLOWING TRANSMITTER CHAR LENGTHS:
* 1 BIT, 2 BITS, 3 BITS, 4 BITS, 5 BITS, 6 BITS, 7 BITS, AND 8 BITS.
* (FOR EXAMPLE, A 5-BIT CHAR REQUIRES 5 CLOCK CYCLES TO BE TRANSMITTED).
* TWO TERMINATING FLAGS ARE SENT AFTER THE DATA.

TEST 29 - TXDATA BIT TEST - CHAR MODE, NO CRC

4752
4753
4754
4755
4756
4757
4758
4759
4760
4761
4762
4763
4764
4765
4766
4767
4768
4769
4770
4771
4772
4773
4774
4775
4776
4777
4778
4779
4780
4781
4782
4783
4784
4785
4786
4787
4788
4789
4790
4791
4792
4793
4794
4795
4796
4797
4798
4799
4800
4801
4802
4803
4804
4805
4806
4807

*
* THE LINE UNIT IS INITIALIZED AND A MSG IS INITIATED (USING STEPLU) WITH CRC-
* 16 SELECTED IN CHAR MODE. TWO SYNCHS, 000,125,252,377,000, AND 2 TERMINATING
* SYNCHS ARE THEN SENT. THE PROGRAM CHECKS EACH BIT OF THE TRANSMITTED
* DATA CHARS, BY MONITORING TXDATA (REG 17) AS THE DATA IS CLOCKED OUT OF
* THE USYRT TRANSMITTER.
;*****

;*****
TEST 30 - USYRT RECEIVER MSG TEST - CHAR MODE, CRC

*
* THE LINE UNIT IS INITIALIZED AND A MESSAGE IS INITIATED (USING STEPLU) WITH
* LULP (REG 12) SET TO LOOP THE DATA INTERNALLY IN THE USYRT, AND WITH CRC-16
* SELECTED. TWO SYNCHS, 000,125,252,377,000, AND FOUR TERMINATING SYNCHS ARE
* SENT. THE PROGRAM MONITORS IACT, AND THE RCV'D CHARS AND CRC BYTES ARE READ
* FROM AX0-15 AND COMPARED TO EXPECTED VALUES. THE PROGRAM THEN CHECKS FOR IACT
* = 0 AFTER RECEIVER SHUTDOWN.
;*****

;*****
TEST 31 - USYRT RECEIVER MSG TEST - BIT MODE, CRC

*
* THE LINE UNIT IS INITIALIZED AND A MESSAGE IS INITIATED (USING STEPLU) WITH
* LULP (REG 12) SET TO LOOP THE DATA INTERNALLY IN THE USYRT, AND WITH CRC-
* CCITT-1. TWO FLAGS, 000,125,252,377,000, AND TWO TERMINATING FLAGS ARE THEN
* SENT. THE PROGRAM MONITORS IACT, RSOM, AND THE RCV'D CHARS ARE READ
* FROM AX0-15 AND COMPARED TO EXPECTED VALUES. THE PROGRAM THEN CHECKS FOR
* IACT = 0, SETS IC TO CLEAR THE RECEIVER, AND CHECKS FOR IACT STILL = 0.
;*****

;*****
TEST 32 - USYRT RECEIVER MSG TEST - CHAR MODE, NO CRC

*
* THE LINE UNIT IS INITIALIZED AND A MESSAGE IS INITIATED (USING STEPLU) WITH
* LULP (REG 12) SET TO LOOP THE DATA INTERNALLY IN THE USYRT, AND WITH NO
* ERROR DETECTION. TWO SYNCHS, 000,125,252,377,000, AND TWO SYNCHS ARE
* THEN SENT. THE PROGRAM MONITORS IACT, AND THE RECEIVED CHARS ARE READ FROM
* AX0-15 AND COMPARED TO EXPECTED VALUES. THE PROGRAM THEN CHECKS FOR IACT
* STILL = 0, SETS IC TO CLEAR THE RECEIVER, AND CHECKS FOR IACT = 0.
;*****

4808
4809
4810
4811
4812
4813
4814
4815
4816
4817
4818
4819
4820
4821
4822
4823
4824
4825
4826
4827
4828
4829
4830
4831
4832
4833
4834
4835
4836
4837
4838
4839
4840
4841
4842
4843
4844
4845
4846
4847
4848
4849
4850
4851
4852
4853
4854
4855
4856
4857
4858
4859
4860
4861
4862
4863

```
*****  
: TEST 33 - USYRT RECEIVER MSG TEST - BIT MODE, NO CRC  
*  
* THE LINE UNIT IS INITIALIZED AND A MESSAGE IS INITIATED (USING STEPLU) WITH  
* LULP (REG 12) SET TO LOOP THE DATA INTERNALLY IN THE USYRT, AND WITH ERROR  
* DETECTION INHIBITED. TWO FLAGS, 000,125,252,377,000, AND TWO TERMINATING FLAGS  
* ARE THEN SENT. THE PROGRAM MONITORS IACT, RSOM, AND THE RCV'D CHARS ARE  
* READ FROM AX0-15 AND COMPARED TO EXPECTED VALUES. THE PROGRAM THEN CHECKS FOR  
* IACT = 0, SETS IC TO CLEAR THE RECEIVER, AND CHECKS FOR IACT STILL = 0.  
:*****
```

```
*****  
: TEST 34 - SILO-DISABLED TRANSMITTER LOAD TEST  
*  
* THIS TEST DISABLES THE SILOS, LOADS A 125 CHARACTER INTO THE TX SILO, AND  
* READS AX1-15 AND CHECKS THAT THE DATA DID NOT GET LOADED INTO THE USYRT TX  
* BUFFER.  
:*****
```

```
*****  
: TEST 35 - SILO-DISABLED MESSAGE TEST - BIT MODE, NO CRC  
*  
* THE LINE UNIT IS INITIALIZED AND A MESSAGE IS INITIATED (USING STEPLU)  
* WITH LULP (REG 12) SET TO LOOP THE DATA INTERNALLY IN THE USYRT, WITH SILO  
* DISABLE SET, AND WITH NO ERROR DETECTION. TWO FLAGS, 000,125,252, AND  
* TERMINATING FLAGS ARE THEN SENT BY LOADING THE TRANSMITTED CHARS INTO  
* REG AX1. THE PROGRAM MONITORS OACT, IACT, RSOM, REOM, ORDY, OCOR, ICIR,  
* IRDY, AND THE RECEIVED CHARS ARE READ FROM AX0 AND COMPARED TO EXPECTED  
* VALUES.  
:*****
```

```
*****  
: TEST 36 - RECEIVER BUFFER TEST - CHAR MODE, CRC  
*  
* FIRST, A MASTER CLEAR IS DONE AND THE PROGRAM CHECKS FOR ICIR = 1 AND IRDY  
* = 0. THEN, 2 SOM CHARS ARE LOADED AND CLOCKED INTO THE USYRT, AND 64  
* BYTES OF A 256-BYTE BINARY COUNT DATA PATTERN (000-377) ARE LOADED INTO  
* THE TX SILO.  
* THE LINE UNIT IS THEN CLOCKED UNTIL IRDY = 1, AND THE PROGRAM CHECKS FOR  
* THIS TO OCCUR WITHIN 40-43 CYCLES. THE PROGRAM READS THE RCV SILO, CHECKS THE  
* CHAR FOR 000, AND CHECKS FOR IRDY = 0 AGAIN.  
* THE LINE UNIT IS THEN CLOCKED IN GROUPS OF 8 CYCLES, AND AFTER EACH, THE  
* PROGRAM CHECKS FOR ICIR = 1, IRDY = 1, UNTIL THE 64TH GROUP, AFTER WHICH
```

4864
4865
4866
4867
4868
4869
4870
4871
4872
4873
4874
4875
4876
4877
4878
4879
4880
4881
4882
4883
4884
4885
4886
4887
4888
4889
4890
4891
4892
4893
4894
4895
4896
4897
4898
4899
4900
4901
4902
4903
4904
4905
4906
4907
4908
4909
4910
4911
4912
4913
4914
4915
4916
4917
4918
4919

* IT CHECKS FOR ICIR = 0, IRDY = 1. THE SECOND DATA CHAR IS READ FROM THE
* RECEIVER SILO AND COMPARED TO 001. THEN, THE PROGRAM CHECKS FOR ICIR = 1,
* IRDY = 1 AGAIN.
* THE REST OF THE BINARY COUNT DATA BYTES ARE CYCLED 8 CLOCKS AND READ AND
* COMPARED A BYTE AT A TIME.
;*****

;*****
* TEST 37 - RECEIVER CHAR LENGTH TIMING TEST - CHAR MODE, NO CRC
*
* THE LINE UNIT IS PLACED IN CHAR MODE, WITH NO ERROR DETECTION, AND A MSG IS
* INITIATED WITH 2 SYNCH CHARS. NEXT, FIFTEEN 000 CHARS ARE LOADED INTO THE
* TRANSMITTER SILO. THE LINE UNIT IS THEN CLOCKED USING STEPLU WITH LULOOP
* SET, WHILE THE RECEIVER CHAR LENGTH IS SET TO THE FOLLOWING VALUES : 5,6,7,8.
* FOR EACH RCV CHAR LENGTH, THE PROGRAM CHECKS TO MAKE SURE THET USYRT RECEIVER
* FLAGS OCCUR THE PROPER NO. OF CYCLES APART, FOR EACH RCV CHAR LENGTH.
* (FOR EXAMPLE A 5-BIT CHAR TAKES 5 CLOCK CYCLES TO BE RECEIVED). A MASTER
* CLEAR IS THEN DONE TO TERMINATE THE OPERATION.
;*****

;*****
* TEST 38 - RECEIVER CHAR LENGTH TIMING TEST - BIT MODE, NO CRC
*
* THE LINE UNIT IS PLACED IN BIT MODE WITH NO ERROR DETECTION, AND A MESSAGE IS
* INITIATED WITH 2 FLAG CHARS. NEXT, FIFTEEN 000 CHARS ARE LOADED INTO THE
* TRANSMITTER SILO. THE LINE UNIT IS THEN CLOCKED USING STEPLU WITH LULOOP
* SET, WHILE THE RCV CHAR LENGTH IS SET TO THE FOLLOWING VALUES : 8,8,8,7,6,5,
* 4,3,2,1. FOR EACH RCV CHAR LENGTH, THE PROGRAM CHECKS TO MAKE SURE THAT THE
* USYRT RECEIVER FLAGS OCCUR THE PROPER NO. OF CYCLES APART, FOR EACH RCV
* CHAR LENGTH. (FOR EXAMPLE, A 5 BIT CHAR TAKES 5 CLOCK CYCLES TO BE RECEIVED).
* A MASTER CLEAR IS THEN DONE TO TERMINATE THE OPERATION.
;*****

;*****
* TEST 39 - TRANSMITTER UNDERRUN ERROR, IDLE MARKING, CHAR MODE, NO CRC
*
* THE LINE UNIT IS PLACED IN CHAR MODE, AND THE IDLE BIT IS SET. THEN, A
* MSG IS INITIATED, A 000 CHAR IS SENT, AND THE TX BUFFER IS NOT SERVICED
* IN RESPONSE TO THE USYRT TX BUFFER EMPTY FLAG, WHICH CAUSES A TX UNDERRUN
* ERROR. THEN, THE RECEIVER IS CLOCKED AND CHECKED FOR TWO 377 CHARS TO BE
* RECEIVED (LINE MARKING) BEFORE SHUTTING DOWN WITH A MASTER CLEAR.
;*****

4920
4921
4922
4923
4924
4925
4926
4927
4928
4929
4930
4931
4932
4933
4934
4935
4936
4937
4938
4939
4940
4941
4942
4943
4944
4945
4946
4947
4948
4949
4950
4951
4952
4953
4954
4955
4956
4957
4958
4959
4960
4961
4962
4963
4964
4965
4966
4967
4968
4969
4970
4971
4972
4973
4974
4975

```
*****  
; TEST 40 - MSG TERMINATION WITH GA CHARS - BIT MODE,NO CRC  
*  
* THE DEVICE IS ENABLED FOR TRANSMIT AND RECEIVE, AND A MESSAGE IS  
* INITIATED IN BIT MODE.  
* 2 FLAG CHARACTERS ARE SENT, FOLLOWED BY  
* THE FOLLOWING DATA CHARACTERS : 000, 125, 252, 377, 000. THEN THE LOOP  
* MODE BIT (STRIP) IS SET AND 2 TERMINATING GO-AHEAD CHARACTERS ARE  
* SENT. EACH USYRT RCV FLAG IS TIMED AS IT IS RECEIVED, AND THE 5 DATA  
* WORDS ARE READ AND COMPARED TO EXPECTED VALUES.  
* ALSO, THE FIRST GA CHAR IS CHECKED BY SCANNING THE TXDATA BIT AS THE GA  
* IS BEING TRANSMITTED (GA = 376 OCTAL).  
* THE TEST ALSO CHECKS FOR SETTING OF RAB AND EBLK  
* IN LOOP MODE.  
*****  
  
*****  
; TEST 41 - IDLE SYNCHS TEST - CHAR MODE  
*  
* THE DEVICE IS ENABLED FOR TRANSMIT AND RECEIVE, AND A MESSAGE IS  
* INITIATED IN CHAR MODE. 24(DEC) SYNCHS ARE SENT.  
* EACH SYNCH IS TIMED AS IT IS RECEIVED, AND THE BITS ARE CHECKED  
* FOR A VALID SYNCH CHAR FOR EACH OF THE 22 SYNCHS WHICH FOLLOW  
* THE FIRST TWO (THESE PERFORM SYNCHRONIZATION, AND ARE NOT READ).  
* WHILE THE LAST SYNCH IS BEING TRANSMITTED, OC IS SET, AND THE  
* NEXT CHAR RCV'D AFTER THE SYNCH IS CHECKED TO BE 377 (LINE MARKING).  
* THEN, A MASTER CLEAR IS ISSUED.  
* THE SYNCH CHAR USED IS 226 (OCTAL).  
*****  
  
*****  
; TEST 42 - STRIP SYNCH TEST  
*  
* THE DEVICE IS ENABLED FOR TRANSMIT AND RECEIVE, AND A MESSAGE IS  
* INITIATED IN CHAR MODE AND WITH THE STRIP SYNCH  
* BIT SET. THEN 24 (DEC) SYNCHS ARE SENT  
* FOLLOWED BY THE FOLLOWING DATA CHARACTERS : 377, 000, 125, 252,  
* AND THEN 2 TERMINATING SYNCHS.  
* EACH OF THE 23 SYNCHS AFTER THE FIRST ARE CHECKED AT THE TRANSMITTER OUTPUT,  
* BY SCANNING THE TXDATA BIT.  
* EACH USYRT RCV FLAG IS TIMED AS IT IS RECEIVED, AND THE 4 DATA WORDS  
* ARE READ AND COMPARED TO EXPECTED VALUES.  
* FINALLY, THE LINE UNIT IS CLOCKED FOR SEVERAL CHAR TIMES, AND A CHECK  
* IS MADE FOR OACT = 0 (TEOM SHOULD CAUSE TX ENABLE TO DROP).  
* THE ABOVE TEST IS REPEATED FOR EACH OF THE FOLLOWING SYNCH CHAR DATA
```

4976
4977
4978
4979
4980
4981
4982
4983
4984
4985
4986
4987
4988
4989
4990
4991
4992
4993
4994
4995
4996
4997
4998
4999
5000
5001
5002
5003
5004
5005
5006
5007
5008
5009
5010
5011
5012
5013
5014
5015
5016
5017
5018
5019
5020
5021
5022
5023
5024
5025
5026
5027
5028
5029
5030
5031

* PATTERNS : 226,000,125,252,376,177.
;*****

8.1 DATA PATTERNS USED

***** DATA PATTERN A *****

PATA:
.BYTE 125
.BYTE 252
.BYTE 000
.BYTE 377
.BYTE 001
.BYTE 002
.BYTE 004
.BYTE 010
.BYTE 020
.BYTE 040
.BYTE 100
.BYTE 200
.BYTE 376
.BYTE 375
.BYTE 373
.BYTE 367
.BYTE 357
.BYTE 337
.BYTE 277
.BYTE 177

***** DATA PATTERN B *****

PATB:
.BYTE 000
.BYTE 000
.BYTE 040
.BYTE 100
.BYTE 220
.BYTE 000
.BYTE 000
.BYTE 051

***** DATA PATTERN C *****

PATC:
.BYTE 020
.BYTE 020
.BYTE 020

***** DATA PATTERN D *****

PATD:
.BYTE 000
.BYTE 040
.BYTE 000

5032
5033
5034
5035
5036
5037
5038
5039
5040
5041
5042
5043
5044
5045
5046
5047
5048
5049
5050
5051
5052
5053
5054
5055
5056
5057
5058
5059
5060
5061
5062
5063
5064
5065
5066
5067
5068
5069
5070
5071
5072
5073
5074
5075
5076
5077
5078
5079
5080
5081
5082
5083
5084
5085
5086
5087

***** DATA PATTERN E *****

PATE:
.BYTE 000
.BYTE 120
.BYTE 020
.BYTE 100
.BYTE 120
.BYTE 000

***** DATA PATTERN F *****

PATF:
.BYTE 050
.BYTE 051
.BYTE 050

***** DATA PATTERN G *****

PATG:
.BYTE 000
.BYTE 000
.BYTE 240
.BYTE 120
.BYTE 177
.BYTE 000
.BYTE 000
.BYTE 001

***** DATA PATTERN H *****

PATH:
.BYTE 000
.BYTE 000
.BYTE 377
.BYTE 017
.BYTE 377
.BYTE 377
.BYTE 375
.BYTE 377

***** DATA PATTERN I *****

PATI:
.BYTE 000
.BYTE 000
.BYTE 000
.BYTE 000
.BYTE 000
.BYTE 103
.BYTE 000
.BYTE 000

***** DATA PATTERN J *****

PATJ:
.BYTE 000
.BYTE 000
.BYTE 010
.BYTE 002
.BYTE 004

5088 .BYTE 103
5089 .BYTE 001
5090 .BYTE 100

***** DATA PATTERN K *****

PATK: 5093 .BYTE 000
5094 .BYTE 000
5095 .BYTE 377
5096 .BYTE 377
5097 .BYTE 125
5098 .BYTE 125
5099 .BYTE 252
5100 .BYTE 252
5101 .BYTE 001
5102 .BYTE 000
5103 .BYTE 002
5104 .BYTE 000
5105 .BYTE 004
5106 .BYTE 000
5107 .BYTE 010
5108 .BYTE 000
5109 .BYTE 020
5110 .BYTE 000
5111 .BYTE 040
5112 .BYTE 000
5113 .BYTE 100
5114 .BYTE 000
5115 .BYTE 200
5116 .BYTE 000
5117 .BYTE 000
5118 .BYTE 001
5119 .BYTE 000
5120 .BYTE 002
5121 .BYTE 000
5122 .BYTE 004
5123 .BYTE 000
5124 .BYTE 010
5125 .BYTE 000
5126 .BYTE 020
5127 .BYTE 000
5128 .BYTE 040
5129 .BYTE 000
5130 .BYTE 100
5131 .BYTE 000
5132 .BYTE 200
5133 .BYTE 376
5134 .BYTE 377
5135 .BYTE 375
5136 .BYTE 377
5137 .BYTE 373
5138 .BYTE 377
5139 .BYTE 367
5140 .BYTE 377
5141 .BYTE 357
5142 .BYTE 377
5143 .BYTE 337

| | | |
|------|-------|-----|
| 5144 | .BYTE | 377 |
| 5145 | .BYTE | 277 |
| 5146 | .BYTE | 377 |
| 5147 | .BYTE | 177 |
| 5148 | .BYTE | 377 |
| 5149 | .BYTE | 377 |
| 5150 | .BYTE | 376 |
| 5151 | .BYTE | 377 |
| 5152 | .BYTE | 375 |
| 5153 | .BYTE | 377 |
| 5154 | .BYTE | 373 |
| 5155 | .BYTE | 377 |
| 5156 | .BYTE | 367 |
| 5157 | .BYTE | 377 |
| 5158 | .BYTE | 357 |
| 5159 | .BYTE | 377 |
| 5160 | .BYTE | 337 |
| 5161 | .BYTE | 377 |
| 5162 | .BYTE | 277 |
| 5163 | .BYTE | 377 |
| 5164 | .BYTE | 177 |

***** DATA PATTERN L *****

PATL:

| | |
|-------|-----|
| .BYTE | 000 |
| .BYTE | 000 |
| .BYTE | 377 |
| .BYTE | 377 |
| .BYTE | 000 |
| .BYTE | 000 |

***** DATA PATTERN M *****

PATM:

| | |
|-------|-----|
| .BYTE | 000 |
| .BYTE | 020 |
| .BYTE | 000 |
| .BYTE | 000 |
| .BYTE | 200 |
| .BYTE | 000 |
| .BYTE | 000 |
| .BYTE | 051 |

***** DATA PATTERN N *****

PATN:

| | |
|-------|-----|
| .BYTE | 000 |
| .BYTE | 000 |
| .BYTE | 000 |
| .BYTE | 125 |
| .BYTE | 000 |
| .BYTE | 252 |
| .BYTE | 000 |
| .BYTE | 377 |
| .BYTE | 005 |
| .BYTE | 000 |
| .BYTE | 012 |
| .BYTE | 000 |

5199

5200 .BYTE 017
5201 .BYTE 000

***** DATA PATTERN O *****
PATO:

5202 .BYTE 000
5203 .BYTE 041
5204 .BYTE 004
5205 .BYTE 010
5206 .BYTE 020
5207 .BYTE 040
5208 .BYTE 100
5209 .BYTE 101
5210 .BYTE 200
5211 .BYTE 201
5212 .BYTE 300
5213 .BYTE 111
5214 .BYTE 301
5215 .BYTE 375
5216
5217
5218
5219

***** DATA PATTERN P *****
PATP:

5220 .BYTE 000
5221 .BYTE 113
5222 .BYTE 200
5223 .BYTE 040
5224 .BYTE 020
5225 .BYTE 010
5226 .BYTE 001
5227 .BYTE 104
5228 .BYTE C07
5229 .BYTE 105
5230 .BYTE 007
5231 .BYTE 144
5232 .BYTE 107
5233 .BYTE 157
5234
5235
5236

***** DATA PATTERN U *****
PATU:

5237 .BYTE 000
5238 .BYTE 000
5239 .BYTE 001
5240 .BYTE 000
5241 .BYTE 013
5242 .BYTE 000
5243 .BYTE 011
5244 .BYTE 000
5245 .BYTE 021
5246 .BYTE 000
5247 .BYTE 101
5248 .BYTE 000
5249 .BYTE 301
5250 .BYTE 000
5251 .BYTE 000
5252 .BYTE 001
5253 .BYTE 000
5254 .BYTE 000
5255 .BYTE 002

| | | |
|------|-------|-----|
| 5256 | .BYTE | 000 |
| 5257 | .BYTE | 004 |
| 5258 | .BYTE | 000 |
| 5259 | .BYTE | 040 |
| 5260 | .BYTE | 000 |
| 5261 | .BYTE | 100 |
| 5262 | .BYTE | 000 |
| 5263 | .BYTE | 200 |
| 5264 | .BYTE | 000 |
| 5265 | .BYTE | 346 |
| 5266 | .BYTE | 000 |
| 5267 | .BYTE | 345 |
| 5268 | .BYTE | 000 |
| 5269 | .BYTE | 343 |
| 5270 | .BYTE | 000 |
| 5271 | .BYTE | 307 |
| 5272 | .BYTE | 000 |
| 5273 | .BYTE | 247 |
| 5274 | .BYTE | 000 |
| 5275 | .BYTE | 147 |

***** DATA PATTERN V *****

| | | |
|------|-------------|-----|
| 5276 | | |
| 5277 | | |
| 5278 | PATV: .BYTE | 000 |
| 5279 | .BYTE | 000 |
| 5280 | .BYTE | 333 |
| 5281 | .BYTE | 000 |
| 5282 | .BYTE | 331 |
| 5283 | .BYTE | 000 |
| 5284 | .BYTE | 323 |
| 5285 | .BYTE | 000 |
| 5286 | .BYTE | 313 |
| 5287 | .BYTE | 000 |
| 5288 | .BYTE | 233 |
| 5289 | .BYTE | 000 |
| 5290 | .BYTE | 133 |
| 5291 | .BYTE | 000 |
| 5292 | .BYTE | 000 |
| 5293 | .BYTE | 001 |
| 5294 | .BYTE | 000 |
| 5295 | .BYTE | 002 |
| 5296 | .BYTE | 000 |
| 5297 | .BYTE | 004 |
| 5298 | .BYTE | 000 |
| 5299 | .BYTE | 040 |
| 5300 | .BYTE | 000 |
| 5301 | .BYTE | 100 |
| 5302 | .BYTE | 000 |
| 5303 | .BYTE | 200 |
| 5304 | .BYTE | 000 |
| 5305 | .BYTE | 346 |
| 5306 | .BYTE | 000 |
| 5307 | .BYTE | 345 |
| 5308 | .BYTE | 000 |
| 5309 | .BYTE | 343 |
| 5310 | .BYTE | 000 |
| 5311 | .BYTE | 307 |

| | | |
|------|-------|-----|
| 5312 | .BYTE | 000 |
| 5313 | .BYTE | 247 |
| 5314 | .BYTE | 000 |
| 5315 | .BYTE | 147 |
| 5316 | | |
| 5317 | | |
| 5318 | | |
| 5319 | | |
| 5320 | | |
| 5321 | | |

5323
5324
5325
5326
5327
5328
5329
5330
5331
5332
5333
5334
5335
5336
5337
5338
5339
5340
5341
5342
5343
5344
5345
5346
5347
5348
5349
5350
5351
5352
5353
5354
5355
5356
5357
5358
5359
5360
5361
5362
5363
5364
5365
5366
5367
5368
5369
5370
5371
5372
5373
5374
5375
5376
5377
5378

9.0 ERROR INFORMATION

9.1 ERROR REPORTING

ERRORS ARE REPORTED BY THE PROGRAM AS THEY OCCUR (IF NOT INHIBITED). THE REPORT CONFORMS TO THE DIAGNOSTIC SUPERVISOR ERROR REPORT FORMAT, AND CONSISTS OF A DESCRIPTION OF THE ERROR, THE TEST NUMBER, SUBTEST NUMBER, PC OF THE ERROR CALL, DEVICE ADDRESS, AND BASIC AND EXTENDED ERROR INFORMATION.

THE FOLLOWING EXAMPLE PROVIDES A TYPICAL ERROR REPORT, WHICH DESCRIBES AN "IRDY NOT SET" ERROR, AND PROVIDES THE PC OF THE ERROR CALL AND THE PC OF THE CALL TO THE SUBROUTINE REPORTING IT, THE FAILING REGISTER NAME, AND DEVICE REGISTER CONTENTS :

CZDMR DVC FTL ERR 00017 ON UNIT 00 TST 034 SUB 000 PC: 006210
IRDY NOT SET
PC OF SUBR CALL: 030044
DEVICE CSR ADDRESS : 160170

FAILING REG: INBUS/OUTBUS REG 12

LINE UNIT INBUS REGS:
REG10 REG11 REG12 REG13
000 120 000 257
REG14 REG15 REG16 REG17
024 377 377 035

LINE UNIT EXTENDED REGS:
AX0-15 AX0-16 AX1-15 AX1-16
000 000 000 000
AX2-15 AX2-16 AX3-15 AX3-16
000 000 000 000

FOR OTHER ERRORS, THE REPORT MAY BE MORE EXTENSIVE, AND REQUIRE ADDITIONAL DATA TO BE REPORTED.

IF EXTENDED ERROR INFORMATION HAD BEEN INHIBITED USING THE IXE FLAG PRIOR TO RUNNING THE TEST, THE ABOVE ERROR WOULD HAVE BEEN REPORTED IN THE FOLLOWING SHORTENED FORM :

CZDMR DVC FTL ERR 00017 ON UNIT 00 TST 034 SUB 000 PC:006210
IRDY NOT SET
PC OF SUBR CALL: 030044
DEVICE CSR ADDRESS : 160170

FAILING REG: INBUS/OUTBUS REG 12

5379
5380
5381
5382
5383
5384
5385
5386
5387

2

```
5389          .TITLE CZDMRA M8203 STATIC DIAG #1
5398          002000          .-2000
5399
5400
5401
5402
5403
5404
5405
5406 002000          .MCALL SVC          ; INITIALIZE SUPERVISOR MACROS
5407          SVC
5408
5409
5410
5411
5412 002000          BGNMOD LU1MOD
5413
5414
5415          000001          $LSTIN= 1
5416          000001          $LSTTAG= 1
5417          000001          SVCINS= 1          ; LIST INSTRUCTIONS, SHIFTED RIGHT
5418          000001          SVCTST= 1          ; LIST TEST TAGS, SHIFTED RIGHT
5419          000001          SVCSUB= 1          ; LIST SUBTEST TAGS, SHIFTED RIGHT
5420          000001          SVCGBL= 1          ; LIST GLOBAL TAGS, SHIFTED RIGHT
5421          000001          SVCTAG= 1          ; LIST OTHER TAGS, SHIFTED RIGHT
5422
5423          ; CHANGE THE VALUES OF THE SVC... SYMBOLS TO BE ZERO IF YOU WISH
5424          ; TO ALIGN THE MACRO CALLS AND THEIR EXPANSIONS. CHANGE THE
5425          ; SYMBOLS TO BE MINUS-ONE TO NOT LIST THE EXPANSIONS. YOU MAY
5426          ; CHANGE THE SYMBOLS AT ANY POINT IN YOUR PROGRAM.
5427
5428
```

5430
5431
5432
5433
5434
5435
5436 002000
5437
5439
5440
5441
5442
5443
5445
5446 002000
(4) 002000
(4) 002000 103
(4) 002001 132
(4) 002002 104
(4) 002003 115
(4) 002004 122
(6) 002005 000
(6) 002006 000
(5) 002007 000
(5) 002010
(4) 002010 101
(5) 002011
(4) 002011 060
(5) 002012
(4) 002012 000000
(5) 002014
(4) 002014 000055
(5) 002016
(4) 002016 035514
(5) 002020
(4) 002020 000000
(5) 002022
(4) 002022 002252
(5) 002024
(4) 002024 000000
(5) 002026
(4) 002026 036172
(5) 002030
(4) 002030 000000
(5) 002032
(4) 002032 000000
(5) 002034
(4) 002034 000000
(5) 002036
(4) 002036 000000
(5) 002040
(4) 002040 002124
(5) 002042
(4) 002042 000000
(5) 002044
(4) 002044 000000

.SBTTL PROGRAM HEADER
:++
: THE PROGRAM HEADER IS THE INTERFACE BETWEEN
: THE DIAGNOSTIC PROGRAM AND THE SUPERVISOR.
:--

POINTER BGNAU,BGNDU

:XX
: IF ANY OPTIONAL POINTERS ARE TO BE USED IN THE 'HEADER', CHANGE
: 'POINTER' TO CONTAIN THE CORRECT ARGUMENTS. IF ALL OPTIONAL
: POINTERS ARE TO BE USED, CHANGE 'POINTER' TO BE 'POINTER ALL'.
:XX

HFADER CZDMR,A,0,45.,0

LSNAME::
.ASCII /C/
.ASCII /Z/
.ASCII /D/
.ASCII /M/
.ASCII /R/
.BYTE 0
.BYTE 0
.BYTE 0
LSREV::
.ASCII /A/
LSDEPO::
.ASCII /O/
LSUNIT::
.WORD 0
LSTIML::
.WORD 45.
LSHPCP::
.WORD LSHARD
LSSPCP::
.WORD 0
LSHPTP::
.WORD LSHW
LSSPTP::
.WORD 0
LSLADP::
.WORD L\$LAST
L\$STA::
.WORD 0
L\$CO::
.WORD 0
LSDTYP::
.WORD 0
L\$APT::
.WORD 0
LSDTP::
.WORD 0
L\$EXP1::
.WORD L\$DISPATCH
L\$EXP2::
.WORD 0
.WORD 0

5462
5463
5464
5465
5466
5467
5468
5469 002122
(4) 002122 000052
(3) 002124
(6) 002124 021610
(6) 002126 021672
(6) 002130 021756
(6) 002132 022102
(6) 002134 022204
(6) 002136 022324
(6) 002140 022444
(6) 002142 022620
(6) 002144 023054
(6) 002146 023172
(6) 002150 023310
(6) 002152 023426
(6) 002154 023544
(6) 002156 024174
(6) 002160 024526
(6) 002162 025016
(6) 002164 025226
(6) 002166 025432
(6) 002170 025636
(6) 002172 026066
(6) 002174 026320
(6) 002176 026572
(6) 002200 027250
(6) 002202 027370
(6) 002204 027500
(6) 002206 027674
(6) 002210 030240
(6) 002212 030450
(6) 002214 031000
(6) 002216 031100
(6) 002220 031450
(6) 002222 032012
(6) 002224 032262
(6) 002226 032624
(6) 002230 032754
(6) 002232 033666
(6) 002234 034150
(6) 002236 034350
(6) 002240 034650
(6) 002242 034740
(6) 002244 035102
(6) 002246 035250

.SBTTL DISPATCH TABLE

:/ THE DISPATCH TABLE CONTAINS THE STARTING ADDRESS OF EACH TEST.
:/ IT IS USED BY THE SUPERVISOR TO DISPATCH TO EACH TEST.

DISPATCH 42

.WORD 42
LSDISPATCH: :
.WORD T1
.WORD T2
.WORD T3
.WORD T4
.WORD T5
.WORD T6
.WORD T7
.WORD T8
.WORD T9
.WORD T10
.WORD T11
.WORD T12
.WORD T13
.WORD T14
.WORD T15
.WORD T16
.WORD T17
.WORD T18
.WORD T19
.WORD T20
.WORD T21
.WORD T22
.WORD T23
.WORD T24
.WORD T25
.WORD T26
.WORD T27
.WORD T28
.WORD T29
.WORD T30
.WORD T31
.WORD T32
.WORD T33
.WORD T34
.WORD T35
.WORD T36
.WORD T37
.WORD T38
.WORD T39
.WORD T40
.WORD T41
.WORD T42

5470
5472
5473
5474

:XX
: CHANGE THE ARGUMENT OF 'DISPATCH' TO BE THE
: NUMBER OF HARDWARE TESTS IN YOUR PROGRAM.

5483
5484
5485
5486
5487
5488
5489
5490
5491
(3)
(3)
(3)
5492
5493
5494
5495
5496
5497
5498
5499
5500
5501
5502
5503
5504
5505
(3)
5506
5507
5508
5509
5510

.SBTTL DEFAULT HARDWARE P-TABLE

:/ THE DEFAULT HARDWARE P-TABLE CONTAINS DEFAULT VALUES OF
:/ THE TEST-DEVICE PARAMETERS. THE STRUCTURE OF THIS TABLE
:/ IS IDENTICAL TO THE STRUCTURE OF THE RUN-TIME P-TABLE.
:////////////////////

BGNHW DFPTBL

.WORD L10000-L\$HW/2

L\$HW::
DFPTBL::

.WORD 7
.WORD 160170
.WORD 300
.WORD 5000
.WORD 3
.WORD 056
.WORD 000
.WORD 000
.WORD 0
.WORD 4
.WORD 1

:MICROPROCESSOR TYPE = M8207
:DMC11 OR KMC11 CSR UNIBUS ADDRESS
:DMC11 OR KMC11 INTERRUPT VECTOR
:DMC11 OR KMC11 INTERRUPT PRIORITY LEVEL = 5
:LINE UNIT = M8203
:SWITCH PACK #1 (REG 11)
:SWITCH PACK #2 (REG 15)
:SWITCH PACK #3 (REG 16)
:H3254&H3255 USED
:BAUD RATE = 56 K
:RUN SWITCH ON MICROPROCESSOR IS ON

ENDHW

L10000:

5512
5513
5514
5515
5516
5517
5518
5519
5520
5521
5522
5523
5524
5525
5526
5527
5528

002300
(3) 002300 000000
(3) 002302
(3) 002302

```
.SBTTL SOFTWARE P-TABLE  
:////////////////////////////////////  
:// THE SOFTWARE P-TABLE CONTAINS THE VALUES OF THE PROGRAM  
:// PARAMETERS THAT CAN BE CHANGED BY THE OPERATOR.  
:////////////////////////////////////
```

BGNSW SFPTBL

.WORD L10001-L\$SW/2
L\$SW::
SFPTBL::

ENDSW

L10001:

5530
5531
5532
5533
5534
5535
5536
5537
5538
5539
5540
5541
5542
5543
5544
5545
5546
5547
5548

002302

.SBTTL GLOBAL EQUATES SECTION

:/ THE GLOBAL EQUATES SECTION CONTAINS PROGRAM EQUATES THAT
:/ ARE USED IN MORE THAN ONE TEST.
:////

EQUALS

: BIT DIFINITIONS

(1) 100000
(1) 040000
(1) 020000
(1) 010000
(1) 004000
(1) 002000
(1) 001000
(1) 000400
(1) 000200
(1) 000100
(1) 000040
(1) 000020
(1) 000010
(1) 000004
(1) 000002
(1) 000001

BIT15== 100000
BIT14== 40000
BIT13== 20000
BIT12== 10000
BIT11== 4000
BIT10== 2000
BIT09== 1000
BIT08== 400
BIT07== 200
BIT06== 100
BIT05== 40
BIT04== 20
BIT03== 10
BIT02== 4
BIT01== 2
BIT00== 1

(1) 001000
(1) 000400
(1) 000200
(1) 000100
(1) 000040
(1) 000020
(1) 000010
(1) 000004
(1) 000002
(1) 000001

BIT9== BIT09
BIT8== BIT08
BIT7== BIT07
BIT6== BIT06
BIT5== BIT05
BIT4== BIT04
BIT3== BIT03
BIT2== BIT02
BIT1== BIT01
BIT0== BIT00

: EVENT FLAG DEFINITIONS
: EF32:EF17 RESERVED FOR SUPERVISOR TO PROGRAM COMMUNICATION

(1) 000040 EF.START== 32. ; START COMMAND WAS ISSUED
(1) 000037 EF.RESTART== 31. ; RESTART COMMAND WAS ISSUED

: CONTINUE COMMAND WAS ISSUED
: A NEW PASS HAS BEEN STARTED
: A POWER-FAIL/POWER-UP OCCURRED

(1) 000036 EF.CONTINUE== 30.
(1) 000035 EF.NEW== 29.
(1) 000034 EF.PWR== 28.
(1) :
(1) :
(1) : PRIORITY LEVEL DEFINITIONS
(1) :
(1) 000340 PRI07== 340
(1) 000300 PRI06== 300
(1) 000240 PRI05== 240
(1) 000200 PRI04== 200
(1) 000140 PRI03== 140
(1) 000100 PRI02== 100
(1) 000040 PRI01== 40
(1) 000000 PRI00== 0

(1) :
(1) : OPERATOR FLAG BITS
(1) :
(1) 000004 EVL== 4
(1) 000010 LOT== 10
(1) 000020 ADR== 20
(1) 000040 IDU = 40
(1) 000100 ISR== 100
(1) 000200 UAM== 200
(1) 000400 BOE== 400
(1) 001000 PNT== 1000
(1) 002000 PRI== 2000
(1) 004000 IXE== 4000
(1) 010000 IBE== 10000
(1) 020000 IER== 20000
(1) 040000 LOE== 40000
(1) 100000 HOE== 100000

5550
5551
5552
5553
5554
5555
5556
5557
5558
5559
5560
5561
5562
5563
5564
5565
5566
5567
5568
5569
5570
5571
5572
5573

::*****
:* PROGRAM EVENT FLAG DEFINITIONS
:*****

::*****
:* MAINTENANCE REGISTER - BSEL1
:*****

RUN = BIT7
MCLR = BIT6
STEPLU = BIT4
LULOOP = BIT3
ROMO = BIT2
ROMI = BIT1
STEPMP = BIT0

::*****

```
5574      ;* OBUS REG 10 - TRANSMITTER BUFFER
5575      ;*****
5576      000200 TX7      = BIT7
5577      000100 TX6      = BIT6
5578      000040 TX5      = BIT5
5579      000020 TX4      = BIT4
5580      000010 TX3      = BIT3
5581      000004 TX2      = BIT2
5582      000002 TX1      = BIT1
5583      000001 TX0      = BIT0
5584
5585      ;*****
5586      ;* OBUS REG 11
5587      ;*****
5588      000200 OC       = BIT7
5589      000010 GOAH    = BIT3
5590      000004 ABORT   = BIT2
5591      000002 EOM     = BIT1
5592      000001 SOM     = BIT0
5593
5594      ;*****
5595      ;* OBUS REG 12
5596      ;*****
5597      000200 IC       = BIT7
5598      000100 BPOLL   = BIT6
5599      000040 LULP    = BIT5
5600
5601      ;*****
5602      ;* OBUS REG 13
5603      ;*****
5604      000200 POLL    = BIT7
5605      000100 DTR     = BIT6
5606      000040 SELFR   = BIT5
5607      000020 HDX     = BIT4
5608      000010 MAINT1  = BIT3
5609      000004 MAINT2  = BIT2
5610      000002 SELSBY  = BIT1
5611
5612      ;*****
5613      ;* OBUS REG 14
5614      ;*****
5615      000100 TXEN    = BIT6
5616      000040 DISSI   = BIT5
5617      000020 RDAX    = BIT4
5618      000010 WAX     = BIT3
5619      000004 ENAX    = BIT2
5620      000002 AX2     = BIT1
5621      000001 AX1     = BIT0
5622
5623      ;*****
5624      ;* OBUS REG 17
5625      ;*****
5626      000200 CRC2    = BIT7
5627      000100 CRC1    = BIT6
5628      000040 IDLE   = BIT5
5629      000020 SECA    = BIT4
```

5630 000010 STRIP = BIT3
5631 000004 RDALL = BIT2
5632 000002 IERR = BIT1
5633 000001 DDCMP = BIT0

5634
5635 :*****
5636 :* IBUS REG 10 - RECEIVER BUFFER
5637 :*****
5638 000200 RX7 = BIT7
5639 000100 RX6 = BIT6
5640 000040 RX5 = BIT5
5641 000020 RX4 = BIT4
5642 000010 RX3 = BIT3
5643 000004 RX2 = BIT2
5644 000002 RX1 = BIT1
5645 000001 RX0 = BIT0

5646
5647 :*****
5648 :* IBUS REG 11
5649 :*****
5650 000200 OC = BIT7
5651 000100 OACT = BIT6
5652 000040 SW3 = BIT5
5653 000020 ORDY = BIT4
5654 000010 SW2 = BIT3
5655 000004 SW1 = BIT2
5656 000002 SW0 = BIT1
5657 000001 UNRR = BIT0

5658
5659 :*****
5660 :* IBUS REG 12
5661 :*****
5662 000200 IC = BIT7
5663 000100 IACT = BIT6
5664 000040 LULP = BIT5
5665 000020 IRDY = BIT4
5666 000010 OVRP = BIT3
5667 000004 RAB = BIT2
5668 000002 EBLK = BIT1
5669 000001 BCC = BIT0

5670
5671 :*****
5672 :* IBUS REG 13
5673 :*****
5674 000200 RING = BIT7
5675 000100 DTR = BIT6
5676 000040 RTS = BIT5
5677 000020 HDX = BIT4
5678 000010 MODR = BIT3
5679 000004 CS = BIT2
5680 000002 STBY = BIT1
5681 000001 CARR = BIT0

5682
5683 :*****
5684 :* IBUS REG 14
5685 :*****

| | | | | |
|------|--------|-------|---|------|
| 5686 | 000200 | READY | = | BIT7 |
| 5687 | 000100 | TXEN | = | BIT6 |
| 5688 | 000040 | DISSI | = | BIT5 |
| 5689 | 000020 | RDAX | = | BIT4 |
| 5690 | 000010 | WAX | = | BIT3 |
| 5691 | 000004 | ENAX | = | BIT2 |
| 5692 | 000002 | AX2 | = | BIT1 |
| 5693 | 000001 | AX1 | = | BIT0 |

;* IBUS REG 17

| | | | | |
|------|--------|--------|---|------|
| 5698 | 000200 | SIGR | = | BIT7 |
| 5699 | 000100 | SIGQ | = | BIT6 |
| 5700 | 000040 | TXDATA | = | BIT5 |
| 5701 | 000020 | OCOR | = | BIT4 |
| 5702 | 000010 | ICIR | = | BIT3 |
| 5703 | 000004 | TESTMD | = | BIT2 |
| 5704 | 000002 | MCLK | = | BIT1 |
| 5705 | 000001 | DDCMP | = | BIT0 |

;* AX0-15 - USYRT REG 0 (READ ONLY)

| | | | | |
|------|--------|-----|---|------|
| 5710 | 000200 | RX7 | = | BIT7 |
| 5711 | 000100 | RX6 | = | BIT6 |
| 5712 | 000040 | RX5 | = | BIT5 |
| 5713 | 000020 | RX4 | = | BIT4 |
| 5714 | 000010 | RX3 | = | BIT3 |
| 5715 | 000004 | RX2 | = | BIT2 |
| 5716 | 000002 | RX1 | = | BIT1 |
| 5717 | 000001 | RX0 | = | BIT0 |

;* AX0-16 - USYRT REG 1 (READ ONLY)

| | | | | |
|------|--------|-------|---|------|
| 5722 | 000200 | RERR | = | BIT7 |
| 5723 | 000100 | ASBC2 | = | BIT6 |
| 5724 | 000040 | ASBC1 | = | BIT5 |
| 5725 | 000020 | ASBC0 | = | BIT4 |
| 5726 | 000010 | ROR | = | BIT3 |
| 5727 | 000004 | RABT | = | BIT2 |
| 5728 | 000002 | REOM | = | BIT1 |
| 5729 | 000001 | RSOM | = | BIT0 |

;* AX1-15 - USYRT REG 2

| | | | | |
|------|--------|-----|---|------|
| 5734 | 000200 | TX7 | = | BIT7 |
| 5735 | 000100 | TX6 | = | BIT6 |
| 5736 | 000040 | TX5 | = | BIT5 |
| 5737 | 000020 | TX4 | = | BIT4 |
| 5738 | 000010 | TX3 | = | BIT3 |
| 5739 | 000004 | TX2 | = | BIT2 |
| 5740 | 000002 | TX1 | = | BIT1 |
| 5741 | 000001 | TX0 | = | BIT0 |


```
5742  
5743  
5744  
5745  
5746      000200  
5747      000010  
5748      000004  
5749      000002  
5750      000001  
5751  
5752  
5753  
5754  
5755      000200  
5756      000100  
5757      000040  
5758      000020  
5759      000010  
5760      000004  
5761      000002  
5762      000001  
5763      000226  
5764  
5765  
5766  
5767  
5768      000200  
5769      000100  
5770      000040  
5771      000020  
5772      000010  
5773      000004  
5774      000002  
5775      000001  
5776  
5777  
5778  
5779  
5780      000200  
5781      000100  
5782      000020  
5783      000010  
5784      000002  
5785      000001  
5786      000332  
5787  
5788  
5789  
5790  
5791      000200  
5792      000100  
5793      000040  
5794      000004  
5795      000002  
5796      000001  
5797
```

```
*****  
* AX1-16 - USYRT REG 3  
*****  
TERR      = BIT7  
TXGA      = BIT3  
TXAB      = BIT2  
TEOM      = BIT1  
TSOM      = BIT0  
  
*****  
* AX2-15 - USYRT REG 4  
*****  
SYN7      = BIT7  
SYN6      = BIT6  
SYN5      = BIT5  
SYN4      = BIT4  
SYN3      = BIT3  
SYN2      = BIT2  
SYN1      = BIT1  
SYN0      = BIT0  
SYNCH     = 226  
  
*****  
* AX2-16 - USYRT REG 5  
*****  
APA       = BIT7  
DDC       = BIT6  
STR       = BIT5  
SEC       = BIT4  
IDL       = BIT3  
CRCTY2    = BIT2  
CRCTY1    = BIT1  
CRCTY0    = BIT0  
  
*****  
* AX3-15 - USYRT REG 6  
*****  
I422     = BIT7  
XYZ      = BIT6  
V35     = BIT4  
INTGRL   = BIT3  
OP       = BIT1  
TEST     = BIT0  
AX315U   = I422.XYZ!V35!INTGRL.OP  
  
*****  
* AX3-16 - USYRT REG 7  
*****  
TXLEN2    = BIT7  
TXLEN1    = BIT6  
TXLENO    = BIT5  
RXLEN2    = BIT2  
RXLEN1    = BIT1  
RXLENO    = BIT0
```

5798
5799
5800
5801
5802
5803
5804
5805 004000
5806 002000
5807 001000
5808 000400
5809
5810
5811
5812
5813
5814
5815
5816 004000
5817 002000
5818 001000
5819 000400
5820
5821
5822
5823
5824
5825
5826
5827
5828 002302
5829 002304
5830 002306
5831 002310
5832 002312
5833 002314
5834 002316
5835 002320
5836 002322
5837 002324
5838 002326
5839 002330
5840 002332
5841 002334
5842 002336
5843 002340
5844
5845
5846
5847
5848
5849 100000
5850
5851 100000
5852 100000
5853

```
*****  
;* TX CONTROL BITS DEFINED ON WORD BASIS  
*****  
TXGOA = BIT11  
TXABT = BIT10  
TXEOM = BIT9  
TXSOM = BIT8
```

```
*****  
;* RCV CONTROL BITS DEFINED ON WORD BASIS  
*****  
RXOVR = BIT11  
RXABT = BIT10  
RXEBL = BIT9  
RXBCC = BIT8
```

```
*****  
;* ADDRESS EQUATES FOR REGISTER STORAGE TABLE (LUREG:)  
*****  
LUR10 = LUREG+0 ;LINE UNIT IBUS REG 10  
LUR11 = LUREG+2 ;LINE UNIT IBUS REG 11  
LUR12 = LUREG+4 ;LINE UNIT IBUS REG 12  
LUR13 = LUREG+6 ;LINE UNIT IBUS REG 13  
LUR14 = LUREG+10 ;LINE UNIT IBUS REG 14  
LUR15 = LUREG+12 ;LINE UNIT IBUS REG 15  
LUR16 = LUREG+14 ;LINE UNIT IBUS REG 16  
LUR17 = LUREG+16 ;LINE UNIT IBUS REG 17  
AX0.15 = LUREG+20 ;USYRT REG 0  
AX0.16 = LUREG+22 ;USYRT REG 1  
AX1.15 = LUREG+24 ;USYRT REG 2  
AX1.16 = LUREG+26 ;USYRT REG 3  
AX2.15 = LUREG+30 ;USYRT REG 4  
AX2.16 = LUREG+32 ;USYRT REG 5  
AX3.15 = LUREG+34 ;USYRT REG 6  
AX3.16 = LUREG+36 ;USYRT REG 7
```

```
CHPCHK = BIT15  
BCCCHK = BIT15  
CRCCHK = BIT15
```

5854
5855
5856
5857
5858
5859
5860
5861
5862
5863
5864
5865
5866
5867
5868
5869
5870
5871
5872
5873
5874

021000
122000
121000

```
*****  
;* MICROINSTRUCTION DEFINITIONS  
*****  
MVI0X  = 021000      ;MOVE IBUS TO OBUS*  
MVI0X  = 122000      ;MOVE IBUS* TO OBUS  
MVI0X  = 121000      ;MOVE IBUS* TO OBUS*
```

```
***** ERROR1 BIT FLAG DEFINITIONS *****  
RRDYTO = BIT0  
WRDYTO = BIT1
```

```
5876 .SBTTL GLOBAL DATA SECTION
5877
5878 :////////////////////////////////////////////////////////////////////
5879 :/ THE GLOBAL DATA SECTION CONTAINS DATA THAT ARE USED
5880 :/ IN MORE THAN ONE TEST.
5881 :////////////////////////////////////////////////////////////////////
5882
5883 :*****
5884 :* STORAGE FOR DEVICE REGISTERS
5885 :*****
5886 002302 000020 LUREG: .BLKW 16.
5887
5888 :*****
5889 :* MISCELLANEOUS STORAGE
5890 :*****
5891 002342 000000 SCRACH: .WORD 0 ;GEN'L PURPOSE SCRATCH WORD
5892 002344 000000 LOGDEV: .WORD 0 ;LOGICAL DEVICE NUMBER
5893 002346 000000 PSTACK: .WORD 0 ;CONTAINS BASE LEVEL PROGRAM STACK POINTER
5894 002350 000000 PRIOR: .WORD 0 ;CPU PRIORITY FOR PRINTOUT
5895 002352 000000 SUBRPC: .WORD 0 ;PC OF SUBR CALL FOR ERROR REPORTS
5896 002354 000000 INTFLG: .WORD 0 ;INTERRUPT RECEIVED FLAGS
5897 ; BIT 0 FOR TX, BIT 1 FOR RCV
5898 002356 000000 ERRFLG: .WORD 0 ;SUBROUTINE ERROR FLAG
5899 002360 000000 TIMFLG: .WORD 0 ;EVENT TIME-OUT FLAG
5900 002362 000000 RETADR: .WORD 0 ;SUBR ERROR RETURN ADDRESS
5901 002364 000000 REDBYT: .WORD 0 ;LO BYTE CONTAINS BYTE READ FROM LU REG
5902 002366 000000 WRIBYT: .WORD 0 ;LO BYTE CONTAINS BYTE TO LOAD INTO LU REG
5903 002370 000000 RAX15: .WORD 0 ;LO BYTE CONTAINS BYTE READ FROM REG 15
5904 002372 000000 RAX16: .WORD 0 ;LO BYTE CONTAINS BYTE READ FROM REG 16
5905 002374 000000 WAX15: .WORD 0 ;LO BYTE CONTAINS BYTE TO LOAD INTO REG 15
5906 002376 000000 WAX16: .WORD 0 ;LO BYTE CONTAINS BYTE TO LOAD INTO REG 16
5907 002400 000000 REGNUM: .WORD 0 ;NUMBER (10-17) OF LINE UNIT REG BEING TESTED
5908 002402 000000 AXNUM: .WORD 0 ;NUMBER (0-7) OF EXTENDED REG BYTE BEING TESTED
5909 002404 000000 GOODAT: .WORD 0 ;STORAGE FOR EXPECTED DATA
5910 002406 000000 BADDAT: .WORD 0 ;STORAGE FOR ACTUAL DATA
5911 002410 000000 LOADAT: .WORD 0 ;CONTAINS TEST DATA LOADED INTO REG
5912 002412 000000 FRSTIM: .WORD 0 ;FLAG=0 IF PROGRAM JUST LOADED
5913 002414 000000 SAVE4: .WORD 0 ;SAVE LOC 4 HERE (ERROR TRAP VECTOR)
5914 002416 000000 SAVE6: .WORD 0 ;SAVE LOC 6 HERE (ERROR TRAP VECTOR)
5915 002420 000000 ERROR1: .WORD 0 ;SUBR ERROR BIT FLAGS (DEF'D IN GLOBAL EQUATES)
5916 002422 000000 TXWORD: .WORD 0 ;BITS 0-11 CONTAIN DATA TO LOAD INTO TX SILO
5917 002424 000000 RXWORD: .WORD 0 ;BITS 0-11 CONTAIN DATA READ FROM RCV SILO
5918 002426 000000 DISILO: .WORD 0 ;CONTAINS CURRENT STATE OF DISSI IN BIT 5
5919 002430 000000 CHPTYP: .WORD 0 ;USYRT CHIP TYPE, =0 FOR SIG, ELSE =1
5920 002432 000000 SAVLEN: .WORD 0 ;SAVED TX AND RCV CHAR LENGTHS
5921 002434 000000 DEVMAP: .WORD 0 ;BIT MAP OF ACTIVE DEVICES
5922 002436 000000 DEVPTR: .WORD 0 ;DEVICE MAP BIT POINTER
5923 002440 000000 UNIT: .WORD 0 ;CONTAINS UNIT NO. (1 TO N)
5924 002442 000000 TSTNUM: .WORD 0 ;CONTAINS TEST NUMBER FOR SOME TESTS
5925 002444 000000 STARES: .WORD 0 ;FLAG=0 IF FIRST PASS AFTER STA OR RES
5926
5927 :***** CURRENT DEVICE PARAMETERS *****
5928 002446 160170 MPCSR: .WORD 160170 ;POINTER TO MICROPROCESSOR CSR'S
5929 002450 160171 BSEL1: .WORD 160171 ;POINTER TO BSEL1
5930 002452 160172 BSEL2: .WORD 160172 ;POINTER TO BSEL2
5931 002454 BSEL4.
```

```
5932 002454 160174 SEL4: .WORD 160174 ;POINTER TO SEL4
5933 002456 160176 SEL6: .WORD 160176 ;POINTER TO SEL6
5934 002460 000300 MPIVEC: .WORD 300 ;MICROPROCESSOR INPUT INTERRUPT VECTOR
5935 002462 000304 MPOVEC: .WORD 304 ;MICROPROCESSOR OUTPUT INTERRUPT VECTOR
5936 002464 000240 MPRIOR: .WORD 240 ;MICROPROCESSOR DEVICE PRIORITY
5937 002466 000000 LUSWI1: .WORD 0 ;LINE UNIT SWITCH PACK #1
5938 002470 000000 LUSWI2: .WORD 0 ;LINE UNIT SWITCH PACK #2
5739 002472 000000 LUSWI3: .WORD 0 ;LINE UNIT SWITCH PACK #3
5940 002474 000000 TSTCON: .WORD 0 ;TEST CONNECTOR INDICATOR
5941 002476 000000 RUNINH: .WORD 0 ;RUN SWITCH INDICATOR
5942
5943 ;***** STORAGE FOR DATA READ IN ADDRESS TESTS *****
5944 002500 000 REDDAT: .BYTE 0
5945 002501 000 .BYTE 0
5946 002502 000 .BYTE 0
5947 002503 000 .BYTE 0
5948 002504 000 .BYTE 0
5949 002505 000 .BYTE 0
5950 002506 000 .BYTE 0
5951 002507 000 .BYTE 0
5952
5953 ;:***** GEN'L PURPOSE SCRATCH STORAGE *****
5954 002510 000000 REG0: .WORD 0
5955 002512 000000 REG1: .WORD 0
5956 002514 000000 REG2: .WORD 0
5957 002516 000000 REG3: .WORD 0
5958 002520 000000 REG4: .WORD 0
5959 002522 000000 REG5: .WORD 0
5960 002524 000000 REG6: .WORD 0
5961 002526 000000 REG7: .WORD 0
5962
5963 ;:***** SCRATCH STORAGE FOR MESSAGE REPORTING *****
5964 002530 000000 TMP0: .WORD 0
5965 002532 000000 TMP1: .WORD 0
5966 002534 000000 TMP2: .WORD 0
5967 002536 000000 TMP3: .WORD 0
5968 002540 000000 TMP4: .WORD 0
5969 002542 000000 TMP5: .WORD 0
5970 002544 000000 TMP6: .WORD 0
5971 002546 000000 TMP7: .WORD 0
5972
5973 ;***** INBUS LU REG BIT MASKS FOR UNPREDICTABLE BITS *****
5974 002550 UPBITS:
5975 002550 000 .BYTE 000 ;MASK FOR REG 10
5976 002551 056 .BYTE 056 ;MASK FOR REG 11
5977 002552 000 .BYTE 000 ;MASK FOR REG 12
5978 002553 257 .BYTE 257 ;MASK FOR REG 13
5979 002554 100 .BYTE 100 ;MASK FOR REG 14
5980 002555 377 .BYTE 377 ;MASK FOR REG 15
5981 002556 377 .BYTE 377 ;MASK FOR REG 16
5982 002557 306 .BYTE 306 ;MASK FOR REG 17
5983
5984 002560 200 R14NRW: .BYTE 200 ;REG 14 NON-R/W BITS
5985
5986 ;***** MASKS FOR EXTENDED REGISTER NON-READ/WRITE BITS *****
5987 002561 ANBITS:
```

| | | | | | |
|------|--------|-----|-------|-----|------------------|
| 5988 | 002561 | 377 | .BYTE | 377 | :MASK FOR AX0-15 |
| 5989 | 002562 | 377 | .BYTE | 377 | :MASK FOR AX0-16 |
| 5990 | 002563 | 000 | .BYTE | 000 | :MASK FOR AX1-15 |
| 5991 | 002564 | 360 | .BYTE | 360 | :MASK FOR AX1-16 |
| 5992 | 002565 | 000 | .BYTE | 000 | :MASK FOR AX2-15 |
| 5993 | 002566 | 000 | .BYTE | 000 | :MASK FOR AX2-16 |
| 5994 | 002567 | 004 | .BYTE | 004 | :MASK FOR AX3-15 |
| 5995 | 002570 | 030 | .BYTE | 030 | :MASK FOR AX3-16 |

5996
5997 ;***** DATA PATTERN A *****
5998 PATA:

| | | | | | |
|------|--------|-----|-------|-----|--|
| 5998 | 002571 | | .BYTE | 125 | |
| 5999 | 002571 | 125 | .BYTE | 125 | |
| 6000 | 002572 | 252 | .BYTE | 252 | |
| 6001 | 002573 | 000 | .BYTE | 000 | |
| 6002 | 002574 | 377 | .BYTE | 377 | |
| 6003 | 002575 | 001 | .BYTE | 001 | |
| 6004 | 002576 | 002 | .BYTE | 002 | |
| 6005 | 002577 | 004 | .BYTE | 004 | |
| 6006 | 002600 | 010 | .BYTE | 010 | |
| 6007 | 002601 | 020 | .BYTE | 020 | |
| 6008 | 002602 | 040 | .BYTE | 040 | |
| 6009 | 002603 | 100 | .BYTE | 100 | |
| 6010 | 002604 | 200 | .BYTE | 200 | |
| 6011 | 002605 | 376 | .BYTE | 376 | |
| 6012 | 002606 | 375 | .BYTE | 375 | |
| 6013 | 002607 | 373 | .BYTE | 373 | |
| 6014 | 002610 | 367 | .BYTE | 367 | |
| 6015 | 002611 | 357 | .BYTE | 357 | |
| 6016 | 002612 | 337 | .BYTE | 337 | |
| 6017 | 002613 | 277 | .BYTE | 277 | |
| 6018 | 002614 | 177 | .BYTE | 177 | |

6019
6020 ;***** DATA PATTERN B *****
6021 PATB:

| | | | | | |
|------|--------|-----|-------|-----|--|
| 6021 | 002615 | | .BYTE | 000 | |
| 6022 | 002615 | 000 | .BYTE | 000 | |
| 6023 | 002616 | 000 | .BYTE | 000 | |
| 6024 | 002617 | 040 | .BYTE | 040 | |
| 6025 | 002620 | 100 | .BYTE | 100 | |
| 6026 | 002621 | 220 | .BYTE | 220 | |
| 6027 | 002622 | 000 | .BYTE | 000 | |
| 6028 | 002623 | 000 | .BYTE | 000 | |
| 6029 | 002624 | 051 | .BYTE | 051 | |

6030
6031 ;***** DATA PATTERN C *****
6032 PATC:

| | | | | | |
|------|--------|-----|-------|-----|--|
| 6032 | 002625 | | .BYTE | 020 | |
| 6033 | 002625 | 020 | .BYTE | 020 | |
| 6034 | 002626 | 020 | .BYTE | 020 | |
| 6035 | 002627 | 020 | .BYTE | 020 | |

6036
6037 ;***** DATA PATTERN D *****
6038 PATD:

| | | | | | |
|------|--------|-----|-------|-----|--|
| 6038 | 002630 | | .BYTE | 000 | |
| 6039 | 002630 | 000 | .BYTE | 000 | |
| 6040 | 002631 | 040 | .BYTE | 040 | |
| 6041 | 002632 | 000 | .BYTE | 000 | |

6042
6043 ;***** DATA PATTERN E *****

6044 002633
6045 002633 000
6046 002634 120
6047 002635 020
6048 002636 100
6049 002637 120
6050 002640 000
6051
6052
6053 002641
6054 002641 050
6055 002642 051
6056 002643 050
6057
6058
6059 002644
6060 002644 000
6061 002645 000
6062 002646 240
6063 002647 120
6064 002650 177
6065 002651 000
6066 002652 000
6067 002653 001
6068
6069
6070 002654
6071 002654 000
6072 002655 000
6073 002656 377
6074 002657 017
6075 002660 377
6076 002661 377
6077 002662 375
6078 002663 377
6079
6080
6081 002664
6082 002664 000
6083 002665 000
6084 002666 000
6085 002667 000
6086 002670 000
6087 002671 103
6088 002672 000
6089 002673 000
6090
6091
6092 002674
6093 002674 000
6094 002675 000
6095 002676 010
6096 002677 002
6097 002700 004
6098 002701 103
6099 002702 001

PATE:
.BYTE 000
.BYTE 120
.BYTE 020
.BYTE 100
.BYTE 120
.BYTE 000
:***** DATA PATTERN F *****
PATF:
.BYTE 050
.BYTE 051
.BYTE 050
:***** DATA PATTERN G *****
PATG:
.BYTE 000
.BYTE 000
.BYTE 240
.BYTE 120
.BYTE 177
.BYTE 000
.BYTE 000
.BYTE 001
:***** DATA PATTERN H *****
PATH:
.BYTE 000
.BYTE 000
.BYTE 377
.BYTE 017
.BYTE 377
.BYTE 377
.BYTE 377
.BYTE 375
.BYTE 377
:***** DATA PATTERN I *****
PATI:
.BYTE 000
.BYTE 000
.BYTE 000
.BYTE 000
.BYTE 000
.BYTE 103
.BYTE 000
.BYTE 000
:***** DATA PATTERN J *****
PATJ:
.BYTE 000
.BYTE 000
.BYTE 010
.BYTE 002
.BYTE 004
.BYTE 103
.BYTE 001

| | | | | |
|------|--------|-----|-------|-----|
| 6100 | 002703 | 100 | .BYTE | 100 |
| 6101 | | | | |
| 6102 | | | | |
| 6103 | 002704 | 000 | | |
| 6104 | 002705 | 000 | | |
| 6105 | 002706 | 377 | | |
| 6106 | 002707 | 377 | | |
| 6107 | 002710 | 125 | | |
| 6108 | 002711 | 125 | | |
| 6109 | 002712 | 252 | | |
| 6110 | 002713 | 252 | | |
| 6111 | 002714 | 001 | | |
| 6112 | 002715 | 000 | | |
| 6113 | 002716 | 002 | | |
| 6114 | 002717 | 000 | | |
| 6115 | 002720 | 004 | | |
| 6116 | 002721 | 000 | | |
| 6117 | 002722 | 010 | | |
| 6118 | 002723 | 000 | | |
| 6119 | 002724 | 020 | | |
| 6120 | 002725 | 000 | | |
| 6121 | 002726 | 040 | | |
| 6122 | 002727 | 000 | | |
| 6123 | 002730 | 100 | | |
| 6124 | 002731 | 000 | | |
| 6125 | 002732 | 200 | | |
| 6126 | 002733 | 000 | | |
| 6127 | 002734 | 000 | | |
| 6128 | 002735 | 001 | | |
| 6129 | 002736 | 000 | | |
| 6130 | 002737 | 002 | | |
| 6131 | 002740 | 000 | | |
| 6132 | 002741 | 004 | | |
| 6133 | 002742 | 000 | | |
| 6134 | 002743 | 010 | | |
| 6135 | 002744 | 000 | | |
| 6136 | 002745 | 020 | | |
| 6137 | 002746 | 000 | | |
| 6138 | 002747 | 040 | | |
| 6139 | 002750 | 000 | | |
| 6140 | 002751 | 100 | | |
| 6141 | 002752 | 000 | | |
| 6142 | 002753 | 200 | | |
| 6143 | 002754 | 376 | | |
| 6144 | 002755 | 377 | | |
| 6145 | 002756 | 375 | | |
| 6146 | 002757 | 377 | | |
| 6147 | 002760 | 373 | | |
| 6148 | 002761 | 377 | | |
| 6149 | 002762 | 367 | | |
| 6150 | 002763 | 377 | | |
| 6151 | 002764 | 357 | | |
| 6152 | 002765 | 377 | | |
| 6153 | 002766 | 337 | | |
| 6154 | 002767 | 377 | | |
| 6155 | 002770 | 277 | | |

***** DATA PATTERN K *****

PATK: .BYTE 000
.BYTE 000
.BYTE 377
.BYTE 377
.BYTE 125
.BYTE 125
.BYTE 252
.BYTE 252
.BYTE 001
.BYTE 000
.BYTE 002
.BYTE 000
.BYTE 004
.BYTE 000
.BYTE 010
.BYTE 000
.BYTE 020
.BYTE 000
.BYTE 040
.BYTE 000
.BYTE 100
.BYTE 000
.BYTE 200
.BYTE 000
.BYTE 000
.BYTE 001
.BYTE 000
.BYTE 002
.BYTE 000
.BYTE 004
.BYTE 000
.BYTE 010
.BYTE 000
.BYTE 020
.BYTE 000
.BYTE 040
.BYTE 000
.BYTE 100
.BYTE 000
.BYTE 200
.BYTE 376
.BYTE 377
.BYTE 375
.BYTE 377
.BYTE 373
.BYTE 377
.BYTE 367
.BYTE 377
.BYTE 357
.BYTE 377
.BYTE 337
.BYTE 377
.BYTE 277

| | | | | |
|------|--------|-----|-------|-----|
| 6156 | 002771 | 377 | .BYTE | 377 |
| 6157 | 002772 | 177 | .BYTE | 177 |
| 6158 | 002773 | 377 | .BYTE | 377 |
| 6159 | 002774 | 377 | .BYTE | 377 |
| 6160 | 002775 | 376 | .BYTE | 376 |
| 6161 | 002776 | 377 | .BYTE | 377 |
| 6162 | 002777 | 375 | .BYTE | 375 |
| 6163 | 003000 | 377 | .BYTE | 377 |
| 6164 | 003001 | 373 | .BYTE | 373 |
| 6165 | 003002 | 377 | .BYTE | 377 |
| 6166 | 003003 | 367 | .BYTE | 367 |
| 6167 | 003004 | 377 | .BYTE | 377 |
| 6168 | 003005 | 357 | .BYTE | 357 |
| 6169 | 003006 | 377 | .BYTE | 377 |
| 6170 | 003007 | 337 | .BYTE | 337 |
| 6171 | 003010 | 377 | .BYTE | 377 |
| 6172 | 003011 | 277 | .BYTE | 277 |
| 6173 | 003012 | 377 | .BYTE | 377 |
| 6174 | 003013 | 177 | .BYTE | 177 |

6175
6176
6177 003014
6178 003014 000
6179 003015 000
6180 003016 377
6181 003017 377
6182 003020 000
6183 003021 000

***** DATA PATTERN L *****
PATL:
.BYTE 000
.BYTE 000
.BYTE 377
.BYTE 377
.BYTE 000
.BYTE 000

6184
6185
6186 003022
6187 003022 000
6188 003023 020
6189 003024 000
6190 003025 000
6191 003026 200
6192 003027 000
6193 003030 000
6194 003031 051

***** DATA PATTERN M *****
PATM:
.BYTE 000
.BYTE 020
.BYTE 000
.BYTE 000
.BYTE 200
.BYTE 000
.BYTE 000
.BYTE 051

6195
6196
6197 003032
6198 003032 000
6199 003033 000
6200 003034 000
6201 003035 125
6202 003036 000
6203 003037 252
6204 003040 000
6205 003041 377
6206 003042 005
6207 003043 000
6208 003044 012
6209 003045 000
6210 003046 017
6211 003047 000

***** DATA PATTERN N *****
PATN:
.BYTE 000
.BYTE 000
.BYTE 000
.BYTE 125
.BYTE 000
.BYTE 252
.BYTE 000
.BYTE 377
.BYTE 005
.BYTE 000
.BYTE 012
.BYTE 000
.BYTE 017
.BYTE 000

6212
6213
6214 003050
6215 003050 000
6216 003051 041
6217 003052 004
6218 003053 010
6219 003054 020
6220 003055 040
6221 003056 100
6222 003057 101
6223 003060 200
6224 003061 201
6225 003062 300
6226 003063 111
6227 003064 301
6228 003065 375
6229

***** DATA PATTERN O *****
PATO:

.BYTE 000
.BYTE 041
.BYTE 004
.BYTE 010
.BYTE 020
.BYTE 040
.BYTE 100
.BYTE 101
.BYTE 200
.BYTE 201
.BYTE 300
.BYTE 111
.BYTE 301
.BYTE 375

6230
6231 003066
6232 003066 000
6233 003067 113
6234 003070 200
6235 003071 040
6236 003072 020
6237 003073 010
6238 003074 001
6239 003075 104
6240 003076 007
6241 003077 105
6242 003100 007
6243 003101 144
6244 003102 107
6245 003103 157
6246

***** DATA PATTERN P *****
PATP:

.BYTE 000
.BYTE 113
.BYTE 200
.BYTE 040
.BYTE 020
.BYTE 010
.BYTE 001
.BYTE 104
.BYTE 007
.BYTE 105
.BYTE 007
.BYTE 144
.BYTE 107
.BYTE 157

6247
6248 003104 000
6249 003105 000
6250 003106 001
6251 003107 000
6252 003110 013
6253 003111 000
6254 003112 011
6255 003113 000
6256 003114 021
6257 003115 000
6258 003116 101
6259 003117 000
6260 003120 301
6261 003121 000
6262 003122 000
6263 003123 001
6264 003124 000
6265 003125 002
6266 003126 000
6267 003127 004

***** DATA PATTERN U *****
PATU:

.BYTE 000
.BYTE 000
.BYTE 001
.BYTE 000
.BYTE 013
.BYTE 000
.BYTE 011
.BYTE 000
.BYTE 021
.BYTE 000
.BYTE 101
.BYTE 000
.BYTE 301
.BYTE 000
.BYTE 000
.BYTE 001
.BYTE 000
.BYTE 002
.BYTE 000
.BYTE 004

| | | | | |
|------|--------|-----|-------|-----|
| 6268 | 003130 | 000 | .BYTE | 000 |
| 6269 | 003131 | 040 | .BYTE | 040 |
| 6270 | 003132 | 000 | .BYTE | 000 |
| 6271 | 003133 | 100 | .BYTE | 100 |
| 6272 | 003134 | 000 | .BYTE | 000 |
| 6273 | 003135 | 200 | .BYTE | 200 |
| 6274 | 003136 | 000 | .BYTE | 000 |
| 6275 | 003137 | 346 | .BYTE | 346 |
| 6276 | 003140 | 000 | .BYTE | 000 |
| 6277 | 003141 | 345 | .BYTE | 345 |
| 6278 | 003142 | 000 | .BYTE | 000 |
| 6279 | 003143 | 343 | .BYTE | 343 |
| 6280 | 003144 | 000 | .BYTE | 000 |
| 6281 | 003145 | 307 | .BYTE | 307 |
| 6282 | 003146 | 000 | .BYTE | 000 |
| 6283 | 003147 | 247 | .BYTE | 247 |
| 6284 | 003150 | 000 | .BYTE | 000 |
| 6285 | 003151 | 147 | .BYTE | 147 |

:***** PATTERN V *****
PATV:

| | | | | |
|------|--------|-----|-------|-----|
| 6288 | 003152 | 000 | .BYTE | 000 |
| 6289 | 003153 | 000 | .BYTE | 000 |
| 6290 | 003154 | 333 | .BYTE | 333 |
| 6291 | 003155 | 000 | .BYTE | 000 |
| 6292 | 003156 | 331 | .BYTE | 331 |
| 6293 | 003157 | 000 | .BYTE | 000 |
| 6294 | 003160 | 323 | .BYTE | 323 |
| 6295 | 003161 | 000 | .BYTE | 000 |
| 6296 | 003162 | 313 | .BYTE | 313 |
| 6297 | 003163 | 000 | .BYTE | 000 |
| 6298 | 003164 | 233 | .BYTE | 233 |
| 6299 | 003165 | 000 | .BYTE | 000 |
| 6300 | 003166 | 133 | .BYTE | 133 |
| 6301 | 003167 | 000 | .BYTE | 000 |
| 6302 | 003170 | 000 | .BYTE | 000 |
| 6303 | 003171 | 001 | .BYTE | 001 |
| 6304 | 003172 | 000 | .BYTE | 000 |
| 6305 | 003173 | 002 | .BYTE | 002 |
| 6306 | 003174 | 000 | .BYTE | 000 |
| 6307 | 003175 | 004 | .BYTE | 004 |
| 6308 | 003176 | 000 | .BYTE | 000 |
| 6309 | 003177 | 040 | .BYTE | 040 |
| 6310 | 003200 | 000 | .BYTE | 000 |
| 6311 | 003201 | 100 | .BYTE | 100 |
| 6312 | 003202 | 000 | .BYTE | 000 |
| 6313 | 003203 | 200 | .BYTE | 200 |
| 6314 | 003204 | 000 | .BYTE | 000 |
| 6315 | 003205 | 346 | .BYTE | 346 |
| 6316 | 003206 | 000 | .BYTE | 000 |
| 6317 | 003207 | 345 | .BYTE | 345 |
| 6318 | 003210 | 000 | .BYTE | 000 |
| 6319 | 003211 | 343 | .BYTE | 343 |
| 6320 | 003212 | 000 | .BYTE | 000 |
| 6321 | 003213 | 307 | .BYTE | 307 |
| 6322 | 003214 | 000 | .BYTE | 000 |
| 6323 | 003215 | 247 | .BYTE | 247 |

6324 003216 000 .BYTE 000
6325 003217 147 .BYTE 147

6326
6327 003220 ENDPAT:
6328 .EVEN
6329
6330
6331
6332

6333
6334
6335

;*** TEST MESSAGES TO BE TRANSMITTED ***

6336 003220 000400 MSG1: TXSOM
6337 003222 000400 TXSOM
6338 003224 000000 000
6339 003226 000125 125
6340 003230 000252 252
6341 003232 000377 377
6342 003234 000000 000
6343 003236 001000 TXEOM
6344 003240 001000 TXEOM
6345 003242 001000 TXEOM
6346 003244 001000 TXEOM

6347
6348
6349
6350
6351
6352
6353

MSG4: 377
000
125
252
TXEOM
TXEOM

6354
6355
6356
6357
6358

6359
6360 003262 000100

;*** RECEIVED DATA BUFFER (64. WORDS) ***
RCVBUF: .BLKW 64.

6361
6362
6363
6364
6365
6366
6367
6368
6369
6370

6407
6408
6409
6410
6411
6412
6413
6414
6415
6416
6417
6418
6419
6420
6421
6422
6423
6424
6425
6426
6427
6428
6429
6430
6431
6432
6433
6434
6435
6436
6437
6438
6439
6440
6441
6442
6443
6444
6445
6446
6447
6448
6449
6450
6451
6452
6453
6454
6455
6456
6457
6458
6459
6460
6461
6462

.SBTTL GLOBAL SUBROUTINES

:/ THE GLOBAL SUBROUTINES ARE CALLED BY MORE THAN ONE TEST
:/

* STPCLK - THIS SUBROUTINE FORCES THE DMC11 OR KMC11 MICROPROCESSOR TO
* EXECUTE AN INSTRUCTION WHICH IS PASSED IN THE WORD FOLLOWING THE CALL.

```
STPCLK:
BISB #ROMO!ROMI,@BSEL1 ;SET ROMO, ROMI BITS IN BSEL1
MOV @ (SP),@SEL6 ;PUT INSTRUCTION INTO SEL6
BISB #ROMO!ROMI!STEPMP,@BSEL1 ;SET ROMO, ROMI, STEPMP IN BSEL1
BICB #ROMO!ROMI!STEPMP,@BSEL1 ;CLEAR ROMO, ROMI, STEPMP IN BSEL1
ADD #2,(SP) ;FIX UP RETURN PC
RTS PC ;RETURN
```

* MSTCLR - THIS SUBROUTINE ISSUES A MASTER CLEAR AND SETS LULoop

```
MSTCLR:
MOV R1,-(SP) ;SAVE R1
MOVB #MCLR,@BSEL1 ;SET MASTER CLEAR BIT
BICB #RUN!MCLR,@BSEL1 ;CLEAR RUN AND MCLR BITS
MOV #20.,R1 ;INITIALIZE STALL COUNTER
2$:
NOP ;STALL IN LOOP FOR SEVERAL MICRO-SEC
DEC R1
BNE 2$
BISB #LULoop,@BSEL1 ;SET LU LOOP
MOV (SP)+,R1 ;RESTORE R1
CLR SAVLEN ;CLEAR CHAR LENGTH FROM SETUP
RTS PC ;RETURN
```

* READLU - THIS SUBROUTINE FORCES THE DMC11 OR KMC11 MICROPROCESSOR
* TO EXECUTE AN INSTRUCTION WHICH READS THE LINE UNIT REG WHOSE
* NUMBER IS PASSED IN REGNUM, INTO REDBYT.

```
READLU:
MOV R1,-(SP) ;SAVE R1
MOV REGNUM,R1 ;GET LINE UNIT REG NUMBER
ASL R1 ;SHIFT INTO SOURCE BITS 4-7
ASL R1
```

003540
003540 152777 000006 176702
003546 017677 000000 176702
003554 152777 000007 176666
003562 142777 000007 176660
003570 062716 000002
003574 000207

003576
003576 010146
003600 112777 000100 176642
003606 142777 000300 176634
003614 012701 000024
003620 000240
003622 005301
003624 001375
003626 152777 000010 176614
003634 012601
003636 005037 002432
003642 000207

003644
003644 010146
003646 013701 002400
003652 006301
003654 006301

```
6463 003656 006301 ASL R1
6464 003660 006301 ASL R1
6465 003662 052701 000004 BIS #4,R1 ;SET DESTINATION = BSEL4
6466 003666 052701 021000 BIS #MVIOX,R1 ;SET REST OF MOVE INSTRUCTION
6467 003672 010137 003702 MOV R1,2$ ;SET INSTRUCTION AS SUBROUTINE ARGUMENT
6468 003676 004737 003540 JSR PC,STPCLK ;EXECUTE MOVE INSTRUCTION
6469 003702 000000 2$: .WORD 0 ;INSTRUCTION GOES HERE
6470 003704 117737 176544 002364 MOVB @BSEL4,REDBYT ;GET LU REG CONTENTS INTO REDBYT
6471 003712 105037 002365 CLRB REDBYT+1 ;CLR HI BYTE OF STORAGE
6472 003716 012601 MOV (SP)+,R1 ;RESTORE R1
6473 003720 000207 RTS PC ;RETURN
6474
6475
6476
6477
6478
6479
```

* WRITLU - THIS SUBROUTINE FORCES THE DMC11 OR KMC11 MICROPROCESSOR TO
* EXECUTE AN INSTRUCTION WHICH LOADS THE BYTE CONTAINED IN WRIBYT
* INTO THE LU REG WHOSE NUMBER IS PASSED IN REGNUM,

```
6484 003722 WRITLU:
6485 003722 010146 MOV R1,-(SP) ;SAVE R1
6486 003724 013701 002400 MOV REGNUM,R1 ;GET LINE UNIT REG NUMBER
6487 003730 052701 000100 BIS #100,R1 ;SET SOURCE = BSEL4
6488 003734 052701 122000 BIS #MVIXO,R1 ;SET REST OF MOVE INSTRUCTION
6489 003740 010137 003762 MOV R1,2$ ;SET INSTRUCTION AS SUBROUTINE ARGUMENT
6490 003744 105037 002367 CLRB WRIBYT+1 ;CLR HI BYTE OF STORAGE
6491 003750 113777 002366 176476 MOVB WRIBYT,@BSEL4 ;LOAD BYTE INTO BSEL4
6492 003756 004737 003540 JSR PC,STPCLK ;EXECUTE MOVE INSTRUCTION
6493 003762 000000 2$: .WORD 0
6494 003764 012601 MOV (SP)+,R1 ;RESTORE R1
6495 003766 000207 RTS PC ;RETURN
6496
6497
6498
6499
6500
```

* GETREG - THIS SUBROUTINE READS THE LINE UNIT REGISTERS 10-17 INTO THE
* REGISTER STORAGE TABLE (LUREG:).

```
6504 003770 GETREG:
6505 003770 010146 MOV R1,-(SP) ;SAVE R1
6506 003772 013746 002400 MOV REGNUM,-(SP) ;SAVE CURRENT REG NO.
6507 003776 012701 002302 MOV #LUR10,R1 ;INIT POINTER TO REG STORAGE TABLE
6508 004002 012737 000010 002400 MOV #10,REGNUM ;INIT LU REG NO. TO 10
6509 004010 004737 003644 3$: JSR PC,READLU ;READ A LINE UNIT REG
6510 004014 113721 002364 MOVB REDBYT,(R1)+ ;PUT BYTE READ INTO TABLE
6511 004020 105021 CLRB (R1)+ ;CLEAR UPPER BYTE OF TABLE ENTRY
6512 004022 005237 002400 INC REGNUM ;INCREMENT REG NO.
6513 004026 023727 002400 000020 CMP REGNUM,#20 ;SEE IF ALL REGS READ YET
6514 004034 002765 BLT 3$ ;BR IF NOT
6515 004036 012637 002400 MOV (SP)+,REGNUM ;RESTORE CURRENT REG NO.
6516 004042 012601 MOV (SP)+,R1 ;RESTORE R1
6517 004044 000207 RTS PC ;RETURN
6518
```

6519
6520
6521
6522
6523
6524
6525
6526
6527
6528
6529
6530
6531 004046
6532 004046 152777 000006 176374
6533 004054 017677 000000 176374
6534 004062 152777 000206 176360
6535 004070 062716 000002
6536 004074 000207
6537
6538
6539
6540
6541
6542
6543
6544
6545
6546
6547
6548
6549 004076 010146
6550 004100 013746 002400
6551 004104 042737 000001 002420
6552 004112 012737 000014 002400
6553 004120 113737 002402 002366
6554 004126 006237 002366
6555 004132 152737 000024 002366
6556 004140 053737 002426 002366
6557 004146 004737 003722
6558 004152 005001
6559 004154 004737 003644 6\$:
6560 004160 132737 000200 002364
6561 004166 001006
6562 004170 005201
6563 004172 001370
6564 004174 052737 000001 002420
6565 004202 000424
6566 004204 012737 000015 002400 9\$:
6567 004212 004737 003644
6568 004216 113737 002364 002370
6569 004224 105037 002371
6570 004230 012737 000016 002400
6571 004236 004737 003644
6572 004242 113737 002364 002372
6573 004250 105037 002373
6574 004254 012637 002400 12\$:

```
*****  
* LOOPIN - THIS SUBROUTINE PLACES THE MICROPROCESSOR IN A LOOP ON AN  
* INSTRUCTION, BY MOVING THE INSTRUCTION FROM THE WORD FOLLOWING THE CALL  
* INTO SEL6, AND SETTING RUN AND ROMI IN BSEL1. THE SUBROUTINE RETURNS  
* WITH THE MICROPROCESSOR STUCK IN THE LOOP, AND IF IT IS DESIRED TO  
* TERMINATE THE LOOP, THE PDP-11 PROGRAM MUST CLEAR THE RUN BIT IN  
* BSEL1, OR CALL SUBROUTINE MSTCLR TO DO THIS.  
*****
```

```
LOOPIN:  
BISB #ROMO!ROMI,@BSEL1 ;SET ROMO, ROMI BITS IN BSEL1  
MOV @ (SP),@SEL6 ;PUT MICROINSTRUCTION INTO SEL6  
BISB #RUN.ROMO!ROMI,@BSEL1 ;SET RUN, ROMO, ROMI IN BSEL1  
ADD #2,(SP) ;FIX UP RETURN PC  
RTS PC ;RETURN WITH MICROPROCESSOR STUCK IN SINGLE  
; INSTRUCTION LOOP
```

```
*****  
* READAX - THIS SUBROUTINE READS THE USYRT REG PAIR WHOSE NUMBER (0-3)  
* IS PASSED IN BITS 1,2 OF AXNUM ON ENTRY, AND RETURNS THE BYTES READ IN  
* RAX15 AND RAX16. IF THE LINE UNIT DOES NOT RESPOND WITH READY IN REG 14,  
* RRDYTO BIT IS SET IN ERROR1 ON RETURN.  
*****
```

```
READAX: MOV R1,-(SP) ;SAVE R1  
MOV REGNUM,-(SP) ;STORE CURRENT REG NO.  
BIC #RRDYTO,ERROR1 ;CLEAR ERROR BIT  
MOV #14,REGNUM ;SET LU REG NO. = 14  
MOVB AXNUM,WRIBYT ;SET UP AX REG NO. BITS  
ASR WRIBYT  
BISB #RDAX!ENAX,WRIBYT ;SET UP BITS TO LOAD INTO REG 14  
BIS DISILO,WRIBYT ;SET PROPER STATE OF DISSI BIT  
JSR PC,WRITLU ;SET RDAX AND ENAX IN REG 14  
CLR R1 ;INIT TIMER  
JSR PC,READLU ;READ REG 14  
BITB #READY,REDBYT ;SEE IF READY BIT SET IN REG 14 YET  
BNE 9$ ;BR IF READY SET  
INC R1 ;INCR TIMER  
BNE 6$ ;BR IF TIMER DIDN'T TIME OUT YET  
BIS #RRDYTO,ERROR1 ;SET ERROR FLAG FOR TIME OUT ON READ RDY  
BR 12$ ;BR TO RETURN  
9$: MOV #15,REGNUM ;SET REG NO. = 15  
JSR PC,READLU ;READ REG 15  
MOVB REDBYT,RAX15 ;STORE REG AX-15  
CLRB RAX15+1 ;CLR HI BYTE OF STORAGE  
MOV #16,REGNUM ;SET REG NO. = 16  
JSR PC,READLU ;READ REG 16  
MOVB REDBYT,RAX16 ;STORE REG AX-16  
CLRB RAX16+1 ;CLR HI BYTE OF STORAGE  
12$: MOV (SP)+,REGNUM ;RESTORE CURRENT REG NO.
```



```
6575 004260 012601          MOV    (SP)+,R1      ;RESTORE R1
6576 004262 000207          RTS     PC           ;RETURN
6577
6578
6579
6580
6581
6582
6583
6584
6585
6586
6587 004264 010146          ;*****
6588 004266 013746 002400      ;* WRITAX - THIS SUBROUTINE WRITES THE USYRT REG PAIR WHOSE NUMBER (0-3) IS
6589 004272 042737 000002 002420      ;* PASSED IN BITS 1,2 OF AXNUM ON ENTRY, WITH THE DATA FROM WAX15 AND
6590 004300 012737 000014 002400      ;* WAX16. IF LINE UNIT DOES NOT RESPOND WITH READY IN REG 14, WRDYTO BIT
6591 004306 113737 002402 002366      ;* IS SET IN ERROR1 ON RETURN.
6592 004314 006237 002366      ;*****
6593 004320 053737 002426 002366      WRITAX: MOV    R1,-(SP)      ;SAVE R1
6594 004326 004737 003722      MOV    REGNUM,-(SP)     ;SAVE CURRENT REG NO.
6595 004332 012737 000015 002400      BIC    #WRDYTO,ERROR1  ;CLEAR ERROR BIT
6596 004340 105037 002375      MOV    #14,REGNUM      ;SET LU REG NO. = 14
6597 004344 113737 002374 002366      MOVB   AXNUM,WRIBYT    ;SET AX REG NO. BITS
6598 004352 004737 003722      ASR    WRIBYT
6599 004356 005237 002400      BIS    DISILO,WRIBYT   ;SET PROPER STATE OF DISSI BIT
6600 004362 105037 002377      JSR    PC,WRITLU      ;SET AX NO. BITS IN REG 14
6601 004366 113737 002376 002366      MOV    #15,REGNUM      ;SET REG NO. = 15
6602 004374 004737 003722      CLRB   WAX15+1        ;CLR HI BYTE OF STORAGE
6603 004400 012737 000014 002400      MOVB   WAX15,WRIBYT   ;SET UP BYTE TO WRITE INTO REG 15
6604 004406 113737 002402 002366      JSR    PC,WRITLU      ;WRITE BYTE INTO REG 15
6605 004414 006237 002366      INC    REGNUM          ;SET REG NO. = 16
6606 004420 152737 000014 002366      CLRB   WAX16+1        ;CLR HI BYTE OF STORAGE
6607 004426 053737 002426 002366      MOVB   WAX16,WRIBYT   ;SET UP BYTE TO WRITE INTO REG 16
6608 004434 004737 003722      JSR    PC,WRITLU      ;WRITE BYTE INTO REG 16
6609 004440 005001          CLR    R1              ;SET UP BITS TO LOAD INTO REG 14
6610 004442 004737 003644 6$:      JSR    PC,READLU      ;SET PROPER STATE OF DISSI BIT
6611 004446 132737 000200 002364      BITB   #READY,REDBYT  ;SET ENAX AND WAX IN REG 14
6612 004454 001005          BNE    9$              ;INIT PROGRAM TIMER
6613 004456 005201          INC    R1              ;READ REG 14
6614 004460 001370          BNE    6$              ;SEE IF READY BIT SET IN REG 14 YET
6615 004462 052737 000002 002420      BIS    #ENAX!WAX,WRIBYT ;BR IF READY SET
6616 004470 012637 002400 9$:      MOV    (SP)+,REGNUM    ;INCR TIMER
6617 004474 012601          MOV    (SP)+,R1       ;BR IF TIMER DIDN'T TIME OUT YET
6618 004476 000207          RTS     PC             ;SET ERROR FLAG BIT FOR TIME OUT ON WRITE RDY
6619
6620
6621
6622
6623
6624
6625
6626
6627
6628 004500 010146          ;*****
6629 004502 013746 002402      ;* GETALL - THIS SUBROUTINE READS THE LINE UNIT REGS 10-17 AND THE EXTENDED
6630 004506 012737 014411 002530      ;* REGISTERS AX0-AX3 INTO REGISTER STORAGE TABLE (LUREG:).
6630 004506 012737 014411 002530      ;*****
6630 004506 012737 014411 002530      GETALL: MOV    R1,-(SP)      ;SAVE R1
6630 004506 012737 014411 002530      MOV    AXNUM,-(SP)     ;SAVE CURRENT AX REG BYTE NO.
6630 004506 012737 014411 002530      MOV    #DH5,IMPO      ;SET AX LO BYTE NO.
```

```

6631 004514 032737 000001 002402      BIT      #BIT0,AXNUM      ;SEE IF LO OR HI BYTE
6632 004522 001403                BEQ      1$              ;BR IF LO BYTE
6633 004524 012737 014414 002530      MOV      #DH6,TMP0      ;SET AX HI BYTE NO.
6634 004532 004737 003770      1$:     JSR      PC,GETREG  ;READ AND STORE REGS 10-17
6635 004536 142777 000010 175704      BICB    #LULOOP,@BSEL1 ;CLEAR LULOOP
6636 004544 012701 002322      MOV      #AX0.15,R1    ;INIT POINTER TO REG STORAGE TABLE
6637 004550 005037 002402      CLR     AXNUM          ;INIT AX REG BYTE NO. TO 0
6638 004554 004737 004076      3$:     JSR      PC,READAX  ;READ 2 AX REG BYTES
6639 004560 113721 002370      MOV     RAX15,(R1)+    ;PUT LO BYTE READ INTO TABLE
6640 004564 105021                CLRB    (R1)+          ;CLEAR UPPER BYTE OF TABLE ENTRY
6641 004566 113721 002372      MOV     RAX16,(R1)+    ;PUT HI BYTE READ INTO TABLE
6642 004572 105021                CLRB    (R1)+          ;CLEAR UPPER BYTE OF TABLE ENTRY
6643 004574 062737 000002 002402      ADD     #2,AXNUM       ;INCR AX REG BYTE NO.
6644 004602 023727 002402 000010      CMP     AXNUM,#10     ;SEE IF ALL REGS READ YET
6645 004610 002761                BLT     3$              ;BR IF NOT
6646 004617 012637 002402      MOV     (SP)+,AXNUM    ;RESTORE CURRENT AX REG BYTE NO.
6647 004616 012601                MOV     (SP)+,R1      ;RESTORE R1
6648 004620 013737 002402 002532      MOV     AXNUM,TMP1
6649 004626 006237 002532      ASR     TMP1          ;GET EXTENDED REG NO. FOR PRINTOUT
6650 004632 000207                RTS     PC              ;RETURN

```

```

6651
6652
6653
6654
6655

```

```

:*****
;* OSIRDY - THIS SUBROUTINE CHECKS FOR THE PROPER STATES OF ORDY (REG 11)
;* AND OCOR (REG 17) AND REPORTS AN ERROR IF EITHER IS NOT PROPERLY SET
;* AS PASSED IN BIT 0 (ORDY) AND BIT 1 (OCOR) OF THE WORD FOLLOWING THE
;* CALL.
;* IF AN ERROR OCCURS, A RETURN IS MADE TO THE TEST, AT THE ADDRESS IN
;* RETADR.
:*****

```

```

6664 004634 013746 002400      OSIRDY: MOV     REGNUM,-(SP) ;SAVE LU REG NO.
6665 004640 013746 002352      MOV     SUBRPC,-(SP)
6666 004644 005737 002352      TST     SUBRPC          ;SEE IF THIS IS A NESTED CALL
6667 004650 001006                BNE     1$              ;BR IF YES
6668 004652 016637 000004 002352      MOV     4(SP),SUBRPC
6669 004660 162737 000004 002352      SUB     #4,SUBRPC
6670 004666 012737 000011 002400      1$:     MOV     #11,REGNUM    ;GET PC OF SUBROUTINE CALL
6671 004674 004737 003644      JSR     PC,READLU      ;SET REG NO. TO 11
6672 004700 032776 000001 000004      BIT     #BIT0,@4(SP)   ;READ REG 11
6673 004706 001413                BEQ     3$              ;GET EXPECTED STATE OF ORDY
6674 004710 132737 000020 002364      BITB   #ORDY,REDBYT   ;BR IF EXPECTED ORDY = 0
6675 004716 001022                BNE     9$              ;SEE IF ORDY = 1
6676 004720 004737 004500      JSR     PC,GETALL     ;BR IF ORDY = 1
6677      ;REPORT ORDY NOT SET ;GET REGS FOR PRINTOUT
6678      ERRDF 7,EM7,ERR4

```

```

TRAP  C$ERDF
.WORD 7
.WORD EM7
.WORD ERR4

```

```

(4) 004724 104455
(5) 004726 000007
(5) 004730 012350
(5) 004732 015576
6679 004734 000451
6680 004736 132737 000020 002364 3$:
6681 004744 001407
6682 004746 004737 004500

```

```

BR 16$ ;TAKE ERROR RETURN
BITB #ORDY,REDBYT ;SEE IF ORDY = 0
BEQ 9$ ;BR IF ORDY = 0
JSR PC,GETALL ;GET REGS FOR PRINTOUT

```

```
6683 ;REPORT ORDY NOT CLEARED
6684 004752 ERRDF 8,EM8,ERR4
(4) 004752 104455 TRAP C$ERDF
(5) 004754 000010 .WORD 8
(5) 004756 012365 .WORD EM8
(5) 004760 015576 .WORD ERR4
6685 004762 000436 BR 16$ ;TAKE ERROR RETURN
6686 004764 012737 000017 002400 9$: MOV #17,REGNUM ;SET REG NO. = 17
6687 004772 004737 003644 JSR PC,READLU ;READ LU REG 17
6688 004776 132776 000002 000004 BITB #BIT1,@4(SP) ;GET EXPECTED STATE OF OCOR
6689 005004 001413 BEQ 12$ ;BR IF EXPECTED OCOR 0
6690 005006 132737 000020 002364 BITB #OCOR,REDBYT ;SEE IF OCOR = 1
6691 005014 001031 BNE 20$ ;BR IF OCOR = 1
6692 005016 004737 004500 JSR PC,GETALL ;GET REGS FOR PRINTOUT
6693 ;REPORT OCOR NOT SET
6694 005022 ERRDF 9,EM9,ERR4
(4) 005022 104455 TRAP C$ERDF
(5) 005024 000011 .WORD 9
(5) 005026 012406 .WORD EM9
(5) 005030 015576 .WORD ERR4
6695 005032 000412 BR 16$ ;TAKE ERROR RETURN
6696 005034 132737 000020 002364 12$: BITB #OCOR,REDBYT ;SEE IF OCOR = 0
6697 005042 001416 BEQ 20$ ;BR IF OCOR = 0
6698 005044 004737 004500 JSR PC,GETALL ;GET REGS FOR PRINTOUT
6699 ;REPORT OCOR NOT CLEARED
6700 005050 ERRDF 10,EM10,ERR4
(4) 005050 104455 TRAP C$ERDF
(5) 005052 000012 .WORD 10
(5) 005054 012423 .WORD EM10
(5) 005056 015576 .WORD ERR4
6701 005060 016637 000002 002400 16$: MOV 2(SP),REGNUM ;RESTORE LU REG NO.
6702 005066 013706 002346 MOV PSTACK,SP ;RESTORE STACK POINTER TO BASE LEVEL
6703 005072 013746 002362 MOV RETADR,-(SP) ;FIX ERROR RETURN PC
6704 005076 000407 BR 23$
6705 005100 062766 000002 000004 20$: ADD #2,4(SP) ;FIX UP ERROR-FREE RETURN PC
6706 005106 012637 002352 MOV (SP)+,SUBRPC
6707 005112 012637 002400 MOV (SP)+,REGNUM ;RESTORE LU REG NO.
6708 005116 000207 23$: RTS PC ;RETURN
6709
6710
6711
6712
6713
6714
6715 ;*****
6716 ;* WAIT50 - THIS SUBROUTINE STALLS FOR AT LEAST 50 MICRO-SEC, AND THEN RETURNS.
6717 ;*****
6717 005120 010146 WAIT50: MOV R1,-(SP) ;SAVE R1
6718 005122 012701 000310 MOV #200.,R1 ;INIT COUNTER
6719 005126 005301 3$: DEC R1 ;DECREMENT COUNTER
6720 005130 001376 BNE 3$ ;BR IF NOT DONE YET
6721 005132 012601 MOV (SP)+,R1 ;RESTORE R1
6722 005134 000207 RTS PC ;RETURN
6723
6724
6725
6726
```

6727
6728
6729
6730
6731 005136 000240
6732 005140 000240
6733 005142 000240
6734 005144 000207
6735
6736
6737
6738
6739
6740
6741
6742
6743
6744 005146 013746 002400
6745 005152 042737 170000 002422
6746 005160 012737 000011 002400
6747 005166 113737 002423 002366
6748 005174 004737 003722
6749 005200 012737 000010 002400
6750 005206 113737 002422 002366
6751 005214 004737 003722
6752 005220 012637 002400
6753 005224 000207
6754
6755
6756
6757
6758
6759
6760
6761
6762
6763
6764
6765 005226 010146
6766 005230 017601 000002
6767 005234 001426
6768 005236 100006
6769 005240 042701 100000
6770 005244 005737 002430
6771 005250 001401
6772 005252 005301
6773 005254 152777 000010 175166 2\$:
6774 005262 152777 000020 175160 3\$:
6775 005270 004737 005136
6776 005274 142777 000020 175146
6777 005302 004737 005136
6778 005306 005301
6779 005310 001364
6780 005312 062766 000002 000002 6\$:
6781 005320 012601
6782 005322 000207

;* STALL - THIS SUBROUTINE STALLS FOR ABOUT A MICRO-SEC.

STALL: NOP
NOP
NOP
RTS PC

;* LDTXSI - THIS SUBROUTINE LOADS THE TX SILO (REGS 10,11) WITH THE DATA PASSED
;* IN BITS 0-11 OF TXWORD.

LDTXSI: MOV REGNUM,-(SP) ;SAVE LU REG NO.
BIC #170000,TXWORD ;CLEAR UNUSED BITS
MOV #11,REGNUM ;SET REG NO. = 11
MOVB TXWORD+1,WRIBYT ;SET DATA TO BE WRITTEN INTO REG 11
JSR PC,WRITLU ;LOAD DATA INTO REG 11
MOV #10,REGNUM ;SET REG NO. = 10
MOVB TXWORD,WRIBYT ;SET DATA TO BE WRITTEN INTO REG 10
JSR PC,WRITLU ;LOAD DATA INTO REG 10
MOV (SP)+,REGNUM ;RESTORE LU REG NO.
RTS PC ;RETURN

;* STPLU - THIS SUBROUTINE CLOCKS THE LINE UNIT FOR THE NO. OF CYCLES PASSED
;* IN BITS 0-14 OF THE WORD FOLLOWING THE CALL.
;* IF BIT 15 = 1, A CHECK IS MADE TO DETERMINE IF THE USYRT CHIP TYPE
;* REQUIRES DECREMENTING THE NO. OF CYCLES BY 1.

STPLU: MOV R1,-(SP) ;SAVE R1
MOV @2(SP),R1 ;GET DESIRED NO. OF CYCLES
BEQ 6\$;IF DESIRED CYCLES = 0, RETURN
BPL 2\$;BR IF CHIP TYPE CHECK NOT NECESSARY
BIC #BIT15,R1 ;CLEAR FLAG BIT
TST CHPTYP ;SEE IF SIG USYRT
BEQ 2\$;BR IF YES
DEC R1 ;DECREMENT CYCLE COUNT
BISB #LULOOP,@BSEL1 ;SET LU LOOP BIT
BISB #STEPLU,@BSEL1 ;SET THE STEPLU BIT (CLOCK THE TRANSMITTER)
JSR PC,STALL ;STALL
BICB #STEPLU,@BSEL1 ;CLEAR THE STEPLU BIT (CLOCK THE RECEIVER)
JSR PC,STALL ;STALL
DEC R1 ;DECREMENT CYCLE COUNTER
BNE 3\$;BR IF NOT DONE YET
ADD #2,2(SP) ;FIX UP RETURN PC
MOV (SP)+,R1 ;RESTORE R1
RTS PC ;RETURN

6783
6784
6785
6786
6787
6788
6789
6790
6791
6792
6793 005324 013746 002400
6794 005330 013746 002352
6795 005334 005737 002352
6796 005340 001006
6797 005342 016637 000004 002352
6798 005350 162737 000004 002352
6799 005356 012737 000011 002400
6800 005364 004737 003644
6801 005370 032776 000001 000004
6802 005376 001413
6803 005400 132737 000100 002364
6804 005406 001031
6805 005410 004737 004500
6806
6807 005414
(4) 005414 104455
(5) 005416 000013
(5) 005420 012444
(5) 005422 015576
6808 005424 000412
6809 005426 132737 000100 002364
6810 005434 001416
6811 005436 004737 004500
6812
6813 005442
(4) 005442 104455
(5) 005444 000014
(5) 005446 012461
(5) 005450 015576
6814 005452 016637 000002 002400
6815 005460 013706 002346
6816 005464 013746 002362
6817 005470 000407
6818 005472 062766 000002 000004
6819 005500 012637 002352
6820 005504 012637 002400
6821 005510 000207
6822
6823
6824
6825
6826
6827
6828
6829
6830

;* OACTIV - THIS SUBROUTINE CHECKS FOR THE PROPER STATE OF OACT (REG 11) AND
;* REPORTS AN ERROR IF IT IS NOT PROPERLY SET TO THE STATE OF BIT 0 IN THE
;* WORD FOLLOWING THE CALL.

OACTIV: MOV REGNUM,-(SP) ;SAVE LU REG NO.
MOV SUBRPC,-(SP)
TST SUBRPC ;SEE IF THIS IS A NESTED CALL
BNE 1\$;BR IF YES
MOV 4(SP),SUBRPC
SUB #4,SUBRPC ;GET PC OF SUBROUTINE CALL
1\$: MOV #11,REGNUM ;SET REG NO. = 11
JSR PC,READLU ;READ REG 11
BIT #BIT0,@4(SP) ;GET EXPECTED STATE OF OACT
BEQ 3\$;BR IF EXPECTED OACT = 0
BITB #OACT,REDBYT ;SEE IF OACT = 1
BNE 9\$;BR IF OACT = 1
JSR PC,GETALL ;GET REGS FOR PRINTOUT
;REPORT OACT NOT SET
ERRDF 11,EM11,ERR4

TRAP C\$ERDF
.WORD 11
.WORD EM11
.WORD ERR4

BR 6\$;TAKE ERROR RETURN
3\$: BITB #OACT,REDBYT ;SEE IF OACT = 0
BEQ 9\$;BR IF OACT = 0
JSR PC,GETALL ;GET REGS FOR PRINTOUT
;REPORT OACT NOT CLEARED
ERRDF 12,EM12,ERR4

TRAP C\$ERDF
.WORD 12
.WORD EM12
.WORD ERR4

6\$: MOV 2(SP),REGNUM ;RESTORE LU REG NO.
MOV PSTACK,SP ;RESTORE PROGRAM STACK TO BASE LEVEL
MOV RETADR,-(SP) ;FIX UP ERROR RETURN PC
BR 12\$
9\$: ADD #2,4(SP) ;FIX UP ERROR-FREE RETURN PC
MOV (SP)+,SUBRPC
MOV (SP)+,REGNUM ;RESTORE LU REG NO.
12\$: RTS PC ;RETURN

;* INITRN - THIS SUBROUTINE INITIATES TRANSMISSION OF A MESSAGE, BY DOING A
;* MASTER CLEAR, LOADING AX2-15 AND REG 17 WITH THE DATA PASSED IN THE 2
;* WORDS FOLLOWING THE CALL, LOADING 2 SOM CHARS INTO THE TX SILO, AND

```

6831 ;*   CLOCKING THE LINE UNIT UNTIL THE FIRST SYNCH OR FLAG HAS BEEN SERIALIZED
6832 ;*   IN THE USYRT. THE PROGRAM MONITORS ORDY,OCOR, AND OACT FOR VALID STATES,
6833 ;*   THROUGHOUT THE PROCESS.
6834 ;*   IF THE SUBROUTINE DETECTS AN ERROR, A RETURN IS MADE TO THE TEST, AT THE
6835 ;*   ADDRESS CONTAINED IN RETADR.
6836 ;*****
6837 005512 010146      INITRN: MOV     R1,-(SP)      ;SAVE R1
6838 005514 013746 002400  MOV     REGNUM,-(SP)  ;SAVE LU REG NO.
6839 005520 013746 002402  MOV     AXNUM,-(SP)   ;SAVE AX BYTE NO.
6840 005524 016637 000006 002352  MOV     6(SP),SUBRPC
6841 005532 162737 000004 002352  SUB     #4,SUBRPC     ;GET PC OF SUBR CALL
6842 005540 004737 003576  JSR     PC,MSTCLR     ;ISSUE A MASTER CLEAR
6843 005544 004737 004634  JSR     PC,OSIRDY     ;CHECK ORDY=1, OCOR=0
6844 005550 000001      1
6845 005552 004737 005324  JSR     PC,OACTIV     ;CHK OACT=0
6846 005556 000000      0
6847 005560 012737 000004 002402  MOV     #4,AXNUM      ;SET AX BYTE NO. = 4 FOR AX2
6848 005566 117637 000006 002374  MOVVB  @6(SP),WAX15    ;SET DATA BYTE TO LOAD INTO AX2-15
6849 005574 012737 000400 002422  MOV     #TXSOM,TXWORD ;SET TSOM BIT
6850 005602 113737 002374 002422  MOVVB  WAX15,TXWORD   ;SET SYNCH CHAR
6851 005610 005037 002376  CLR     WAX16
6852 005614 004737 004264  JSR     PC,WRITAX     ;LOAD AX2
6853 005620 012737 000017 002400  MOV     #17,REGNUM    ;SET REG NO. = 17
6854 005626 062766 000002 000006  ADD     #2,6(SP)      ;INCR POINTER TO NEXT DATA BYTE
6855 005634 117637 000006 002366  MOVVB  @6(SP),WRIBYT  ;SET DATA BYTE TO LOAD INTO REG 17
6856 005642 004737 003722  JSR     PC,WRITLU     ;LOAD REG 17
6857 005646 004737 005146  JSR     PC,LDTXSI     ;LOAD THE SILO WITH SOM CHAR
6858 005652 004737 005146  JSR     PC,LDTXSI     ;LOAD ANOTHER SOM INTO SILO
6859 005656 004737 005120  JSR     PC,WAIT50     ;WAIT FOR DATA TO RIPPLE
6860 005662 004737 004634  JSR     PC,OSIRDY     ;CHK ORDY=1, OCOR=1
6861 005666 000003      3
6862 005670 004737 005324  JSR     PC,OACTIV     ;CHK FOR OACT = 0
6863 005674 000000      0
6864 005676 005001      CLR     R1            ;INIT CYCLE COUNTER
6865 005700 012737 000011 002400  MOV     #11,REGNUM    ;SET LU REG NO. = 11
6866 005706 152777 000010 174534 6$: BISB  #LULOOP,@BSEL1  ;SET LINE UNIT LOOP BIT
6867 005714 152777 000020 174526  BISB  #STEPLU,@BSEL1 ;SET CLOCK BIT
6868 005722 004737 005136  JSR     PC,STALL      ;STALL FOR MICRO-SEC
6869 005726 004737 003644  JSR     PC,READLU     ;READ REG 11
6870 005732 132737 000100 002364  BITB  #OACT,REDBYT   ;SEE IF OACT = 1 YET
6871 005740 001014      BNE     9$           ;BR IF OACT = 1
6872 005742 142777 000020 174500  BICB  #STEPLU,@BSEL1 ;CLEAR CLOCK BIT
6873 005750 004737 005136  JSR     PC,STALL      ;STALL FOR A MICRO-SEC
6874 005754 005201      INC     R1            ;INCR CYCLE COUNT
6875 005756 020127 000003  CMP     R1,#3         ;SEE IF 3 CYCLES DONE YET
6876 005762 002751      BLT     6$           ;BR IF NOT
6877 005764 004737 005324  JSR     PC,OACTIV     ;CHK FOR OACT = 1
6878 005770 000001      1
6879 005772 012737 000017 002400 9$: MOV     #17,REGNUM    ;SET REG NO. = 17
6880 006000 005037 002430  CLR     CHPTYP        ;CLEAR USYRT CHIP INDICATOR
6881 006004 004737 003644  JSR     PC,READLU     ;READ REG 17
6882 006010 132737 000020 002364  BITB  #OCOR,REDBYT   ;CHK FOR OCOR CLEARED YET
6883 006016 001403      BEQ     12$         ;BR IF YES - IT IS SIG CHIP
6884 006020 012737 000001 002430  MOV     #1,CHPTYP     ;SET INDICATOR FOR OTHER CHIP TYPE
6885 006026 142777 000020 174414 12$: BICB  #STEPLU,@BSEL1 ;CLEAR CLOCK BIT
6886 006034 004737 005136  JSR     PC,STALL      ;STALL FOR MICRO-SEC

```

```

6887 006040 004737 004634 JSR PC,OSIRDY ;CHK FOR ORDY = 1, OCOR = 0
6888 006044 000001 1 ADD #2,6(SP) ;FIX UP RETURN PC
6889 006046 062766 000002 000006 ADD (SP)+,AXNUM ;RESTORE AX BYTE NO.
6890 006054 012637 002402 MOV (SP)+,REGNUM ;RESTORE LU REG NO.
6891 006060 012637 002400 MOV (SP)+,R1 ;RESTORE R1
6892 006064 012601 MOV (SP)+,R1 ;RESTORE R1
6893 006066 005037 002352 CLR SUBRPC ;CLEAR SUBR CALL PC
6894 006072 000207 RTS PC ;RETURN
6895
6896
6897
6898
6899
6900

```

```

6901
6902
6903
6904
6905
6906
6907
6908
6909
6910

```

 * TXCHAR - THIS SUBROUTINE INITIATES TRANSMISSION OF A CHARACTER, BY LOADING
 * THE TX SILO WITH DATA PASSED IN BITS 0-11 OF THE WORD FOLLOWING THE CALL
 * AND CLOCKS THE LINE UNIT WITH THE NUMBER OF CYCLES PASSED IN BITS 0-14
 * OF THE SECOND WORD FOLLOWING THE CALL. IF BIT 15 = 1, A CHK IS MADE TO
 * DETERMINE IF THE USYRT CHIP TYPE REQUIRES DECREMENTING THE NO. OF CYCLES
 * BY 1. THE PROGRAM CHECKS FOR VALID STATES OF ORDY,
 * OCOR, AND OACT THROUGHOUT THE PROCESS.
 * IF AN ERROR IS DETECTED, A RETURN IS MADE TO THE TEST, AT THE ADDRESS
 * CONTAINED IN RETADR.

```

6911 006074 010146 TXCHAR: MOV R1,-(SP) ;SAVE R1
6912 006076 010246 MOV R2,-(SP) ;SAVE R2
6913 006100 016637 000004 002352 MOV 4(SP),SUBRPC
6914 006106 162737 000004 002352 SUB #4,SUBRPC ;GET PC OF SUBR CALL
6915 006114 017637 000004 002422 MOV @4(SP),TXWORD ;GET DATA TO BE TRANSMITTED
6916 006122 004737 005146 JSR PC,LDTXSI ;LOAD THE TX SILO WITH THE DATA
6917 006126 004737 005120 JSR PC,WAIT50 ;WAIT FOR DATA TO RIPPLE DOWN SILO
6918 006132 062766 000002 000004 ADD #2,4(SP) ;INCR POINTER
6919 006140 005001 CLR R1 ;INIT CYCLE COUNT
6920 006142 017602 000004 MOV @4(SP),R2 ;GET DESIRED NO. OF CYCLES
6921 006146 005702 TST R2 ;SEE IF CHIP TYPE CHK SHOULD BE MADE
6922 006150 100006 BPL 9$ ;BR IF NOT
6923 006152 042702 100000 BIC #BIT15,R2 ;CLEAR FLAG BIT
6924 006156 005737 002430 TST CHPTYP ;SEE IF SIG USYRT
6925 006162 001401 BEQ 9$ ;BR IF YES
6926 006164 005302 DEC R2 ;DECREMENT NO. OF CYCLES
6927 006166 004737 005324 9$: JSR PC,OACTIV ;CHK OACT = 1
6928 006172 000001 1 CMP R1,R2 ;SEE IF REQUIRED CYCLES DONE YET
6929 006174 020102 BEQ 12$ ;BR IF YES
6930 006176 001410 JSR PC,OSIRDY ;CHK ORDY=1, OCOR=1
6931 006200 004737 004634 3 JSR PC,STPLU ;STEP LU ONE CYCLE
6932 006204 000003 1 INC R1 ;INCR CYCLE COUNT
6933 006206 004737 005226 BR 9$
6934 006212 000001 12$: JSR PC,OSIRDY ;CHK ORDY=1, OCOR=0
6935 006214 005201 1 ADD #2,4(SP) ;FIX UP RETURN PC
6936 006216 000763 CLR SUBRPC ;CLEAR SUBR CALL PC
6937 006220 004737 004634 12$: MOV (SP)+,R2 ;RESTORE R2
6938 006224 000001 1 MOV (SP)+,R1 ;RESTORE R1
6939 006226 062766 000002 000004
6940 006234 005037 002352
6941 006240 012602
6942 006242 012601

```

6943 006244 000207

RTS PC ;RETURN

6944
6945
6946
6947
6948
6949

6950
6951
6952

* ENDTRN - THIS SUBROUTINE CLEARS THE TRANSMITTER BY SETTING OC. THE PROGRAM
* WAITS FOR 50 US, AND CHECKS FOR ORDY=1, OCOR=0, OACT=0, RTS=0.

6953 006246 010146
6954 006250 013746 002400
6955 006254 016637 000004 002352
6956 006262 162737 000004 002352
6957 006270 012737 000011 002400
6958 006276 012737 000200 002366
6959 006304 004737 003722
6960 006310 004737 005120
6961 006314 004737 004634
6962 006320 000001
6963 006322 004737 005324
6964 006326 000000
6965 006330 012737 000013 002400
6966 006336 004737 003644
6967 006342 032737 000040 002364
6968 006350 001406
6969 006352 004737 004500

ENDTRN: MOV R1,-(SP) ;SAVE R1
MOV REGNUM,-(SP) ;SAVE LU REG NO.
MOV 4(SP),SUBRPC
SUB #4,SUBRPC ;GET PC OF SUBROUTINE CALL
MOV #11,REGNUM ;SET LU REG NO. = 11
MOV #OC,WRIBYT ;SET OC IN DATA
JSR PC,WRITLU ;SET OC IN REG 11
JSR PC,WAIT50 ;STALL FOR >50 US.
JSR PC,OSIRDY ;CHK ORDY=1, OCOR=0
1
JSR PC,OACTIV ;CHK OACT = 0
0
MOV #13,REGNUM ;SET REG NO. = 13
JSR PC,READLU ;READ REG 13
BIT #RTS,REDBYT ;CHK FOR RTS = 0
BEQ 3\$;BR IF RTS = 0
JSR PC,GETALL ;GET REGS FOR PRINTOUT

6970
6971
(4)
(5)
(5)
(5)

006356 104455
006360 000101
006362 014220
006364 015576

;REPORT RTS NOT CLEARED
ERRDF 65,EM65,ERR4

TRAP C\$ERDF
.WORD 65
.WORD EM65
.WORD ERR4

6972 006366 005037 002352
6973 006372 012637 002400
6974 006376 012601
6975 006400 000207

3\$: CLR SUBRPC ;CLEAR SUBR CALL PC
MOV (SP)+,REGNUM ;RESTORE LU REG NO.
MOV (SP)+,R1 ;RESTORE R1
RTS PC ;RETURN

6976
6977
6978
6979
6980
6981

6982
6983
6984
6985
6986
6987
6988

* ISIRDY - THIS SUBROUTINE CHECKS FOR THE PROPER STATES OF ICIR (REG 17)
* AND IRDY (REG 12) AND REPORTS AN ERROR IF EITHER IS NOT PROPERLY SET
* AS PASSED IN BIT 0 (ICIR) AND BIT 1 (IRDY) OF THE WORD FOLLOWING THE
* CALL.
* IF AN ERROR OCCURS, A RETURN IS MADE TO THE TEST AT THE ADDRESS
* IN RETADR.

6989 006402 013746 002400
6990 006406 013746 002352
6991 006412 005737 002352
6992 006416 001006
6993 006420 016637 000004 002352
6994 006426 162737 000004 002352

ISIRDY: MOV REGNUM,-(SP) ;SAVE LU REG NO.
MOV SUBRPC,-(SP)
TST SUBRPC ;SEE IF THIS IS A NESTED CALL
BNE 1\$;BR IF YES
MOV 4(SP),SUBRPC
SUB #4,SUBRPC ;GET PC OF SUBR CALL


```

6995 006434 012737 000012 002400 1$:  MOV    #12,REGNUM      ;SET REG NO. TO 12
6996 006442 004737 003644          JSR    PC,READLU      ;READ REG 12
6997 006446 032776 000002 000004    BIT    #BIT1,@4(SP)   ;GET EXPECTED STATE OF IRDY
6998 006454 001413          BEQ    3$             ;BR IF EXPECTED IRDY = 0
6999 006456 132737 000020 002364    BITB   #IRDY,REDBYT   ;SEE IF IRDY = 1
7000 006464 001022          BNE    9$             ;BR IF IRDY = 1
7001 006466 004737 004500          JSR    PC,GETALL     ;GET REGS FOR PRINTOUT
7002          ;REPORT IRDY NOT SET
7003          ERRDF 17,EM17,ERR4
7004          (4) 006472 104455          TRAP  C$ERDF
7005          (5) 006474 000021          .WORD 17
7006          (5) 006476 012617          .WORD EM17
7007          (5) 006500 015576          .WORD ERR4
7008 006502 000451          BR     16$           ;TAKE ERROR EXIT
7009 006504 132737 000020 002364 3$:  BITB   #IRDY,REDBYT   ;SEE IF IRDY = 0
7010 006512 001407          BEQ    9$             ;BR IF IRDY = 0
7011 006514 004737 004500          JSR    PC,GETALL     ;GET REGS FOR PRINTOUT
7012          ;REPORT IRDY NOT CLEARED
7013          ERRDF 18,EM18,ERR4
7014          (4) 006520 104455          TRAP  C$ERDF
7015          (5) 006522 000022          .WORD 18
7016          (5) 006524 012634          .WORD EM18
7017          (5) 006526 015576          .WORD ERR4
7018 006530 000436          BR     16$           ;TAKE ERROR RETURN
7019 006532 012737 000017 002400 9$:  MOV    #17,REGNUM      ;SET REG NO. = 17
7020 006540 004737 003644          JSR    PC,READLU      ;READ REG 17
7021 006544 132776 000001 000004    BITB   #BIT0,@4(SP)   ;GET EXPECTED STATE OF ICIR
7022 006552 001413          BEQ    12$            ;BR IF EXPECTED ICIR = 0
7023 006554 132737 000010 002364    BITB   #ICIR,REDBYT   ;SEE IF ICIR = 1
7024 006562 001031          BNE    20$            ;BR IF ICIR = 1
7025 006564 004737 004500          JSR    PC,GETALL     ;GET REGS FOR PRINTOUT
7026          ;REPORT ICIR NOT SET
7027          ERRDF 19,EM19,ERR4
7028          (4) 006570 104455          TRAP  C$ERDF
7029          (5) 006572 000023          .WORD 19
7030          (5) 006574 012655          .WORD EM19
7031          (5) 006576 015576          .WORD ERR4
7032 006600 000412          BR     16$           ;TAKE ERROR RETURN
7033 006602 132737 000010 002364 12$: BITB   #ICIR,REDBYT   ;SEE IF ICIR = 0
7034 006610 001416          BEQ    20$            ;BR IF ICIR = 0
7035 006612 004737 004500          JSR    PC,GETALL     ;GET REGS FOR PRINTOUT
7036          ;REPORT ICIR NOT CLEARED
7037          ERRDF 20,EM20,ERR4
7038          (4) 006616 104455          TRAP  C$ERDF
7039          (5) 006620 000024          .WORD 20
7040          (5) 006622 012672          .WORD EM20
7041          (5) 006624 015576          .WORD ERR4
7042 006626 016637 000002 002400 16$: MOV    2(SP),REGNUM    ;RESTORE LU REG NO.
7043 006634 013706 002346          MOV    PSTACK,SP     ;RESTORE STACK POINTER TO BASE LEVEL
7044 006640 013746 002362          MOV    RETADR,-(SP)  ;FIX ERROR RETURN PC
7045 006644 000407          BR     23$
7046 006646 062766 000002 000004 20$: ADD    #2,4(SP)       ;FIX UP ERROR-FREE RETURN PC
7047 006654 012637 002352          MOV    (SP)+,SUBRPC
7048 006660 012637 002400          MOV    (SP)+,REGNUM
7049 006664 000207          23$: RTS    PC        ;RESTORE LU REG NO.
7050          ;RETURN
  
```

```
7035
7036
7037
7038
7039
7040
7041
7042
7043
7044
7045
7046 006666 013746 002400
7047 006672 013746 002352
7048 006676 005737 002352
7049 006702 001006
7050 006704 016637 000004 002352
7051 006712 162737 000004 002352
7052 006720 012737 000012 002400 1$:
7053 006726 004737 003644
7054 006732 032776 000001 000004
7055 006740 001413
7056 006742 132737 000100 002364
7057 006750 001031
7058 006752 004737 004500
7059
7060 006756
(4) 006756 104455
(5) 006760 000025
(5) 006762 012713
(5) 006764 015576
7061 006766 000412
7062 006770 132737 000100 002364 3$:
7063 006776 001416
7064 007000 004737 004500
7065
7066 007004
(4) 007004 104455
(5) 007006 000026
(5) 007010 012730
(5) 007012 015576
7067 007014 016637 000002 002400 6$:
7068 007022 013706 002346
7069 007026 013746 002362
7070 007032 000407
7071 007034 062766 000002 000004 9$:
7072 007042 012637 002352
7073 007046 012637 002400
7074 007052 000207 12$:
7075
7076
7077
7078
7079
7080
7081
7082

;*****
;* IACTIV - THIS SUBROUTINE CHECKS FOR THE PROPER STATE OF IACT (REG 12) AND
;* REPORTS AN ERROR IF IT IS NOT PROPERLY SET TO THE STATE OF BIT 0 IN THE
;* WORD FOLLOWING THE CALL.
;* IF AN ERROR OCCURS, A RETURN IS MADE TO THE TEST AT THE ADDRESS IN
;* RETADR.
;*****
IACTIV: MOV REGNUM,-(SP) ;SAVE LU REG NO.
MOV SUBRPC,-(SP)
TST SUBRPC ;SEE IF THIS IS A NESTED CALL
BNE 1$ ;BR IF YES
MOV 4(SP),SUBRPC
SUB #4,SUBRPC ;GET PC OF SUBR CALL
1$: MOV #12,REGNUM ;SET REG NO. = 12
JSR PC,READLU ;READ REG 12
BIT #BIT0,@4(SP) ;GET EXPECTED STATE OF IACT
BEQ 3$ ;BR IF EXPECTED IACT = 0
BITB #IACT,REDBYT ;SEE IF IACT = 1
BNE 9$ ;BR IF IACT = 1
JSR PC,GETALL ;GET REGS FOR PRINTOUT
;REPORT IACT NOT SET
ERRDF 21,EM21,ERR4
TRAP C$ERDF
.WORD 21
.WORD EM21
.WORD ERR4
BR 6$ ;TAKE ERROR EXIT
BITB #IACT,REDBYT ;SEE IF IACT = 0
BEQ 9$ ;BR IF IACT = 0
JSR PC,GETALL ;GET REGS FOR PRINTOUT
;REPORT IACT NOT CLEARED
ERRDF 22,EM22,ERR4
TRAP C$ERDF
.WORD 22
.WORD EM22
.WORD ERR4
6$: MOV 2(SP),REGNUM ;RESTORE LU REG NO.
MOV PSTACK,SP ;RESTORE PROGRAM STACK TO BASE LEVEL
MOV RETADR,-(SP) ;FIX UP ERROR RETURN PC
BR 12$
9$: ADD #2,4(SP) ;FIX UP ERROR-FREE RETURN PC
MOV (SP)+,SUBRPC
MOV (SP)+,REGNUM ;RESTORE LU REG NO.
12$: RTS PC ;RETURN

;*****
;* RSEOM - THIS SUBROUTINE CHECKS FOR THE PROPER STATES OF RSOM AND REOM IN
;* AX0-16, AND REPORTS AN ERROR IF EITHER IS NOT SET TO THE STATE PASSED IN BITS
```

```
7083 ;* 0,1, RESPECTIVELY, OF THE WORD FOLLOWING THE CALL.
7084 ;* IF AN ERROR OCCURS, A RETURN IS MADE TO THE TEST AT THE ADDRESS IN RETADR.
7085 ;*****
7086 007054 013746 002402 RSEOM: MOV AXNUM,-(SP) ;SAVE AX BYTE NO.
7087 007060 013746 002352 MOV SUBRPC,-(SP)
7088 007064 005737 002352 TST SUBRPC ;SEE IF THIS IS A NESTED CALL
7089 007070 001006 BNE 1$ ;BR IF YES
7090 007072 016637 000004 002352 MOV 4(SP),SUBRPC
7091 007100 162737 000004 002352 SUB #4,SUBRPC ;GET PC OF SUBR CALL
7092 007106 012737 000001 002402 1$: MOV #1,AXNUM ;SET AX BYTE NO. FOR AX0-16
7093 007114 004737 004076 JSR PC,READAX ;READ AX0
7094 007120 032776 000001 000004 BIT #BIT0,@4(SP) ;GET EXPECTED STATE OF RSOM
7095 007126 001413 BEQ 3$ ;BR IF EXPECTED RSOM = 0
7096 007130 132737 000001 002372 BITB #RSOM,RAX16 ;SEE IF RSOM = 1
7097 007136 001022 BNE 9$ ;BR IF RSOM = 1
7098 007140 004737 004500 JSR PC,GETALL ;GET REGS FOR PRINTOUT
7099 ;REPORT RSOM NOT SET
7100 007144 ERRDF 29,EM29,ERR6
(4) 007144 104455 TRAP C$ERDF
(5) 007146 000035 .WORD 29
(5) 007150 013147 .WORD EM29
(5) 007152 016766 .WORD ERR6
7101 007154 000444 BR 16$ ;TAKE ERROR EXIT
7102 007156 132737 000001 002372 3$: BITB #RSOM,RAX16 ;SEE IF RSOM = 0
7103 007164 001407 BEQ 9$ ;BR IF RSOM = 0
7104 007166 004737 004500 JSR PC,GETALL ;GET REGS FOR PRINTOUT
7105 ;REPORT RSOM NOT CLEARED
7106 007172 ERRDF 28,EM28,ERR6
(4) 007172 104455 TRAP C$ERDF
(5) 007174 000034 .WORD 28
(5) 007176 013126 .WORD EM28
(5) 007200 016766 .WORD ERR6
7107 007202 000431 BR 16$ ;TAKE ERROR RETURN
7108 007204 132776 000002 000004 9$: BITB #BIT1,@4(SP) ;GET EXPECTED STATE OF REOM
7109 007212 001413 BEQ 12$ ;BR IF EXPECTED REOM = 0
7110 007214 132737 000002 002372 BITB #REOM,RAX16 ;SEE IF REOM = 1
7111 007222 001031 BNE 20$ ;BR IF REOM = 1
7112 007224 004737 004500 JSR PC,GETALL ;GET REGS FOR PRINTOUT
7113 ;REPORT REOM NOT SET
7114 007230 ERRDF 31,EM31,ERR6
(4) 007230 104455 TRAP C$ERDF
(5) 007232 000037 .WORD 31
(5) 007234 013205 .WORD EM31
(5) 007236 016766 .WORD ERR6
7115 007240 000412 BR 16$ ;TAKE ERROR RETURN
7116 007242 132737 000002 002372 12$: BITB #REOM,RAX16 ;SEE IF REOM = 0
7117 007250 001416 BEQ 20$ ;BR IF REOM = 0
7118 007252 004737 004500 JSR PC,GETALL ;GET REGS FOR PRINTOUT
7119 ;REPORT REOM NOT CLEARED
7120 007256 ERRDF 30,EM30,ERR6
(4) 007256 104455 TRAP C$ERDF
(5) 007260 000036 .WORD 30
(5) 007262 013164 .WORD EM30
(5) 007264 016766 .WORD ERR6
7121 007266 016637 000002 002402 16$: MOV 2(SP),AXNUM ;RESTORE AX BYTE NO.
7122 007274 013706 002346 MOV PSTACK,SP ;RESTORE STACK POINTER TO BASE LEVEL
```

```
7123 007300 013746 002362      MOV    RETADR,-(SP)    ;FIX ERROR RETURN PC
7124 007304 000407                BR      23$
7125 007306 062766 000002 000004 20$:  ADD    #2,4(SP)        ;FIX UP ERROR-FREE RETURN PC
7126 007314 012637 002352                MOV    (SP)+,SUBRPC
7127 007320 012637 002402                MOV    (SP)+,AXNUM    ;RESTORE AX BYTE NO.
7128 007324 000207 23$:   RTS      PC        ;RETURN
```

```
7129
7130
7131
7132
7133
7134
```

```
*****
;* RDRXSI - THIS SUBROUTINE READS THE RCV SILO (REGS 10,12) AND RETURNS THE
;* SILO ENTRY IN BITS 0-11 OF RXWORD.
*****
```

```
7137
7138 007326 013746 002400      RDRXSI: MOV    REGNUM,-(SP)    ;SAVE LU REG NO.
7139 007332 012737 000012 002400      MOV    #12,REGNUM     ;SET REG NO. = 12
7140 007340 004737 003644      JSR    PC,READLU     ;READ LU REG 12
7141 007344 113737 002364 002425      MOV    REDBYT,RXWORD+1 ;GET HI BITS OF SILO ENTRY
7142 007352 042737 170000 002424      BIC    #170000,RXWORD  ;CLEAR UNUSED BITS
7143 007360 012737 000010 002400      MOV    #10,REGNUM    ;SET REG NO. = 10
7144 007366 004737 003644      JSR    PC,READLU     ;READ REG 10
7145 007372 113737 002364 002424      MOV    REDBYT,RXWORD  ;GET LOW BITS OF SILO ENTRY
7146 007400 012637 002400      MOV    (SP)+,REGNUM  ;RESTORE LU REG NO.
7147 007404 000207      RTS      PC        ;RETURN
```

```
7148
7149
7150
7151
7152
7153
```

```
*****
;* RCV1ST - THIS SUBROUTINE RECEIVES THE FIRST CHAR OF A MESSAGE, AND MONITORS
;* STATUS OF THE RECEIVER. FIRST, A CHECK IS MADE FOR IACT = 0, IRDY = 0,
;* ICIR = 1, AND RSOM = 0. THEN, THE LINE UNIT IS CLOCKED USING
;* STEPLU UNTIL IRDY = 1. THE PROGRAM CHECKS FOR THIS TO OCCUR WITHIN 3
;* CYCLES AFTER THE NO. OF CYCLES PASSED IN THE WORD FOLLOWING THE CALL.
;* IF AN ERROR OCCURS, A RETURN IS MADE TO THE TEST, AT THE ADDRESS
;* CONTAINED IN RETADR.
*****
```

```
7161
7162 007406 010146      RCV1ST: MOV    R1,-(SP)    ;SAVE R1
7163 007410 010346      MOV    R3,-(SP)    ;SAVE R3
7164 007412 013746 002400      MOV    REGNUM,-(SP) ;SAVE LU REG NO.
7165 007416 016637 000006 002352      MOV    6(SP),SUBRPC
7166 007424 162737 000004 002352      SUB    #4,SUBRPC    ;GET PC OF SUBROUTINE CALL
7167 007432 012737 000012 002400      MOV    #12,REGNUM  ;SET LU REG NO. = 12
7168 007440 005001      CLR    R1          ;INIT CYCLE COUNT TO 0
7169 007442 017603 000006      MOV    @6(SP),R3   ;GET CYCLE COUNT LIMIT
7170 007446 062703 000003      ADD    #3,R3
7171 007452 005776 000006      TST    @6(SP)     ;SEE IF DESIRED CYCLES - 0
7172 007456 001414      BEQ    8$         ;BR IF YES
7173 007460 004737 006666      JSR    PC,IACTIV  ;CHK FOR IACT = 0
7174 007464 000000      0
7175 007466 004737 006402      JSR    PC,ISIRDY  ;CHK FOR ICIR = 1, IRDY = 0
7176 007472 000001      1
7177 007474 004737 007054      JSR    PC,RSEOM   ;CHK RSOM = 0, REOM = 0 IN AX0-16
7178 007500 000000      0
```

```

7179 007502 004737 005226      6$: JSR PC,STPLU      ;CLOCK LU FOR 1 CYCLE
7180 007506 000001              1
7181 007510 004737 005120      8$: JSR PC,WAIT50     ;ALLOW SILO DATA TO RIPPLE
7182 007514 005201              INC R1              ;INCREMENT CYCLE COUNT
7183 007516 004737 003644      JSR PC,READLU      ;READ REG 12
7184 007522 132737 000020 002364 BITB #IRDY,REDBYT   ;SEE IF IRDY = 1 YET
7185 007530 001005              BNE 9$             ;BR IF IRDY = 1
7186 007532 020103              CMP R1,R3          ;SEE IF LIMIT EXCEEDED
7187 007534 002762              BLT 6$             ;BR IF NOT YET
7188 007536 004737 006402      JSR PC,ISIRDY     ;CHK FOR ICIR = 1, IRDY = 1
7189 007542 000003              3
7190 007544 020176 000006      9$: CMP R1,@6(SP)    ;SEE IF LESS THAN REQUIRED CYCLES
7191 007550 002003              BGE 12$           ;BR IF NOT
7192 007552 004737 006402      JSR PC,ISIRDY     ;CHK FOR ICIR = 1, IRDY = 0
7193 007556 000001              1
7194 007560 004737 006666      12$: JSR PC,IACTIV    ;CHK FOR IACT = 1
7195 007564 000001              1
7196 007566 004737 006402      JSR PC,ISIRDY     ;CHK FOR ICIR = 1, IRDY = 1
7197 007572 000003              3
7198 007574 062766 000002 000006 ADD #2,6(SP)       ;FIX UP RETURN PC
7199 007602 012637 002400      MOV (SP)+,REGNUM  ;RESTORE LU REG NO.
7200 007606 012603              MOV (SP)+,R3      ;RESTORE R3
7201 007610 012601              MOV (SP)+,R1      ;RESTORE R1
7202 007612 005037 002352      CLR SUBRPC        ;CLEAR SUBR CALL PC
7203 007616 000207              RTS PC            ;RETURN

```

```

7204
7205
7206
7207
7208
7209
7210
7211
7212
7213
7214

```

 ;* STPERR - THIS SUBROUTINE LOADS THE CONTENTS OF THE FIRST WORD FOLLOWING THE
 ;* CALL INTO REG 17, AND SETS THE IERR BIT, AND CLOCKS THE LINE UNIT
 ;* FOR THE NO. OF CYCLES PASSED IN THE 2ND WORD FOLLOWING THE CALL. THEN,
 ;* IT RESTORES REG 17 TO ITS ORIGINAL CONTENTS, CLEARING THE IERR BIT.

```

7215 007620 013746 002400      STPERR: MOV REGNUM,-(SP) ;SAVE LU REG NO.
7216 007624 012737 000017 002400 MOV #17,REGNUM     ;SET LU REG NO. = 17
7217 007632 017637 000002 002366 MOV @2(SP),WRIBYT
7218 007640 152737 000002 002366 BISB #IERR,WRIBYT
7219 007646 004737 003722      JSR PC,WRITLU     ;SET IERR BIT IN REG 17
7220 007652 062766 000002 000002 ADD #2,2(SP)       ;INCREMENT SUBR ARGUMENT POINTER
7221 007660 017637 000002 007672 MOV @2(SP),3$     ;GET DESIRED NO. OF CYCLES
7222 007666 004737 005226      JSR PC,STPLU     ;CLOCK LU FOR DESIRED NO. OF CYCLES
7223 007672 000000              .WORD 0          ;NO. OF CYCLES GOES HERE
7224 007674 142737 000002 002366 BICB #IERR,WRIBYT
7225 007702 004737 003722      JSR PC,WRITLU     ;CLEAR IERR BIT IN REG 17
7226 007706 062766 000002 000002 ADD #2,2(SP)       ;FIX UP RETURN PC
7227 007714 012637 002400      MOV (SP)+,REGNUM  ;RESTORE LU REG NO.
7228 007720 000207              RTS PC            ;RETURN

```

```

7229
7230
7231
7232
7233
7234

```

```

7235      ;* CKDATA - THIS SUBROUTINE READS THE RCV SILO AND COMPARES THE SILO ENTRY
7236      ;* TO BITS 0-11 OF THE FIRST WORD FOLLOWING THE CALL. IF THERE IS A
7237      ;* MISMATCH, THE ERROR IS REPORTED AND A RETURN IS MADE TO THE TEST AT THE
7238      ;* ADDRESS CONTAINED IN RETADR. IF BIT 15 = 0 IN THE FIRST WORD
7239      ;* FOLLOWING THE CALL, THE SUBROUTINE WILL NOT CHECK THE BCC BIT (SILO
7240      ;* BIT 8). IF THERE ARE NO ERRORS, THE LINE UNIT IS CLOCKED FOR THE
7241      ;* NUMBER OF CYCLES PASSED IN THE SECOND WORD FOLLOWING THE CALL.
7242      ;*****
7243      CKDATA: MOV     R1,-(SP)           ;SAVE R1
7244      MOV     REGNUM,-(SP)         ;SAVE LU REG NO.
7245      MOV     4(SP),SUBRPC
7246      SUB     #4,SUBRPC           ;GET PC OF SUBR CALL
7247      MOV     @4(SP),R1          ;GET EXPECTED SILO ENTRY
7248      BIC     #170000,R1         ;CLEAR UNUSED BITS FOR COMPARE
7249      JSR     PC,RDRXSI          ;READ RCV SILO
7250      CMP     SAVLEN,#TXLEN2!TXLEN1!TXLENO!RXLEN2!RXLEN1!RXLENO
7251      BNE     4$
7252      BIC     #BIT7,R1           ;BR IF CHAR LENGTH NOT = 7
7253      BIC     #BIT7,RXWORD       ;MASK OFF BIT 8TH BIT
7254      4$:  CMPB    R1,RXWORD       ;COMPARE EXPECTED BITS 0-7 TO ACTUAL
7255      BEQ     6$
7256      CLR     GOODAT
7257      MOVB   R1,GOODAT          ;GET EXPECTED DATA
7258      CLR     BADDAT
7259      MOVB   RXWORD,BADDAT      ;GET ACTUAL DATA
7260      MOV     #11,REGNUM         ;SET REG NO. = 11
7261      JSR     PC,READLU          ;READ REG 11
7262      BITB   #UNRR,REDBYT       ;SEE IF TX UNDERRUN ERROR
7263      BEQ     5$
7264      JSR     PC,GETALL          ;GET REGS FOR PRINTOUT
7265      ;REPORT TX UNDERRUN ERROR
7266      ERRDF  54,EM54,ERR4
7266      (4) 010060 104455
7266      (5) 010062 000066
7266      (5) 010064 014162
7266      (5) 010066 015576
7267      (4) 010070 000137 010552
7268      (5) 010074 012737 000010 002400
7269      (5) 010102 004737 004500
7270      ;REPORT RCV'D DATA MISCOMPARE
7271      ERRDF  34,EM34,ERR8
7271      (4) 010106 104455
7271      (5) 010110 000042
7271      (5) 010112 013274
7271      (5) 010114 020076
7272      (4) 010116 000137 010552
7273      (5) 010122 000301
7274      (5) 010124 012737 000012 002400
7275      (5) 010132 120137 002425
7276      (5) 010136 001002
7277      (5) 010140 000137 010526
7278      (5) 010144 005037 002404
7279      (5) 010150 110137 002404
7280      (5) 010154 005037 002406
7281      (5) 010160 113737 002425 002406
7282      (5) 010166 032776 100000 000004
7282      BIT     #BCCCHK,@4(SP)    ;SEE IF BCC SHOULD BE IGNORED
  
```

```

TRAP  C$ERDF
.WORD 54
.WORD EM54
.WORD ERR4

TRAP  C$ERDF
.WORD 34
.WORD EM34
.WORD ERR8
  
```



```
(5) 010400 013414
(5) 010402 020076
7321 010404 000137 010552
7322 010410 132737 000004 002425 16$: JMP 36$ ;TAKE ERROR EXIT
7323 010416 001010 :BITB #RAB,RXWORD+1 ;SEE IF ACTUAL BIT = 1
7324 010420 004737 004500 :BNE 18$ ;BR IF YES
7325 :JSR PC,GETALL ;GET REGS FOR PRINTOUT
7326 010424 ;REPORT RAB NOT SET
ERRDF 40,EM40,ERR8
(4) 010424 104455 TRAP C$ERDF
(5) 010426 000050 .WORD 40
(5) 010430 013434 .WORD EM40
(5) 010432 020076 .WORD ERR8
7327 010434 000137 010552
7328 010440 18$: JMP 36$ ;TAKE ERROR EXIT
7329 010440 132701 000010 :BITB #OVRR,R1 ;SEE IF EXPECTED BIT - 1
7330 010444 001014 :BNE 20$ ;BR IF YES
7331 010446 132737 000010 002425 :BITB #OVRR,RXWORD+1 ;SEE IF ACTUAL BIT = 0
7332 010454 001424 :BEQ 22$ ;BR IF YES
7333 010456 004737 004500 :JSR PC,GETALL ;GET REGS FOR PRINTOUT
7334 :REPORT OVRR NOT CLEARED
ERRDF 41,EM41,ERR8
(4) 010462 104455 TRAP C$ERDF
(5) 010464 000051 .WORD 41
(5) 010466 013450 .WORD EM41
(5) 010470 020076 .WORD ERR8
7336 010472 000137 010552
7337 010476 132737 000010 002425 20$: JMP 36$ ;TAKE ERROR EXIT
7338 010504 001010 :BITB #OVRR,RXWORD+1 ;SEE IF ACTUAL BIT 1
7339 010506 004737 004500 :BNE 22$ ;BR IF YES
7340 :JSR PC,GETALL ;GET REGS FOR PRINTOUT
7341 :REPORT OVRR NOT SET
ERRDF 42,EM42,ERR8
(4) 010512 104455 TRAP C$ERDF
(5) 010514 000052 .WORD 42
(5) 010516 013471 .WORD EM42
(5) 010520 020076 .WORD ERR8
7342 010522 000137 010552
7343 010526 22$: JMP 36$ ;TAKE ERROR EXIT
7344 010526 062766 000002 000004 :ADD #2,4(SP) ;INCR SUBROUTINE ARGUMENT POINTER
7345 010534 017637 000004 010546 :MOV @4(SP),24$ ;GET DESIRED CYCLE COUNT
7346 010542 004737 005226 :JSR PC,STPLU ;CLOCK LU FOR DESIRED CYCLES
7347 010546 000000 24$: .WORD 0
7348 010550 000407 38$: BR 38$ ;TAKE ERROR-FREE EXIT
7349 010552 011637 002400 :MOV (SP),REGNUM ;RESTORE LU REG NO.
7350 010556 013706 002346 :MOV PSTACK,SP ;RESTORE PROGRAM STACK TO BASE LEVEL
7351 010562 013746 002362 :MOV RETADR,-(SP) ;FIX UP ERROR RETURN PC
7352 010566 000406 40$: BR 40$
7353 010570 062766 000002 000004 38$: ADD #2,4(SP) ;FIX UP ERROR-FREE RETURN PC
7354 010576 012637 002400 :MOV (SP)+,REGNUM ;RESTORE LU REG NO.
7355 010602 012601 :MOV (SP)+,R1 ;RESTORE R1
7356 010604 005037 002352 40$: CLR SUBRPC ;CLEAR SUBROUTINE PC
7357 010610 000207 :RTS PC ;RETURN
7358
7359
7360
7361
7362
```


7363
7364
7365
7366
7367
7368 010612 013746 002402
7369 010616 013746 002400
7370 010622 012737 000004 002402
7371 010630 017637 000004 002374
7372 010636 005037 002376
7373 010642 004737 004264
7374 010646 012737 000017 002400
7375 010654 062766 000002 000004
7376 010662 017637 000004 002366
7377 010670 004737 003722
7378 010674 012737 000006 002402
7379 010702 062766 000002 000004
7380 010710 017637 000004 002374
7381 010716 062766 000002 000004
7382 010724 017637 000004 002376
7383 010732 013737 002376 002432
7384 010740 004737 004264
7385 010744 062766 000002 000004
7386 010752 012637 002400
7387 010756 012637 002402
7388 010762 005037 002352
7389 010766 000207

```
*****  
: * SETUP - THIS SUBROUTINE LOADS THE FIRST WORD AFTER THE CALL INTO AX2-15  
: * (SYNCH CHAR), LOADS THE SECOND WORD AFTER THE CALL INTO REG 17  
: * LOADS THE THIRD WORD INTO AX3-15, AND LOADS THE FOURTH INTO AX3-16.  
*****  
SETUP: MOV AXNUM, -(SP) ;SAVE AX BYTE NO.  
MOV REGNUM, -(SP) ;SAVE LU REG NO.  
MOV #4, AXNUM ;SET AX BYTE NO. FOR AX2  
MOV @4(SP), WAX15  
CLR WAX16  
JSR PC, WRITAX ;SET SYNCH CHAR IN AX2-15, CLEAR AX2-16  
MOV #17, REGNUM ;SET LU REG NO. = 17  
ADD #2, 4(SP) ;INCREMENT ARGUMENT POINTER  
MOV @4(SP), WRIBYT  
JSR PC, WRITLU ;LOAD REG 17  
MOV #6, AXNUM ;SET AX BYTE NO. FOR AX3  
ADD #2, 4(SP) ;INCREMENT ARGUMENT POINTER  
MOV @4(SP), WAX15 ;INCR ARGUMENT POINTER  
ADD #2, 4(SP)  
MOV @4(SP), WAX16  
JSR PC, WRITAX ;STORE TX AND RCV CHAR LENGTH  
ADD #2, 4(SP) ;LOAD AX3-15, AX3-16  
MOV (SP)+, REGNUM ;FIX RETURN PC  
MOV (SP)+, AXNUM ;RESTORE LU REG NO.  
CLR SUBRPC ;RESTORE AX BYTE NO.  
RTS PC ;CLEAR SUBROUTINE PC STORAGE  
 ;RETURN
```

7390
7391
7392
7393
7394
7395
7396
7397
7398
7399
7400 010770 010146
7401 010772 010246
7402 010774 017601 000004
7403 011000 062766 000002 000004
7404 011006 017602 000004
7405 011012 062766 000002 000004
7406 011020 012137 002422
7407 011024 004737 005146
7408 011030 005302
7409 011032 001372
7410 011034 004737 005120
7411 011040 012602
7412 011042 012601
7413 011044 000207
7414
7415
7416
7417
7418

```
*****  
: * LODMSG - THIS SUBROUTINE LOADS THE NO. OF WORDS PASSED IN THE SECOND WORD  
: * FOLLOWING THE CALL FROM THE MSG BUFFER WHOSE ADDRESS IS IN THE FIRST  
: * WORD FOLLOWING THE CALL, INTO THE TRANSMITTER SILO.  
*****  
LODMSG: MOV R1, -(SP) ;SAVE R1  
MOV R2, -(SP) ;SAVE R2  
MOV @4(SP), R1 ;GET MSG POINTER INTO R1  
ADD #2, 4(SP) ;INCR ARG POINTER  
MOV @4(SP), R2 ;GET WORD COUNT INTO R2  
ADD #2, 4(SP) ;FIX UP RETURN PC  
6$: MOV (R1)+, TXWORD ;GET NEXT MSG WORD  
JSR PC, LDIXSI ;LOAD A WORD INTO TX SILO  
DEC R2 ;DECR COUNT  
BNE 6$ ;BR IF NOT DONE YET  
JSR PC, WAIT50 ;WAIT FOR SILO TO RIPPLE  
MOV (SP)+, R2 ;RESTORE R2  
MOV (SP)+, R1 ;RESTORE R1  
RTS PC ;RETURN
```

7419
 7420
 7421
 7422
 7423
 7424
 7425
 7426 011046 010146
 7427 011050 016637 000002 002352
 7428 011056 162737 000004 002352
 7429 011064 004737 007326
 7430 011070 004737 005120
 7431 011074 005001
 7432 011076 020176 000002 9\$:
 7433 011102 001410
 7434 011104 004737 006402
 7435 011110 000001
 7436 011112 004737 005226
 7437 011116 000001
 7438 011120 005201
 7439 011122 000765
 7440 011124 004737 006402 12\$:
 7441 011130 000003
 7442 011132 062766 000002 000002
 7443 011140 005037 002352
 7444 011144 012601
 7445 011146 000207

```

*****
;* RXCHAR - THIS SUBROUTINE READS THE RCV SILO AND CLOCKS THE LINE UNIT.
;* FIRST, IT READS THE CHAR FROM THE RCV SILO, AND CHECKS FOR ICIR
;* = 1, IRDY = 0. IT THEN CLOCKS THE LINE UNIT FOR THE NO. OF CYCLES
;* PASSED IN THE WORD FOLLOWING THE CALL, AND THEN CHECKS FOR ICIR
;* = 1, IRDY = 1.
*****
RXCHAR: MOV R1,-(SP) ;SAVE R1
        MOV 2(SP),SUBRPC
        SUB #4,SUBRPC ;GET PC OF SUBR CALL
        JSR PC,RDRXSI ;READ RCV SILO
        JSR PC,WAIT50 ;ALLOW SILO TO RIPPLE
        CLR R1 ;INIT CYCLE COUNT
9$:     CMP R1,@2(SP) ;SEE IF REQUIRED CYCLES DONE YET
        BEQ 12$ ;BR IF YES
        JSR PC,ISIRDY ;CHK ICIR = 1, IRDY = 0
        1
        JSR PC,STPLU ;CLK LU 1 CYCLE
        1
        INC R1 ;INCR CYCLE COUNT
        BR 9$
12$:    JSR PC,ISIRDY ;CHK ICIR = 1 IRDY - 1
        3
        ADD #2,2(SP) ;FIX RETURN PC
        CLR SUBRPC ;CLEAR SUBR CALL PC
        MOV (SP)+,R1 ;RESTORE R1
        RTS PC ;RETURN
  
```

7446
 7447
 7448
 7449
 7450
 7451
 7452
 7453
 7454
 7455
 7456
 7457
 7458 011150 013746 002400
 7459 011154 010146
 7460 011156 016637 000004 002352
 7461 011164 162737 000004 002352
 7462 011172 012701 000001
 7463 011176 012737 000017 002400
 7464 011204 004737 005226 4\$:
 7465 011210 000001
 7466 011212 004737 003644
 7467 011216 030176 000004
 7468 011222 001013
 7469 011224 132737 000040 002364
 7470 011232 001422
 7471 011234 004737 004500
 7472
 7473 011240
 (4) 011240 104455

```

*****
;* CKTBIT - THIS SUBROUTINE CLOCKS THE LINE UNIT FOR 8 CYCLES, AND AFTER EACH
;* CYCLE, THE TXDATA BIT IN REG 17 IS EXAMINED, AND COMPARED TO A BIT OF
;* THE CHAR PASSED IN THE WORD FOLLOWING THE CALL.
;* IF AN ERROR OCCURS, A RETURN IS MADE TO THE TEST AT THE ADDRESS IN
;* RETADR.
*****
CKTBIT: MOV REGNUM,-(SP) ;SAVE LU REG NO.
        MOV R1,-(SP) ;SAVE R1
        MOV 4(SP),SUBRPC
        SUB #4,SUBRPC ;GET PC OF SUBROUTINE CALL
        MOV #BIT0,R1 ;INIT BIT POINTER
        MOV #17,REGNUM ;SET REG NO. = 17
4$:     JSR PC,STPLU ;CLOCK LINE UNIT 1 CYCLE
        1
        JSR PC,READLU ;READ REG 17
        BIT R1,@4(SP) ;SEE IF EXPECTED BIT - 1
        BNE 9$ ;BR IF YES
        BITB #TXDATA,REDBYT ;SEE IF ACTUAL BIT = 0
        BEQ 12$ ;BR IF YES
        JSR PC,GETALL ;GET REGS FOR PRINTOUT
;REPORT TXDATA NOT CLEARED
ERRDF 32,EM32,ERR4
  
```

TRAP C\$ERDF

```
(5) 011242 000040 .WORD 32
(5) 011244 013222 .WORD EM32
(5) 011246 015576 .WORD ERR4
7474 011250 000420
7475 011252 132737 000040 002364 9$: BR 20$ ;TAKE ERROR RETURN
7476 011260 001007 BITB #TXDATA,REDBYT ;SEE IF ACTUAL BIT - 1
7477 011262 004737 004500 BNE 12$ ;BR IF YES
7478 ;REPORT TXDATA BIT NOT SET ;GET REGS FOR PRINTOUT
7479 011266 ERRDF 33,EM33,ERR4
(4) 011266 104455 TRAP C$ERDF
(5) 011270 000041 .WORD 33
(5) 011272 013251 .WORD EM33
(5) 011274 015576 .WORD ERR4
7480 011276 000405
7481 011300 006301 12$: BR 20$ ;TAKE ERROR EXIT
7482 011302 020127 000400 ASL R1 ;SHIFT BIT POINTER
7483 011306 001336 CMP R1,#400 ;SEE IF 8 BITS SCANNED YET
7484 011310 000405 BNE 4$ ;BR IF NO
7485 011312 013706 002346 20$: BR 22$
7486 011316 013746 002362 MOV PSTACK,SP ;RESTORE PROGRAM STACK POINTER TO BASE LEVEL
7487 011322 000406 MOV RETADR,-(SP) ;FIX UP RETURN PC
7488 011324 062766 000002 000004 22$: BR 40$
7489 011332 012601 ADD #2,4(SP) ;FIX UP ERROR-FREE RETURN PC
7490 011334 012637 002400 MOV (SP)+,R1 ;RESTORE R1
7491 011340 005037 002352 40$: MOV (SP)+,REGNUM ;RESTORE REG NO.
7492 011344 000207 CLR SUBRPC ;CLEAR SUBR CALL PC
7493 RTS PC ;RETURN
```

7494
7495
7496
7497
7498
7499

```
:*****
:* LDMSG1 - THIS SUBROUTINE LOADS THE TRANSMITTER SILO WITH MSG1, AND LOADS
:* THE DATA CHARS INTO THE RCV MSG BUFFER (RCVBUF:), AS EXPECTED DATA
:* FOR LATER COMPARISON.
:*****
```

```
LDMSG1: MOV R1,-(SP) ;SAVE R1
MOV R2,-(SP) ;SAVE R2
JSR PC,LODMSG ;LOAD MSG1 INTO TX SILO
MSG1
9.
7508 011362 012701 003224 MOV #MSG1+4,R1 ;GET POINTER TO MSG1
7509 011366 012702 003262 MOV #RCVBUF,R2 ;GET POINTER TO MSG BUF
7510 011372 012122 3$: MOV (R1)+,(R2)+ ;LOAD A CHAR INTO MSG BUF
7511 011374 020127 003236 CMP R1,#MSG1+14. ;SEE IF DID LAST DATA CHAR YET
7512 011400 103774 BLO 3$ ;BR IF NOT
7513 011402 052762 100400 177776 BIS #CRCCHK!RXBCC,-2(R2) ;SET EXPECTED BCC
7514 011410 012726 000160 MOV #160,(SP)+ ;LOAD HI CRC BYTE
7515 011414 012726 000034 MOV #034,(SP)+ ;LOAD LO CRC BYTE
7516 011420 012602 MOV (SP)+,R2 ;RESTORE R2
7517 011422 012601 MOV (SP)+,R1 ;RESTORE R1
7518 011424 000207 RTS PC ;RETURN
```

7519
7520
7521
7522

7523

```

7525          .SBTTL  GLOBAL ERROR REPORT SECTION
7526
7527          ://////////
7528          :/      THE GLOBAL ERROR REPORT SECTION CONTAINS ERROR MESSAGES
7529          :/      THAT ARE USED IN MORE THAN ONE TEST.
7530          ://////////
7531
7532
7533 011426 052045 047445 022466 FMT1:  .ASCIZ  /%T%06%N/
      011434 000116
7534 011436 047045 040445 040506 FMT2:  .ASCIZ  /%N%AFAILING REG: /
      011444 046111 047111 020107
      011452 042522 035107 000040
7535 011460 040445 054105 042520 FMT3:  .ASCIZ  /%AEXPECTED: %03%S5%AACTUAL: %03%N/
      011466 052103 042105 020072
      011474 047445 022463 032523
      011502 040445 041501 052524
      011510 046101 020072 047445
      011516 022463 000116
7536 011522 047045 052045 047045 FMT4:  .ASCIZ  /%N%T%N%T%N/
      011530 052045 047045 000
7537 011535 045 031517 051445 FMT5:  .ASCIZ  /%05%S5%03%S5%03%S5%03%N/
      011542 022465 031517 051445
      011550 022465 031517 051445
      011556 022465 031517 047045
      011564 000
7538 011565 045 032123 047445 FMT6:  .ASCIZ  /%S4%03%S5%03%S5%03%S5%03%N/
      011572 022463 032523 047445
      011600 022463 032523 047445
      011606 022463 032523 047445
      011614 022463 000116
7539 011620 052045 047445 022462 FMT7:  .ASCIZ  /%T%02%N/
      011626 000116
7540 011630 040445 054105 042524 FMT8:  .ASCIZ  /%AEXTENDED REG AX%01%A-%T%N/
      011636 042116 042105 051040
      011644 043505 040440 022530
      011652 030517 040445 022455
      011660 022524 000116
7541 011664 052045 047045 000 FMT9:  .ASCIZ  /%T%N/
7542 011671 045 050101 020103 FMT10: .ASCIZ  /%APC OF SUBR CALL: %06%N/
      011676 043117 051440 041125
      011704 020122 040503 046114
      011712 020072 047445 022466
      011720 000116
7543 011722 040445 042522 020107 FMT11: .ASCIZ  /%AREG %02%A LOADED WITH: %03%N/
      011730 047445 022462 020101
      011736 047514 042101 042105
      011744 053440 052111 035110
      011752 022440 031517 047045
      011760 000
7544 011761 045 022516 052101 FMT19: .ASCIZ  /%N%ATEST %D2%A NOT RUN%N/
      011766 051505 020124 042045
      011774 022462 020101 047516
      012002 020124 052522 022516
      012010 000116
7545 012012 047045 040445 046120 FMT24: .ASCIZ  /%N%PLEASE INSURE RUN SWITCH ON MICROPROCESSOR IS ON%N/

```

| | | | | | | |
|------|--------|--------|--------|--------|-------|---|
| | 012020 | 040505 | 042523 | 044440 | | |
| | 012026 | 051516 | 051125 | 020105 | | |
| | 012034 | 052522 | 020116 | 053523 | | |
| | 012042 | 052111 | 044103 | 047440 | | |
| | 012050 | 020116 | 044515 | 051103 | | |
| | 012056 | 050117 | 047522 | 042503 | | |
| | 012064 | 051523 | 051117 | 044440 | | |
| | 012072 | 020123 | 047117 | 047045 | | |
| | 012100 | 000 | | | | |
| 7546 | | | | | | |
| 7547 | | | | | | |
| 7548 | | | | | | |
| 7549 | 012101 | 103 | 051123 | 040440 | EM1: | .ASCIZ /CSR ADDRESS TIME-OUT (SELO)/ |
| | 012106 | 042104 | 042522 | 051523 | | |
| | 012114 | 052040 | 046511 | 026505 | | |
| | 012122 | 052517 | 020124 | 051450 | | |
| | 012130 | 046105 | 024460 | 000 | | |
| 7550 | 012135 | 122 | 043505 | 047040 | EM2: | .ASCIZ /REG NOT INITIALIZED BY MST CLR/ |
| | 012142 | 052117 | 044440 | 044516 | | |
| | 012150 | 044524 | 046101 | 055111 | | |
| | 012156 | 042105 | 041040 | 020131 | | |
| | 012164 | 051515 | 020124 | 046103 | | |
| | 012172 | 000122 | | | | |
| 7551 | 012174 | 042522 | 020107 | 044515 | EM3: | .ASCIZ /REG MISCOMPARE/ |
| | 012202 | 041523 | 046517 | 040520 | | |
| | 012210 | 042522 | 000 | | | |
| 7552 | 012213 | 122 | 043505 | 047040 | EM4: | .ASCIZ /REG NOT INITIALIZED BY UNIBUS RESET (INIT)/ |
| | 012220 | 052117 | 044440 | 044516 | | |
| | 012226 | 044524 | 046101 | 055111 | | |
| | 012234 | 042105 | 041040 | 020131 | | |
| | 012242 | 047125 | 041111 | 051525 | | |
| | 012250 | 051040 | 051505 | 052105 | | |
| | 012256 | 024040 | 047111 | 052111 | | |
| | 012264 | 000051 | | | | |
| 7553 | 012266 | 040515 | 047111 | 020124 | EM5: | .ASCIZ /MAINT CLK BIT STUCK AT 0/ |
| | 012274 | 046103 | 020113 | 044502 | | |
| | 012302 | 020124 | 052123 | 041525 | | |
| | 012310 | 020113 | 052101 | 030040 | | |
| | 012316 | 000 | | | | |
| 7554 | 012317 | 115 | 044501 | 052116 | EM6: | .ASCIZ /MAINT CLK BIT STUCK AT 1/ |
| | 012324 | 041440 | 045514 | 041040 | | |
| | 012332 | 052111 | 051440 | 052524 | | |
| | 012340 | 045503 | 040440 | 020124 | | |
| | 012346 | 000061 | | | | |
| 7555 | 012350 | 051117 | 054504 | 047040 | EM7: | .ASCIZ /ORDY NOT SET/ |
| | 012356 | 052117 | 051440 | 052105 | | |
| | 012364 | 000 | | | | |
| 7556 | 012365 | 117 | 042122 | 020131 | EM8: | .ASCIZ /ORDY NOT CLEARED/ |
| | 012372 | 047516 | 020124 | 046103 | | |
| | 012400 | 040505 | 042522 | 000104 | | |
| 7557 | 012406 | 041517 | 051117 | 047040 | EM9: | .ASCIZ /OCOR NOT SET/ |
| | 012414 | 052117 | 051440 | 052105 | | |
| | 012422 | 000 | | | | |
| 7558 | 012423 | 117 | 047503 | 020122 | EM10: | .ASCIZ /OCOR NOT CLEARED/ |
| | 012430 | 047516 | 020124 | 046103 | | |
| | 012436 | 040505 | 042522 | 000104 | | |

| | | | | | | | |
|------|--------|--------|--------|--------|-------|--------|-------------------------------|
| 7559 | 012444 | 040517 | 052103 | 047040 | EM11: | .ASCIZ | /OACT NOT SET/ |
| | 012452 | 052117 | 051440 | 052105 | | | |
| | 012460 | 000 | | | | | |
| 7560 | 012461 | 117 | 041501 | 020124 | EM12: | .ASCIZ | /OACT NOT CLEARED/ |
| | 012466 | 047516 | 020124 | 046103 | | | |
| | 012474 | 040505 | 042522 | 000104 | | | |
| 7561 | 012502 | 047125 | 051122 | 047040 | EM13: | .ASCIZ | /UNRR NOT CLEARED BY SOM/ |
| | 012510 | 052117 | 041440 | 042514 | | | |
| | 012516 | 051101 | 042105 | 041040 | | | |
| 7562 | 012524 | 020131 | 047523 | 000115 | EM14: | .ASCIZ | /UNRR NOT SET/ |
| | 012532 | 047125 | 051122 | 047040 | | | |
| | 012540 | 052117 | 051440 | 052105 | | | |
| | 012546 | 000 | | | | | |
| 7563 | 012547 | 125 | 051116 | 020122 | EM15: | .ASCIZ | /UNRR NOT CLEARED BY OC/ |
| | 012554 | 047516 | 020124 | 046103 | | | |
| | 012562 | 040505 | 042522 | 020104 | | | |
| | 012570 | 054502 | 047440 | 000103 | | | |
| 7564 | 012576 | 047125 | 051122 | 047040 | EM16: | .ASCIZ | /UNRR NOT CLEARED/ |
| | 012604 | 052117 | 041440 | 042514 | | | |
| | 012612 | 051101 | 042105 | 000 | | | |
| 7565 | 012617 | 111 | 042122 | 020131 | EM17: | .ASCIZ | /IRDY NOT SET/ |
| | 012624 | 047516 | 020124 | 042523 | | | |
| | 012632 | 000124 | | | | | |
| 7566 | 012634 | 051111 | 054504 | 047040 | EM18: | .ASCIZ | /IRDY NOT CLEARED/ |
| | 012642 | 052117 | 041440 | 042514 | | | |
| | 012650 | 051101 | 042105 | 000 | | | |
| 7567 | 012655 | 111 | 044503 | 020122 | EM19: | .ASCIZ | /ICIR NOT SET/ |
| | 012662 | 047516 | 020124 | 042523 | | | |
| | 012670 | 000124 | | | | | |
| 7568 | 012672 | 041511 | 051111 | 047040 | EM20: | .ASCIZ | /ICIR NOT CLEARED/ |
| | 012700 | 052117 | 041440 | 042514 | | | |
| | 012706 | 051101 | 042105 | 000 | | | |
| 7569 | 012713 | 111 | 041501 | 020124 | EM21: | .ASCIZ | /IACT NOT SET/ |
| | 012720 | 047516 | 020124 | 042523 | | | |
| | 012726 | 000124 | | | | | |
| 7570 | 012730 | 040511 | 052103 | 047040 | EM22: | .ASCIZ | /IACT NOT CLEARED/ |
| | 012736 | 052117 | 041440 | 042514 | | | |
| | 012744 | 051101 | 042105 | 000 | | | |
| 7571 | 012751 | 104 | 051523 | 020111 | EM23: | .ASCIZ | /DSSI NOT CLEARED/ |
| | 012756 | 047516 | 020124 | 046103 | | | |
| | 012764 | 040505 | 042522 | 000104 | | | |
| 7572 | 012772 | 051504 | 044523 | 047040 | EM24: | .ASCIZ | /DSSI NOT SET/ |
| | 013000 | 052117 | 051440 | 052105 | | | |
| | 013006 | 000 | | | | | |
| 7573 | 013007 | 104 | 051523 | 020111 | EM25: | .ASCIZ | /DSSI NOT CLEARED BY MST CLR/ |
| | 013014 | 047516 | 020124 | 046103 | | | |
| | 013022 | 040505 | 042522 | 020104 | | | |
| | 013030 | 054502 | 046440 | 052123 | | | |
| | 013036 | 041440 | 051114 | 000 | | | |
| 7574 | 013043 | 111 | 041516 | 051117 | EM26: | .ASCIZ | /INCORRECT DATA CHAR RCV'D/ |
| | 013050 | 042522 | 052103 | 042040 | | | |
| | 013056 | 052101 | 020101 | 044103 | | | |
| | 013064 | 051101 | 051040 | 053103 | | | |
| | 013072 | 042047 | 000 | | | | |
| 7575 | 013075 | 111 | 041516 | 051117 | EM27: | .ASCIZ | /INCORRECT CRC BYTE RCV'D/ |
| | 013102 | 042522 | 052103 | 041440 | | | |

| | | | | | | | |
|------|--------|--------|--------|--------|-------|--------|--------------------------|
| | 013110 | 041522 | 041040 | 052131 | | | |
| | 013116 | 020105 | 041522 | 023526 | | | |
| | 013124 | 000104 | | | | | |
| 7576 | 013126 | 051522 | 046517 | 047040 | EM28: | .ASCIZ | /RSOM NOT CLEARED/ |
| | 013134 | 052117 | 041440 | 042514 | | | |
| | 013142 | 051101 | 042105 | 000 | | | |
| 7577 | 013147 | 122 | 047523 | 020115 | EM29: | .ASCIZ | /RSOM NOT SET/ |
| | 013154 | 047516 | 020124 | 042523 | | | |
| | 013162 | 000124 | | | | | |
| 7578 | 013164 | 042522 | 046517 | 047040 | EM30: | .ASCIZ | /REOM NOT CLEARED/ |
| | 013172 | 052117 | 041440 | 042514 | | | |
| | 013200 | 051101 | 042105 | 000 | | | |
| 7579 | 013205 | 122 | 047505 | 020115 | EM31: | .ASCIZ | /REOM NOT SET/ |
| | 013212 | 047516 | 020124 | 042523 | | | |
| | 013220 | 000124 | | | | | |
| 7580 | 013222 | 054124 | 040504 | 040524 | EM32: | .ASCIZ | /TXDATA BIT NOT CLEARED/ |
| | 013230 | 041040 | 052111 | 047040 | | | |
| | 013236 | 052117 | 041440 | 042514 | | | |
| | 013244 | 051101 | 042105 | 000 | | | |
| 7581 | 013251 | 124 | 042130 | 052101 | EM33: | .ASCIZ | /TXDATA BIT NOT SET/ |
| | 013256 | 020101 | 044502 | 020124 | | | |
| | 013264 | 047516 | 020124 | 042523 | | | |
| | 013272 | 000124 | | | | | |
| 7582 | 013274 | 041522 | 023526 | 020104 | EM34: | .ASCIZ | /RCV'D DATA MISCOMPARE/ |
| | 013302 | 040504 | 040524 | 046440 | | | |
| | 013310 | 051511 | 047503 | 050115 | | | |
| | 013316 | 051101 | 000105 | | | | |
| 7583 | 013322 | 041502 | 020103 | 047516 | EM35: | .ASCIZ | /BCC NOT CLEARED/ |
| | 013330 | 020124 | 046103 | 040505 | | | |
| | 013336 | 042522 | 000104 | | | | |
| 7584 | 013342 | 041502 | 020103 | 047516 | EM36: | .ASCIZ | /BCC NOT SET/ |
| | 013350 | 020124 | 042523 | 000124 | | | |
| 7585 | 013356 | 041105 | 045514 | 047040 | EM37: | .ASCIZ | /EBLK NOT CLEARED/ |
| | 013364 | 052117 | 041440 | 042514 | | | |
| | 013372 | 051101 | 042105 | 000 | | | |
| 7586 | 013377 | 105 | 046102 | 020113 | EM38: | .ASCIZ | /EBLK NOT SET/ |
| | 013404 | 047516 | 020124 | 042523 | | | |
| | 013412 | 000124 | | | | | |
| 7587 | 013414 | 040522 | 020102 | 047516 | EM39: | .ASCIZ | /RAB NOT CLEARED/ |
| | 013422 | 020124 | 046103 | 040505 | | | |
| | 013430 | 042522 | 000104 | | | | |
| 7588 | 013434 | 040522 | 020102 | 047516 | EM40: | .ASCIZ | /RAB NOT SET/ |
| | 013442 | 020124 | 042523 | 000124 | | | |
| 7589 | 013450 | 053117 | 051122 | 047040 | EM41: | .ASCIZ | /OVRR NOT CLEARED/ |
| | 013456 | 052117 | 041440 | 042514 | | | |
| | 013464 | 051101 | 042105 | 000 | | | |
| 7590 | 013471 | 117 | 051126 | 020122 | EM42: | .ASCIZ | /OVRR NOT SET/ |
| | 013476 | 047516 | 020124 | 042523 | | | |
| | 013504 | 000124 | | | | | |
| 7591 | 013506 | 053523 | 050040 | 041501 | EM43: | .ASCIZ | /SW PACK #1 INCORRECT/ |
| | 013514 | 020113 | 030443 | 044440 | | | |
| | 013522 | 041516 | 051117 | 042522 | | | |
| | 013530 | 052103 | 000 | | | | |
| 7592 | 013533 | 123 | 020127 | 040520 | EM44: | .ASCIZ | /SW PACK #2 INCORRECT/ |
| | 013540 | 045503 | 021440 | 020062 | | | |
| | 013546 | 047111 | 047503 | 051122 | | | |

| | | | | | | |
|------|--------|--------|--------|--------|-------|---|
| 7593 | 013554 | 041505 | 000124 | | | |
| | 013560 | 053523 | 050040 | 041501 | EM45: | .ASCIZ /SW PACK #3 INCORRECT/ |
| | 013566 | 020113 | 031443 | 044440 | | |
| | 013574 | 041516 | 051117 | 042522 | | |
| | 013602 | 052103 | 000 | | | |
| 7594 | 013605 | 122 | 053103 | 051440 | EM46: | .ASCIZ /RCV SILO NOT CLEARED BY IC/ |
| | 013612 | 046111 | 020117 | 047516 | | |
| | 013620 | 020124 | 046103 | 040505 | | |
| | 013626 | 042522 | 020104 | 054502 | | |
| | 013634 | 044440 | 000103 | | | |
| 7595 | 013640 | 051501 | 042523 | 041115 | EM47: | .ASCIZ /ASSEMB BIT COUNT INCORRECT/ |
| | 013646 | 041040 | 052111 | 041440 | | |
| | 013654 | 052517 | 052116 | 044440 | | |
| | 013662 | 041516 | 051117 | 042522 | | |
| | 013670 | 052103 | 000 | | | |
| 7596 | 013673 | 117 | 042104 | 053040 | EM48: | .ASCIZ /ODD VRC PARITY BIT NOT SET/ |
| | 013700 | 041522 | 050040 | 051101 | | |
| | 013706 | 052111 | 020131 | 044502 | | |
| | 013714 | 020124 | 047516 | 020124 | | |
| | 013722 | 042523 | 000124 | | | |
| 7597 | 013726 | 042117 | 020104 | 051126 | EM49: | .ASCIZ /ODD VRC PARITY BIT NOT CLEARED/ |
| | 013734 | 020103 | 040520 | 044522 | | |
| | 013742 | 054524 | 041040 | 052111 | | |
| | 013750 | 047040 | 052117 | 041440 | | |
| | 013756 | 042514 | 051101 | 042105 | | |
| | 013764 | 000 | | | | |
| 7598 | 013765 | 105 | 042526 | 020116 | EM50: | .ASCIZ /EVEN VRC PARITY BIT NOT SET/ |
| | 013772 | 051126 | 020103 | 040520 | | |
| | 014000 | 044522 | 054524 | 041040 | | |
| | 014006 | 052111 | 047040 | 052117 | | |
| | 014014 | 051440 | 052105 | 000 | | |
| 7599 | 014021 | 105 | 042526 | 020116 | EM51: | .ASCIZ /EVEN VRC PARITY BIT NOT CLEARED/ |
| | 014026 | 051126 | 020103 | 040520 | | |
| | 014034 | 044522 | 054524 | 041040 | | |
| | 014042 | 052111 | 047040 | 052117 | | |
| | 014050 | 041440 | 042514 | 051101 | | |
| | 014056 | 042105 | 000 | | | |
| 7600 | 014061 | 122 | 040505 | 054504 | EM52: | .ASCIZ /READY NOT SET AFTER AX REG WRITE/ |
| | 014066 | 047040 | 052117 | 051440 | | |
| | 014074 | 052105 | 040440 | 052106 | | |
| | 014102 | 051105 | 040440 | 020130 | | |
| | 014110 | 042522 | 020107 | 051127 | | |
| | 014116 | 052111 | 000105 | | | |
| 7601 | 014122 | 042522 | 042101 | 020131 | EM53: | .ASCIZ /READY NOT SET AFTER AX REG READ/ |
| | 014130 | 047516 | 020124 | 042523 | | |
| | 014136 | 020124 | 043101 | 042524 | | |
| | 014144 | 020122 | 054101 | 051040 | | |
| | 014152 | 043505 | 051040 | 040505 | | |
| | 014160 | 000104 | | | | |
| 7602 | 014162 | 054124 | 052440 | 042116 | EM54: | .ASCIZ /TX UNDERRUN ERROR/ |
| | 014170 | 051105 | 052522 | 020116 | | |
| | 014176 | 051105 | 047522 | 000122 | | |
| 7603 | 014204 | 052122 | 020123 | 047516 | EM60: | .ASCIZ /RTS NOT SET/ |
| | 014212 | 020124 | 042523 | 000124 | | |
| 7604 | 014220 | 052122 | 020123 | 047516 | EM65: | .ASCIZ /RTS NOT CLEARED/ |
| | 014226 | 020124 | 046103 | 040505 | | |

```

014234 042522 000104
7605
7606
7607
7608 014240 047111 052502 027523 DH1: .ASCIZ &INBUS/OUTBUS REG &
      014246 052517 041124 051525
      014254 051040 043505 000040
7609 014262 044514 042516 052440 DH2: .ASCIZ /LINE UNIT INBUS REGS :/
      014270 044516 020124 047111
      014276 052502 020123 042522
      014304 051507 035040 000
7610 014311 122 043505 030061 DH3: .ASCIZ /REG10 REG11 REG12 REG13/
      014316 020040 051040 043505
      014324 030461 020040 051040
      014332 043505 031061 020040
      014340 051040 043505 031461
      014346 000
7611 014347 040 020040 051040 DH4: .ASCIZ / REG14 REG15 REG16 REG17/
      014354 043505 032061 020040
      014362 051040 043505 032461
      014370 020040 051040 043505
      014376 033061 020040 051040
      014404 043505 033461 000
7612 014411 061 000065 DH5: .ASCIZ /15/
7613 014414 033061 000 DH6: .ASCIZ /16/
7614 014417 114 047111 020105 DH7: .ASCIZ /LINE UNIT EXTENDED REGS :/
      014424 047125 052111 042440
      014432 052130 047105 042504
      014440 020104 042522 051507
      014446 035040 000
7615 014451 101 030130 030455 DH8: .ASCIZ /AX0-15 AX0-16 AX1-15 AX1-16/
      014456 020065 040440 030130
      014464 030455 020066 040440
      014472 030530 030455 020065
      014500 040440 030530 030455
      014506 000066
7616 014510 020040 020040 054101 DH9: .ASCIZ / AX2-15 AX2-16 AX3-15 AX3-16/
      014516 026462 032461 020040
      014524 054101 026462 033061
      014532 020040 054101 026463
      014540 032461 020040 054101
      014546 026463 033061 000
  
```

```

7617
7618 014554 .EVEN
7619
7620
7621
7622
7623
  
```

```

7624 014554 BGNMSG ERR1
(3) 014554
7625 014554 PRINTB #FMT1,#ADDRES,MPCSR
(9) 014554 013746 002446
(8) 014560 012746 035560
(7) 014564 012746 011426
(6) 014570 012746 000003
  
```

ERR1::

```

MOV MPCSR,-(SP)
MOV #ADDRES,-(SP)
MOV #FMT1,-(SP)
MOV #3,-(SP)
  
```

| | | | | | | | |
|------|--------|--------|--------|--------------------------------------|---------|------|---------------|
| (3) | 014574 | 010600 | | | | MOV | SP,R0 |
| (4) | 014576 | 104414 | | | | TRAP | C\$PNTB |
| (4) | 014600 | 062706 | 000010 | | | ADD | #10,SP |
| 7626 | 014604 | | | ENDMSG | | | |
| (3) | 014604 | | | | L10002: | | |
| (3) | 014604 | 104423 | | | | TRAP | C\$MSG |
| 7627 | | | | | | | |
| 7628 | | | | | | | |
| 7629 | | | | | | | |
| 7630 | 014606 | | | BGNMSG ERR2 | | | |
| (3) | 014606 | | | | ERR2:: | | |
| 7631 | 014606 | | | PRINTB #FMT1,#ADDRES,MPCSR | | | |
| (9) | 014606 | 013746 | 002446 | | | MOV | MPCSR,-(SP) |
| (8) | 014612 | 012746 | 035560 | | | MOV | #ADDRES,-(SP) |
| (7) | 014616 | 012746 | 011426 | | | MOV | #FMT1,-(SP) |
| (6) | 014622 | 012746 | 000003 | | | MOV | #3,-(SP) |
| (3) | 014626 | 010600 | | | | MOV | SP,R0 |
| (4) | 014630 | 104414 | | | | TRAP | C\$PNTB |
| (4) | 014632 | 062706 | 000010 | | | ADD | #10,SP |
| 7632 | 014636 | | | PRINTB #FMT2 | | | |
| (7) | 014636 | 012746 | 011436 | | | MOV | #FMT2,-(SP) |
| (6) | 014642 | 012746 | 000001 | | | MOV | #1,-(SP) |
| (3) | 014646 | 010600 | | | | MOV | SP,R0 |
| (4) | 014650 | 104414 | | | | TRAP | C\$PNTB |
| (4) | 014652 | 062706 | 000004 | | | ADD | #4,SP |
| 7633 | 014656 | | | PRINTB #FMT7,#DH1,REGNUM | | | |
| (9) | 014656 | 013746 | 002400 | | | MOV | REGNUM,-(SP) |
| (8) | 014662 | 012746 | 014240 | | | MOV | #DH1,-(SP) |
| (7) | 014666 | 012746 | 011620 | | | MOV | #FMT7,-(SP) |
| (6) | 014672 | 012746 | 000003 | | | MOV | #3,-(SP) |
| (3) | 014676 | 010600 | | | | MOV | SP,R0 |
| (4) | 014700 | 104414 | | | | TRAP | C\$PNTB |
| (4) | 014702 | 062706 | 000010 | | | ADD | #10,SP |
| 7634 | 014706 | | | PRINTB #FMT3,GOODAT,BADDAT | | | |
| (9) | 014706 | 013746 | 002406 | | | MOV | BADDAT,-(SP) |
| (8) | 014712 | 013746 | 002404 | | | MOV | GOODAT,-(SP) |
| (7) | 014716 | 012746 | 011460 | | | MOV | #FMT3,-(SP) |
| (6) | 014722 | 012746 | 000003 | | | MOV | #3,-(SP) |
| (3) | 014726 | 010600 | | | | MOV | SP,R0 |
| (4) | 014730 | 104414 | | | | TRAP | C\$PNTB |
| (4) | 014732 | 062706 | 000010 | | | ADD | #10,SP |
| 7635 | 014736 | | | PRINTX #FMT4,#DH2,#DH3 | | | |
| (9) | 014736 | 012746 | 014311 | | | MOV | #DH3,-(SP) |
| (8) | 014742 | 012746 | 014262 | | | MOV | #DH2,-(SP) |
| (7) | 014746 | 012746 | 011522 | | | MOV | #FMT4,-(SP) |
| (6) | 014752 | 012746 | 000003 | | | MOV | #3,-(SP) |
| (3) | 014756 | 010600 | | | | MOV | SP,R0 |
| (4) | 014760 | 104415 | | | | TRAP | C\$PNTX |
| (4) | 014762 | 062706 | 000010 | | | ADD | #10,SP |
| 7636 | 014766 | | | PRINTX #FMT5,LUR10,LUR11,LUR12,LUR13 | | | |
| (11) | 014766 | 013746 | 002310 | | | MOV | LUR13,-(SP) |
| (10) | 014772 | 013746 | 002306 | | | MOV | LUR12,-(SP) |
| (9) | 014776 | 013746 | 002304 | | | MOV | LUR11,-(SP) |
| (8) | 015002 | 013746 | 002302 | | | MOV | LUR10,-(SP) |
| (7) | 015006 | 012746 | 011535 | | | MOV | #FMT5,-(SP) |
| (6) | 015012 | 012746 | 000005 | | | MOV | #5,-(SP) |

(3) 015016 010600
(4) 015020 104415
(4) 015022 062706 000014
7637 015026
(8) 015026 012746 014347
(7) 015032 012746 011664
(6) 015036 012746 000002
(3) 015042 010600
(4) 015044 104415
(4) 015046 062706 000006
7638 015052
(11) 015052 013746 002320
(10) 015056 013746 002316
(9) 015062 013746 002314
(8) 015066 013746 002312
(7) 015072 012746 011565
(6) 015076 012746 000005
(3) 015102 010600
(4) 015104 104415
(4) 015106 062706 000014
7639 015112
(3) 015112
(3) 015112 104423
7640
7641
7642
7643
7644
7645 015114
(3) 015114
7646 015114
(9) 015114 013746 002446
(8) 015120 012746 035560
(7) 015124 012746 011426
(6) 015130 012746 000003
(3) 015134 010600
(4) 015136 104414
(4) 015140 062706 000010
7647 015144
(7) 015144 012746 011436
(6) 015150 012746 000001
(3) 015154 010600
(4) 015156 104414
(4) 015160 062706 000004
7648 015164
(9) 015164 013746 002530
(8) 015170 013746 002532
(7) 015174 012746 011630
(6) 015200 012746 000003
(3) 015204 010600
(4) 015206 104414
(4) 015210 062706 00001C
7649 015214
(9) 015214 013746 002406
(8) 015220 013746 002404
(7) 015224 012746 011460

PRINTX #FMT9,#DH4

PRINTX #FMT6,LUR14,LUR15,LUR16,LUR17

ENDMSG

BGNMSG ERR3

PRINTB #FMT1,#ADDRES,MPCSR

PRINTB #FMT2

PRINTB #FMT8,TMP1,TMP0

PRINTB #FMT3,GOODAT,BADDAT

MOV SP,RO
TRAP C\$PNTX
ADD #14,SP

MOV #DH4,-(SP)
MOV #FMT9,-(SP)
MOV #2,-(SP)
MOV SP,RO
TRAP C\$PNTX
ADD #6,SP

MOV LUR17,-(SP)
MOV LUR16,-(SP)
MOV LUR15,-(SP)
MOV LUR14,-(SP)
MOV #FMT6,-(SP)
MOV #5,-(SP)
MOV SP,RO
TRAP C\$PNTX
ADD #14,SP

L10U03: TRAP C\$MSG

ERR3::

MOV MPCSR,-(SP)
MOV #ADDRES,-(SP)
MOV #FMT1,-(SP)
MOV #3,-(SP)
MOV SP,RO
TRAP C\$PNTB
ADD #10,SP

MOV #FMT2,-(SP)
MOV #1,-(SP)
MOV SP,RO
TRAP C\$PNTB
ADD #4,SP

MOV TMP0,-(SP)
MOV TMP1,-(SP)
MOV #FMT8,-(SP)
MOV #3,-(SP)
MOV SP,RO
TRAP C\$PNTB
ADD #10,SP

MOV BADDAT,-(SP)
MOV GOODAT,-(SP)
MOV #FMT3,-(SP)

(6) 015230 012746 000003
(3) 015234 010600
(4) 015236 104414
(4) 015240 062706 000010
7650 015244
(9) 015244 012746 014311
(8) 015250 012746 014262
(7) 015254 012746 011522
(6) 015260 012746 000003
(3) 015264 010600
(4) 015266 104415
(4) 015270 062706 000010
7651 015274
(11) 015274 013746 002310
(10) 015300 013746 002306
(9) 015304 013746 002304
(8) 015310 013746 002302
(7) 015314 012746 011535
(6) 015320 012746 000005
(3) 015324 010600
(4) 015326 104415
(4) 015330 062706 000014
7652 015334
(8) 015334 012746 014347
(7) 015340 012746 011664
(6) 015344 012746 000002
(3) 015350 010600
(4) 015352 104415
(4) 015354 062706 000006
7653 015360
(11) 015360 013746 002320
(10) 015364 013746 002316
(9) 015370 013746 002314
(8) 015374 013746 002312
(7) 015400 012746 011565
(6) 015404 012746 000005
(3) 015410 010600
(4) 015412 104415
(4) 015414 062706 000014
7654 015420
(9) 015420 012746 014451
(8) 015424 012746 014417
(7) 015430 012746 011522
(6) 015434 012746 000003
(3) 015440 010600
(4) 015442 104415
(4) 015444 062706 000010
7655 015450
(11) 015450 013746 002330
(10) 015454 013746 002326
(9) 015460 013746 002324
(8) 015464 013746 002322
(7) 015470 012746 011535
(6) 015474 012746 000005
(3) 015500 010600
(4) 015502 104415

PRINTX #FMT4,#DH2,#DH3

PRINTX #FMT5,LUR10,LUR11,LUR12,LUR13

PRINTX #FMT9,#DH4

PRINTX #FMT6,LUR14,LUR15,LUR16,LUR17

PRINTX #FMT4,#DH7,#DH8

PRINTX #FMT5,AX0.15,AX0.16,AX1.15,AX1.16

MOV #3,-(SP)
MOV SP,R0
TRAP C\$PNTB
ADD #10,SP

MOV #DH3,-(SP)
MOV #DH2,-(SP)
MOV #FMT4,-(SP)
MOV #3,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #10,SP

MOV LUR13,-(SP)
MOV LUR12,-(SP)
MOV LUR11,-(SP)
MOV LUR10,-(SP)
MOV #FMT5,-(SP)
MOV #5,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #14,SP

MOV #DH4,-(SP)
MOV #FMT9,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #6,SP

MOV LUR17,-(SP)
MOV LUR16,-(SP)
MOV LUR15,-(SP)
MOV LUR14,-(SP)
MOV #FMT6,-(SP)
MOV #5,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #14,SP

MOV #DH8,-(SP)
MOV #DH7,-(SP)
MOV #FMT4,-(SP)
MOV #3,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #10,SP

MOV AX1.16,-(SP)
MOV AX1.15,-(SP)
MOV AX0.16,-(SP)
MOV AX0.15,-(SP)
MOV #FMT5,-(SP)
MOV #5,-(SP)
MOV SP,R0
TRAP C\$PNTX

(4) 015504 062706 000014
7656 015510
(8) 015510 012746 014510
(7) 015514 012746 011664
(6) 015520 012746 000002
(3) 015524 010600
(4) 015526 104415
(4) 015530 062706 000006
7657 015534
(11) 015534 013746 002340
(10) 015540 013746 002336
(9) 015544 013746 002334
(8) 015550 013746 002332
(7) 015554 012746 011565
(6) 015560 012746 000005
(3) 015564 010600
(4) 015566 104415
(4) 015570 062706 000014
7658 015574
(3) 015574
(3) 015574 104423
7659
7660
7661
7662
7663
7664 015576
(3) 015576
7665 015576
(8) 015576 013746 002352
(7) 015602 012746 011671
(6) 015606 012746 000002
(3) 015612 010600
(4) 015614 104414
(4) 015616 062706 000006
7666 015622
(9) 015622 013746 002446
(8) 015626 012746 035560
(7) 015632 012746 011426
(6) 015636 012746 000003
(3) 015642 010600
(4) 015644 104414
(4) 015646 062706 000010
7667 015652
(7) 015652 012746 011436
(6) 015656 012746 000001
(3) 015662 010600
(4) 015664 104414
(4) 015666 062706 000004
7668 015672
(9) 015672 013746 002400
(8) 015676 012746 014240
(7) 015702 012746 011620
(6) 015706 012746 000003
(3) 015712 010600
(4) 015714 104414

PRINTX #FMT9,#DH9

PRINTX #FMT6,AX2.15,AX2.16,AX3.15,AX3.16

ENDMSG

BGNMSG ERR4

PRINTB #FMT10,SUBRPC

PRINTB #FMT1,#ADDRES,MPCSR

PRINTB #FMT2

PRINTB #FMT7,#DH1,REGNUM

ADD #14,SP

MOV #DH9,-(SP)
MOV #FMT9,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #6,SP

MOV AX3.16,-(SP)
MOV AX3.15,-(SP)
MOV AX2.16,-(SP)
MOV AX2.15,-(SP)
MOV #FMT6,-(SP)
MOV #5,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #14,SP

L10004: TRAP C\$MSG

ERR4::

MOV SUBRPC,-(SP)
MOV #FMT10,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C\$PNTB
ADD #6,SP

MOV MPCSR,-(SP)
MOV #ADDRES,-(SP)
MOV #FMT1,-(SP)
MOV #3,-(SP)
MOV SP,R0
TRAP C\$PNTB
ADD #10,SP

MOV #FMT2,-(SP)
MOV #1,-(SP)
MOV SP,R0
TRAP C\$PNTB
ADD #4,SP

MOV REGNUM,-(SP)
MOV #DH1,-(SP)
MOV #FMT7,-(SP)
MOV #3,-(SP)
MOV SP,R0
TRAP C\$PNTB

| | | | | | | | | | | | |
|------|--------|--------|--------|--------|-----------------------------------|--|--|------|--------------|--|--|
| (4) | 015716 | 062706 | 000010 | | | | | | | | |
| 7669 | 015722 | | | PRINTX | #FMT4,#DH2,#DH3 | | | ADD | #10,SP | | |
| (9) | 015722 | 012746 | 014311 | | | | | MOV | #DH3,-(SP) | | |
| (8) | 015726 | 012746 | 014262 | | | | | MOV | #DH2,-(SP) | | |
| (7) | 015732 | 012746 | 011522 | | | | | MOV | #FMT4,-(SP) | | |
| (6) | 015736 | 012746 | 000003 | | | | | MOV | #3,-(SP) | | |
| (3) | 015742 | 010600 | | | | | | MOV | SP,R0 | | |
| (4) | 015744 | 104415 | | | | | | TRAP | C\$PNTX | | |
| (4) | 015746 | 062706 | 000010 | | | | | ADD | #10,SP | | |
| 7670 | 015752 | | | PRINTX | #FMT5,LUR10,LUR11,LUR12,LUR13 | | | | | | |
| (11) | 015752 | 013746 | 002310 | | | | | MOV | LUR13,-(SP) | | |
| (10) | 015756 | 013746 | 002306 | | | | | MOV | LUR12,-(SP) | | |
| (9) | 015762 | 013746 | 002304 | | | | | MOV | LUR11,-(SP) | | |
| (8) | 015766 | 013746 | 002302 | | | | | MOV | LUR10,-(SP) | | |
| (7) | 015772 | 012746 | 011535 | | | | | MOV | #FMT5,-(SP) | | |
| (6) | 015776 | 012746 | 000005 | | | | | MOV | #5,-(SP) | | |
| (3) | 016002 | 010600 | | | | | | MOV | SP,R0 | | |
| (4) | 016004 | 104415 | | | | | | TRAP | C\$PNTX | | |
| (4) | 016006 | 062706 | 000014 | | | | | ADD | #14,SP | | |
| 7671 | 016012 | | | PRINTX | #FMT9,#DH4 | | | | | | |
| (8) | 016012 | 012746 | 014347 | | | | | MOV | #DH4,-(SP) | | |
| (7) | 016016 | 012746 | 011664 | | | | | MOV | #FMT9,-(SP) | | |
| (6) | 016022 | 012746 | 000002 | | | | | MOV | #2,-(SP) | | |
| (3) | 016026 | 010600 | | | | | | MOV | SP,R0 | | |
| (4) | 016030 | 104415 | | | | | | TRAP | C\$PNTX | | |
| (4) | 016032 | 062706 | 000006 | | | | | ADD | #6,SP | | |
| 7672 | 016036 | | | PRINTX | #FMT6,LUR14,LUR15,LUR16,LUR17 | | | | | | |
| (11) | 016036 | 013746 | 002320 | | | | | MOV | LUR17,-(SP) | | |
| (10) | 016042 | 013746 | 002316 | | | | | MOV | LUR16,-(SP) | | |
| (9) | 016046 | 013746 | 002314 | | | | | MOV | LUR15,-(SP) | | |
| (8) | 016052 | 013746 | 002312 | | | | | MOV | LUR14,-(SP) | | |
| (7) | 016056 | 012746 | 011563 | | | | | MOV | #FMT6,-(SP) | | |
| (6) | 016062 | 012746 | 000005 | | | | | MOV | #5,-(SP) | | |
| (3) | 016066 | 010600 | | | | | | MOV | SP,R0 | | |
| (4) | 016070 | 104415 | | | | | | TRAP | C\$PNTX | | |
| (4) | 016072 | 062706 | 000014 | | | | | ADD | #14,SP | | |
| 7673 | 016076 | | | PRINTX | #FMT4,#DH7,#DH8 | | | | | | |
| (9) | 016076 | 012746 | 014451 | | | | | MOV | #DH8,-(SP) | | |
| (8) | 016102 | 012746 | 014417 | | | | | MOV | #DH7,-(SP) | | |
| (7) | 016106 | 012746 | 011522 | | | | | MOV | #FMT4,-(SP) | | |
| (6) | 016112 | 012746 | 000003 | | | | | MOV | #3,-(SP) | | |
| (3) | 016116 | 010600 | | | | | | MOV | SP,R0 | | |
| (4) | 016120 | 104415 | | | | | | TRAP | C\$PNTX | | |
| (4) | 016122 | 062706 | 000010 | | | | | ADD | #10,SP | | |
| 7674 | 016126 | | | PRINTX | #FMT5,AX0.15,AX0.16,AX1.15,AX1.16 | | | | | | |
| (11) | 016126 | 013746 | 002330 | | | | | MOV | AX1.16,-(SP) | | |
| (10) | 016132 | 013746 | 002326 | | | | | MOV | AX1.15,-(SP) | | |
| (9) | 016136 | 013746 | 002324 | | | | | MOV | AX0.16,-(SP) | | |
| (8) | 016142 | 013746 | 002322 | | | | | MOV | AX0.15,-(SP) | | |
| (7) | 016146 | 012746 | 011535 | | | | | MOV | #FMT5,-(SP) | | |
| (6) | 016152 | 012746 | 000005 | | | | | MOV | #5,-(SP) | | |
| (3) | 016156 | 010600 | | | | | | MOV | SP,R0 | | |
| (4) | 016160 | 104415 | | | | | | TRAP | C\$PNTX | | |
| (4) | 016162 | 062706 | 000014 | | | | | ADD | #14,SP | | |
| 7675 | 016166 | | | PRINTX | #FMT9,#DH9 | | | | | | |
| (8) | 016166 | 012746 | 014510 | | | | | MOV | #DH9,-(SP) | | |

(7) 016172 012746 011664
(6) 016176 012746 000002
(3) 016202 010600
(4) 016204 104415
(4) 016206 062706 000006
7676 016212
(11) 016212 013746 002340
(10) 016216 013746 002336
(9) 016222 013746 002334
(8) 016226 013746 002332
(7) 016232 012746 011565
(6) 016236 012746 000005
(3) 016242 010600
(4) 016244 104415
(4) 016246 062706 000014
7677 016252
(3) 016252
(3) 016252 104423
7678
7679
7680
7681
7682
7683 016254
(3) 016254
7684 016254
(9) 016254 013746 002446
(8) 016260 012746 035560
(7) 016264 012746 011426
(6) 016270 012746 000003
(3) 016274 010600
(4) 016276 104414
(4) 016300 062706 000010
7685 016304
(9) 016304 013746 002410
(8) 016310 013746 002400
(7) 016314 012746 011722
(6) 016320 012746 000003
(3) 016324 010600
(4) 016326 104414
(4) 016330 062706 000010
7686 016334
(7) 016334 012746 011436
(6) 016340 012746 000001
(3) 016344 010600
(4) 016346 104414
(4) 016350 062706 000004
7687 016354
(9) 016354 013746 002530
(8) 016360 013746 002532
(7) 016364 012746 011630
(6) 016370 012746 000003
(3) 016374 010600
(4) 016376 104414
(4) 016400 062706 000010
7688 016404

PRINTX #FMT6,AX2.15,AX2.16,AX3.15,AX3.16

ENDMSG

BGNMSG ERR5

PRINTB #FMT1,#ADDRES,MPCSR

PRINTB #FMT11,REGNUM,LOADAT

PRINTB #FMT2

PRINTB #FMT8,TMP1,TMP0

PRINTB #FMT3,GOODAT,BADDAT

MOV #FMT9,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #6,SP

MOV AX3.16,-(SP)
MOV AX3.15,-(SP)
MOV AX2.16,-(SP)
MOV AX2.15,-(SP)
MOV #FMT6,-(SP)
MOV #5,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #14,SP

L10005:
TRAP C\$MSG

ERR5::

MOV MPCSR,-(SP)
MOV #ADDRES,-(SP)
MOV #FMT1,-(SP)
MOV #3,-(SP)
MOV SP,R0
TRAP C\$PNTB
ADD #10,SP

MOV LOADAT,-(SP)
MOV REGNUM,-(SP)
MOV #FMT11,-(SP)
MOV #3,-(SP)
MOV SP,R0
TRAP C\$PNTB
ADD #10,SP

MOV #FMT2,-(SP)
MOV #1,-(SP)
MOV SP,R0
TRAP C\$PNTB
ADD #4,SP

MOV TMP0,-(SP)
MOV TMP1,-(SP)
MOV #FMT8,-(SP)
MOV #3,-(SP)
MOV SP,R0
TRAP C\$PNTB
ADD #10,SP

| | | | | | | | |
|------|--------|--------|--------|--------|-----------------------------------|------|--------------|
| (9) | 016404 | 013746 | 002406 | | | MOV | BADDAT,-(SP) |
| (8) | 016410 | 013746 | 002404 | | | MOV | GOODAT,-(SP) |
| (7) | 016414 | 012746 | 011460 | | | MOV | #FMT3,-(SP) |
| (6) | 016420 | 012746 | 000003 | | | MOV | #3,-(SP) |
| (3) | 016424 | 010600 | | | | MOV | SP,RO |
| (4) | 016426 | 104414 | | | | TRAP | C\$PNTB |
| (4) | 016430 | 062706 | 000010 | | | ADD | #10,SP |
| 7689 | 016434 | | | PRINTX | #FMT4,#DH2,#DH3 | | |
| (9) | 016434 | 012746 | 014311 | | | MOV | #DH3,-(SP) |
| (8) | 016440 | 012746 | 014262 | | | MOV | #DH2,-(SP) |
| (7) | 016444 | 012746 | 011522 | | | MOV | #FMT4,-(SP) |
| (6) | 016450 | 012746 | 000003 | | | MOV | #3,-(SP) |
| (3) | 016454 | 010600 | | | | MOV | SP,RO |
| (4) | 016456 | 104415 | | | | TRAP | C\$PNTX |
| (4) | 016460 | 062706 | 000010 | | | ADD | #10,SP |
| 7690 | 016464 | | | PRINTX | #FMT5,LUR10,LUR11,LUR12,LUR13 | | |
| (11) | 016464 | 013746 | 002310 | | | MOV | LUR13,-(SP) |
| (10) | 016470 | 013746 | 002306 | | | MOV | LUR12,-(SP) |
| (9) | 016474 | 013746 | 002304 | | | MOV | LUR11,-(SP) |
| (8) | 016500 | 013746 | 002302 | | | MOV | LUR10,-(SP) |
| (7) | 016504 | 012746 | 011535 | | | MOV | #FMT5,-(SP) |
| (6) | 016510 | 012746 | 000005 | | | MOV | #5,-(SP) |
| (3) | 016514 | 010600 | | | | MOV | SP,RO |
| (4) | 016516 | 104415 | | | | TRAP | C\$PNTX |
| (4) | 016520 | 062706 | 000014 | | | ADD | #14,SP |
| 7691 | 016524 | | | PRINTX | #FMT9,#DH4 | | |
| (8) | 016524 | 012746 | 014347 | | | MOV | #DH4,-(SP) |
| (7) | 016530 | 012746 | 011664 | | | MOV | #FMT9,-(SP) |
| (6) | 016534 | 012746 | 000002 | | | MOV | #2,-(SP) |
| (3) | 016540 | 010600 | | | | MOV | SP,RO |
| (4) | 016542 | 104415 | | | | TRAP | C\$PNTX |
| (4) | 016544 | 062706 | 000006 | | | ADD | #6,SP |
| 7692 | 016550 | | | PRINTX | #FMT6,LUR14,LUR15,LUR16,LUR17 | | |
| (11) | 016550 | 013746 | 002320 | | | MOV | LUR17,-(SP) |
| (10) | 016554 | 013746 | 002316 | | | MOV | LUR16,-(SP) |
| (9) | 016560 | 013746 | 002314 | | | MOV | LUR15,-(SP) |
| (8) | 016564 | 013746 | 002312 | | | MOV | LUR14,-(SP) |
| (7) | 016570 | 012746 | 011565 | | | MOV | #FMT6,-(SP) |
| (6) | 016574 | 012746 | 000005 | | | MOV | #5,-(SP) |
| (3) | 016600 | 010600 | | | | MOV | SP,RO |
| (4) | 016602 | 104415 | | | | TRAP | C\$PNTX |
| (4) | 016604 | 062706 | 000014 | | | ADD | #14,SP |
| 7693 | 016610 | | | PRINTX | #FMT4,#DH7,#DH8 | | |
| (9) | 016610 | 012746 | 014451 | | | MOV | #DH8,-(SP) |
| (8) | 016614 | 012746 | 014417 | | | MOV | #DH7,-(SP) |
| (7) | 016620 | 012746 | 011522 | | | MOV | #FMT4,-(SP) |
| (6) | 016624 | 012746 | 000003 | | | MOV | #3,-(SP) |
| (3) | 016630 | 010600 | | | | MOV | SP,RO |
| (4) | 016632 | 104415 | | | | TRAP | C\$PNTX |
| (4) | 016634 | 062706 | 000010 | | | ADD | #10,SP |
| 7694 | 016640 | | | PRINTX | #FMT5,AX0.15,AX0.16,AX1.15,AX1.16 | | |
| (11) | 016640 | 013746 | 002330 | | | MOV | AX1.16,-(SP) |
| (10) | 016644 | 013746 | 002326 | | | MOV | AX1.15,-(SP) |
| (9) | 016650 | 013746 | 002324 | | | MOV | AX0.16,-(SP) |
| (8) | 016654 | 013746 | 002322 | | | MOV | AX0.15,-(SP) |
| (7) | 016660 | 012746 | 011535 | | | MOV | #FMT5,-(SP) |

(6) 016664 012746 000005
(3) 016670 010600
(4) 016672 104415
(4) 016674 062706 000014
7695 016700
(8) 016700 012746 014510
(7) 016704 012746 011664
(6) 016710 012746 000002
(3) 016714 010600
(4) 016716 104415
(4) 016720 062706 000006
7696 016724
(11) 016724 013746 002340
(10) 016730 013746 002336
(9) 016734 013746 002334
(8) 016740 013746 002332
(7) 016744 012746 011565
(6) 016750 012746 000005
(3) 016754 010600
(4) 016756 104415
(4) 016760 062706 000014
7697 016764
(3) 016764
(3) 016764 104423
7698
7699
7700
7701
7702
7703 016766
(3) 016766
7704 016766
(8) 016766 013746 002352
(7) 016772 012746 011671
(6) 016776 012746 000002
(3) 017002 010600
(4) 017004 104414
(4) 017006 062706 000006
7705 017012
(9) 017012 013746 002446
(8) 017016 012746 035560
(7) 017022 012746 011426
(6) 017026 012746 000003
(3) 017032 010600
(4) 017034 104414
(4) 017036 062706 000010
7706 017042
(7) 017042 012746 011436
(6) 017046 012746 000001
(3) 017052 010600
(4) 017054 104414
(4) 017056 062706 000004
7707 017062
(9) 017062 013746 002530
(8) 017066 013746 002532
(7) 017072 012746 011630

PRINTX #FMT9,#DH9

PRINTX #FMT6,AX2.15,AX2.16,AX3.15,AX3.16

ENDMSG

BGNMSG ERR6

PRINTB #FMT10,SUBRPC

PRINTB #FMT1,#ADDRES,MPCSR

PRINTB #FMT2

PRINTB #FMT8,TMP1,TMP0

MOV #5,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #14,SP

MOV #DH9,-(SP)
MOV #FMT9,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #6,SP

MOV AX3.16,-(SP)
MOV AX3.15,-(SP)
MOV AX2.16,-(SP)
MOV AX2.15,-(SP)
MOV #FMT6,-(SP)
MOV #5,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #14,SP

L10006:

TRAP C\$MSG

ERR6::

MOV SUBRPC,-(SP)
MOV #FMT10,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C\$PNTB
ADD #6,SP

MOV MPCSR,-(SP)
MOV #ADDRES,-(SP)
MOV #FMT1,-(SP)
MOV #3,-(SP)
MOV SP,R0
TRAP C\$PNTB
ADD #10,SP

MOV #FMT2,-(SP)
MOV #1,-(SP)
MOV SP,R0
TRAP C\$PNTB
ADD #4,SP

MOV TMP0,-(SP)
MOV TMP1,-(SP)
MOV #FMT8,-(SP)

| | | | | | | |
|------|--------|--------|--------|--------|-----------------------------------|--------------|
| (6) | 017076 | 012746 | 000003 | | MOV | #3,-(SP) |
| (3) | 017102 | 010600 | | | MOV | SP,R0 |
| (4) | 017104 | 104414 | | | TRAP | C\$PNTB |
| (4) | 017106 | 062706 | 000010 | | ADD | #10,SP |
| 7708 | 017112 | | | PRINTX | #FMT4,#DH2,#DH3 | |
| (9) | 017112 | 012746 | 014311 | | MOV | #DH3,-(SP) |
| (8) | 017116 | 012746 | 014262 | | MOV | #DH2,-(SP) |
| (7) | 017122 | 012746 | 011522 | | MOV | #FMT4,-(SP) |
| (6) | 017126 | 012746 | 000003 | | MOV | #3,-(SP) |
| (3) | 017132 | 010600 | | | MOV | SP,R0 |
| (4) | 017134 | 104415 | | | TRAP | C\$PNTX |
| (4) | 017136 | 062706 | 000010 | | ADD | #10,SP |
| 7709 | 017142 | | | PRINTX | #FMT5,LUR10,LUR11,LUR12,LUR13 | |
| (11) | 017142 | 013746 | 002310 | | MOV | LUR13,-(SP) |
| (10) | 017146 | 013746 | 002306 | | MOV | LUR12,-(SP) |
| (9) | 017152 | 013746 | 002304 | | MOV | LUR11,-(SP) |
| (8) | 017156 | 013746 | 002302 | | MOV | LUR10,-(SP) |
| (7) | 017162 | 012746 | 011535 | | MOV | #FMT5,-(SP) |
| (6) | 017166 | 012746 | 000005 | | MOV | #5,-(SP) |
| (3) | 017172 | 010600 | | | MOV | SP,R0 |
| (4) | 017174 | 104415 | | | TRAP | C\$PNTX |
| (4) | 017176 | 062706 | 000014 | | ADD | #14,SP |
| 7710 | 017202 | | | PRINTX | #FMT9,#DH4 | |
| (8) | 017202 | 012746 | 014347 | | MOV | #DH4,-(SP) |
| (7) | 017206 | 012746 | 011664 | | MOV | #FMT9,-(SP) |
| (6) | 017212 | 012746 | 000002 | | MOV | #2,-(SP) |
| (3) | 017216 | 010600 | | | MOV | SP,R0 |
| (4) | 017220 | 104415 | | | TRAP | C\$PNTX |
| (4) | 017222 | 062706 | 000006 | | ADD | #6,SP |
| 7711 | 017226 | | | PRINTX | #FMT6,LUR14,LUR15,LUR16,LUR17 | |
| (11) | 017226 | 013746 | 002320 | | MOV | LUR17,-(SP) |
| (10) | 017232 | 013746 | 002316 | | MOV | LUR16,-(SP) |
| (9) | 017236 | 013746 | 002314 | | MOV | LUR15,-(SP) |
| (8) | 017242 | 013746 | 002312 | | MOV | LUR14,-(SP) |
| (7) | 017246 | 012746 | 011565 | | MOV | #FMT6,-(SP) |
| (6) | 017252 | 012746 | 000005 | | MOV | #5,-(SP) |
| (3) | 017256 | 010600 | | | MOV | SP,R0 |
| (4) | 017260 | 104415 | | | TRAP | C\$PNTX |
| (4) | 017262 | 062706 | 000014 | | ADD | #14,SP |
| 7712 | 017266 | | | PRINTX | #FMT4,#DH7,#DH8 | |
| (9) | 017266 | 012746 | 014451 | | MOV | #DH8,-(SP) |
| (8) | 017272 | 012746 | 014417 | | MOV | #DH7,-(SP) |
| (7) | 017276 | 012746 | 011522 | | MOV | #FMT4,-(SP) |
| (6) | 017302 | 012746 | 000003 | | MOV | #3,-(SP) |
| (3) | 017306 | 010600 | | | MOV | SP,R0 |
| (4) | 017310 | 104415 | | | TRAP | C\$PNTX |
| (4) | 017312 | 062706 | 000010 | | ADD | #10,SP |
| 7713 | 017316 | | | PRINTX | #FMT5,AX0.15,AX0.16,AX1.15,AX1.16 | |
| (11) | 017316 | 013746 | 002330 | | MOV | AX1.16,-(SP) |
| (10) | 017322 | 013746 | 002326 | | MOV | AX1.15,-(SP) |
| (9) | 017326 | 013746 | 002324 | | MOV | AX0.16,-(SP) |
| (8) | 017332 | 013746 | 002322 | | MOV | AX0.15,-(SP) |
| (7) | 017336 | 012746 | 011535 | | MOV | #FMT5,-(SP) |
| (6) | 017342 | 012746 | 000005 | | MOV | #5,-(SP) |
| (3) | 017346 | 010600 | | | MOV | SP,R0 |
| (4) | 017350 | 104415 | | | TRAP | C\$PNTX |

(4) 017352 062706 000014
7714 017356
(8) 017356 012746 014510
(7) 017362 012746 011664
(6) 017366 012746 000002
(3) 017372 010600
(4) 017374 104415
(4) 017376 062706 000006
7715 017402
(11) 017402 013746 002340
(10) 017406 013746 002336
(9) 017412 013746 002334
(8) 017416 013746 002332
(7) 017422 012746 011565
(6) 017426 012746 000005
(3) 017432 010600
(4) 017434 104415
(4) 017436 062706 000014
7716 017442
(3) 017442
(3) 017442 104423
7717
7718
7719
7720
7721
7722 017444
(3) 017444
7723 017444
(9) 017444 013746 002446
(8) 017450 012746 035560
(7) 017454 012746 011426
(6) 017460 012746 000003
(3) 017464 010600
(4) 017466 104414
(4) 017470 062706 000010
7724 017474
(7) 017474 012746 011436
(6) 017500 012746 000001
(3) 017504 010600
(4) 017506 104414
(4) 017510 062706 000004
7725 017514
(9) 017514 013746 002400
(8) 017520 012746 014240
(7) 017524 012746 011620
(6) 017530 012746 000003
(3) 017534 010600
(4) 017536 104414
(4) 017540 062706 000010
7726 017544
(9) 017544 012746 014311
(8) 017550 012746 014262
(7) 017554 012746 011522
(6) 017560 012746 000003
(3) 017564 010600

PRINTX #FMT9,#DH9

PRINTX #FMT6,AX2.15,AX2.16,AX3.15,AX3.16

ENDMSG

BGNMSG ERR7

PRINTB #FMT1,#ADDRFS,MPCSR

PRINTB #FMT2

PRINTB #FMT7,#DH1,REGNUM

PRINTX #FMT4,#DH2,#DH3

ADD #14,SP
MOV #DH9,-(SP)
MOV #FMT9,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #6,SP
MOV AX3.16,-(SP)
MOV AX3.15,-(SP)
MOV AX2.16,-(SP)
MOV AX2.15,-(SP)
MOV #FMT6,-(SP)
MOV #5,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #14,SP
L10007: TRAP C\$MSG
ERR7::
MOV MPCSR,-(SP)
MOV #ADDRFS,-(SP)
MOV #FMT1,-(SP)
MOV #3,-(SP)
MOV SP,R0
TRAP C\$PNTB
ADD #10,SP
MOV #FMT2,-(SP)
MOV #1,-(SP)
MOV SP,R0
TRAP C\$PNTB
ADD #4,SP
MOV REGNUM,-(SP)
MOV #DH1,-(SP)
MOV #FMT7,-(SP)
MOV #3,-(SP)
MOV SP,R0
TRAP C\$PNTB
ADD #10,SP
MOV #DH3,-(SP)
MOV #DH2,-(SP)
MOV #FMT4,-(SP)
MOV #3,-(SP)
MOV SP,R0

| | | | | | | |
|------|--------|--------|--------|--|------|--------------|
| (4) | 017566 | 104415 | | | TRAP | C\$PNTX |
| (4) | 017570 | 062706 | 000010 | | ADD | #10,SP |
| 7727 | 017574 | | | PRINTX #FMT5,LUR10,LUR11,LUR12,LUR13 | | |
| (11) | 017574 | 013746 | 002310 | | MOV | LUR13,-(SP) |
| (10) | 017600 | 013746 | 002306 | | MOV | LUR12,-(SP) |
| (9) | 017604 | 013746 | 002304 | | MOV | LUR11,-(SP) |
| (8) | 017610 | 013746 | 002302 | | MOV | LUR10,-(SP) |
| (7) | 017614 | 012746 | 011535 | | MOV | #FMT5,-(SP) |
| (6) | 017620 | 012746 | 000005 | | MOV | #5,-(SP) |
| (3) | 017624 | 010600 | | | MOV | SP,R0 |
| (4) | 017626 | 104415 | | | TRAP | C\$PNTX |
| (4) | 017630 | 062706 | 000014 | | ADD | #14,SP |
| 7728 | 017634 | | | PRINTX #FMT9,#DH4 | | |
| (8) | 017634 | 012746 | 014347 | | MOV | #DH4,-(SP) |
| (7) | 017640 | 012746 | 011664 | | MOV | #FMT9,-(SP) |
| (6) | 017644 | 012746 | 000002 | | MOV | #2,-(SP) |
| (3) | 017650 | 010600 | | | MOV | SP,R0 |
| (4) | 017652 | 104415 | | | TRAP | C\$PNTX |
| (4) | 017654 | 062706 | 000006 | | ADD | #6,SP |
| 7729 | 017660 | | | PRINTX #FMT6,LUR14,LUR15,LUR16,LUR17 | | |
| (11) | 017660 | 013746 | 002320 | | MOV | LUR17,-(SP) |
| (10) | 017664 | 013746 | 002316 | | MOV | LUR16,-(SP) |
| (9) | 017670 | 013746 | 002314 | | MOV | LUR15,-(SP) |
| (8) | 017674 | 013746 | 002312 | | MOV | LUR14,-(SP) |
| (7) | 017700 | 012746 | 011565 | | MOV | #FMT6,-(SP) |
| (6) | 017704 | 012746 | 000005 | | MOV | #5,-(SP) |
| (3) | 017710 | 010600 | | | MOV | SP,R0 |
| (4) | 017712 | 104415 | | | TRAP | C\$PNTX |
| (4) | 017714 | 062706 | 000014 | | ADD | #14,SP |
| 7730 | 017720 | | | PRINTX #FMT4,#DH7,#DH8 | | |
| (9) | 017720 | 012746 | 014451 | | MOV | #DH8,-(SP) |
| (8) | 017724 | 012746 | 014417 | | MOV | #DH7,-(SP) |
| (7) | 017730 | 012746 | 011522 | | MOV | #FMT4,-(SP) |
| (6) | 017734 | 012746 | 000003 | | MOV | #3,-(SP) |
| (3) | 017740 | 010600 | | | MOV | SP,R0 |
| (4) | 017742 | 104415 | | | TRAP | C\$PNTX |
| (4) | 017744 | 062706 | 000010 | | ADD | #10,SP |
| 7731 | 017750 | | | PRINTX #FMT5,AX0.15,AX0.16,AX1.15,AX1.16 | | |
| (11) | 017750 | 013746 | 002330 | | MOV | AX1.16,-(SP) |
| (10) | 017754 | 013746 | 002326 | | MOV | AX1.15,-(SP) |
| (9) | 017760 | 013746 | 002324 | | MOV | AX0.16,-(SP) |
| (8) | 017764 | 013746 | 002322 | | MOV | AX0.15,-(SP) |
| (7) | 017770 | 012746 | 011535 | | MOV | #FMT5,-(SP) |
| (6) | 017774 | 012746 | 000005 | | MOV | #5,-(SP) |
| (3) | 020000 | 010600 | | | MOV | SP,R0 |
| (4) | 020002 | 104415 | | | TRAP | C\$PNTX |
| (4) | 020004 | 062706 | 000014 | | ADD | #14,SP |
| 7732 | 020010 | | | PRINTX #FMT9,#DH9 | | |
| (8) | 020010 | 012746 | 014510 | | MOV | #DH9,-(SP) |
| (7) | 020014 | 012746 | 011664 | | MOV | #FMT9,-(SP) |
| (6) | 020020 | 012746 | 000002 | | MOV | #2,-(SP) |
| (3) | 020024 | 010600 | | | MOV | SP,R0 |
| (4) | 020026 | 104415 | | | TRAP | C\$PNTX |
| (4) | 020030 | 062706 | 000006 | | ADD | #6,SP |
| 7733 | 020034 | | | PRINTX #FMT6,AX2.15,AX2.16,AX3.15,AX3.16 | | |
| (11) | 020034 | 013746 | 002340 | | MOV | AX3.16,-(SP) |

| | | | | | | | |
|------|--------|--------|--------|--------|---------------------|--------|---------------|
| (10) | 020040 | 013746 | 002336 | | | MOV | AX3.15,-(SP) |
| (9) | 020044 | 013746 | 002334 | | | MOV | AX2.16,-(SP) |
| (8) | 020050 | 013746 | 002332 | | | MOV | AX2.15,-(SP) |
| (7) | 020054 | 012746 | 011565 | | | MOV | #FMT6,-(SP) |
| (6) | 020060 | 012746 | 000005 | | | MOV | #5,-(SP) |
| (3) | 020064 | 010600 | | | | MOV | SP,R0 |
| (4) | 020066 | 104415 | | | | TRAP | C\$PNTX |
| (4) | 020070 | 062706 | 000014 | | | ADD | #14,SP |
| 7734 | 020074 | | | ENDMSG | | | |
| (3) | 020074 | | | | L10010: | | |
| (3) | 020074 | 104423 | | | TRAP | C\$MSG | |
| 7735 | | | | | | | |
| 7736 | | | | | | | |
| 7737 | | | | | | | |
| 7738 | | | | | | | |
| 7739 | | | | | | | |
| 7740 | 020076 | | | BGNMSG | ERR8 | | |
| (3) | 020076 | | | | | | |
| 7741 | 020076 | | | PPINTB | #FMT10,SUBRPC | ERR8:: | |
| (8) | 020076 | 013746 | 002352 | | | MOV | SUBRPC,-(SP) |
| (7) | 020102 | 012746 | 011671 | | | MOV | #FMT10,-(SP) |
| (6) | 020106 | 012746 | 000002 | | | MOV | #2,-(SP) |
| (3) | 020112 | 010600 | | | | MOV | SP,R0 |
| (4) | 020114 | 104414 | | | | TRAP | C\$PNTB |
| (4) | 020116 | 062706 | 000006 | | | ADD | #6,SP |
| 7742 | 020122 | | | PRINTB | #FMT1,#ADDRES,MPCSR | | |
| (9) | 020122 | 013746 | 002446 | | | MOV | MPCSR,-(SP) |
| (8) | 020126 | 012746 | 035560 | | | MOV | #ADDRES,-(SP) |
| (7) | 020132 | 012746 | 011426 | | | MOV | #FMT1,-(SP) |
| (6) | 020136 | 012746 | 000003 | | | MOV | #3,-(SP) |
| (3) | 020142 | 010600 | | | | MOV | SP,R0 |
| (4) | 020144 | 104414 | | | | TRAP | C\$PNTB |
| (4) | 020146 | 062706 | 000010 | | | ADD | #10,SP |
| 7743 | 020152 | | | PRINTB | #FMT2 | | |
| (7) | 020152 | 012746 | 011436 | | | MOV | #FMT2,-(SP) |
| (6) | 020156 | 012746 | 000001 | | | MOV | #1,-(SP) |
| (3) | 020162 | 010600 | | | | MOV | SP,R0 |
| (4) | 020164 | 104414 | | | | TRAP | C\$PNTB |
| (4) | 020166 | 062706 | 000004 | | | ADD | #4,SP |
| 7744 | 020172 | | | PRINTB | #FMT7,#DH1,REGNUM | | |
| (9) | 020172 | 013746 | 002400 | | | MOV | REGNUM,-(SP) |
| (8) | 020176 | 012746 | 014240 | | | MOV | #DH1,-(SP) |
| (7) | 020202 | 012746 | 011620 | | | MOV | #FMT7,-(SP) |
| (6) | 020206 | 012746 | 000003 | | | MOV | #3,-(SP) |
| (3) | 020212 | 010600 | | | | MOV | SP,R0 |
| (4) | 020214 | 104414 | | | | TRAP | C\$PNTB |
| (4) | 020216 | 062706 | 000010 | | | ADD | #10,SP |
| 7745 | 020222 | | | PRINTB | #FMT3,GOODAT,BADDAT | | |
| (9) | 020222 | 013746 | 002406 | | | MOV | BADDAT,-(SP) |
| (8) | 020226 | 013746 | 002404 | | | MOV | GOODAT,-(SP) |
| (7) | 020232 | 012746 | 011460 | | | MOV | #FMT3,-(SP) |
| (6) | 020236 | 012746 | 000003 | | | MOV | #3,-(SP) |
| (3) | 020242 | 010600 | | | | MOV | SP,R0 |
| (4) | 020244 | 104414 | | | | TRAP | C\$PNTB |
| (4) | 020246 | 062706 | 000010 | | | ADD | #10,SP |
| 7746 | 020252 | | | PRINTX | #FMT4,#DH2,#DH3 | | |

(9) 020252 012746 014311
(8) 020256 012746 014262
(7) 020262 012746 011522
(6) 020266 012746 000003
(3) 020272 010600
(4) 020274 104415
(4) 020276 062706 000010
7747 020302
(11) 020302 013746 002310
(10) 020306 013746 002306
(9) 020312 013746 002304
(8) 020316 013746 002302
(7) 020322 012746 011535
(6) 020326 012746 000005
(3) 020332 010600
(4) 020334 104415
(4) 020336 062706 000014
7748 020342
(8) 020342 012746 014347
(7) 020346 012746 011664
(6) 020352 012746 000002
(3) 020356 010600
(4) 020360 104415
(4) 020362 062706 000006
7749 020366
(11) 020366 013746 002320
(10) 020372 013746 002316
(9) 020376 013746 002314
(8) 020402 013746 002312
(7) 020406 012746 011565
(6) 020412 012746 000005
(3) 020416 010600
(4) 020420 104415
(4) 020422 062706 000014
7750 020426
(9) 020426 012746 014451
(8) 020432 012746 014417
(7) 020436 012746 011522
(6) 020442 012746 000003
(3) 020446 010600
(4) 020450 104415
(4) 020452 062706 000010
7751 020456
(11) 020456 013746 002330
(10) 020462 013746 002326
(9) 020466 013746 002324
(8) 020472 013746 002322
(7) 020476 012746 011535
(6) 020502 012746 000005
(3) 020506 010600
(4) 020510 104415
(4) 020512 062706 000014
7752 020516
(8) 020516 012746 014510
(7) 020522 012746 011664
(6) 020526 012746 000002

PRINTX #FMT5,LUR10,LUR11,LUR12,LUR13

PRINTX #FMT9,#DH4

PRINTX #FMT6,LUR14,LUR15,LUR16,LUR17

PRINTX #FMT4,#DH7,#DH8

PRINTX #FMT5,AX0.15,AX0.16,AX1.15,AX1.16

PRINTX #FMT9,#DH9

MOV #DH3,-(SP)
MOV #DH2,-(SP)
MOV #FMT4,-(SP)
MOV #3,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #10,SP

MOV LUR13,-(SP)
MOV LUR12,-(SP)
MOV LUR11,-(SP)
MOV LUR10,-(SP)
MOV #FMT5,-(SP)
MOV #5,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #14,SP

MOV #DH4,-(SP)
MOV #FMT9,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #6,SP

MOV LUR17,-(SP)
MOV LUR16,-(SP)
MOV LUR15,-(SP)
MOV LUR14,-(SP)
MOV #FMT6,-(SP)
MOV #5,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #14,SP

MOV #DH8,-(SP)
MOV #DH7,-(SP)
MOV #FMT4,-(SP)
MOV #3,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #10,SP

MOV AX1.16,-(SP)
MOV AX1.15,-(SP)
MOV AX0.16,-(SP)
MOV AX0.15,-(SP)
MOV #FMT5,-(SP)
MOV #5,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #14,SP

MOV #DH9,-(SP)
MOV #FMT9,-(SP)
MOV #2,-(SP)

| | | | | | | | |
|------|--------|--------|--------|--------|-----------------------------------|---------|---------------|
| (3) | 020532 | 010600 | | | | MOV | SP,R0 |
| (4) | 020534 | 104415 | | | | TRAP | C\$PNTX |
| (4) | 020536 | 062706 | 000006 | | | ADD | #6,SP |
| 7753 | 020542 | | | PRINTX | #FMT6,AX2.15,AX2.16,AX3.15,AX3.16 | | |
| (11) | 020542 | 013746 | 002340 | | | MOV | AX3.16,-(SP) |
| (10) | 020546 | 013746 | 002336 | | | MOV | AX3.15,-(SP) |
| (9) | 020552 | 013746 | 002334 | | | MOV | AX2.16,-(SP) |
| (8) | 020556 | 013746 | 002332 | | | MOV | AX2.15,-(SP) |
| (7) | 020562 | 012746 | 011565 | | | MOV | #FMT6,-(SP) |
| (6) | 020566 | 012746 | 000005 | | | MOV | #5,-(SP) |
| (3) | 020572 | 010600 | | | | MOV | SP,R0 |
| (4) | 020574 | 104415 | | | | TRAP | C\$PNTX |
| (4) | 020576 | 062706 | 000014 | | | ADD | #14,SP |
| 7754 | 020602 | | | ENDMSG | | | |
| (3) | 020602 | | | | | | |
| (3) | 020602 | 104423 | | | | L10011: | TRAP C\$MSG |
| 7755 | | | | | | | |
| 7756 | | | | | | | |
| 7757 | | | | | | | |
| 7758 | | | | | | | |
| 7759 | | | | | | | |
| 7760 | 020604 | | | BGNMSG | ERR9 | | |
| (3) | 020604 | | | | | | |
| 7761 | 020604 | | | PRINTB | #FMT1,#ADDRES,MPCSR | | |
| (9) | 020604 | 013746 | 002446 | | | MOV | MPCSR,-(SP) |
| (8) | 020610 | 012746 | 035560 | | | MOV | #ADDRES,-(SP) |
| (7) | 020614 | 012746 | 011426 | | | MOV | #FMT1,-(SP) |
| (6) | 020620 | 012746 | 000003 | | | MOV | #3,-(SP) |
| (3) | 020624 | 010600 | | | | MOV | SP,R0 |
| (4) | 020626 | 104414 | | | | TRAP | C\$PNTB |
| (4) | 020630 | 062706 | 000010 | | | ADD | #10,SP |
| 7762 | 020634 | | | PRINTB | #FMT2 | | |
| (7) | 020634 | 012746 | 011436 | | | MOV | #FMT2,-(SP) |
| (6) | 020640 | 012746 | 000001 | | | MOV | #1,-(SP) |
| (3) | 020644 | 010600 | | | | MOV | SP,R0 |
| (4) | 020646 | 104414 | | | | TRAP | C\$PNTB |
| (4) | 020650 | 062706 | 000004 | | | ADD | #4,SP |
| 7763 | 020654 | | | PRINTB | #FMT7,#DH1,REGNUM | | |
| (9) | 020654 | 013746 | 002400 | | | MOV | REGNUM,-(SP) |
| (8) | 020660 | 012746 | 014240 | | | MOV | #DH1,-(SP) |
| (7) | 020664 | 012746 | 011620 | | | MOV | #FMT7,-(SP) |
| (6) | 020670 | 012746 | 000003 | | | MOV | #3,-(SP) |
| (3) | 020674 | 010600 | | | | MOV | SP,R0 |
| (4) | 020676 | 104414 | | | | TRAP | C\$PNTB |
| (4) | 020700 | 062706 | 000010 | | | ADD | #10,SP |
| 7764 | 020704 | | | PRINTX | #FMT4,#DH2,#DH3 | | |
| (9) | 020704 | 012746 | 014311 | | | MOV | #DH3,-(SP) |
| (8) | 020710 | 012746 | 014262 | | | MOV | #DH2,-(SP) |
| (7) | 020714 | 012746 | 011522 | | | MOV | #FMT4,-(SP) |
| (6) | 020720 | 012746 | 000003 | | | MOV | #3,-(SP) |
| (3) | 020724 | 010600 | | | | MOV | SP,R0 |
| (4) | 020726 | 104415 | | | | TRAP | C\$PNTX |
| (4) | 020730 | 062706 | 000010 | | | ADD | #10,SP |
| 7765 | 020734 | | | PRINTX | #FMT5,LUR10,LUR11,LUR12,LUR13 | | |
| (11) | 020734 | 013746 | 002310 | | | MOV | LUR13,-(SP) |
| (10) | 020740 | 013746 | 002306 | | | MOV | LUR12,-(SP) |

(9) 020744 013746 002304
(8) 020750 013746 002302
(7) 020754 012746 011535
(6) 020760 012746 000005
(3) 020764 010600
(4) 020766 104415
(4) 020770 062706 000014
7766 020774
(8) 020774 012746 014347
(7) 021000 012746 011664
(6) 021004 012746 000002
(3) 021010 010600
(4) 021012 104415
(4) 021014 062706 000006
7767 021020
(11) 021020 013746 002320
(10) 021024 013746 002316
(9) 021030 013746 002314
(8) 021034 013746 002312
(7) 021040 012746 011565
(6) 021044 012746 000005
(3) 021050 010600
(4) 021052 104415
(4) 021054 062706 000014
7768 021060
(3) 021060
(3) 021060 104423
7769
7770
7771
7772
7773

PRINTX #FMT9,#DH4

PRINTX #FMT6,LUR14,LUR15,LUR16,LUR17

ENDMSG

MOV LUR11,-(SP)
MOV LUR10,-(SP)
MOV #FMT5,-(SP)
MOV #5,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #14,SP

MOV #DH4,-(SP)
MOV #FMT9,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #6,SP

MOV LUR17,-(SP)
MOV LUR16,-(SP)
MOV LUR15,-(SP)
MOV LUR14,-(SP)
MOV #FMT6,-(SP)
MOV #5,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #14,SP

L10012:

TRAP C\$MSG

7775
7776
7777
7778
7779
7780
7781
7782
7783
(3)
7784
7785
(3)
(3)
7786
7787
7788
7789
7790

.SBTTL REPORT CODING SECTION

:/ THE REPORT CODING SECTION CONTAINS THE
:/ 'PRINTS' CALLS THAT GENERATE STATISTICAL REPORTS.

021062
021062
021062
021062
021062 104425

BGNRPT
ENDRPT

LSRPT::
L10013: TRAP CSRPT

7792
7793
7794
7795
7796
7797
7798
7799
7800
7801
7802
7803
7804
7805
7806
7807
7808

021064
(3) 021064
021064 177777
021066 177777
021070 177777
021072

.SBTTL LOAD DEVICE PROTECTION TABLE
:////////////////////
:/ THIS TABLE IDENTIFIES THE LOAD DEVICE TO THE SUPERVISOR, SO THAT IT CAN BE
:/ PROTECTED FROM TESTING, IF DESIRED.
:////////////////////

BGNPROT
.WORD -1 :DON'T CHK CSR ADRS
.WORD -1 :DON'T CHK MASSBUS UNIT NO.
.WORD -1 :DON'T CHK DRIVE NO.
ENDPROT
L\$PROT::

```
7810 .SBTTL INITIALIZE SECTION
7811
7812 ;////////////////////////////////////
7813 ;/ THE INITIALIZE SECTION CONTAINS THE CODING THAT IS PERFORMED
7814 ;/ AT THE BEGINNING OF THE TEST SEQUENCE ON THE NEXT UNIT.
7815 ;////////////////////////////////////
7816
7817 021072 BGNINIT
(3) 021072 LSINIT::
7818
7819 021072 010637 002346 MOV SP,PSTACK ;SAVE BASE-LEVEL STACK POINTER
7820 021076 005037 002352 CLR SUBRPC ;CLEAR SUBR CALL PC
7821 021102 005037 002426 CLR DISILO ;CLEAR CURRENT STATE OF DISSI
7822 021106 005037 002430 CLR CHPTYP ;CLEAR USYRT CHIP TYPE INDICATOR
7823 021112 005037 002420 CLR ERROR1 ;CLEAR ERROR FLAG
7824 021116 005037 002432 CLR SAVLEN ;CLEAR CHAR LENGTH FROM SETUP
7825 021122 005737 002412 TST FRSTIM ;SEE IF FIRST TIME THROUGH AFTER LOAD
7826 021126 001007 BNE 6$ ;BR IF NOT
7827 021130 013737 000004 002414 MOV @#4,SAVE4 ;SAVE ERROR TRAP VECTOR
7828 021136 013737 000006 002416 MOV @#6,SAVE6
7829 021144 000406 BR 9$
7830 021146 013737 002414 000004 6$: MOV SAVE4,@#4 ;RESTORE ERROR TRAP VECTOR
7831 021154 013737 002416 000006 MOV SAVE6,@#6
7832 021162 012737 000001 002412 9$: MOV #1,FRSTIM ;MARK FLAG FOR NEXT TIME THROUGH
7833 ;SEE IF PROGRAM JUST STARTED, BR IF YES
7834 021170 READEF #EF.START
(3) 021170 012700 000040 MOV #EF.START,R0
(3) 021174 104447 TRAP C$REFG
7835 021176 BCOMPLETE STARST BCS STARST
(2) 021176 103415 ;SEE IF PROGRAM JUST RESTARTED, BR IF YES
7836 ;SEE IF PROGRAM JUST RESTARTED, BR IF YES
7837 021200 READEF #EF.RESTART
(3) 021200 012700 000037 MOV #EF.RESTART,R0
(3) 021204 104447 TRAP C$REFG
7838 021206 BCOMPLETE STARST BCS STARST
(2) 021206 103411 ;SEE IF THIS IS A NEW PASS, BR IF YES
7839 ;SEE IF THIS IS A NEW PASS, BR IF YES
7840 021210 READEF #EF.NEW
(3) 021210 012700 000035 MOV #EF.NEW,R0
(3) 021214 104447 TRAP C$REFG
7841 021216 BCOMPLETE NEWST BCS NEWST
(2) 021216 103411 ;SEE IF PROGRAM WAS JUST CONTINUED
7842 ;SEE IF PROGRAM WAS JUST CONTINUED
7843 021220 READEF #EF.CONTINUE
(3) 021220 012700 000036 MOV #EF.CONTINUE,R0
(3) 021224 104447 TRAP C$REFG
7844 021226 BCOMPLETE ENDIT BCS ENDIT
(2) 021226 103504
7845 021230 000414 BR GETPRM
7846 021232 STARST:
7847 021232 005037 002444 CLR STARES ;CLEAR FLAG TO SHOW JUST HAD STA OR RES
7848 ;CLEAR DEVICE MAP
7849 021236 005037 002434 CLR DEVMAP
7850 021242 NEWST:
7851 021242 012737 177777 002344 MOV #-1,LOGDEV ;RESET LOGICAL DEVICE TO -1
7852 021250 005237 002444 INC STARES ;INCR NO. OF PASSES SINCE STA OR RES
```

```
7853 021254 012737 000001 002436      MOV    #BIT0,DEVPTR      ;INIT DEVICE MAP BIT POINTER
7854                                     ; GET UNIBUS ADDRESS, VECTOR, PRIORITY LEVEL, SWITCH PACKS, TEST
7855                                     ; CONNECTOR INFORMATION FOR THIS LOGICAL DEVICE
7856 021262                                     GETPRM:
7857 021262 005237 002344      INC    LOGDEV            ;INCREMENT LOGICAL DEVICE NUMBER
7858 021266 023737 002344 002012      CMP    LOGDEV,L$UNIT    ;SEE IF MAXIMUM UNIT NO. EXCEEDED
7859 021274 002362                                     BGE    NEWST             ;BR IF YES
7860 021276                                     GPWARD LOGDEV,R1       ;GET P-TABLE POINTER INTO R1
(3) 021276 013700 002344                                     MOV    LOGDEV,RO
(3) 021302 104442                                     TRAP   C$GPHRD
(3) 021304 010001                                     MOV    RO,R1
7861 021306      BCOMPLETE      10$      ;BR IF DEVICE AVAILABLE
(2) 021306 103403                                     BCS   10$
7862 021310 006337 002436      ASL    DEVPTR           ;SHIFT DEVICE POINTER
7863 021314 000762      BR     GETPRM          ;SKIP THIS DEVICE
7864 021316 053737 002436 002434 10$:  BIS    DEVPTR,DEVMAP   ;SET BIT FOR THIS DEVICE
7865 021324 006337 002436      ASL    DEVPTR           ;SHIFT BIT POINTER
7866 021330 062701 000002      ADD    #2,R1           ;INCREMENT R1 PAST MICROPROCESSOR TYPE
7867 021334 011137 002446      MOV    (R1),MPCSR      ;STORE POINTER TO MICROPROCESSOR CSR'S
7868 021340 011137 002450      MOV    (R1),BSEL1
7869 021344 005237 002450      INC    BSEL1           ;GET POINTER TO BSEL1 (MAINTENANCE REGISTER)
7870 021350 013737 002450 002452      MOV    BSEL1,BSEL2
7871 021356 005237 002452      INC    BSEL2           ;GET POINTER TO BSEL2
7872 021362 011137 002454      MOV    (R1),SEL4
7873 021366 062737 000004 002454      ADD    #4,SEL4         ;GET POINTER TO SEL4
7874 021374 012137 002456      MOV    (R1)+,SEL6
7875 021400 062737 000006 002456      ADD    #6,SEL6         ;STORE POINTER TO SEL6
7876 021406 011137 002460      MOV    (R1),MPIVEC     ;GET MICROPROCESSOR INPUT INTRPT VECTOR
7877 021412 012137 002462      MOV    (R1)+,MPOVEC
7878 021416 062737 000004 002462      ADD    #4,MPOVEC       ;GET MICROPROCESSOR OUTPUT INTRPT VECTOR
7879 021424 012137 002464      MOV    (R1)+,MPRIOR    ;GET MICROPROCESSOR DEVICE PRIORITY
7880 021430 062701 000014      ADD    #14,R1          ;POINT R1 TO RUN SWITCH INDICATOR
7881 021434 011137 002476      MOV    (R1),RUNINH     ;GET STATE OF MICROPROCESSOR RUN SWITCH
7882 021440      ENDIT:
7883 021440      ENDINIT
(3) 021440                                     L10015:
(3) 021440 104411                                     TRAP   C$INIT
7884
7885
7886
7887
7888
7889
```

```
7891 .SBTTL AUTO DROP UNIT SECTION
7892
7893 :////////////////////////////////////////////////////////////////////
7894 :// THE AUTO DROP CODING DETERMINES WHETHER OR NOT THE DEVICE WHOSE P-TABLE
7895 :// WAS JUST OBTAINED IS READY FOR TESTING, AND IT IS DROPPED IF NOT READY.
7896 :////////////////////////////////////////////////////////////////////
7897 021442 BGNAUTO
(3) 021442
7898
7899 :ESTABLISH CPU PRIORITY = 7
021442 SETPRI #PRI07
(3) 021442 012700 000340
(3) 021446 104441
7900 021450 012737 021472 000004 MOV #6$,@#4 ;SET UP NON-EXISTENT MEMORY ERROR TRAP VECTOR
7901 021456 012737 000340 000006 MOV #PRI07,@#6 TRAP #PRI07,RO
7902 021464 005777 160756 TST @MPCSR ;ADDRESS SELO
7903 021470 000405 BR 9$ ;TAKE THIS BR IF DEVICE RESPONDS
7904
7905 021472 062706 000004 ;COME HERE IF DEVICE CSR IS NON-EXISTENT
6$: ADD #4,SP ;CLEAN UP THE STACK POINTER
(3) 021476 013700 002344 DODU LOGDEV ;DROP THIS UNIT FROM TESTING
(3) 021502 104451
7907 021504 013737 002414 000004 9$: MOV SAVE4,@#4 ;RESTORE ERROR TRAP VECTOR
7908 021512 013737 002416 000006 MOV SAVE6,@#6 TRAP LOGDEV,RO
7909 021520
(3) 021520
(3) 021520 104461 L10016: TRAP C$DODU
7910
7911
7912
7913
7914
```

7916
7917
7918
7919
7920
7921
7922
7923
(3)
7924
7925
7926
(3)
(3)
7927
7928
7929
7930
7931

.SBTTL CLEANUP CODING SECTION

:/ THE CLEANUP CODING SECTION CONTAINS THE CODING THAT IS PERFORMED
:/ AT THE END OF THE TEST SEQUENCE ON A PARTICULAR UNIT.

021522
021522

BGNCLN

L\$CLEAN::

021522
021522
021522 104412

ENDCLN

L10017: TRAP C\$CLEAN

7933
7934
7935
7936
7937
7938
7939
7940
(3)
7941
7942
(3)
7943
7944
(8)
(7)
(6)
(3)
(4)
(4)
7945
(3)
(3)
7946
7947

7948
7949
7950
7951
7952
7953

021524
021524

021524 104433

021526 013746 002344
021532 012746 021554
021536 012746 000002
021542 010600
021544 104417
021546 062706 000006

021552
021552 104453

021554 047045 040445 047125
021562 052111 022440 031104
021570 040445 042040 047522
021576 050120 042105 047045
021604 000
021606

```
.SBTTL DROP UNIT SECTION
://////
:// THE DROP-UNIT SECTION CONTAINS THE CODING THAT CAUSES A DEVICE
:// TO NO LONGER BE TESTED.
://////

BGNDU
LSDU::
:ISSUE UNIBUS RESET TO CLEAN UP
BRESET
:PRINT 'UNIT XX DROPPED'
PRINTF #FMT27,LOGDEV
TRAP C$RESET
MOV LOGDEV,-(SP)
MOV #FMT27,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C$PNTF
ADD #6,SP

ENDDU
L10020:
TRAP C$DU

FMT27: .ASCIZ /%N%UNIT %D2%A DROPPED%/

.EVEN
```


7955
7956
7957
7958
7959
7960
7961
7962
7963
(3)
7964
(3)
(3)
7965
7966
7967
7968
7969
7970

.SBTTL ADD UNIT SECTION

:/ THE ADD-UNIT SECTION CONTAINS THE CODING THAT CAUSES A DEVICE
:/ TO BE (A) TESTED FOR THE FIRST TIME, OR (B) RESUMED IN TESTING. IF
:/ 'EF.AUNIT' IS SET, THE UNIT WILL BE TESTED AS A NEW UNIT.

021606
021606
021606
021606
021606 104452

BGNAU
ENDAU

LSAU::
L10021: TRAP CSAU

7972
7973
7974
7975
7976
7977
7978
7979
7980
7981
7982
7983
7984
7985 021610
(3) 021610
7986
7987 021610
(3) 021610 012700 000340
(3) 021614 104441
7988 021616 012737 021640 000004
7989 021624 012737 000340 000006
7990 021632 005777 160610
7991 021636 000406
7992
7993 021640 062706 000004
7994
7995 021644
(4) 021644 104455
(5) 021646 000001
(5) 021650 012101
(5) 021652 014554
7996 021654 013737 002414 000004
7997 021662 013737 002416 000006
7998 021670
(3) 021670
(3) 021670 104401
7999
8000
8001
8002
8003
8004
8005
8006
8007
8008
8009
8010 021672
(3) 021672
8011 021672 012737 000014 002400
8012 021700 004737 003576
8013 021704 004737 003644
8014 021710 123737 002364 003026
8015 021716 001416
8016 021720 005037 002404
8017 021724 113737 003026 002404

```
.SBTTL HARDWARE TESTS

:*****
.SBTTL TEST 1 - MICROPROCESSOR CSR ADDRESSING TEST (SELO)
:*
:* THIS TEST ADDRESSES THE FIRST MICROPROCESSOR CSR (SELO), TO MAKE SURE
:* THAT A NON-EXISTENT MEMORY TIME-OUT TRAP DOES NOT OCCUR WHILE
:* ATTEMPTING TO ADDRESS THE MICROPROCESSOR.
:*****
BGNTST
T1::
;ESTABLISH CPU PRIORITY = 7
SETPRI #PRI07
MOV #PRI07,R0
TRAP C$SPRI
;SET UP NON-EXISTENT MEMORY ERROR TRAP VECTOR
MOV #6$,@#4
MOV #PRI07,@#6
TST @MPCSR ;ADDRESS SELO
BR 9$ ;TAKE THIS BR IF DEVICE RESPONDS
;COME HERE IF DEVICE CSR IS NON-EXISTENT
6$: ADD #4,SP ;CLEAN UP THE STACK POINTER
;REPORT CSR ADDRESS TIME-OUT
ERRDF 1,EM1,ERR1
TRAP C$ERDF
.WORD 1
.WORD EM1
.WORD ERR1
9$: MOV SAVE4,@#4 ;RESTORE ERROR TRAP VECTOR
MOV SAVE6,@#6
ENDTST
L10022:
TRAP C$SETST
```

```
:*****
.SBTTL TEST 2 - IBUS/OUTBUS REG 14 INITIALIZATION TEST
:*
:* MASTER CLEAR (MCLR) IS SET IN THE MICROPROCESSOR, IBUS REG 14 IS READ
:* AND COMPARED TO 200.
:*****
BGNTST
T2::
MOV #14,REGNUM ;SET LU REG NO. = 14
JSR PC,MSTCLR ;ISSUE MASTER CLEAR
JSR PC,READLU ;READ REG 14
CMPB REDBYT,PATM+4 ;CHK FOR INITIALIZED STATE
BEQ 6$ ;BR IF YES
CLR GOODAT ;SET EXPECTED REG CONTENTS = 000
MOVB PATM+4,GOODAT ;SET EXPECTED DATA
```

```

8018 021732 013737 002364 002406      MOV      REDBYT,BADDAT      ;SET ACTUAL REG CONTENTS
8019 021740 004737 003770              JSR      PC,GETREG        ;GET REGS FOR PRINTOUT
8020                                ;REPORT REG NOT CLEARED BY MASTER CLEAR
8021 021744                                ERRDF   2,EM2,ERR2
      (4) 021744 104455                                TRAP    C$ERDF
      (5) 021746 000002                                .WORD  2
      (5) 021750 012135                                .WORD  EM2
      (5) 021752 014606                                .WORD  ERR2
8022 021754                                6$:
8023 021754                                ENDTST
      (3) 021754                                L10023:
      (3) 021754 104401                                TRAP    C$SETST
  
```

```

8024
8025
8026
8027
8028
8029
8030  ;*****
8031  ;SBITL      TEST 3 - INBUS/OUTBUS REG 14 READ/WRITE BIT TEST
8032  ;*
8033  ;* WRITE, READ, AND COMPARE ALL WORDS OF DATA PATTERN A INTO REG 14,
8034  ;* A BYTE AT A TIME. NON-R/W BITS ARE MASKED OFF TO 0 BEFORE WRITING AND
8035  ;* READING.
8036  ;* DATA PATTERN A = 125,252,000,377,001,002,004,010,020,040,100,200,376,
8037  ;* 375,373,367,357,337,277,177.
8038  ;*****
      BGNTST
  
```

```

8039 021756 004737 003576              JSR      PC,MSTCLR        ;ISSUE MASTER CLEAR      T3::
8040 021762 012737 000014 002400      MOV      #14,REGNUM      ;SET LU REG NO. = 14
8041 021770 012701 002571              MOV      #PATA,R1        ;GET POINTER TO DATA PAT IN R1
8042 021774                                3$:
8043 021774                                BGNSEG
      (3) 021774 104404                                TRAP    C$BSEG
8044 021776 111137 002366              MOV      (R1),WRIBYT     ;GET A BYTE OF PAT A
8045 022002 143737 002560 002366      BIC      R14NRW,WRIBYT  ;MASK OFF NON-READ/WRITE BITS
8046 022010 004737 003722              JSR      PC,WRITLU       ;WRITE DATA BYTE INTO REG 14
8047 022014 004737 003644              JSR      PC,READLU       ;READ DATA BYTE FROM REG 14
8048 022020 143737 002560 002364      BIC      R14NRW,REDBYT  ;MASK OFF NON-READ/WRITE BITS
8049 022026 123737 002364 002366      CMP      REDBYT,WRIBYT  ;COMPARE BYTE READ TO BYTE WRITTEN
8050 022034 001414                                BEQ      6$              ;BR IF BYTES MATCH
8051 022036 013737 002366 002404      MOV      WRIBYT,GOODAT   ;SET EXPECTED REG CONTENTS
8052 022044 013737 002364 002406      MOV      REDBYT,BADDAT   ;SET ACTUAL REG CONTENTS
8053 022052 004737 003770              JSR      PC,GETREG        ;GET REGS FOR PRINTOUT
8054                                ;REPORT LINE UNIT REG MISCOMPARE
      (4) 022056 104455                                ERRDF   3,EM3,ERR2
      (5) 022060 000003                                TRAP    C$ERDF
      (5) 022062 012174                                .WORD  3
      (5) 022064 014606                                .WORD  EM3
8056 022066                                6$:
8057 022066                                ENDSEG
      (3) 022066                                10000$:
      (3) 022066 104405                                TRAP    C$ESEG
8058 022070 005201                                INC      R1
8059 022072 020127 002615              CMP      R1,#PATB        ;INCREMENT DATA PATTERN POINTER
                                ;SEE IF ALL WORDS OF PATTERN A USED YET
  
```

8060 022076 103736
8061 022100
(3) 022100
(3) 022100 104401

ENDTST BLO 3\$;BR IF NOT DONE YET

L10024: TRAP C\$ETST

8062
8063
8064
8065
8066

8067
8068
8069
8070
8071

:SBTTL TEST 4 - REG 14 MASTER CLEAR TEST
:* WRITE 377 INTO REG 14, ISSUE MASTER CLEAR, READ REG 14 AND COMPARE
:* TO 200.

8072 022102
(3) 022102

BGNTST

T4::

8073 022102 004737 003576
8074 022106 012737 000014 002400
8075 022114 112737 000377 002366
8076 022122 004737 003722
8077 022126 004737 003576
8078 022132 004737 003644
8079 022136 123737 002364 003026
8080 022144 001416
8081 022146 005037 002404
8082 022152 113737 003026 002404
8083 022160 013737 002364 002406
8084 022166 004737 003770

JSR PC,MSTCLR ;PERFORM MASTER CLEAR
MOV #14,REGNUM ;SET LU REG NO. = 14
MOVB #377,WRIBYT ;SET DATA BYTE = 377
JSR PC,WRITLU ;WRITE 377 INTO REG 14
JSR PC,MSTCLR ;ISSUE MASTER CLEAR
JSR PC,READLU ;READ REG 14
CMPB REDBYT,PATM+4 ;CHK FOR INIT'D STATE
BEQ 6\$;BR IF REG GOT CLEARED
CLR GOODAT
MOVB PATM+4,GOODAT ;SET EXPECTED DATA
MOV REDBYT,BADDAT ;SET ACTUAL DATA
JSR PC,GETREG ;GET REGS FOR PRINTOUT
:REPORT REG NOT CLEARED BY MASTER CLEAR
ERRDF 2,EM2,ERR2

8085
8086 022172
(4) 022172 104455
(5) 022174 000002
(5) 022176 012135
(5) 022200 014606

TRAP C\$ERDF
.WORD 2
.WORD EM2
.WORD ERR2

8087 022202
8088 022202
(3) 022202
(3) 022202 104401

6\$:
ENDTST

L10025: TRAP C\$ETST

8089
8090
8091
8092
8093

8094
8095
8096
8097
8098

:SBTTL TEST 5 - REG 14 UNIBUS RESET (INIT) TEST
:* WRITE 377 INTO REG 14, ISSUE UNIBUS RESET (INIT), READ REG 14 AND COMPARE
:* TO 200.

8099 022204
(3) 022204
8100 022204 004737 003576
8101 022210 012737 000014 002400
8102 022216 112737 000377 002366
8103 022224 004737 003722
8104 022230
(3) 022230 104433

BGNTST

T5::

JSR PC,MSTCLR ;PERFORM MASTER CLEAR
MOV #14,REGNUM ;SET LU REG NO. = 14
MOVB #377,WRIBYT ;SET DATA BYTE = 377
JSR PC,WRITLU ;WRITE 377 INTO REG 14
BRESET ;ISSUE UNIBUS RESET (INIT)

TRAP C\$RESET

```

8105 022232 142777 000200 160210      BICB  #RUN,@BSEL1 ;CLEAR RUN BIT
8106 022240 012701 000024      MOV   #20.,R1     ;INIT LOOP COUNTER
8107 022244 000240      2$: NOP
8108 022246 005301      DEC   R1         ;DECR COUNTER
8109 022250 001375      BNE   2$        ;BR TO STALL
8110 022252 004737 003644      JSR   PC,READLU ;READ REG 14
8111 022256 123737 002364 003026      CMPB  REDBYT,PATM+4 ;CHK FOR INIT'D STATE
8112 022264 001416      BEQ   6$        ;BR IF REG GOT CLEARED
8113 022266 005037 002404      CLR   GOODAT
8114 022272 113737 003026 002404      MOVB  PATM+4,GOODAT ;SET EXPECTED DATA
8115 022300 013737 002364 002406      MOV   REDBYT,BADDAT ;SET ACTUAL DATA
8116 022306 004737 003770      JSR   PC,GETREG  ;GET REGS FOR PRINTOUT
8117      ;REPORT REG NOT CLEARED BY UNIBUS RESET (INIT)
8118 022312      ERRDF  4,EM4,ERR2
(4) 022312 104455
(5) 022314 000004
(5) 022316 012213
(5) 022320 014606
8119 022322      6$:
8120 022322      ENDTST
(3) 022322
(3) 022322 104401
8121
8122
8123
8124
8125
8126
8127
8128
8129
8130
8131
8132
8133
8134
8135
8136 022324
(3) 022324
8137 022324 004737 003576 002400      JSR   PC,MSTCLR ;ISSUE A MASTER CLEAR
8138 022330 012737 000014      MOV   #14,REGNUM ;SET LU REG NO. = 14
8139 022336 013701 002400      MOV   REGNUM,R1  ;SET DESTINATION OBUS* REG 14
8140 022342 052701 000100      BIS   #100,R1    ;SET SOURCE = BSEL4
8141 022346 052701 121000      BIS   #MVIXOX,R1 ;SET REST OF MOVE INSTRUCTION
8142 022352 010137 022370      MOV   R1,2$     ;SET INSTRUCTION AS SUBROUTINE ARGUMENT
8143 022356 112777 000041 160070      MOVB  #041,@BSEL4 ;SET DATA BYTE = 041
8144 022364 004737 003540      JSR   PC,STPCLK ;EXECUTE MOVE INSTRUCTION
8145 022370 000000      2$: .WORD 0 ;INSTRUCTION GOES HERE
8146 022372 004737 003644      JSR   PC,READLU ;READ LU REG 14
8147 022376 123737 002364 003026      CMPB  REDBYT,PATM+4 ;CHECK FOR LU REG 14 UNCHANGED
8148 022404 001416      BEQ   4$        ;BR IF LU REG 14 UNCHANGED
8149 022406 005037 002404      CLR   GOODAT   ;SET EXPECTED DATA
8150 022412 113737 003026 002404      MOVB  PATM+4,GOODAT ;SET ACTUAL DATA
8151 022420 013737 002364 002406      MOV   REDBYT,BADDAT ;GET REGS FOR PRINTOUT
8152 022426 004737 003770      JSR   PC,GETREG
8153      ;REPORT REGISTER MISCOMPARE
    
```

TRAP C\$ERDF
 .WORD 4
 .WORD EM4
 .WORD ERR2

L10026:
 TRAP C\$ETST

```

:*****
:SBTTL TEST 6 - LINE UNIT FALSE SELECTION TEST
:*
:* FIRST, A MASTER CLEAR IS PERFORMED. THEN, THE PROGRAM SINGLE-STEPS THE
:* MICROPROCESSOR THROUGH AN INSTRUCTION WHICH LOADS 041 (OCT) INTO THE MAR
:* REGISTER (OBUS* ADRS 14). THEN, THE LINE UNIT REGISTER 14 IS READ AND CHECKED
:* TO BE UNAFFECTED (STILL = 0). THIS TEST IS INTENDED TO DETECT A FALSE
:* SELECTION OF THE LINE UNIT REGISTERS, WHEN THE LINE UNIT IS NOT BEING
:* ACCESSED.
:*****
:BGNTST
    
```

T6: :

8154 022432
(4) 022432 104455
(5) 022434 000003
(5) 022436 012174
(5) 022440 014606
8155 022442
8156 022442
(3) 022442
(3) 022442 104401

ERRDF 3,EM3,ERR2

TRAP C\$ERDF
.WORD 3
.WORD EM3
.WORD ERR2

4\$:
ENDTST

L10027:
TRAP C\$ETST

8157
8158
8159
8160
8161
8162
8163
8164
8165
8166
8167
8168
8169
8170
8171
8172

```
*****
:SBTTL TEST 7 - INBUS REG MASTER CLEAR TEST
:
:* FIRST, ALL READ/WRITE BITS OF REGS 10-17 ARE SET BY LOADING A
:* DIFFERENT WORD OF PATTERN G INTO EACH REG. THEN,
:* A MASTER CLEAR IS ISSUED AND EACH REG IS READ AND COMPARED TO A WORD OF
:* PATTERN M, WHICH CONTAINS THE INITIALIZED STATES OF THE REGS. (UNPREDICTABLE
:* BITS ARE MASKED OFF TO 0 BEFORE COMPARISON).
:* PATTERN G = 000,000,240,120,177,000,000,001
:* PATTERN M = 000,020,000,000,200,000,000,051
*****
```

8173 022444
(3) 022444
8174 022444 004737 003576
8175 022450 012737 000010 002400
8176 022456 012701 002644
8177 022462 112137 002366
8178 022466 004737 003722
8179 022472 005237 002400
8180 022476 020127 002654
8181 022502 103767
8182 022504 004737 003576
8183 022510 012737 000010 002400
8184 022516 012702 003022
8185 022522 012701 002550
8186 022526
8187 022526 004737 003644
8188 022532 142137 002364
8189 022536 123712 002364
8190 022542 001417
8191 022544 005037 002404
8192 022550 111237 002404
8193 022554 013737 002364 002406
8194 022562 004737 003770
8195

BGNTST

T7::

```
JSR PC,MSTCLR ;ISSUE MASTER CLEAR
MOV #10,REGNUM ;INIT REG NO. TO 10
MOV #PATG,R1 ;INIT DATA PATTERN POINTER
2$: MOVB (R1)+,WRIBYT ;SET DATA PATTERN BYTE TO BE WRITTEN
JSR PC,WRITLU ;WRITE BYTE INTO REG
INC REGNUM ;INCREMENT REG NO. FOR WRITING
CMP R1,#PATH ;SEE IF ALL BYTES WRITTEN YET
BLO 2$ ;BR IF NOT DONE WRITING YET
JSR PC,MSTCLR ;ISSUE MASTER CLEAR
MOV #10,REGNUM ;INIT LU REG NO. TO 10
MOV #PATM,R2 ;INIT DATA PATTERN POINTER
MOV #UPBITS,R1 ;INIT POINTER TO UNPREDICTABLE BITS
3$: JSR PC,READLU ;READ A LINE UNIT REG
BICB (R1)+,REDBYT ;MASK OUT UNPREDICTABLE BITS FOR THIS REG
CMPB REDBYT,(R2) ;COMPARE MASKED DATA TO EXPECTED
BEQ 6$ ;BR IF DATA READ IS OK
CLR GOODAT
MOVB (R2),GOODAT ;SET EXPECTED DATA
MOV REDBYT,BADDAT ;SET ACTUAL DATA
JSR PC,GETREG ;GET REGS FOR PRINTOUT
;REPORT REG NOT CLEARED BY MASTER CLEAR
```

ERRDF 2,EM2,ERR2

TRAP C\$ERDF
.WORD 2
.WORD EM2
.WORD ERR2

ESCAPE TST

TRAP C\$ESCAPE

8196 022566
(4) 022566 104455
(5) 022570 000002
(5) 022572 012135
(5) 022574 014606
8197 022576
(3) 022576 104410

```

(3) 022600 000016 .WORD L10030-
8198 022602 005237 002400 6$: INC REGNUM ;INCREMENT REG NO.
8199 022606 005202 INC R2 ;INCR DATA PATTERN POINTER
8200 022610 020227 003032 CMP R2,#PATN ;SEE IF ALL DONE YET
8201 022614 103744 BLO 3$ ;BR IF NOT DONE YET
8202 022616 ENDTST
(3) 022616 L10030:
(3) 022616 104401 TRAP C$ETST

```

```

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
8232
8233
8234
8235
8236
8237
8238
8239
8240
8241
8242
8243
8244
8245
8246
8247
8248
(4)

```

```

:*****
:SBTTL TEST 8 - REGISTER 10-17 ADDRESSING TEST
:
:* FIRST, A MASTER CLEAR IS ISSUED. THEN,
:* WRITE A DIFFERENT WORD OF DATA PATTERN B INTO EACH OF REGS 10-17,
:* AND AFTER EACH WRITE, READ AND COMPARE ALL REGS TO EXPECTED VALUES.
:* UNPREDICTABLE BITS ARE MASKED OFF TO 0 WHEN READ FOR COMPARISON.
:* PATTERN B = 000,000,040,100,220,000,000,051
:*****
BGNST

```

```

T8::
JSR PC,MSTCLR ;ISSUE MASTER CLEAR
MOV #REDDAT,R1 ;INIT POINTER TO EXPECTED DATA AREA
MOV #PATM,R2 ;GET POINTER TO PATTERN M
MOV #8,R3 ;SET COUNTER
3$: MOVB (R2)+,(R1)+ ;LOAD BYTE OF PATRN INTO EXPECTED DATA AREA
DEC R3 ;DECR COUNTER
BNE 3$ ;BR IF NOT DONE LOADING YET
CLR R1 ;INIT DATA PATTERN INDEX FOR WRITING
6$: MOV R1,REGNUM
ADD #10,REGNUM ;GET REG NO. FOR WRITING
MOVB PATB(R1),WRIBYT ;SET DATA BYTE TO BE WRITTEN
MOVB WRIBYT,REDDAT(R1) ;SET EXPECTED DATA FOR READ
BICB UPBITS(R1),REDDAT(R1) ;MASK OUT UNPREDICTABLE BITS
JSR PC,WRITLU ;WRITE DATA BYTE INTO REG
CLR R3 ;INIT DATA PAT INDEX FOR READS
9$: MOV R3,REGNUM
ADD #10,REGNUM ;GET REG NO. FOR READING
JSR PC,READLU ;READ A LINE UNIT REG
BICB UPBITS(R3),REDBYT ;MASK OUT UNPREDICTABLE BITS
CMP REGNUM,#11 ;SEE IF READING REG 11
BNE 10$ ;BR IF NOT
BICB #ORDY,REDBYT ;MASK ORDY BIT IN ACTUAL BYTE
BICB #ORDY,REDDAT(R3) ;MASK ORDY BIT IN EXPECTED BYTE
10$: CMPB REDBYT,REDDAT(R3) ;COMPARE BYTE READ TO EXPECTED
BEQ 12$ ;BR IF DATA MATCHES
CLR GOODAT
MOVB REDDAT(R3),GOODAT ;SET EXPECTED DATA
MOV REDBYT,BADAT ;SET ACTUAL DATA
JSR PC,GETREG ;READ AND STORE REGS 10-17 FOR PRINTOUT
;REPORT REG MISC COMPARE
ERRDF 3,EM3,ERR2

```

TRAP C\$ERDF

```

(5) 023020 000003 .WORD 3
(5) 023022 012174 .WORD EM3
(5) 023024 014606 .WORD ERR2
8249 023026 ESCAPE TST
(3) 023026 104410 TRAP C$ESCAPE
(3) 023030 000022 .WORD L10031-
8250 023032 005203 12$: INC R3 ;INCREMENT DATA PATTERN INDEX FOR READING
8251 023034 020327 000010 CMP R3,#10 ;SEE IF ALL REGS READ YET
8252 023040 002724 BLT 9$ ;BR IF NOT
8253 023042 005201 INC R1 ;INCREMENT DATA PAT INDEX FOR WRITING
8254 023044 020127 000010 CMP R1,#10 ;SEE IF ALL REGS WRITTEN YET
8255 023050 002677 BLT 6$ ;BR IF NOT
8256 023052 ENDTST
(3) 023052 L10031: TRAP C$SETST
(3) 023052 104401 TRAP C$SETST
8257
8258
8259
8260
8261
8262
8263 :*****
8264 :SBTTL TEST 9 - REG 11 READ/WRITE BIT TEST
8265 :*
8266 :* WRITE, READ, AND COMPARE EACH WORD OF DATA PATTERN C INTO REG 11 :
8267 :* DATA PATTERN C = 020,020,020.
8268 :*****
8268 023054 BGNST
(3) 023054 T9::
8269 023054 004737 003576 JSR PC,MSTCLR ;ISSUE MASTER CLEAR
8270 023060 012737 000011 002400 MOV #11,REGNUM ;SET LU REG NO. = 11
8271 023066 012701 002625 MOV #PATC,R1 ;GET POINTER TO DATA PAT IN R1
8272 023072 3$:
8273 023072 BGNSEG TRAP C$BSEG
(3) 023072 104404 TRAP C$BSEG
8274 023074 111137 002366 MOVB (R1),WRIBYT ;GET A BYTE OF PAT C
8275 023100 004737 003722 JSR PC,WRITLU ;WRITE DATA BYTE INTO REG 11
8276 023104 004737 003644 JSR PC,READLU ;READ DATA BYTE FROM REG 11
8277 023110 143737 002551 002364 BICB UPBITS+1,REDBYT ;MASK OUT UNPREDICTABLE BITS
8278 023116 123737 002364 002366 CMPB REDBYT,WRIBYT ;COMPARE BYTE READ TO BYTE WRITTEN
8279 023124 001414 BEQ 6$ ;BR IF BYTES MATCH
8280 023126 013737 002366 002404 MOV WRIBYT,GOODAT ;SET EXPECTED REG CONTENTS
8281 023134 013737 002364 002406 MOV REDBYT,BADDAT ;SET ACTUAL REG CONTENTS
8282 023142 004737 003770 JSR PC,GETREG ;GET REGS FOR PRINTOUT
8283 ;REPORT LINE UNIT REG MISCOMPARE
8284 023146 ERRDF 3,EM3,ERR2
(4) 023146 104455 TRAP C$ERDF
(5) 023150 000003 .WORD 3
(5) 023152 012174 .WORD EM3
(5) 023154 014606 .WORD ERR2
8285 023156 6$:
8286 023156 ENDSEG
(3) 023156 10000$: TRAP C$ESEG
(3) 023156 104405 TRAP C$ESEG
8287 023160 005201 INC R1 ;INCREMENT DATA PATTERN POINTER
8288 023162 020127 002630 CMP R1,#PATD ;SEE IF ALL WORDS OF PATTERN C USED YET
8289 023166 103741 BLO 3$ ;BR IF NOT DONE YET

```



```

8290 023170          ENDTST
      (3) 023170
      (3) 023170 104401          L10032: TRAP C$ETST
8291
8292
8293
8294
8295
8296
8297
8298
8299
8300
8301
8302 023172          BGNTST
      (3) 023172

```

```

*****
:SBTTL      TEST 10 - REG 12 READ/WRITE BIT TEST
:
:* WRITE, READ, AND COMPARE EACH WORD OF DATA PATTERN D INTO REG 12 :
:* DATA PATTERN D = 000,040,000.
*****

```

```

8303 023172 004737 003576          JSR      PC,MSTCLR      ;ISSUE MASTER CLEAR      T10::
8304 023176 012737 000012 002400   MOV      #12,REGNUM    ;SET LU REG NO. - 12
8305 023204 012701 002630          MOV      #PATD,R1      ;GET POINTER TO DATA PAT IN R1
8306 023210          3$:
8307 023210          BGNSEG
      (3) 023210 104404          TRAP      C$BSEG
8308 023212 111137 002366          MOVB     (R1),WRIBYT    ;GET A BYTE OF PAT D
8309 023216 004737 003722          JSR      PC,WRITLU     ;WRITE DATA BYTE INTO REG 12
8310 023222 004737 003644          JSR      PC,READLU     ;READ DATA BYTE FROM REG 12
8311 023226 143737 002552 002364   BICB     UPBITS+2,REDBYT ;MASK OUT UNPREDICTABLE BITS
8312 023234 123737 002364 002366   CMPB     REDBYT,WRIBYT  ;COMPARE BYTE READ TO BYTE WRITTEN
8313 023242 001414          BEQ      6$           ;BR IF BYTES MATCH
8314 023244 013737 002366 002404   MOV      WRIBYT,GOODAT ;SET EXPECTED REG CONTENTS
8315 023252 013737 002364 002406   MOV      REDBYT,BADDAT ;SET ACTUAL REG CONTENTS
8316 023260 004737 003770          JSR      PC,GETREG     ;GET REGS FOR PRINTOUT
8317          :REPORT LINE UNIT REG MISCOMPARE
8318 023264          ERRDF 3,EM3,ERR2
      (4) 023264 104455          TRAP      C$ERDF
      (5) 023266 000003          .WORD    3
      (5) 023270 012174          .WORD    EM3
      (5) 023272 014606          .WORD    ERR2
8319 023274          6$:
8320 023274          ENDSEG
      (3) 023274          10000$: TRAP      C$ESEG
      (3) 023274 104405
8321 023276 005201          INC      R1            ;INCREMENT DATA PATTERN POINTER
8322 023300 020127 002633          CMP      R1,#PATE     ;SEE IF ALL WORDS OF PATTERN D USED YET
8323 023304 103741          BLO     3$           ;BR IF NOT DONE YET
8324 023306          ENDTST
      (3) 023306          L10033: TRAP      C$ETST
      (3) 023306 104401
8325
8326
8327
8328
8329
8330
8331
8332
8333

```

```

*****
:SBTTL      TEST 11 - REG 13 READ/WRITE BIT TEST
:
:* WRITE, READ, AND COMPARE EACH WORD OF DATA PATTERN E INTO REG 13 :

```

```
8334      ;* DATA PATTERN E = 000,120,020,100,120,000.
8335      ;*****
8336 023310 BGNTST
(3) 023310
8337 023310 004737 003576 JSR PC,MSTCLR ;ISSUE MASTER CLEAR T11::
8338 023314 012737 000013 002400 MOV #13,REGNUM ;SET LU REG NO. = 13
8339 023322 012701 002633 MOV #PATE,R1 ;GET POINTER TO DATA PAT IN R1
8340 023326 3$:
8341 023326 BGNSEG
(3) 023326 104404 TRAP C$BSEG
8342 023330 111137 002366 MOVB (R1),WRIBYT ;GET A BYTE OF PAT E
8343 023334 004737 003722 JSR PC,WRITLU ;WRITE DATA BYTE INTO REG 13
8344 023340 004737 003644 JSR PC,READLU ;READ DATA BYTE FROM REG 13
8345 023344 143737 002553 002364 BICB UPBITS+3,REDBYT ;MASK OUT UNPREDICTABLE BITS
8346 023352 123737 002364 002366 CMPB REDBYT,WRIBYT ;COMPARE BYTE READ TO BYTE WRITTEN
8347 023360 001414 BEQ 6$ ;BR IF BYTES MATCH
8348 023362 013737 002366 002404 MOV WRIBYT,GOODAT ;SET EXPECTED REG CONTENTS
8349 023370 013737 002364 002406 MOV REDBYT,BADDAT ;SET ACTUAL REG CONTENTS
8350 023376 004737 003770 JSR PC,GETREG ;GET REGS FOR PRINTOUT
8351 ;REPORT LINE UNIT REG MISCOMPARE
8352 023402 ERRDF 3,EM3,ERR2
(4) 023402 104455 TRAP C$ERDF
(5) 023404 000003 .WORD 3
(5) 023406 012174 .WORD EM3
(5) 023410 014606 .WORD ERR2
8353 023412 6$:
8354 023412 ENDSEG
(3) 023412 10000$:
(3) 023412 104405 TRAP C$ESEG
8355 023414 005201 INC R1 ;INCREMENT DATA PATTERN POINTER
8356 023416 020127 002641 CMP R1,#PATF ;SEE IF ALL WORDS OF PATTERN E USED YET
8357 023422 103741 BLO 3$ ;BR IF NOT DONE YET
8358 023424 ENDTST
(3) 023424 L10034:
(3) 023424 104401 TRAP C$ETST
8359
8360
8361
8362
8363
8364 ;*****
8365 .SBTTL TEST 12 - REG 17 READ/WRITE BIT TEST
8366 ;*
8367 ;* WRITE, READ, AND COMPARE EACH WORD OF DATA PATTERN F INTO REG 17 :
8368 ;* DATA PATTERN F = 050,051,050.
8369 ;*****
8370 023426 BGNTST
(3) 023426
8371 023426 004737 003576 JSR PC,MSTCLR ;ISSUE MASTER CLEAR T12::
8372 023432 012737 000017 002400 MOV #17,REGNUM ;SET LU REG NO. = 17
8373 023440 012701 002641 MOV #PATF,R1 ;GET POINTER TO DATA PAT IN R1
8374 023444 3$:
8375 023444 BGNSEG
(3) 023444 104404 TRAP C$BSEG
8376 023446 111137 002366 MOVB (R1),WRIBYT ;GET A BYTE OF PAT F
8377 023452 004737 003722 JSR PC,WRITLU ;WRITE DATA BYTE INTO REG 17
```

```

8378 023456 004737 003644 JSR PC,READLU ;READ DATA BYTE FROM REG 17
8379 023462 143737 002557 002364 BICB UPBITS+7,REDBYT ;MASK OUT UNPREDICTABLE BITS
8380 023470 123737 002364 002366 CMPB REDBYT,WRIBYT ;COMPARE BYTE READ TO BYTE WRITTEN
8381 023476 001414 BEQ 6$ ;BR IF BYTES MATCH
8382 023500 013737 002366 002404 MOV WRIBYT,GOODAT ;SET EXPECTED REG CONTENTS
8383 023506 013737 002364 002406 MOV REDBYT,BADDAT ;SET ACTUAL REG CONTENTS
8384 023514 004737 003770 JSR PC,GETREG ;GET REGS FOR PRINTOUT
8385 ;REPORT LINE UNIT REG MISCOMPARE
8386 023520 ERRDF 3,EM3,ERR2
(4) 023520 104455 TRAP C$ERDF
(5) 023522 000003 .WORD 3
(5) 023524 012174 .WORD EM3
(5) 023526 014606 .WORD ERR2
8387 023530 6$:
8388 023530 ENDSEG
(3) 023530 10000$: TRAP C$ESEG
(3) 023530 104405
8389 023532 005201 INC R1 ;INCREMENT DATA PATTERN POINTER
8390 023534 020127 002644 CMP R1,#PATG ;SEE IF ALL WORDS OF PATTERN F USED YET
8391 023540 103741 BLO 3$ ;BR IF NOT DONE YET
8392 023542 ENDTST
(3) 023542 L10035:
(3) 023542 104401 TRAP C$ETST
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
    
```

```

:*****
:SBTTL TEST 13 - MAINTENANCE CLOCK BIT TEST
:
:* FIRST, A MASTER CLEAR IS ISSUED TO INIT ALL REGS. THEN, THE MICROPROCESSOR
:* IS PLACED IN A LOOP ON AN INSTRUCTION, BY SETTING THE INSTRUCTION IN SEL6
:* AND SETTING ROMI AND RUN IN BSEL1. THE INSTRUCTION IS ONE WHICH REPETITIVELY
:* READS LINE UNIT REG 17 INTO BSEL2. THE PDP-11 CAN THEN SCAN BSEL2 TO MONITOR
:* THE MAINTENANCE CLOCK BIT, MCLK. THE FOLLOWING SEQUENCE IS THEN PERFORMED
:* TO MONITOR MCLK :
:* - THE PROGRAM REPEATEDLY CHECKS THE MCLK BIT FOR THE 1 STATE, AND IF IT IS
:* NOT FOUND WITHIN SEVERAL HUNDRED MILLI-SEC (DEPENDING ON THE PROCESSOR)
:* AN ERROR IS REPORTED. (THE MAINTENANCE CLOCK HAS A PERIOD OF 41.6 MICRO-
:* SEC).
:* - THE PROGRAM NEXT REPEATEDLY CHECKS THE MCLK BIT FOR THE 0 STATE, AND IF
:* IT IS NOT FOUND WITHIN SEVERAL HUNDRED MILLI-SEC AN ERROR IS REPORTED.
:* - THE PROGRAM NEXT REPEATEDLY CHECKS MCLK BIT FOR THE 1 STATE AGAIN, AND
:* IF IT IS NOT FOUND WITHIN SEVERAL HUNDRED MILLI-SEC, AN ERROR IS REPORTED.
:*
:* IF THE P-TABLE FOR THIS UNIT INDICATES THAT THE MICROPROCESSOR RUN SWITCH
:* IS NOT ON, THE TEST WILL BE SKIPPED.
:*****
BGNTST
    
```

```

(3) 023544 T13::
8420 023544 004737 003576 JSR PC,MSTCLR ;PERFORM MASTER CLEAR
8421 023550 012737 000015 002442 MOV #13, TSTNUM ;SET TEST NO.
8422 023556 005737 002476 TST RUNINH ;SEE IF RUN SWITCH IS SET
8423 023562 001020 BNE 1$ ;BR IF YES, TO RUN TEST
8424 023564 023727 002444 000001 CMP STARES,#1 ;SEE IF THIS IS FIRST PASS SINCE STA OR RES
    
```

```

8425 023572 001012          BNE      40$          ;BR IF NOT, TO SKIP PRINTING
8426 023574                PRINTF  #FMT19,TSTNUM ;PRINT MSG TO SAY TEST NOT RUN
(8) 023574 013746 002442                MOV      TSTNUM,-(SP)
(7) 023600 012746 011761                MOV      #FMT19,-(SP)
(6) 023604 012746 000002                MOV      #2,-(SP)
(3) 023610 010600                MOV      SP,R0
(4) 023612 104417                TRAP    C$PNTF
(4) 023614 062706 000006                ADD     #6,SP
8427 023620 000137 024166          40$:   JMP     A1          ;GO TO SKIP TEST
8428 023624 012737 000017          1$:   MOV     #17,REGNUM ;SET REG NO. = 17
8429 023632 012701 000360          MOV     #360,R1    ;SET INSTRUCTION SOURCE = INBUS REG 17
8430 023636 052701 000002          BIS     #2,R1      ;SET INSTRUCTION DESTINATION = BSEL2
8431 023642 052701 021000          BIS     #MVIOX,R1 ;SET REST OF MOVE INSTRUCTION
8432 023646 010137 023656          MOV     R1,2$     ;SET INSTRUCTION AS SUBROUTINE ARGUMENT
8433 023652 004737 004046          JSR     PC,LOOPIN ;GET MICROPROCESSOR LOOPING ON MOVE INSTRUCTION
8434 023656 000000          2$:   .WORD  0        ;INSTRUCTION GOES HERE
8435
8436
8437
8438
-----
; WAIT FOR MCLK BIT TO BE SET TO 1
-----
8439 023660 005037 002510          CLR     REGO      ;INIT PROGRAM TIMER
8440 023664 117737 156562          3$:   MOV     @BSEL2,REDBYT ;GET REG 17 INTO REDBYT
8441 023672 132737 000002          BITB   #MCLK,REDBYT ;SEE IF MCLK BIT = 1 YET
8442 023700 001031                BNE     6$        ;BR IF MCLK = 1
8443 023702 005237 002510          INC     REGO      ;INCREMENT TIMER
8444 023706 001366                BNE     3$        ;BR IF PROGRAM TIMER DID NOT TIME OUT YET
8445
8446 023710 012737 000002          MOV     #MCLK,GOODAT ;(TIME OUT = SEVERAL HUNDRED MILLI-SEC)
8447 023716 013737 002364          MOV     REDBYT,BADDAT ;SET EXPECTED REG CONTENTS
8448 023724 004737 003770          JSR     PC,GETREG ;SET ACTUAL REG CONTENTS
8449
8450 023730          ;REPORT MCLK BIT STUCK AT 0
ERRDF  5,EM5,ERR2 ;GET REGS FOR PRINTOUT
(4) 023730 104455                TRAP    C$ERDF
(5) 023732 000005                .WORD  5
(5) 023734 012266                .WORD  EM5
(5) 023736 014606                .WORD  ERR2
8451
8452 023740          ;TYPE 'PLEASE INSURE MICROPROCESSOR RUN SWITCH IS ON'
PRINTF #FMT24
(7) 023740 012746 012012                MOV     #FMT24,-(SP)
(6) 023744 012746 000001                MOV     #1,-(SP)
(3) 023750 010600                MOV     SP,R0
(4) 023752 104417                TRAP    C$PNTF
(4) 023754 062706 000004                ADD     #4,SP
8453 023760 000137 024166          JMP     A1          ;ESCAPE TO END OF TEST
8454
8455
8456
8457
-----
; WAIT FOR MCLK BIT TO BE CLEARED TO 0
-----
8458 023764          6$:
8459 023764 005037 002510          CLR     REGO      ;INIT PROGRAM TIMER
8460 023770 117737 156456          8$:   MOV     @BSEL2,REDBYT ;GET REG 17 INTO REDBYT
8461 023776 132737 000002          BITB   #MCLK,REDBYT ;SEE IF MCLK BIT = 0 YET
8462 024004 001430                BEQ     10$       ;BR IF MCLK = 0
8463 024006 005237 002510          INC     REGO      ;INCREMENT TIMER
8464 024012 001366                BNE     8$        ;BR IF TIMER DID NOT TIME OUT YET
8465 024014 005037 002404          CLR     GOODAT    ;SET EXPECTED REG CONTENTS

```

```
8466 024020 013737 002364 002406      MOV    REDBYT,BADDAT    ;SET ACTUAL REG CONTENTS
8467 024026 004737 003770                JSR    PC,GETREG       ;GET REGS FOR PRINTOUT
8468                                     ;REPORT MCLK BIT STUCK AT 1
8469 024032                                ERRDF  6,EM6,ERR2
(4) 024032 104455                                TRAP  C$ERDF
(5) 024034 000006                                .WORD 6
(5) 024036 012317                                .WORD EM6
(5) 024040 014606                                .WORD ERR2
8470                                     ;TYPE 'PLEASE INSURE MICROPROCESSOR RUN SWITCH IS ON'
8471 024042                                PRINTF #FMT24
(7) 024042 012746 012012                                MOV    #FMT24,-(SP)
(6) 024046 012746 000001                                MOV    #1,-(SP)
(3) 024052 010600                                MOV    SP,R0
(4) 024054 104417                                TRAP  C$PNTF
(4) 024056 062706 000004                                ADD    #4,SP
8472 024062 000137 024166                JMP    A1                ;ESCAPE TO END OF TEST
8473
8474 -----
8475 ; WAIT FOR MCLK BIT TO BE SET TO 1 AGAIN
8476 -----
8477 024066                                10$:
8478 024066 005037 002510                                CLR    REGO                ;INIT PROGRAM TIMER
8479 024072 117737 156354 002364 12$:  MOVB  @BSEL2,REDBYT    ;GET REG 17 INTO REDBYT
8480 024100 132737 000002 002364        BITB  #MCLK,REDBYT    ;SEE IF MCLK BIT = 1 YET
8481 024106 001027                                BNE   A1                ;BE IF MCLK = 1
8482 024110 005237 002510                                INC   REGO                ;INCREMENT TIMER
8483 024114 001366                                BNE   12$                ;BR IF TIMER DID NOT TIME OUT YET
8484 024116 012737 000002 002404        MOV    #MCLK,GOODAT    ;SET EXPECTED REG CONTENTS
8485 024124 013737 002364 002406        MOV    REDBYT,BADDAT    ;SET ACTUAL REG CONTENTS
8486 024132 004737 003770                JSR    PC,GETREG       ;GET REGS FOR PRINTOUT
8487                                     ;REPORT MCLK BIT STUCK AT 0
8488 024136                                ERRDF  5,EM5,ERR2
(4) 024136 104455                                TRAP  C$ERDF
(5) 024140 000005                                .WORD 5
(5) 024142 012266                                .WORD EM5
(5) 024144 014606                                .WORD ERR2
8489                                     ;TYPE 'PLEASE INSURE MICROPROCESSOR RUN SWITCH IS ON'
8490 024146                                PRINTF #FMT24
(7) 024146 012746 012012                                MOV    #FMT24,-(SP)
(6) 024152 012746 000001                                MOV    #1,-(SP)
(3) 024156 010600                                MOV    SP,R0
(4) 024160 104417                                TRAP  C$PNTF
(4) 024162 062706 000004                                ADD    #4,SP
8491 024166                                A1:
8492 024166 004737 003576                JSR    PC,MSTCLR        ;ISSUE MASTER CLEAR
8493 024172                                ENDTST
(3) 024172                                L10036:
(3) 024172 104401                                TRAP  C$ETST
8494
8495
8496
8497
8498
8499
8500
8501
```

```
*****
;SBTTL      TEST 14 - EXTENDED REGISTER MASTER CLEAR TEST
;*
```

```

8502      ; * FIRST, ALL READ/WRITE BITS OF EXTENDED REGS AX0-AX3 ARE SET BY LOADING
8503      ; * A DIFFERENT WORD OF PATTERN H INTO EACH REG. THEN, A MASTER CLEAR IS
8504      ; * ISSUED AND EACH REG IS READ AND COMPARED TO A WORD OF PATTERN I, WHICH
8505      ; * CONTAINS THE INITIALIZED STATES OF ALL THE EXTENDED REGS.
8506      ; *   PATTERN H = 000,000,377,017,377,377,375,377
8507      ; *   PATTERN I = 000,000,000,000,000,103,000,000
8508      ; *****
8509 024174 BGNTST
      (3) 024174
8510 024174 004737 003576      JSR    PC,MSTCLR      ;ISSUE AN INITIAL MASTER CLEAR
8511 024200 005037 002402      CLR    AXNUM         ;INIT AX REG BYTE NO. TO 0
8512 024204 012701 002654      MOV    #PATH,R1     ;INIT DATA PATTERN POINTER
8513 024210 112137 002374      1$:  MOVB  (R1)+,WAX15 ;SET BITS TO LOAD INTO LO BYTE
8514 024214 112137 002376      MOVB  (R1)+,WAX16 ;SET BITS TO LOAD INTO HI BYTE
8515 024220 004737 004264      JSR    PC,WRITAX    ;WRITE EXTENDED REG
8516 024224 032737 000002 002420 BIT    #WRDYTO,ERROR1 ;SEE IF READY FAILED TO SET
8517 024232 001413          BEQ    8$           ;BR IF NOT
8518 024234 012737 000014 002400 MOV    #14,REGNUM    ;SET LU REG NO. = 14
8519 024242 004737 003770      JSR    PC,GETREG    ;GET REGS FOR PRINTOUT
8520      ;REPORT READY NOT SET AFTER AX REG WRITE
8521 024246      ERRDF 52,EM52,ERR9
      (4) 024246 104455
      (5) 024250 000064      TRAP  C$ERDF
      (5) 024252 014061      .WORD 52
      (5) 024254 020604      .WORD EM52
8522 024256      ESCAPE TST      .WORD  ERR9
      (3) 024256 104410
      (3) 024260 000244      TRAP  C$ESCAPE
8523 024262 062737 000002 002402 8$:  ADD    #2,AXNUM      ;INCR REG BYTE NO.
8524 024270 020127 002664      CMP    R1,#PATI     ;SEE IF ALL REGS WRITTEN YET
8525 024274 103745          BLO   1$           ;BR IF NOT DONE WRITING YET
8526 024276 004737 003576      JSR    PC,MSTCLR    ;ISSUE MASTER CLEAR
8527 024302 005037 002402      CLR    AXNUM         ;INIT EXTENDED REG BYTE NO. TO 0
8528 024306 012701 002664      MOV    #PATI,R1     ;INIT DATA PAT POINTER FOR READS
8529 024312 004737 004076      2$:  JSR    PC,READAX   ;READ AN EXTENDED REG
8530 024316 032737 000001 002420 BIT    #RRDYTO,ERROR1 ;SEE IF READY FAILED TO SET
8531 024324 001413          BEQ    10$          ;BR IF NOT
8532 024326 012737 000014 002400 MOV    #14,REGNUM    ;SET LU REG NO. = 14
8533 024334 004737 003770      JSR    PC,GETREG    ;GET REGS FOR PRINTOUT
8534      ;REPORT READY NOT SET AFTER AX REG READ
8535 024340      ERRDF 53,EM53,ERR9
      (4) 024340 104455
      (5) 024342 000065      TRAP  C$ERDF
      (5) 024344 014122      .WORD 53
      (5) 024346 020604      .WORD EM53
8536 024350      ESCAPE TST      .WORD  ERR9
      (3) 024350 104410
      (3) 024352 000152      TRAP  C$ESCAPE
8537 024354 023727 002402 000006 10$:  CMP    AXNUM,#6     ;SEE IF AX3-15
8538 024362 001003          BNE   3$           ;BR IF NOT
8539 024364 142737 000332 002370 BICB  #AX315U,RAX15 ;MASK OFF UNPREDICTABLE BITS
8540 024372 123711 002370      3$:  (MPB  RAX15,(R1)   ;COMPARE LO BYTE TO EXPECTED VAL JE
8541 024376 001417          BEQ    4$           ;BR IF DATA MATCHES
8542 024400 005037 002404      CLR    GOODAT
8543 024404 111137 002404      MOVB  (R1),GOODAT  ;GET EXPECTED DATA BYTE
8544 024410 013737 002370 002406 MOV    RAX15,BADDAT ;GET ACTUAL DATA BYTE

```

```
8545 024416 004737 004500      JSR    PC,GETALL      ;GET REGS FOR PRINTOUT
8546      ;REPORT REG NOT INITIALIZED BY MASTER CLEAR
8547 024422      ERRDF  2,EM2,ERR3
      (4) 024422 104455      TRAP   C$ERDF
      (5) 024424 000002      .WORD  2
      (5) 024426 012135      .WORD  EM2
      (5) 024430 015114      .WORD  ERR3
8548 024432      ESCAPE  TST
      (3) 024432 104410      TRAP   C$ESCAPE
      (3) 024434 000070      .WORD  L10037-
8549 024436 005237 002402      4$:   INC    AXNUM      ;INCREMENT AX BYTE NO.
8550 024442 005201      INC    R1             ;INCREMENT PAT POINTER
8551 024444 123711 002372      CMPB  RAX16,(R1)     ;COMPARE HI BYTE TO EXPECTED VALUE
8552 024450 001417      BEQ   6$             ;BR IF DATA MATCHES
8553 024452 005037 002404      CLR   GOODAT
8554 024456 111137 002404      MOVB  (R1),GOODAT   ;GET EXPECTED DATA BYTE
8555 024462 013737 002372      MOV   RAX16,BADDAT  ;GET ACTUAL DATA BYTE
8556 024470 004737 004500      JSR   PC,GETALL      ;GET REGS FOR PRINTOUT
8557      ;REPORT REG NOT INITIALIZED BY MASTER CLEAR
8558 024474      ERRDF  2,EM2,ERR3
      (4) 024474 104455      TRAP   C$ERDF
      (5) 024476 000002      .WORD  2
      (5) 024500 012135      .WORD  EM2
      (5) 024502 015114      .WORD  ERR3
8559 024504      ESCAPE  TST
      (3) 024504 104410      TRAP   C$ESCAPE
      (3) 024506 000016      .WORD  L10037-
8560 024510 005237 002402      6$:   INC    AXNUM      ;INCR AX BYTE NO.
8561 024514 005201      INC    R1             ;INCR PAT POINTER
8562 024516 020127 002674      CMP   R1,#PATJ      ;SEE IF ALL REGS READ YET
8563 024522 103673      BLO   2$             ;BR IF NOT DONE READING YET
8564 024524      ENDTST
      (3) 024524      L10037:
      (3) 024524 104401      TRAP   C$ETST
8565
8566
8567
8568
8569
8570
8571      ;*****
8572      ;SBTTL      TEST 15 - EXTENDED REGISTER ADDRESSING TEST
8573      ;*
8574      ;* FIRST, ISSUE A MASTER CLEAR TO PUT REGS INTO INITIALIZED STATES SHOWN IN
8575      ;* PATTERN I. THEN, WRITE A DIFFERENT WORD OF PATTERN J INTO EACH EXTENDED (AX)
8576      ;* REG, AND AFTER EACH WRITE, READ AND COMPARE ALL EXTENDED REGS TO EXPECTED
8577      ;* VALUES.
8578      ;* PATTERN I = 000,000,000,000,000,103,000,000
8579      ;* PATTERN J = 000,000,010,002,004,103,001,100
8580      ;*****
8580 024526      BGNTST
      (3) 024526      T15::
8581 024526 004737 003576      JSR   PC,MSTCLR      ;ISSUE MASTER CLEAR
8582 024532 012701 002664      MOV   #PATI,R1       ;INIT POINTER TO PAT I
8583 024536 012702 002500      MOV   #REDDAT,R2     ;INIT POINTER TO EXPECTED DATA AREA
8584 024542 112122      3$:   MOVB  (R1)+,(R2)+   ;MOVE PAT I INTO REDDAT TABLE
8585 024544 020127 002674      CMP   R1,#PATJ
```

```
8586 024550 103774          BLO      3$
8587 024552 005001          CLR      R1              ;INIT INDEX FOR WRITING
8588 024554 010137 002402    6$:      MOV      R1,AXNUM      ;GET AX BYTE NO. FOR WRITING
8589 024560 116137 002674 002374    MOVB     PATJ(R1),WAX15   ;SET LO AX BYTE TO BE WRITTEN
8590 024566 116137 002675 002376    MOVB     PATJ+1(R1),WAX16 ;SET HI AX BYTE TO BE WRITTEN
8591 024574 113761 002374 002500    MOVB     WAX15,REDDAT(R1) ;SET EXPECTED LO BYTE IN TABLE
8592 024602 113761 002376      MOVB     WAX16,REDDAT+1(R1) ;SET EXPECTED HI BYTE IN TABLE
8593 024610 004737 004264    JSR      PC,WRITAX       ;WRITE DATA BYTES INTO EXTENDED REG
8594 024614 005003          CLR      R3              ;INIT INDEX FOR READING
8595 024616 010337 002402    9$:      MOV      R3,AXNUM      ;GET AX BYTE NO. FOR READING
8596 024622 004737 004076    JSR      PC,READAX       ;READ 2 AX BYTES
8597 024626 023727 002402 000006    CMP      AXNUM,#6        ;SEE IF AX3-15
8598 024634 001003          BNE      10$            ;BR IF NOT
8599 024636 142737 000332 002370    BICB     #AX315U,RAX15   ;MASK OFF UNPREDICTABLE BITS
8600 024644 123763 002370 002500    10$:     CMPB     RAX15,REDDAT(R3) ;COMPARE LO BYTE READ TO EXPECTED
8601 024652 001420          BEQ      12$            ;BR IF LO BYTE MATCHES EXPECTED
8602 024654 005037 002404    CLR      GOODAT
8603 024660 116337 002500 002404    MOVB     REDDAT(R3),GOODAT ;GET EXPECTED LO BYTE
8604 024666 013737 002370 002406    MOV      RAX15,BADDAT    ;GET ACTUAL DATA
8605 024674 004737 004500    JSR      PC,GETALL       ;GET REGS FOR PRINTOUT
8606                                     ;REPORT REG MISCCOMPARE
8607 024700          ERRDF   3,EM3,ERR3
(4) 024700 104455          TRAP    C$ERDF
(5) 024702 000003          .WORD  3
(5) 024704 012174          .WORD  EM3
(5) 024706 015114          .WORD  ERR3
8608 024710          ESCAPE  TST
(3) 024710 104410          TRAP    C$ESCAPE
(3) 024712 000102          .WORD  L10040-
8609 024714 005237 002402    12$:     INC      AXNUM           ;INCR AX BYTE NO.
8610 024720 123763 002372 002501    CMPB     RAX16,REDDAT+1(R3) ;COMPARE HI BYTE READ TO EXPECTED
8611 024726 001420          BEQ      15$            ;BR IF HI BYTE MATCHES EXPECTED
8612 024730 005037 002404    CLR      GOODAT
8613 024734 116337 002501 002404    MOVB     REDDAT+1(R3),GOODAT ;GET EXPECTED HI BYTE
8614 024742 013737 002372 002406    MOV      RAX16,BADDAT    ;GET ACTUAL DATA
8615 024750 004737 004500    JSR      PC,GETALL       ;GET REGS FOR PRINTOUT
8616                                     ;REPORT REG MISCCOMPARE
8617 024754          ERRDF   3,EM3,ERR3
(4) 024754 104455          TRAP    C$ERDF
(5) 024756 000003          .WORD  3
(5) 024760 012174          .WORD  EM3
(5) 024762 015114          .WORD  ERR3
8618 024764          ESCAPE  TST
(3) 024764 104410          TRAP    C$ESCAPE
(3) 024766 000026          .WORD  L10040-
8619 024770 062703 000002    15$:     ADD      #2,R3           ;INCR INDEX FOR READS
8620 024774 020327 000010    CMP      R3,#10         ;SEE IF ALL AX BYTES READ YET
8621 025000 002706          BLT     9$              ;BR IF NOT DONE READING YET
8622 025002 062701 000002    ADD      #2,R1           ;INCR INDEX FOR WRITES
8623 025006 020127 000010    CMP      R1,#10         ;SEE IF ALL AX BYTES WRITTEN YET
8624 025012 002660          BLT     6$              ;BR IF NOT DONE WRITING YET
8625 025014          ENDTST
(3) 025014          L10040:
(3) 025014 104401          TRAP    C$ETST
8626
8627
```


8628
8629
8630
8631
8632
8633
8634
8635
8636
8637
8638
8639
8640
8641
8642
8643
8644
8645
8646
(3)
8647
8648
8649
8650
(3)
8651
8652
8653
8654
8655
8656
8657
8658
8659
8660
8661
8662
8663
8664
(4)
(5)
(5)
(5)
8665
(3)
(3)
8666
8667
8668
8669
8670
8671
8672
8673
8674
(4)

025016
025016
025016 004737 003576
025022 012701 002704
025026
025026
(3) 025026 104404
025030 012737 000004 002402
025036 111137 002374
025042 116137 000001 002376
025050 143737 002565 002374
025056 143737 002566 002376
025064 004737 004264
025070 004737 004076
025074 123737 002370 002374
025102 001416
025104 013737 002374 002404
025112 013737 002370 002406
025120 004737 004500
025124
(4) 025124 104455
(5) 025126 000003
(5) 025130 012174
(5) 025132 015114
025134
(3) 025134 104410
(3) 025136 000052
025140 005237 002402
025144 123737 002372 002671
025152 001416
025154 005037 002404
025160 113737 002671 002404
025166 013737 002372 002406
025174 004737 004500
025200
(4) 025200 104455

```
*****  
:SBTTL TEST 16 - REGS 15,16 / AX2-15,AX2-16 READ/WRITE BIT TEST  
:  
* USING REGS 15,16, THE INDIRECT REGS AX2-15,AX2-16 (USYRT REGS 4,5) ARE  
* WRITTEN AND READ USING EACH WORD OF PATTERN K. AX2-15 IS COMPARED  
* TO THE WORD WRITTEN, AND AX2-16 IS ALWAYS COMPARED TO 103. (AX2-16 IS NOT  
* WRITEABLE).  
* PATTERN K =  
* FOR REG 15: 000,377,125,252,001,002,004,010,020,040,100,200,000,000,  
* 000,000,000,000,000,000,376,375,373,367,357,337,277,177,  
* 377,377,377,377,377,377,377,377  
* FOR REG 16: 000,377,125,252,000,000,000,000,000,000,000,000,001,002,  
* 004,010,020,040,100,200,377,377,377,377,377,377,377,377,  
* 376,375,373,367,357,337,277,177.  
*****  
BGNTST  
T16::  
JSR PC,MSTCLR ;ISSUE MASTER CLEAR  
MOV #PATK,R1 ;INIT DATA PATTERN POINTER  
3$:  
BGNSEG  
MOV #4,AXNUM ;SET BYTE NO. = 4  
MOVB (R1),WAX15 ;SET DATA TO WRITE INTO LO BYTE  
MOVB 1(R1),WAX16 ;SET DATA TO WRITE INTO HI BYTE  
BICB ANBITS+4,WAX15 ;MASK OFF NON-READ/WRITE BITS IN LO BYTE  
BICB ANBITS+5,WAX16 ;MASK OFF NON-READ/WRITE BITS IN HI BYTE  
JSR PC,WRITAX ;LOAD DATA INTO AX2-15,AX2-16  
JSR PC,READAX ;READ AX2-15 AND AX2-16  
CMPB RAX15,WAX15 ;COMPARE LO BYTE DATA READ  
BEQ 6$ ;BR IF DATA MATCHES  
MOV WAX15,GOODAT ;SET EXPECTED DATA  
MOV RAX15,BADDAT ;SET ACTUAL DATA  
JSR PC,GETALL ;GET REGS FOR PRINTOUT  
:REPORT REG MISC  
ERRDF 3,EM3,ERR3  
TRAP C$BSEG  
TRAP C$ERDF  
.WORD 3  
.WORD EM3  
.WORD ERR3  
ESCAPE SEG  
TRAP C$ESCAPE  
.WORD 10000$-.  
6$:  
INC AXNUM ;SET AX BYTE NO. = 5  
CMPB RAX16,PATI+5 ;COMPARE HI BYTE DATA READ  
BEQ 9$ ;BR IF DATA MATCHES  
CLR GOODAT  
MOVB PATI+5,GOODAT ;SET EXPECTED DATA  
MOV RAX16,BADDAT ;SET ACTUAL DATA  
JSR PC,GETALL ;GET REGS FOR PRINTOUT  
:REPORT REG MISC  
ERRDF 3,EM3,ERR3  
TRAP C$ERDF
```

```
(5) 025202 000003 .WORD 3
(5) 025204 012174 .WORD EM3
(5) 025206 015114 .WORD ERR3
8675 025210 9$:
8676 025210 ENDSEG
(3) 025210 10000$:
(3) 025210 104405 TRAP C$ESEG
8677 025212 062701 000002 ADD #2,R1 ;INCR PATTERN POINTER
8678 025216 020127 003014 CMP R1,#PATL ;SEE IF ALL DATA WRITTEN YET
8679 025222 103701 BLO 3$ ;BR IF NOT DONE YET
8680 025224
(3) 025224
(3) 025224 104401 L10041:
8681 TRAP C$ETST
8682
8683
8684
8685
8686
8687
8688
8689
8690
8691
8692
8693
8694
8695
8696
8697 025226
(3) 025226
8698 025226 004737 003576 JSR PC,MSTCLR ;ISSUE MASTER CLEAR
8699 025232 012701 003014 MOV #PATL,R1 ;INIT DATA PATTERN POINTER
8700 025236
8701 025236 3$:
(3) 025236 104404 BGNSEG TRAP C$BSEG
8702 025240 012737 000000 002402 MOV #0,AXNUM ;SET BYTE NO. = 0
8703 025246 111137 002374 MOVB (R1),WAX15 ;SET DATA TO WRITE INTO LO BYTE
8704 025252 116137 000001 002376 MOVB 1(R1),WAX16 ;SET DATA TO WRITE INTO HI BYTE
8705 025260 143737 002561 002374 BICB ANBITS+0,WAX15 ;MASK OFF NON-READ/WRITE BITS IN LO BYTE
8706 025266 143737 002562 002376 BICB ANBITS+1,WAX16 ;MASK OFF NON-READ/WRITE BITS IN HI BYTE
8707 025274 004737 004264 JSR PC,WRITAX ;LOAD DATA INTO AX0-15,AX0-16
8708 025300 004737 004076 JSR PC,READAX ;READ AX0-15 AND AX0-16
8709 025304 123737 002370 002374 CMPB RAX15,WAX15 ;COMPARE LO BYTE DATA READ
8710 025312 001416 BEQ 6$ ;BR IF DATA MATCHES
8711 025314 013737 002374 002404 MOV WAX15,GOODAT ;SET EXPECTED DATA
8712 025322 013737 002370 002406 MOV RAX15,BADDAT ;SET ACTUAL DATA
8713 025330 004737 004500 JSR PC,GETALL ;GET REGS FOR PRINTOUT
8714 ;REPORT REG MISCOMPARE
8715 025334 ERRDF 3,EM3,ERR3
(4) 025334 104455 TRAP C$ERDF
(5) 025336 000003 .WORD 3
(5) 025340 012174 .WORD EM3
(5) 025342 015114 .WORD ERR3
8716 025344 ESCAPE SEG
(3) 025344 104410 TRAP C$ESCAPE
```

```
(3) 025346 000046 .WORD 10000$-
8717 025350 005237 002402 6$: INC AXNUM ;SET AX BYTE NO. - 1
8718 025354 123737 002372 002376 CMPB RAX16,WAX16 ;COMPARE HI BYTE DATA READ
8719 025362 001414 BEQ 9$ ;BR IF DATA MATCHES
8720 025364 013737 002376 002404 MOV WAX16,GOODAT ;SET EXPECTED DATA
8721 025372 013737 002372 002406 MOV RAX16,BADDAT ;SET ACTUAL DATA
8722 025400 004737 004500 JSR PC,GETALL ;GET REGS FOR PRINTOUT
8723 ;REPORT REG MISCOMPARE
8724 025404 ERRDF 3,EM3,ERR3
(4) 025404 104455 TRAP C$ERDF
(5) 025406 000003 .WORD 3
(5) 025410 012174 .WORD EM3
(5) 025412 015114 .WORD ERR3
8725 025414 9$:
8726 025414 ENDSEG
(3) 025414 10000$:
(3) 025414 104405 TRAP C$ESEG
8727 025416 062701 000002 ADD #2,R1 ;INCR PATTERN POINTER
8728 025422 020127 003022 CMP R1,#PATM ;SEE IF ALL DATA WRITTEN YET
8729 025426 103703 BLO 3$ ;BR IF NOT DONE YET
8730 025430
8731 (3) 025430 L10042:
8732 (3) 025430 104401 TRAP C$ETST
8733
8734
8735
8736
8737 ;*****
8738 .SBITL TEST 18 - AX1-15,AX1-16 READ/WRITE BIT TEST
8739 ;*
8740 ;* IN THIS TEST, A MASTER CLEAR IS DONE, AND THEN A WRITE, READ, AND COMPARE
8741 ;* ARE PERFORMED IN REGS AX1-15,AX1-16 USING EACH WORD OF PATTERN K.
8742 ;* ANY BITS IN AX1-15,AX1-16 WHICH ARE NOT READ/WRITE ARE MASKED OFF (TO 0)
8743 ;* IN THE EXPECTED VALUE BEFORE COMPARISON.
8744 ;*****
8745 025432 BGNST
8746 (3) 025432 T18::
8747 025432 004737 003576 JSR PC,MSTCLR ;ISSUE MASTER CLEAR
8748 025436 012701 002704 MOV #PATK,R1 ;INIT DATA PATTERN POINTER
8749 (3) 025442 104404 3$: BGNSEG TRAP C$BSEG
8750 025444 012737 000002 002402 MOV #2,AXNUM ;SET BYTE NO. = 2
8751 025452 111137 002374 MOVB (R1),WAX15 ;SET DATA TO WRITE INTO LO BYTE
8752 025456 116137 000001 002376 MOVB 1(R1),WAX16 ;SET DATA TO WRITE INTO HI BYTE
8753 025464 143737 002563 002374 BICB ANBITS+2,WAX15 ;MASK OFF NON-READ/WRITE BITS IN LO BYTE
8754 025472 143737 002564 002376 BICB ANBITS+3,WAX16 ;MASK OFF NON-READ/WRITE BITS IN HI BYTE
8755 025500 004737 004264 JSR PC,WRITAX ;LOAD DATA INTO AX1-15,AX1-16
8756 025504 004737 004076 JSR PC,READAX ;READ AX1-15 AND AX1-16
8757 025510 123737 002370 002374 CMPB RAX15,WAX15 ;COMPARE LO BYTE DATA READ
8758 025516 001416 BEQ 6$ ;BR IF DATA MATCHES
8759 025520 013737 002374 002404 MOV WAX15,GOODAT ;SET EXPECTED DATA
8760 025526 013737 002370 002406 MOV RAX15,BADDAT ;SET ACTUAL DATA
8761 025534 004737 004500 JSR PC,GETALL ;GET REGS FOR PRINTOUT
;REPORT REG MISCOMPARE
```

| | | | | | | | | | | | |
|------|--------|--------|--------|--------|---------|------------|--------------|------------------------------|----------|-----------|--|
| 8762 | 025540 | | | | ERRDF | 3,EM3,ERR3 | | | | | |
| (4) | 025540 | 104455 | | | | | | | TRAP | C\$ERDF | |
| (5) | 025542 | 000003 | | | | | | | .WORD | 3 | |
| (5) | 025544 | 012174 | | | | | | | .WORD | EM3 | |
| (5) | 025546 | 015114 | | | | | | | .WORD | ERR3 | |
| 8763 | 025550 | | | | ESCAPE | SEG | | | | | |
| (3) | 025550 | 104410 | | | | | | | TRAP | C\$ESCAPE | |
| (3) | 025552 | 000046 | | | | | | | .WORD | 10000\$- | |
| 8764 | 025554 | 005237 | 002402 | | 6\$: | INC | AXNUM | :SET AX BYTE NO. = 3 | | | |
| 8765 | 025560 | 123737 | 002372 | 002376 | | CMPB | RAX16,WAX16 | :COMPARE HI BYTE DATA READ | | | |
| 8766 | 025566 | 001414 | | | | BEQ | 9\$ | :BR IF DATA MATCHES | | | |
| 8767 | 025570 | 013737 | 002376 | 002404 | | MOV | WAX16,GOODAT | :SET EXPECTED DATA | | | |
| 8768 | 025576 | 013737 | 002372 | 002406 | | MOV | RAX16,BADDAT | :SET ACTUAL DATA | | | |
| 8769 | 025604 | 004737 | 004500 | | | JSR | PC,GETALL | :GET REGS FOR PRINTOUT | | | |
| 8770 | | | | | :REPORT | REG MISC | COMPARE | | | | |
| 8771 | 025610 | | | | ERRDF | 3,EM3,ERR3 | | | | | |
| (4) | 025610 | 104455 | | | | | | | TRAP | C\$ERDF | |
| (5) | 025612 | 000003 | | | | | | | .WORD | 3 | |
| (5) | 025614 | 012174 | | | | | | | .WORD | EM3 | |
| (5) | 025616 | 015114 | | | | | | | .WORD | ERR3 | |
| 8772 | 025620 | | | | 9\$: | | | | | | |
| 8773 | 025620 | | | | ENDSEG | | | | | | |
| (3) | 025620 | | | | | | | | 10000\$: | | |
| (3) | 025620 | 104405 | | | | | | | TRAP | C\$ESEG | |
| 8774 | 025622 | 062701 | 000002 | | ADD | #2,R1 | | :INCR PATTERN POINTER | | | |
| 8775 | 025626 | 020127 | 003014 | | CMP | R1,#PATL | | :SEE IF ALL DATA WRITTEN YET | | | |
| 8776 | 025632 | 103703 | | | BLO | 3\$ | | :BR IF NOT DONE YET | | | |
| 8777 | 025634 | | | | ENDTST | | | | | | |
| (3) | 025634 | | | | | | | | L10043: | | |
| (3) | 025634 | 104401 | | | | | | | TRAP | C\$ETST | |

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
(3)

```
*****  
:SBTTL TEST 19 - AX3-15,AX3-16 READ/WRITE BIT TEST  
:  
:* IN THIS TEST A MASTER CLEAR IS DONE AND THEN A WRITE, READ, AND COMPARE ARE  
:* PERFORMED IN REGS AX3-15,AX3-16 USING EACH WORD OF PATTERN V FOR WRITING,  
:* AND PATTERN U FOR COMPARING.  
:* ANY BITS IN AX3-15,AX3-16 WHICH ARE NOT READ/WRITE ARE MASKED OFF (TO 0)  
:* IN THE EXPECTED VALUE BEFORE COMPARISON.  
:* PATTERN V =  
:* FOR REG 15 : 000,333,331,323,313,233,133,000,000,000,000,  
:* 000,000,000,000,000,000,000,000  
:* FOR REG 16 : 000,000,000,000,000,000,000,001,002,004,040,  
:* 100,200,346,345,343,307,247,147  
:* PATTERN U -  
:* FOR REG 15 : 000,001,013,011,021,101,301,000,000,000,000,  
:* 000,000,000,000,000,000,000,000  
:* FOR REG 16 : 000,000,000,000,000,000,000,001,002,004,040,  
:* 100,200,346,345,343,307,247,147  
:*****  
:BGNTST
```

```

8803 025636 004737 003576          JSR    PC,MSTCLR          ;ISSUE MASTER CLEAR
8804 025642 142777 000010 154600  BICB   #LULOOP,#BSEL1    ;CLEAR LULOOP
8805 025650 012702 003152          MOV    #PATV,R2          ;INIT PATTERN V POINTER
8806 025654 012701 003104          MOV    #PATU,R1          ;INIT PATTERN U POINTER
8807 025660          3$:
8808 025660          BGNSEG
(3) 025660 104404          TRAP   C$BSEG
8809 025662 012737 000006 002402  MOV    #6,AXNUM          ;SET BYTE NO. = 6
8810 025670 111237 002374          MOVB  (R2),WAX15         ;SET DATA TO WRITE INTO LO BYTE
8811 025674 116237 000001 002376  MOVB  1(R2),WAX16        ;SET DATA TO WRITE INTO HI BYTE
8812 025702 004737 004264          JSR    PC,WRITAX         ;LOAD DATA INTO AX3-15,AX3-16
8813 025706 004737 004076          JSR    PC,READAX        ;READ AX3-15 AND AX3-16
8814 025712 132737 000001 002374  BITB  #TEST,WAX15        ;SEE IF AN INTERFACE IS SELECTED
8815 025720 001003          BNE   4$                ;BR IF YES
8816 025722 142737 000332 002370  BICB  #AX315U,RAX15      ;MASK OFF UNPREDICTABLE BITS
8817 025730 123711 002370 4$:    CMPB  RAX15,(R1)         ;COMPARE LO BYTE DATA READ
8818 025734 001417          BEQ   6$                ;BR IF DATA MATCHES
8819 025736 005037 002404          CLR   GOODAT
8820 025742 111137 002404          MOVB  (R1),GOODAT       ;SET EXPECTED DATA
8821 025746 013737 002370 002406  MOV    RAX15,BADDAT      ;SET ACTUAL DATA
8822 025754 004737 004500          JSR    PC,GETALL        ;GET REGS FOR PRINTOUT
8823          ;REPORT REG MISCMPARE
8824 025760          ERRDF 3,EM3,ERR3
(4) 025760 104455          TRAP   C$ERDF
(5) 025762 000003          .WORD 3
(5) 025764 012174          .WORD EM3
(5) 025766 015114          .WORD ERR3
8825 025770          ESCAPE SEG
(3) 025770 104410          TRAP   C$ESCAPE
(3) 025772 000052          .WORD 10000$.
8826 025774 005237 002402 6$:    INC    AXNUM             ;SET AX BYTE NO. = 7
8827 026000 123761 002372 000001  CMPB  RAX16,1(R1)        ;COMPARE HI BYTE DATA READ
8828 026006 001416          BEQ   9$                ;BR IF DATA MATCHES
8829 026010 005037 002404          CLR   GOODAT
8830 026014 116137 000001 002404  MOVB  1(R1),GOODAT       ;SET EXPECTED DATA
8831 026022 013737 002372 002406  MOV    RAX16,BADDAT      ;SET ACTUAL DATA
8832 026030 004737 004500          JSR    PC,GETALL        ;GET REGS FOR PRINTOUT
8833          ;REPORT REG MISCMPARE
8834 026034          ERRDF 3,EM3,ERR3
(4) 026034 104455          TRAP   C$ERDF
(5) 026036 000003          .WORD 3
(5) 026040 012174          .WORD EM3
(5) 026042 015114          .WORD ERR3
8835 026044          9$:
8836 026044          ENDSEG
(3) 026044          10000$:
(3) 026044 104405          TRAP   C$ESEG
8837 026046 062702 000002          ADD    #2,R2            ;INCR PATTERN V POINTER
8838 026052 062701 000002          ADD    #2,R1            ;INCR PATTERN U POINTER
8839 026056 020127 003152          CMP    R1,#PATV         ;SEE IF ALL DATA WRITTEN YET
8840 026062 103676          BLO   3$                ;BR IF NOT DONE YET
8841 026064          ENDTST
(3) 026064          L10044:
(3) 026064 104401          TRAP   C$ETST
8842
8843

```

8844
8845
8846
8847
8848
8849
8850
8851
8852
8853
8854
8855
8856
8857
8858
(3)
8859
8860
8861
8862
8863
8864
8865
8866
8867
(3)
(3)
8868
8869
(3)
8870
8871
8872
8873
8874
8875
8876
8877
8878
8879
8880
8881
8882
(4)
(5)
(5)
(5)
8883
8884
(3)
(3)
8885
8886
8887
8888
8889

026066
026066
026066 004737 003576
026072 012701 003050
026076 012702 003066
026102 012737 000017 002400
026110 012737 000005 002402

026116
026116 104402
026120
026120 104404
026122 111137 002366
026126 004737 003722
026132 004737 004076
026136 123712 002372
026142 001421
026144 005037 002410
026150 111137 002410
026154 005037 002404
026160 111237 002404
026164 013737 002372 002406
026172 004737 004500

026176
026176 104455
026200 000003
026202 012174
026204 016254
026206
026206 104405
026210 005201
026212 005202
026214 020127 003066
026220 103737
026222

```
*****
:SBITL      TEST 20 - REG 17 - AX2-16 READ/WRITE, MASTER CLEAR TEST
:
: * THIS TEST CONSISTS OF 2 SUBTESTS. IN THE FIRST SUBTEST, EACH BYTE OF PAT 0
: * IS WRITTEN INTO REG 17 AND AFTER EACH WRITE, AX2-16 IS READ AND COMPARED
: * TO A BYTE OF PAT P.
: * PATTERN O = 000,041,004,010,020,040,100,101,200,201,300,111,301,375
: * PATTERN P = 000,113,200,040,020,010,001,104,007,105,007,144,107,157
: * IN THE SECOND SUBTEST, REG 17 IS LOADED WITH 375, A MASTER CLEAR IS ISSUED,
: * AND AX2-16 IS COMPARED TO ITS INITIALIZED STATE (103).
:*****
BGNTST
                                           T20::
      JSR      PC,MSTCLR      ;ISSUE MASTER CLEAR
      MOV      #PATO,R1      ;INIT PAT 0 POINTER
      MOV      #PATP,R2      ;INIT PAT P POINTER
      MOV      #17,REGNUM    ;SET LU REG NO. = 17
      MOV      #5,AXNUM      ;SET AX BYTE NO. = 5 FOR AX2-16
-----
: WRITE REG 17, READ AND COMPARE AX2-16
-----
      BGNSUB
                                           T20.1:
3$: TRAP      C$BSUB
      BGNSEG
                                           TRAP      C$BSEG
      MOVB     (R1),WRIBYT    ;SET BYTE TO BE WRITTEN
      JSR      PC,WRITLU     ;WRITE DATA BYTE INTO REG 17
      JSR      PC,READAX     ;READ AX2
      CMPB     RAX16,(R2)    ;COMPARE AX2-16 TO EXPECTED DATA
      BEQ      6$           ;BR IF DATA MATCHES
      CLR      LOADAT
      MOVB     (R1),LOADAT   ;SET DATA WHICH WAS WRITTEN
      CLR      GOODAT
      MOVB     (R2),GOODAT   ;SET EXPECTED DATA READ
      MOV      RAX16,BADDAT ;SET ACTUAL DATA READ
      JSR      PC,GETALL     ;GET REGS FOR PRINTOUT
:REPORT REG MISCMPARE
      ERRDF   3,EM3,ERR5
                                           TRAP      C$ERDF
                                           .WORD    3
                                           .WORD    EM3
                                           .WORD    ERR5
6$:
      ENDSEG
                                           10000$:
                                           TRAP      C$ESEG
      INC      R1            ;INCR PAT 0 POINTER
      INC      R2            ;INCR PAT P POINTER
      CMP      R1,#PATP     ;SEE IF ALL BYTES LOADED YET
      BLO     3$           ;BR IF NOT
      ENDSUB
```

```
(3) 026222  
(3) 026222 104403  
8890  
8891  
8892  
8893 026224  
(3) 026224  
(3) 026224 104402  
8894 026226 112737 000375 002366  
8895 026234 004737 003722  
8896 026240 004737 003576  
8897 026244 004737 004076  
8898 026250 123737 002372 002671  
8899 026256 001416  
8900 026260 005037 002404  
8901 026264 113737 002671 002404  
8902 026272 013737 002372 002406  
8903 026300 004737 004500  
8904  
8905 026304  
(4) 026304 104455  
(5) 026306 000002  
(5) 026310 012135  
(5) 026312 015114  
8906 026314  
8907 026314  
(3) 026314  
(3) 026314 104403  
8908 026316  
(3) 026316  
(3) 026316 104401  
8909  
8910  
8911  
8912  
8913  
8914  
8915  
8916  
8917  
8918  
8919  
8920  
8921  
8922  
8923  
8924  
8925  
8926 026320  
(3) 026320  
8927 026320 012701 003032  
8928 026324 012737 026566 002362  
8929 026332  
(3) 026332  
(3) 026332 104402  
8930 026334 004737 003576
```

```
L10046: TRAP C$ESUB  
-----  
: LOAD REG 17, DO MASTER CLEAR, READ AND COMPARE AX2-16  
-----  
BGNSUB  
T20.2: TRAP C$BSUB  
MOV #375,WRIBYT ;SET DATA TO BE LOADED  
JSR PC,WRITLU ;LOAD DATA INTO REG 17  
JSR PC,MSTCLR ;PERFORM MASTER CLEAR  
JSR PC,READAX ;READ AX2-15,AX2-16  
CMPB RAX16,PATI+5 ;SEE IF AX2-16 WAS INIT'D CORRECTLY  
BEQ 6$ ;BR IF YES  
CLR GOODAT  
MOVB PATI+5,GOODAT ;SET EXPECTED DATA  
MOV RAX16,BADDAT ;SET ACTUAL DATA  
JSR PC,GETALL ;GET REGS FOR PRINTOUT  
:REPORT REG NOT INITIALIZED BY MASTER CLEAR  
ERRDF 2,EM2,ERR3  
TRAP C$ERDF  
.WORD 2  
.WORD EM2  
.WORD ERR3  
6$:  
ENDSUB  
L10047: TRAP C$ESUB  
ENDTST  
L10045: TRAP C$ETST
```

```
*****  
:SBITL TEST 21 - TRANSMITTER BUFFER DATA TEST  
:* A MASTER CLEAR IS DONE FIRST, AND THEN A BYTE OF PATTERN N IS LOADED INTO  
:* REG 11 AND THE NEXT BYTE IS LOADED TWICE INTO REG 10. THE PROGRAM THEN WAITS  
:* AT LEAST 50 MICRO-SEC, AND THEN IT READS AND COMPARES AX1-15 TO THE BYTE  
:* WHICH WAS LOADED INTO REG 10, AND IT READS AND COMPARES AX1-16 TO THE BYTE  
:* WHICH WAS LOADED INTO REG 11. THIS PROCESS IS REPEATED (INCLUDING THE MASTER  
:* CLEAR) FOR EACH PAIR OF BYTES IN PATTERN N.  
:* PATTERN N =  
:* FOR REG 10: 000,125,252,377,000,000,000  
:* FOR REG 11: 000,000,000,000,005,012,017  
*****
```

```
BGNTST  
T21.:  
MOV #PATN,R1 ;INIT PATTERN POINTER  
MOV #A2,RETADR ;SET SUBROUTINE ERROR RETURN ADDRESS  
T21.1: TRAP C$BSUB  
3$: JSR PC,MSTCLR ;ISSUE MASTER CLEAR
```

```

8931 026340 004737 004634 JSR PC,OSIRDY ;CHECK ORDY AND GCOR FOR EXPECTED STATES
8932 026344 000001 1 MOV #1,REGNUM ;SET LU REG NO. = 11
8933 026346 012737 000011 002400 MOVB (R1),WRIBYT ;SET DATA BYTE TO BE WRITTEN
8934 026354 111137 002366 JSR PC,WRITLU ;WRITE BYTE INTO REG 11
8935 026360 004737 003722 JSR PC,WRITLU ;WRITE BYTE INTO REG 11
8936 026364 012737 000010 002400 MOV #10,REGNUM ;SET LU REG NO. = 10
8937 026372 116137 000001 002366 MOVB 1(R1),WRIBYT ;SET DATA BYTE TO BE WRITTEN
8938 026400 004737 003722 JSR PC,WRITLU ;WRITE BYTE INTO REG 10
8939 026404 004737 003722 JSR PC,WRITLU ;WRITE IT AGAIN (SO 2 ENTRIES ARE IN SILO)
8940 026410 004737 005120 JSR PC,WAIT50 ;WAIT FOR SILO DATA TO RIPPLE
8941 026414 004737 004634 JSR PC,OSIRDY ;CHECK ORDY AND OCOR FOR EXPECTED STATES
8942 026420 000003 3 MOV #2,AXNUM ;SET BYTE NO. FOR AX1-15
8943 026422 012737 000002 002402 JSR PC,READAX ;READ AX1-15, AX1-16
8944 026430 004737 004076 CMPB RAX15,1(R1) ;COMPARE AX1-15 TO EXPECTED
8945 026434 123761 002370 000001 BEQ 6$ ;BR IF MATCH
8946 026442 001420 CLR GOODAT
8947 026444 005037 002404 MOVB 1(R1),GOODAT ;SET EXPECTED DATA
8948 026450 116137 000001 002404 MOV RAX15,BADDAT ;SET ACTUAL DATA
8949 026456 013737 002370 002406 JSR PC,GETALL ;GET REGS FOR PRINTOUT
8950 026464 004737 004500 ;REPORT REG MISCMPARE
8951 ERRDF 3,EM3,ERR3
8952 026470 (4) 026470 104455 TRAP C$ERDF
(5) 026472 000003 .WORD 3
(5) 026474 012174 .WORD EM3
(5) 026476 015114 .WORD ERR3
8953 026500 ESCAPE SUB TRAP C$ESCAPE
(3) 026500 104410 .WORD L10051-.
(3) 026502 000064
8954 026504 005237 002402 6$: INC AXNUM ;INCR AX BYTE NO.
8955 026510 123711 002372 CMPB RAX16,(R1) ;COMPARE AX1-16 TO EXPECTED
8956 026514 001417 BEQ 9$ ;BR IF MATCH
8957 026516 005037 002404 CLR GOODAT
8958 026522 111137 002404 MOVB (R1),GOODAT ;SET EXPECTED DATA
8959 026526 013737 002372 002406 MOV RAX16,BADDAT ;SET ACTUAL DATA
8960 026534 004737 004500 JSR PC,GETALL ;GET REGS FOR PRINTOUT
8961 ;REPORT REG MISCMPARE
8962 ERRDF 3,EM3,ERR3
8962 026540 (4) 026540 104455 TRAP C$ERDF
(5) 026542 000003 .WORD 3
(5) 026544 012174 .WORD EM3
(5) 026546 015114 .WORD ERR3
8963 026550 ESCAPE SUB TRAP C$ESCAPE
(3) 026550 104410 .WORD L10051-.
(3) 026552 000014
8964 026554 062701 000002 9$: ADD #2,R1 ;INCR DATA PATTERN POINTER
8965 026560 020127 003050 CMP R1,#PATO ;SEE IF ALL DATA BYTES WRITTEN YET
8966 026564 103663 BLO 3$ ;BR IF NOT DONE YET
8967 026566 A2:
8968 026566 ENDSUB
(3) 026566 L10051:
(3) 026566 104403 TRAP C$ESUB
8969 026570 ENDTST
(3) 026570 L10050:
(3) 026570 104401 TRAP C$ETST
8970

```


8971
 8972
 8973
 8974
 8975
 8976
 8977
 8978
 8979
 8980
 8981
 8982
 8983
 8984
 8985
 8986
 8987
 8988
 8989
 8990
 8991 026572
 (3) 026572
 8992 026572 012737 027246 002362
 8993
 8994
 8995
 8996 026600 004737 003576
 8997 026604 004737 004634
 8998 026610 000001
 8999
 9000
 9001
 9002 026612 012737 000400 002422
 9003 026620 004737 005146
 9004 026624 004737 005146
 9005 026630 004737 005120
 9006 026634 004737 004634
 9007 026640 000003
 9008
 9009
 9010
 9011 026642 005001
 9012 026644 012737 000017 002400
 9013 026652 004737 005226
 9014 026656 000001
 9015 026660 004737 003644
 9016 026664 132737 000020 002364
 9017 026672 001404
 9018 026674 005201
 9019 026676 020127 000003
 9020 026702 002763
 9021 026704 004737 004634
 9022 026710 000001
 9023
 9024
 9025

```

:*****
.SBTTL      TEST 22 - TRANSMITTER BUFFER SEQUENCING TEST
:
:* FIRST, A MASTER CLEAR IS DONE, AND THE PROGRAM CHECKS FOR ORDY=1, OCOR=0.
:* THEN, 2 TSOM CHARS ARE LOADED INTO THE TX SILO, AND ALLOWED TO RIPPLE
:* DOWN TO THE OUTPUT. THE PROGRAM CHECKS FOR ORDY=1, OCOR=1.
:* NEXT, THE PROGRAM CYCLES THE STEPLU BIT UNTIL OCOR=0 AGAIN, AND CHECKS FOR
:* THIS TO OCCUR WITHIN 3 CYCLES.
:* THE SILO IS THEN FILLED WITH 64 BYTES OF A 256-BYTE BINARY COUNT PATTERN
:* (000-377) AND THE PROGRAM CHECKS FOR ORDY=0 AFTER THE 64TH CHAR IS LOADED.
:* THE PROGRAM CYCLES STEPLU FOR 8 CYCLES AND CHECKS THAT AFTER THE 8TH, ORDY=1.
:* AX1-15 IS READ AND COMPARED TO EXPECTED DATA.
:* THE REST OF THE BINARY COUNT DATA BYTES ARE LOADED, CYCLED 8 CLOCKS, READ AND
:* COMPARED, A BYTE AT A TIME. UPON COMPLETION, THE SILO IS CHECKED TO BE EMPTY
:* WITH ORDY=1, OCOR=0.
:*****
BGNTST
                                           T22::
MOV      #A3,RETADR      ;SET SUBR ERROR RETURN ADDR
-----
: SET MASTER CLEAR, CHECK FOR ORDY=1, OCOR=0
-----
JSR      PC,MSTCLR      ;ISSUE MASTER CLEAR
JSR      PC,OSIRDY      ;CHK ORDY=1, OCOR=0
1
-----
: LOAD 2 SOM CHARS, ALLOW SILO TO RIPPLE, CHK ORDY=1, OCOR=1
-----
MOV      #TXSOM,TXWORD  ;SET DATA TO WRITE INTO SILO
JSR      PC,LDTXSI      ;LOAD THE SILO WITH SOM
JSR      PC,LDTXSI      ;LOAD ANOTHER SOM
JSR      PC,WAIT50      ;WAIT FOR DATA TO RIPPLE
JSR      PC,OSIRDY      ;CHK ORDY=1, OCOR=1
3
-----
: CLOCK LINE UNIT, CHK FOR OCOR = 0 WITHIN 3 CYCLES
-----
CLR      R1              ;INIT CYCLE COUNTER TO 0
MOV      #17,REGNUM     ;SET REG NO. = 17
3$: JSR      PC,STPLU      ;STEP LU 1 CYCLE
1
JSR      PC,READLU      ;READ REG 17
BITB     #OCOR,REDBYT   ;SEE IF OCOR = 0 YET
6$: BEQ     6$           ;BR IF OCOR = 0
INC      R1              ;INCR CYCLE COUNT
CMP      R1,#3          ;SEE IF 3 CYCLES DONE YET
3$: BLT     3$           ;BR IF NO
6$: JSR      PC,OSIRDY   ;CHK ORDY=1, OCOR=0
1
-----
: LOAD 64 BINARY COUNT CHARS INTO SILO, CHK ORDY=0
-----

```

```

9026 026712 005003          CLR      R3          ;INIT PATTERN FOR WRITING
9027 026714 010337 002422 8$:  MOV      R3,TXWORD ;SET DATA TO BE WRITTEN
9028 026720 004737 005146   JSR      PC,LDTXSI  ;LOAD DATA CHAR INTO TX SILO
9029 026724 004737 005120   JSR      PC,WAIT50  ;WAIT FOR DATA TO RIPPLE IN SILO
9030 026730 020327 000077   CMP      R3,#63.    ;SEE IF 64TH CHAR JUST LOADED
9031 026734 001004          BNE      9$         ;BR IF NO
9032 026736 012737 000002 026760  MOV      #2,14$    ;SET UP TO CHK ORDY=0,OCOR=1
9033 026744 000403          BR       12$       ;
9034 026746 012737 000003 026760 9$:  MOV      #3,14$    ;SET UP TO CHK ORDY=1,OCOR=1
9035 026754 004737 004634 12$:  JSR      PC,OSIRDY ;CHK ORDY, OCOR
9036 026760 000000          .WORD   0         ;
9037 026762 005203          INC      R3         ;INCR PATTERN FOR WRITES
9038 026764 020327 000100   CMP      R3,#64.    ;SEE IF 64 CHARS LOADED YET
9039 026770 002751          BLT     8$         ;BR IF NO
9040
9041  ;-----
9042  ; CLOCK LINE UNIT, CHECK ORDY = 1 WITHIN 8 CYCLES
9043 026772 012737 000011 002400  MOV      #11,REGNUM ;SET REG NO. = 11
9044 027000 005001          CLR      R1         ;INIT CYCLE COUNT
9045 027002 004737 005226 16$:  JSR      PC,STPLU  ;CLOCK LU FOR 1 CYCLE
9046 027006 000001          1
9047 027010 004737 003644   JSR      PC,READLU ;READ REG 11
9048 027014 132737 000020 002364  BITB     #ORDY,REDBYT ;SEE IF ORDY = 1 YET
9049 027022 001004          BNE      19$       ;BR IF YES
9050 027024 005201          INC      R1         ;INCR CYCLE COUNT
9051 027026 020127 000010   CMP      R1,#8.    ;SEE IF 8 CYCLES YET
9052 027032 002763          BLT     16$       ;BR IF NOT YET
9053 027034 004737 004634 19$:  JSR      PC,OSIRDY ;CHK ORDY = 1, OCOR = 1
9054 027040 000003          3
9055
9056  ;-----
9057  ; READ AND COMPARE FIRST CHARACTER IN AX1-15
9058 027042 005004          CLR      R4         ;INIT PATTERN FOR READING
9059 027044 012737 000002 002402  MOV      #2,AXNUM  ;SET AX BYTE NO. FOR AX1-15
9060 027052 004737 004076   JSR      PC,READAX ;READ AX1-15
9061 027056 123704 002370   CMPB     RAX15,R4  ;COMPARE AX1-15 TO EXPECTED
9062 027062 001415          BEQ     20$       ;BR IF MATCH
9063 027064 010437 002404   MOV      R4,GOODAT ;SET EXPECTED DATA
9064 027070 013737 002370 002406  MOV      RAX15,BADDAT ;SET ACTUAL DATA
9065 027076 004737 004500   JSR      PC,GETALL ;GET REGS FOR PRINTOUT
9066  ;REPORT REG MISCMPARE
9067 027102          ERRDF  3,EM3,ERR3
9068 027112          ESCAPE TST
9069 027116          TRAP   C$ERDF
9070 027102 104455          .WORD   3
9071 027104 000003          .WORD   EM3
9072 027106 012174          .WORD   ERR3
9073 027110 015114          TRAP   C$ESCAPE
9074 027112 104410          .WORD   L10052-.
9075 027114 000132
9076 027116
20$:
9077  ;-----
9078  ; LOAD AND COMPARE REST OF CHARS, MONITOR ORDY, OCOR
9079
9080 027116 020327 000377 24$:  CMP      R3,#255.  ;SEE IF ALL CHARS LOADED YET
9081 027122 003010          BGT     26$       ;BR IF YES
9082 027124 010337 002422   MOV      R3,TXWORD ;SET DATA TO BE WRITTEN

```

```
9076 027130 004737 005146 JSR PC,LDTXSI ;LOAD DATA CHAR INTO TX SILO
9077 027134 005203 INC R3 ;INCR DATA TO BE WRITTEN
9078 027136 004737 004634 JSR PC,OSIRDY ;CHK ORDY=0, OCOR=1
9079 027142 000002 2
9080 027144 005204 26$: INC R4 ;INCR PAT FOR READING
9081 027146 004737 005226 JSR PC,STPLU ;CLOCK LINE UNIT FOR 8 CYCLES
9082 027152 000010 8.
9083 027154 004737 004076 JSR PC,READAX ;READ AX1-15
9084 027160 123704 002370 CMPB RAX15,R4 ;COMPARE AX1-15 TO EXPECTED
9085 027164 001415 BEQ 27$ ;BR IF MATCH
9086 027166 010437 002404 MOV R4,GOODAT ;SET EXPECTED DATA
9087 027172 013737 002370 002406 MOV RAX15,BADDAT ;SET ACTUAL DATA
9088 027200 004737 004500 JSR PC,GETALL ;GET REGS FOR PRINTOUT
9089
9090 027204 ;REPORT REG MISCOMPARE
(4) 027204 104455 ERRDF 3,EM3,ERR3
(5) 027206 000003 TRAP C$ERDF
(5) 027210 012174 .WORD 3
(5) 027212 015114 .WORD EM3
9091 027214 ESCAPE TST .WORD ERR3
(3) 027214 104410 TRAP C$ESCAPE
(3) 027216 000030 .WORD L10052-.
9092 027220 020427 000377 27$: CMP R4,#255. ;SEE IF WE READ LAST CHAR YET
9093 027224 001004 BNE 29$ ;BR IF NOT
9094 027226 004737 004634 JSR PC,OSIRDY ;CHK ORDY=1, OCOR=0
9095 027232 000001 1
9096 027234 000404 BR A3
9097 027236 004737 004634 29$: JSR PC,OSIRDY ;CHK ORDY=1, OCOR=1
9098 027242 000003 3
9099 027244 000724 BR 24$ ;CONTINUE
9100 027246 A3:
9101 027246 ENDTST
(3) 027246
(3) 027246 104401 L10052: TRAP C$ETST
9102
9103
9104
9105
9106
9107
9108 ;*****
9109 .SBTTL TEST 23 - TX MSG TIMING TEST, CHAR MODE, WITH CRC
9110 ;*
9111 ;* IN THIS TEST, AN ENTIRE MESSAGE IS TRANSMITTED (USING STEPLU AND LULOOP)
9112 ;* AND THE PROGRAM MONITORS THE OCCURRENCE OF USYRT TX BUFFER EMPTY FLAGS
9113 ;* (BY SCANNING ORDY AND OCOR) AND OACT, AT EACH STEP. THE TEST IS DONE IN
9114 ;* CHARACTER ORIENTED PROTOCOL MODE, USING 8-BIT CHARS AND CRC-16.
9115 ;* THE FOLLOWING STEPS ARE DONE:
9116 ;* A MASTER CLEAR IS DONE, AND THE LINE UNIT IS PLACED IN CHAR MODE.
9117 ;* SOM IS SET TWICE TO SEND 2 SYNCH CHARS. THEN, 2 000 CHARS ARE SENT, AND
9118 ;* THEN 2 TERMINATING SYNCHS ARE SENT.
9119 ;* THE TRANSMITTER IS THEN DISABLED, USING OC, AND OACT IS MONITORED FOR THE
9120 ;* CLEARED STATE AFTER THE 3RD SYNCH COMPLETES.
9121 ;*****
9121 027250 BGNTST
(3) 027250
9122 027250 012737 027362 002362 MOV #A4,RETADR ;SET TEST EXIT ADDRESS FOR ERRORS T23::
```

| | | | | | | |
|------|--------|--------|--------|-------------|-----------|--|
| 9123 | 027256 | 004737 | 005512 | JSR | PC,INITRN | ;DO MASTER CLR, LOAD 2 SOM'2 |
| 9124 | 027262 | 000226 | | SYNCH | | |
| 9125 | 027264 | 000011 | | STRIP!DDCMP | | |
| 9126 | 027266 | 004737 | 006074 | JSR | PC,TXCHAR | ;LOAD A 000 CHAR, TX FIRST SYNCH (226) |
| 9127 | 027272 | 000000 | | 000 | | |
| 9128 | 027274 | 100010 | | CHPCHK!8. | | |
| 9129 | 027276 | 004737 | 006074 | JSR | PC,TXCHAR | ;LOAD 2ND 000 CHAR, TX 2ND SYNCH |
| 9130 | 027302 | 000000 | | 000 | | |
| 9131 | 027304 | 000010 | | 8. | | |
| 9132 | 027306 | 004737 | 006074 | JSR | PC,TXCHAR | ;LOAD EOM CHAR, TX FIRST 000 CHAR |
| 9133 | 027312 | 001000 | | TXEOM | | |
| 9134 | 027314 | 000010 | | 8. | | |
| 9135 | 027316 | 004737 | 006074 | JSR | PC,TXCHAR | ;LOAD EOM CHAR, TX 2ND 000 CHAR |
| 9136 | 027322 | 001000 | | TXEOM | | |
| 9137 | 027324 | 000010 | | 8. | | |
| 9138 | 027326 | 004737 | 006074 | JSR | PC,TXCHAR | ;LOAD EOM, TX CRC-16 CHAR |
| 9139 | 027332 | 001000 | | TXEOM | | |
| 9140 | 027334 | 000020 | | 16. | | |
| 9141 | 027336 | 004737 | 006074 | JSR | PC,TXCHAR | ;LOAD EOM, TX FIRST TERMINATING SYNCH |
| 9142 | 027342 | 001000 | | TXEOM | | |
| 9143 | 027344 | 000010 | | 8. | | |
| 9144 | 027346 | 004737 | 006074 | JSR | PC,TXCHAR | ;LOAD EOM, TX 2ND TERMINATING SYNCH |
| 9145 | 027352 | 001000 | | TXEOM | | |
| 9146 | 027354 | 000010 | | 8. | | |
| 9147 | 027356 | 004737 | 006246 | JSR | PC,ENDTRN | ;SET OC, MONITOR OCOR |
| 9148 | 027362 | | | A4: | | |
| 9149 | 027362 | 004737 | 003576 | JSR | PC,MSTCLR | ;ISSUE MASTER CLEAR TO CLEAN UP |
| 9150 | 027366 | | | ENDTST | | |
| (3) | 027366 | | | | | L10053: TRAP C\$ETST |
| (3) | 027366 | 104401 | | | | |

| | | | | | | |
|------|--------|--------|--------|--------|-----|--|
| 9151 | | | | | | |
| 9152 | | | | | | |
| 9153 | | | | | | |
| 9154 | | | | | | |
| 9155 | | | | | | |
| 9156 | | | | | | |
| 9157 | | | | | | |
| 9158 | | | | | | |
| 9159 | | | | | | |
| 9160 | | | | | | |
| 9161 | | | | | | |
| 9162 | | | | | | |
| 9163 | | | | | | |
| 9164 | | | | | | |
| 9165 | | | | | | |
| 9166 | | | | | | |
| 9167 | | | | | | |
| 9168 | | | | | | |
| 9169 | | | | | | |
| 9170 | 027370 | | | | | |
| (3) | 027370 | | | | | |
| 9171 | 027370 | 012737 | 027472 | 002362 | MOV | #A5,RETADR ;SET TEST EXIT ADDRESS FOR ERRORS |
| 9172 | 027376 | 004737 | 005512 | | JSR | PC,INITRN ;DO MASTER CLR, LOAD 2 SOM'2 |
| 9173 | 027402 | 000000 | | | 000 | |
| 9174 | 027404 | 000000 | | | 000 | |
| 9175 | 027406 | 004737 | 006074 | | JSR | PC,TXCHAR ;LOAD A 000 CHAR, TX FIRST FLAG |

```
::*****  
:SBTTL TEST 24 - TX MSG TIMING TEST, BIT MODE, WITH CRC  
:*  
:* IN THIS TEST, AN ENTIRE MESSAGE IS TRANSMITTED (USING STEPLU AND LULOOP)  
:* AND THE PROGRAM MONITORS THE OCCURRENCE OF USYRT TX BUFFER EMPTY FLAGS  
:* (BY SCANNING ORDY AND OCOR) AND OACT, AT EACH STEP. THE TEST IS DONE IN  
:* BIT ORIENTED PROTOCOL MODE, USING 8-BIT CHARS AND CRC-CCITT-1.  
:* THE FOLLOWING STEPS ARE DONE:  
:* A MASTER CLEAR IS DONE, AND THE LINE UNIT IS PLACED IN BIT MODE.  
:* SOM IS SET TWICE TO SEND 2 FLAG CHARS. THEN, 2 000 CHARS ARE SENT, AND  
:* THEN 2 TERMINATING FLAGS ARE SENT.  
:* THE TRANSMITTER IS THEN DISABLED, USING OC, AND OACT IS MONITORED FOR THE  
:* CLEARED STATE.  
:*****  
:BGNTST
```

T24::

| | | | | | | |
|------|--------|--------|--------|-----------|------------|---|
| 9176 | 027412 | 000000 | | OCO | | |
| 9177 | 027414 | 100010 | | CHPCHK!8. | | |
| 9178 | 027416 | 004737 | 006074 | JSR | PC, TXCHAR | ;LOAD 2ND 000 CHAR, TX 2ND FLAG |
| 9179 | 027422 | 000000 | | 000 | | |
| 9180 | 027424 | 000010 | | 8. | | |
| 9181 | 027426 | 004737 | 006074 | JSR | PC, TXCHAR | ;LOAD EOM CHAR, TX FIRST 000 CHAR |
| 9182 | 027432 | 001000 | | TXEOM | | |
| 9183 | 027434 | 000010 | | 8. | | |
| 9184 | 027436 | 004737 | 006074 | JSR | PC, TXCHAR | ;LOAD EOM, TX 2ND 000 CHAR AND CRC-CCITT-1 CHAR |
| 9185 | 027442 | 001000 | | TXEOM | | |
| 9186 | 027444 | 000030 | | 24. | | |
| 9187 | 027446 | 004737 | 006074 | JSR | PC, TXCHAR | ;LOAD EOM, TX FIRST TERMINATING FLAG |
| 9188 | 027452 | 001000 | | TXEOM | | |
| 9189 | 027454 | 000010 | | 8. | | |
| 9190 | 027456 | 004737 | 006074 | JSR | PC, TXCHAR | ;LOAD EOM, TX 2ND TERMINATING FLAG |
| 9191 | 027462 | 001000 | | TXEOM | | |
| 9192 | 027464 | 000010 | | 8. | | |
| 9193 | 027466 | 004737 | 006246 | JSR | PC, ENDTRN | ;SET OC, MONITOR OCOR |
| 9194 | 027472 | | | | | |
| 9195 | 027472 | 004737 | 003576 | JSR | PC, MSTCLR | ;ISSUE MASTER CLEAR TO CLEAN UP |
| 9196 | 027476 | | | | | |
| (3) | 027476 | | | | | |
| (3) | 027476 | 104401 | | | | L10054: TRAP C\$ETS1 |

A5:

ENDTST

L10054: TRAP C\$ETS1

9197
9198
9199
9200
9201
9202
9203
9204
9205
9206
9207
9208
9209
9210
9211
9212
9213
9214
9215
9216
9217

```

:*****
.SBTTL      TEST 25 - TX MSG TIMING TEST, CHAR MODE, WITH NO CRC
:*
:* IN THIS TEST, AN ENTIRE MESSAGE IS TRANSMITTED (USING STEPLU AND LULOOP)
:* AND THE PROGRAM MONITORS THE OCCURRENCE OF USYRT TX BUFFER EMPTY FLAGS
:* (BY SCANNING ORDY AND OCOR) AND OACT, AT EACH STEP. THE TEST IS DONE IN
:* CHARACTER ORIENTED PROTOCOL MODE, USING 8-BIT CHARS AND NO ERROR CHECKING.
:* THE FOLLOWING STEPS ARE DONE:
:* A MASTER CLEAR IS DONE, AND THE LINE UNIT IS PLACED IN CHAR MODE.
:* SOM IS SET TWICE TO SEND 2 SYNCH CHARS. THEN, 2 000 CHARS ARE SENT, AND
:* THEN 2 TERMINATING SYNCHS ARE SENT.
:* THE TEST IS PERFORMED WITH TXEN (REG14, BIT6) SET, AND THE PROGRAM CHECKS
:* THAT THIS HOLDS RTS HIGH PAST THE END OF THE MESSAGE.
:* THE TRANSMITTER IS THEN DISABLED, USING OC, AND OACT IS MONITORED FOR THE
:* CLEARED STATE.
:*****

```

BGNTST

| | | | | | | |
|------|--------|--------|--------|--------|--|-----------------------------------|
| 9218 | 027500 | | | | | |
| (3) | 027500 | | | | | |
| 9219 | 027500 | 012737 | 027666 | 002362 | | T25:: |
| 9220 | 027506 | 004737 | 005512 | | | ;SET TEST EXIT ADDRESS FOR ERRORS |
| 9221 | 027512 | 000226 | | | | ;DO MASTER CLR, LOAD 2 SOM'2 |
| 9222 | 027514 | 000311 | | | | |
| 9223 | 027516 | 012737 | 000014 | 002400 | | |
| 9224 | 027524 | 012737 | 000100 | 002366 | | |
| 9225 | 027532 | 004737 | 003722 | | | |
| 9226 | 027536 | 004737 | 006074 | | | |
| 9227 | 027542 | 000000 | | | | |
| 9228 | 027544 | 100010 | | | | |

```

MOV      #A6,RETADR      ;SET TEST EXIT ADDRESS FOR ERRORS
JSR      PC,INITRN      ;DO MASTER CLR, LOAD 2 SOM'2
SYNCH
CRC2!CRC1!STRIP!DDCMP
MOV      #14,REGNUM      ;SET REG NO. = 14
MOV      #TXEN,WRIBYT    ;SET TXEN BIT
JSR      PC,WRITLU      ;LOAD TXEN BIT IN REG 14
JSR      PC, TXCHAR      ;LOAD A 000 CHAR, TX FIRST SYNCH (226)
000
CHPCHK!8.

```

```

9229 027546 004737 006074 JSR PC,TXCHAR ;LOAD 2ND 000 CHAR, TX 2ND SYNCH
9230 027552 000000 000 8.
9231 027554 000010 JSR PC,TXCHAR ;LOAD EOM CHAR, TX FIRST 000 CHAR
9232 027556 004737 006074 TXEOM 8.
9233 027562 001000 JSR PC,TXCHAR ;LOAD EOM CHAR, TX 2ND 000 CHAR
9234 027564 000010 TXEOM 8.
9235 027566 004737 006074 JSR PC,TXCHAR ;LOAD EOM, TX FIRST TERMINATING SYNCH
9236 027572 001000 TXEOM 8.
9237 027574 000010 JSR PC,TXCHAR ;LOAD EOM, TX 2ND TERMINATING SYNCH
9238 027576 004737 006074 TXEOM 8.
9239 027602 001000 JSR PC,STPLU ;CLK PAST END OF MSG
9240 027604 000010 24.
9241 027606 004737 006074 MOV #13,REGNUM ;SET REG NO. = 13
9242 027612 001000 JSR PC,READLU ;READ REG 13
9243 027614 000010 BIT #RTS,REDBYT ;CHK FOR RTS STILL SET
9244 027616 004737 005226 BNE 3$ ;BR IF RTS SET
9245 027622 000030 JSR PC,GETALL ;GET REGS FOR PRINTOUT
9246 027624 012737 000013 002400 ;REPORT RTS NOT SET
9247 027632 004737 003644 ERRDF 60,EM60,ERR7
9248 027636 032737 000040 002364 TRAP C$ERDF
9249 027644 001006 .WORD 60
9250 027646 004737 004500 .WORD EM60
9251 .WORD ERR7
9252 027652 (4) 027652 104455
9253 027662 004737 006246 3$: JSR PC,ENDTRN ;SET OC, MONITOR OCOR
9254 027666 A6: JSR PC,MSTCLR ;ISSUE MASTER CLEAR TO CLEAN UP
9255 027666 004737 003576
9256 027672 ENDTST
9257 (3) 027672 104401 L10055: TRAP C$ETST
9258
9259
9260
9261
9262 ;*****
9263 .SBTTL TEST 26 - TX UNDERRUN SET AND CLEAR TEST - CHAR MODE
9264 ;*
9265 ;* IN THIS TEST, A TX UNDERRUN ERROR IS FORCED IN EACH OF 2 SITUATIONS,
9266 ;* AND THEN CLEARED DIFFERENTLY IN EACH.
9267 ;* IN THE FIRST, A MESSAGE IS INITIATED, A 000 CHAR IS SENT, AND THE TX
9268 ;* BUFFER IS NOT SERVICED IN RESPONSE TO THE USYRT TX BUFFER EMPTY FLAG,
9269 ;* WHICH CAUSES UNRR TO SET IN REG 11. THEN, SOM IS SET TO CLEAR THE ERROR,
9270 ;* AND THIS IS VERIFIED.
9271 ;* IN THE SECOND SITUATION, A MSG IS INITIATED, A 000 CHAR IS SENT, AND THE TX
9272 ;* BUFFER IS NOT SERVICED IN RESPONSE TO THE USYRT TX BUFFER EMPTY FLAG, WHICH
9273 ;* AGAIN CAUSES UNRR TO SET. THEN, A MASTER CLEAR IS DONE, AND THE UNRR BIT
9274 ;* IS CHECKED TO BE CLEARED.
9275 ;*****
9276 027674 BGNST
9277 (3) 027674 T26:

```

```
9278 ; CAUSE TX UNDERRUN, CHK UNRR = 1; SET SOM, CHK UNRR = 0
9279 -----
9280 027674 012737 030232 002362 MOV #A7,RETADR ;SFT TEST EXIT ADDRESS FOR ERRORS
9281 027702 004737 005512 JSR PC,INITRN ;DO MASTER CLEAR, LOAD 2 SOM'S
9282 027706 000226 SYNCH
9283 027710 000011 STRIP!DDCMP
9284 027712 004737 006074 JSR PC,TXCHAR ;LOAD A 000 CHAR, TX FIRST SYNCH
9285 027716 000000 000
9286 027720 100010 CHPCHK!8.
9287 027722 004737 005226 JSR PC,STPLU ;CLOCK THE TRANSMITTER UNTIL 7 BITS OF
9288 027726 000016 14. ; THE 000 CHAR HAVE BEEN TRANSMITTED
9289 027730 012737 000011 002400 MOV #11,REGNUM ;SET LU REG NO. = 11
9290 027736 004737 003644 JSR PC,READLU ;READ REG 11
9291 027742 132737 000001 002364 BITB #UNRR,REDBYT ;CHK FOR UNRR = 0
9292 027750 001410 BEQ 6$ ;BR IF UNRR = 0
9293 027752 004737 004500 JSR PC,GETALL ;GET REGS FOR PRINTOUT
9294 ;REPORT UNRR NOT CLEARED
9295 027756 104455 ERRDF 16,EM16,ERR7
(4) 027756 104455 TRAP C$ERDF
(5) 027760 000020 .WORD 16
(5) 027762 012576 .WORD EM16
(5) 027764 017444 .WORD ERR7
9296 027766 000137 030232 JMP A7 ;SKIP TO END OF TEST
9297 027772 004737 005226 6$: JSR PC,STPLU ;CLOCK LAST BIT OF 000 CHAR
9298 027776 000003 3
9299 030000 004737 003644 JSR PC,READLU ;READ REG 11
9300 030004 132737 000001 002364 BITB #UNRR,REDBYT ;CHK FOR UNRR = 1
9301 030012 001010 BNE 9$ ;BR IF UNRR = 1
9302 030014 004737 004500 JSR PC,GETALL ;GET REGS FOR PRINTOUT
9303 ;REPORT UNRR NOT SET
9304 030020 104455 ERRDF 14,EM14,ERR7
(4) 030020 104455 TRAP C$ERDF
(5) 030022 000016 .WORD 14
(5) 030024 012532 .WORD EM14
(5) 030026 017444 .WORD ERR7
9305 030030 000137 030232 JMP A7 ;SKIP TO END OF TEST
9306 030034 012737 000400 002422 9$: MOV #TXSOM,TXWORD ;SET SOM CHAR TO BE WRITTEN
9307 030042 004737 005146 JSR PC,LDTXSI ;LOAD SOM CHAR INTO TX SILO
9308 030046 004737 005120 JSR PC,WAIT50 ;WAIT FOR SILO DATA TO RIPPLE
9309 030052 004737 005226 JSR PC,STPLU ;CLOCK LU FOR 2 CYCLES
9310 030056 000002 2
9311 030060 004737 003644 JSR PC,READLU ;READ REG 11
9312 030064 132737 000001 002364 BITB #UNRR,REDBYT ;CHK FOR UNRR = 0
9313 030072 001410 BEQ 12$ ;BR IF UNRR = 0
9314 030074 004737 004500 JSR PC,GETALL ;GET REGS FOR PRINTOUT
9315 ;REPORT UNRR NOT CLEARED BY SOM
9316 030100 104455 ERRDF 13,EM13,ERR7
(4) 030100 104455 TRAP C$ERDF
(5) 030102 000015 .WORD 13
(5) 030104 012502 .WORD EM13
(5) 030106 017444 .WORD ERR7
9317 030110 000137 030232 JMP A7 ;SKIP TO END OF TEST
9318 030114 12$:
9319 -----
9320 ; CAUSE TX UNDERRUN, CHK UNRR = 1; DO MASTER CLR, CHK UNRR = 0
9321 -----
```

```

9322 030114 004737 005512 JSR PC,INITRN ;DO MASTER CLEAR, LOAD 2 SOM'S
9323 030120 000226 SYNCH
9324 030122 000051 IDLE!STRIP.DDCMP
9325 030124 004737 006074 JSR PC, TXCHAR ;LOAD A 000 CHAR, TX FIRST SYNCH
9326 030130 000000 000
9327 030132 100010 CHPCHK!8.
9328 030134 004737 005226 JSR PC,STPLU ;STEP THE LU UNTIL 000 HAS BEN TRANSMITTED
9329 030140 000021 17.
9330 030142 004737 003644 JSR PC,READLU ;READ REG 11
9331 030146 132737 000001 002364 BITB #UNRR,REDBYT ;CHK FOR UNR = 1
9332 030154 001010 BNE 16$ ;BR IF UNRR = 1
9333 030156 004737 004500 JSR PC,GETALL ;GET REGS FOR PRINTOUT
9334 ;REPORT UNRR NOT SET
9335 030162 ERRDF 14,EM14,ERR7
(4) 030162 104455 TRAP C$ERDF
(5) 030164 000016 .WORD 14
(5) 030166 012532 .WORD EM14
(5) 030170 017444 .WORD ERR7
9336 030172 000137 030232 JMP A7 ;SKIP TO END OF TEST
9337 030176 004737 003576 16$: JSR PC,MSTCLR ;ISSUE MASTER CLEAR
9338 030202 004737 003644 JSR PC,READLU ;READ REG 11
9339 030206 132737 000001 002364 BITB #UNRR,REDBYT ;CHK FOR UNRR = 0
9340 030214 001406 BEQ A7 ;BR IF UNRR = 0
9341 030216 004737 004500 JSR PC,GETALL ;GET REGS FOR PRINTOUT
9342 ;REPORT UNRR NOT CLEARED BY OC
9343 030222 ERRDF 15,EM15,ERR7
(4) 030222 104455 TRAP C$ERDF
(5) 030224 000017 .WORD 15
(5) 030226 012547 .WORD EM15
(5) 030230 017444 .WORD ERR7
9344 030232 004737 003576 A7: JSR PC,MSTCLR ;ISSUE CLEAN-UP MASTER CLEAR
9345 030236 ENDTST
(3) 030236 L10056: TRAP C$ETST
(3) 030236 104401

```

```

9346
9347
9348
9349
9350
9351 ;*****
9352 .SBTTL TEST 27 - TRANSMIT CHAR LENGTH TIMING TEST - CHAR MODE, CRC
9353 ;*
9354 ;* THE LINE UNIT IS PLACED IN CHAR MODE (DDCMP) AND A MESSAGE IS INITIATED
9355 ;* WITH AN 8-BIT SYNCH AND A 5-BIT SYNCH CHAR. NEXT, A 000 CHAR IS SENT WITH
9356 ;* EACH OF THE FOLLOWING TX CHAR LENGTHS : 5 BITS, 6 BITS, 7 BITS, 8 BITS.
9357 ;* (FOR EXAMPLE, A 5-BIT CHAR REQUIRES 5 CLOCK CYCLES TO BE TRANSMITTED). TWO
9358 ;* TERMINATING SYNCHS ARE SENT AFTER THE DATA.
9359 ;*****
9360 BGNTST
(3) 030240 T27::
9361 030240 012737 030446 002362 MOV #A8,RETADR ;SET TEST EXIT ADDRESS FOR ERRORS
9362 030246 004737 005512 JSR PC,INITRN ;DO MASTER CLR, LOAD 2 SOM'S
9363 030252 000000 000
9364 030254 000041 IDLE!DDCMP
9365 030256 012737 000006 002402 MOV #6,AXNUM ;SET BYTE NO. - 6 FOR AX3
9366 030264 012737 000000 002374 MOV #000,WAX15 ;SET DATA FOR AX3-15 = 0

```


CZDMRA M8203 STATIC DIAG #1
CZDMRA.P11 18-JUL-79 09:44

MACY11 30A(1052) 18-JUL-79 09:53 PAGE 7-110

TEST 27 - TRANSMIT CHAR LENGTH TIMING TEST - CHAR MODE, CRC

SEQ 0150

```

9367 030272 012737 000240 002376      MOV      #TXLEN2,TXLENO,WAX16 ;SET TX LENGTH = 5 FOR AX3-16
9368 030300 004737 004264      JSR      PC,WRITAX           ;LOAD AX3-15,AX3-16
9369 030304 004737 006074      JSR      PC,TXCHAR           ;LOAD 5-BIT 000 CHAR, TX 8-BIT SYNCH
9370 030310 000000 000000      000
9371 030312 100010 000000      CHPCHK.8.
9372 030314 004737 006074      JSR      PC,TXCHAR           ;LOAD 6-BIT 000 CHAR, TX 5-BIT SYNCH
9373 030320 000000 000000      000
9374 030322 000005 000000      5
9375 030324 012737 000300 002376      MOV      #TXLEN2!TXLEN1,WAX16
9376 030332 004737 004264      JSR      PC,WRITAX           ;SET TX CHAR LENGTH = 6
9377 030336 004737 006074      JSR      PC,TXCHAR           ;LOAD 7-BIT 000 CHAR, TX 5-BIT 000 CHAR
9378 030342 000000 000000      000
9379 030344 000005 000000      5
9380 030346 012737 000340 002376      MOV      #TXLEN2!TXLEN1!TXLENO,WAX16
9381 030354 004737 004264      JSR      PC,WRITAX           ;SET TX CHAR LENGTH = 7
9382 030360 004737 006074      JSR      PC,TXCHAR           ;LOAD 8-BIT 000 CHAR, TX 6-BIT 000 CHAR
9383 030364 000000 000000      000
9384 030366 000006 000000      6
9385 030370 012737 000000 002376      MOV      #000,WAX16
9386 030376 004737 004264      JSR      PC,WRITAX           ;SET TX CHAR LENGTH = 8
9387 030402 004737 006074      JSR      PC,TXCHAR           ;LOAD EOM, TX 7-BIT 000 CHAR
9388 030406 001000 000000      TXEOM
9389 030410 000007 000000      7
9390 030412 004737 006074      JSR      PC,TXCHAR           ;LOAD EOM, TX 8-BIT 000 CHAR
9391 030416 000000 000000      TXEOM
9392 030420 000010 000000      8.
9393 030422 004737 006074      JSR      PC,TXCHAR           ;LOAD EOM, TX CRC-16 CHAR
9394 030426 001000 000000      TXEOM
9395 030430 000020 000000      16.
9396 030432 004737 006074      JSR      PC,TXCHAR           ;LOAD EOM, TX FIRST TERMINATING SYNCH
9397 030436 001000 000000      TXEOM
9398 030440 000010 000000      8.
9399 030442 004737 006246      JSR      PC,ENDTRN          ;CLEAR TRANSMITTER

```

A8:
ENDTST

L10057:
TRAP CSETST

9400 030446
9401 030446
(3) 030446
(3) 030446 104401

9402
9403
9404
9405
9406
9407
9408
9409
9410
9411
9412
9413
9414
9415
9416
9417 030450
(3) 030450
9418 030450 012737 030776 002362
9419 030456 004737 005512

```

*****
.SBTTL      TEST 28 - TRANSMIT CHAR LENGTH TIMING TEST - BIT MODE, CRC
.*
.* THE LINE UNIT IS PLACED IN BIT MODE AND A MESSAGE IS INITIATED
.* WITH 2 FLAG CHARS. NEXT, 2 8-BIT 000 CHAR ARE SENT, FOLLOWED BY 000 CHARS
.* WITH EACH OF THE FOLLOWING TRANSMITTER CHAR LENGTHS:
.* 1 BIT, 2 BITS, 3 BITS, 4 BITS, 5 BITS, 6 BITS, 7 BITS, AND 8 BITS.
.* (FOR EXAMPLE, A 5-BIT CHAR REQUIRES 5 CLOCK CYCLES TO BE TRANSMITTED).
.* TWO TERMINATING FLAGS ARE SENT AFTER THE DATA.
*****
BGNST

```

```

T28::
MOV      #A9,RETADR          ;SET TEST EXIT ADDRESS FOR ERRORS
JSR      PC,INITRN          ;DO MASTER LR, LOAD 2 .COM'S

```

```

9420 030462 000000 000
9421 030464 000000 000
9422 030466 004737 006074 JSR PC,TXCHAR ;LOAD FIRST 8-BIT 000 CHAR, TX FIRST FLAG
9423 030472 000000 000
9424 030474 100010 CHPCHK!8.
9425 030476 004737 006074 JSR PC,TXCHAR ;LOAD 2ND 8-BIT 000 CHAR, TX 2ND FLAG
9426 030502 000000 000
9427 030504 000010 8.
9428 030506 004737 006074 JSR PC,TXCHAR ;LOAD 1-BIT 000 CHAR, TX FIRST 8-BIT 000 CHAR
9429 030512 000000 000
9430 030514 000010 8.
9431 030516 012737 000000 002402 MOV #6,AXNUM ;SET BYTE NO. = 6 FOR AX3
9432 030524 012737 000000 002374 MOV #000,WAX15 ;SET DATA FOR AX3-15 = 0
9433 030532 012737 000040 002376 MOV #TXLENO,WAX16 ;SET TX CHAR LENGTH = 1 FOR AX3-16
9434 030540 004737 004264 JSR PC,WRITAX ;LOAD AX3-15,AX3-16
9435 030544 004737 006074 JSR PC,TXCHAR ;LOAD 2-BIT 000 CHAR, TX 2ND 8-BIT 000 CHAR
9436 030550 000000 000
9437 030552 000010 8.
9438 030554 012737 000100 002376 MOV #TXLEN1,WAX16
9439 030562 004737 004264 JSR PC,WRITAX ;SET TX CHAR LENGTH = 2
9440 030566 004737 006074 JSR PC,TXCHAR ;LOAD 3-BIT 000 CHAR, TX 1-BIT 000 CHAR
9441 030572 000000 000
9442 030574 000001 1
9443 030576 012737 000140 002376 MOV #TXLEN1!TXLENO,WAX16
9444 030604 004737 004264 JSR PC,WRITAX ;SET TX CHAR LENGTH - 3
9445 030610 004737 006074 JSR PC,TXCHAR ;LOAD 4-BIT 000 CHAR, TX 2-BIT 000 CHAR
9446 030614 000000 000
9447 030616 000002 2
9448 030620 012737 000200 002376 MOV #TXLEN2,WAX16
9449 030626 004737 004264 JSR PC,WRITAX ;SET TX CHAR LENGTH = 4
9450 030632 004737 006074 JSR PC,TXCHAR ;LOAD 5-BIT 000 CHAR, TX 3-BIT 000 CHAR
9451 030636 000000 000
9452 030640 000003 3
9453 030642 012737 000240 002376 MOV #TXLEN2!TXLENO,WAX16
9454 030650 004737 004264 JSR PC,WRITAX ;SET TX CHAR LENGTH - 5
9455 030654 004737 006074 JSR PC,TXCHAR ;LOAD 6-BIT 000 CHAR, TX 4-BIT 000 CHAR
9456 030660 000000 000
9457 030662 000004 4
9458 030664 012737 000300 002376 MOV #TXLEN2!TXLEN1,WAX16
9459 030672 004737 004264 JSR PC,WRITAX ;SET TX CHAR LENGTH = 6
9460 030676 004737 006074 JSR PC,TXCHAR ;LOAD 7-BIT 000 CHAR, TX 5-BIT 000 CHAR
9461 030702 000000 000
9462 030704 000005 5
9463 030706 012737 000340 002376 MOV #TXLEN2!TXLEN1!TXLENO,WAX16
9464 030714 004737 004264 JSR PC,WRITAX ;SET TX CHAR LENGTH 7
9465 030720 004737 006074 JSR PC,TXCHAR ;LOAD 8-BIT 000 CHAR, TX 6-BIT 000 CHAR
9466 030724 000000 000
9467 030726 000006 6
9468 030730 012737 000000 002376 MOV #000,WAX16
9469 030736 004737 004264 JSR PC,WRITAX ;SET TX CHAR LENGTH = 8
9470 030742 004737 006074 JSR PC,TXCHAR ;LOAD EOM, TX 7-BIT 000 CHAR
9471 030746 001000 TXEOM
9472 030750 000007 7
9473 030752 004737 006074 JSR PC,TXCHAR ;LOAD EOM, TX 8-BIT 000 CHAR, CRC-CCITT-1 CHAR
9474 030756 001000 TXEOM
9475 030760 000031 25.

```

CZDMRA M8203 STATIC DIAG #1
CZDMRA.P11 18-JUL-79 09:44

MACY11 30A(1052) 18-JUL-79 09:53 PAGE 7-112
TEST 28 - TRANSMIT CHAR LENGTH TIMING TEST - BIT MODE, CRC

SEQ 0152

9476 030762 004737 006074
9477 030766 001000
9478 030770 000010
9479 030772 004737 006246
9480 030776
9481 030776
(3) 030776
(3) 030776 104401

JSR PC, TXCHAR ;LOAD EOM, TX FIRST TERMINATING FLAG
TXEOM
8.
JSR PC, ENDTRN ;CLEAR TRANSMITTER
A9:
ENDTST
L10060: TRAP (SETST

9482
9483
9484
9485
9486
9487
9488
9489
9490
9491
9492
9493
9494
9495

:SBTTL TEST 29 - TXDATA BIT TEST - CHAR MODE, NO CRC
:
:* THE LINE UNIT IS INITIALIZED AND A MSG IS INITIATED (USING STEPLU) WITH CRC-
:* 16 SELECTED IN CHAR MODE. TWO SYNCHS, 000,125,252,377,000, AND 2 TERMINATING
:* SYNCHS ARE THEN SENT. THE PROGRAM CHECKS EACH BIT OF THE TRANSMITTED
:* DATA CHARS, BY MONITORING TXDATA (REG 17) AS THE DATA IS CLOCKED OUT OF
:* THE USYRT TRANSMITTER.
:*****
BGNTST

9496 031000
(3) 031000
9497 031000 004737 005512
9498 031004 000226
9499 031006 000011
9500 031010 012701 003224
9501 031014 010103
9502 031016 012137 002422
9503 031022 004737 005146
9504 031026 020127 003242
9505 031032 103771
9506 031034 004737 005120
9507 031040 004737 005226
9508 031044 100020
9509 031046 011337 031056
9510 031052 004737 011150
9511 031056 000000
9512 031060 062703 000002
9513 031064 020327 003236
9514 031070 103766
9515 031072 004737 003576
9516 031076
(3) 031076
(3) 031076 104401

T29::
JSR PC, INITRN ;DO MASTER CLR, LOAD 2 SOM'S
SYNCH
STRIP!DDCMP
MOV #MSG1+4, R1 ;GET POINTER TO DATA
MOV R1, R3
3\$: MOV (R1)+, TXWORD
JSR PC, LDTXSI ;LOAD A DATA CHAR INTO TX SILO
CMP R1, #MSG1+18. ;SEE IF ALL CHARS LOADED YET
BLO 3\$;BR IF NOT YET
JSR PC, WAIT50 ;WAIT FOR SILO TO RIPPLE
JSR PC, STPLU ;CLOCK LU UNTIL SYNCHS ARE TX'D
CHPCHK!16.
6\$: MOV (R3), 8\$;GET EXPECTED DATA CHAR
JSR PC, CKTBIT ;CHECK TXDATA FOR CHAR BITS
8\$: .WORD 0 ;EXPECTED CHAR GOES HERE
ADD #2, R3 ;INCR PATTERN POINTER
CMP R3, #MSG1+14. ;SEE IF ALL CHARS CHECKED YET
BLO 6\$;BR IF NOT YET
16\$: JSR PC, MSTCLR ;ISSUE MASTER CLEAR TO CLEAN UP
ENDTST
L10061: TRAP (SETST

9517
9518
9519
9520
9521
9522
9523
9524
9525
9526

:SBTTL TEST 30 - USYRT RECEIVER MSG TEST - CHAR MODE, CRC
:
:* THE LINE UNIT IS INITIALIZED AND A MESSAGE IS INITIATED (USING STEPLU) WITH
:* LULP (REG 12) SET TO LOOP THE DATA INTERNALLY IN THE JSYRT, AND WITH CRC-16

9527 ;* SELECTED. TWO SYNCHS, 000,125,252,377,000, AND FOUR TERMINATING SYNCHS ARE
9528 ;* SENT. THE PROGRAM MONITORS IACT, AND THE RCV'D CHARS AND CRC BYTES ARE READ
9529 ;* FROM AX0-15 AND COMPARED TO EXPECTED VALUES. THE PROGRAM THEN CHECKS FOR IACT
9530 ;* = 0 AFTER RECEIVER SHUTDOWN.
9531 ;*****

9532 031100 BGNTST
(3) 031100

9533 031100 012737 031442 002362 MOV #24\$,RETADR ;SET TEST EXIT ADDRESS FOR ERRORS
9534 031106 004737 003576 JSR PC,MSTCLR ;ISSUE MASTER CLEAR
9535 031112 004737 010612 JSR PC,SETUP ;PROGRAM THE USYRT

9536 031116 000226 SYNCH
9537 031120 000013 STRIP'IERR.DDCMP
9538 031122 000000 000
9539 031124 000000 000

9540 031126 012737 000012 002400 MOV #12,REGNUM ;SET LU REG NO. = 12
9541 031134 005037 002402 CLR AXNUM ;SET AX BYTE NO. = 0 FOR AX0

9542 031140 112737 000040 002366 MOVB #LULP,WRIBYT
9543 031146 004737 003722 JSR PC,WRITLU ;SET LULP IN REG 12

9544 031152 012701 003220 MOV #MSG1,R1 ;GET POINTER TO MSG DATA TABLE
9545 031156 012137 002422 3\$: MOV (R1)+,TXWORD ;GET CHAR TO BE LOADED

9546 031162 004737 005146 JSR PC,LDTXSI ;LOAD CHAR INTO TX SILO
9547 031166 020127 003246 CMP R1,#MSG1+22. ;SEE IF ALL MSG CHARS LOADED YET

9548 031172 103771 BLO 3\$;BR IF NOT YET
9549 031174 004737 005120 JSR PC,WAIT50 ;ALLOW DATA TO RIPPLE IN SILO

9550 031200 004737 005226 JSR PC,STPLU ;CLOCK LU FOR 40 CYCLES (UNTIL FIRST
9551 031204 000050 40. ; DATA CHAR IS ABOUT TO BE RECEIVED)

9552 031206 004737 006666 JSR PC,IACTIV ;CHK IACT = 0
9553 031212 000000 0

9554 031214 004737 005226 JSR PC,STPLU ;CLOCK LU UNTIL 1ST DATA CHAR IS RCV'D
9555 031220 000006 6

9556 031222 012701 003224 10\$: MOV #MSG1+4,R1
9557 031226 004737 006666 JSR PC,IACTIV ;CHK IACT = 1

9558 031232 000001 1
9559 031234 004737 004076 JSR PC,READAX ;READ AX0

9560 031240 023721 002370 CMP RAX15,(R1)+ ;COMPARE RCV'D CHAR TO EXPECTED
9561 031244 001415 BEQ 12\$;BR IF RCV'D DATA OK

9562 031246 016137 177776 002404 MOV -2(R1),GOODAT ;GET EXPECTED DATA
9563 031254 013737 002370 002406 MOV RAX15,BADDAT ;GET ACTUAL DATA

9564 031262 004737 004500 JSR PC,GETALL ;GET REGS FOR PRINTOUT
9565 ;REPORT INCORRECT DATA CHAR RCV'D
9566 031266 ERRDF 26,EM26,ERR3

(4) 031266 104455 TRAP C\$ERDF
(5) 031270 000032 .WORD 26
(5) 031272 013043 .WORD EM26
(5) 031274 015114 .WORD ERR3

9567 031276 000461 BR 24\$
9568 031300 004737 005226 12\$: JSR PC,STPLU ;CLOCK LU 8 CYCLES

9569 031304 000010 8.
9570 031306 020127 003236 CMP R1,#MSG1+14. ;SEE IF CHECKING HI CRC BYTE YET

9571 031312 103745 BLO 10\$;BR IF NOT YET
9572 031314 004737 006666 JSR PC,IACTIV ;CHK IACT = 1

9573 031320 000001 1
9574 031322 004737 004076 JSR PC,READAX ;READ AX0

9575 031326 123727 002370 000160 CMPB RAX15,#160 ;CMP RCV'D CHAR TO EXPECTED HI CRC BYTE
9576 031334 001415 BEQ 16\$;BR IF HI CRC BYTE RCV'D OK

9577 031336 012737 000160 002404 MOV #160,GOODAT ;GET EXPECTED DATA

```
9578 031344 013737 002370 002406 14$: MOV RAX15,BADAT ;SET ACTUAL DATA
9579 031352 004737 004500 JSR PC,GETALL ;GET REGS FOR PRINTOUT
9580 ;REPORT INCORRECT CRC BYTE RCV'D
9581 031356 ERRDF 27,EM27,ERR3
(4) 031356 104455 TRAP C$ERDF
(5) 031360 000033 .WORD 27
(5) 031362 013075 .WORD EM27
(5) 031364 015114 .WORD ERR3
9582 031366 000425 BR 24$
9583 031370 004737 005226 16$: JSR PC,STPLU ;CLOCK LU FOR 8 CYCLES
9584 031374 000010 8.
9585 031376 004737 006666 JSR PC,IACTIV ;CHK IACT = 1
9586 031402 000001 1
9587 031404 012737 000034 002404 MOV #034,GOODAT ;GET EXPECTED LO CRC BYTE
9588 031412 004737 004076 JSR PC,READAX ;READ AX0
9589 031416 123727 002370 000034 CMPB RAX15,#034 ;CMP RCV'D CHAR TO EXPECTED LO CRC BYTE
9590 031424 001347 BNE 14$ ;BR IF LO CRC INCORRECT
9591 031426 004737 005226 JSR PC,STPLU ;CLOCK LU 8 CYCLES
9592 031432 000010 8.
9593 031434 004737 006666 JSR PC,IACTIV ;CHK IACT = 0
9594 031440 000000 0
9595 031442 004737 003576 24$: JSR PC,MSTCLR ;ISSUE CLEAN-UP MASTER CLEAR
9596 031446 ENDTST
(3) 031446 L10062: TRAP C$ETST
(3) 031446 104401
```

9597
9598
9599
9600
9601
9602
9603
9604
9605
9606
9607
9608
9609
9610
9611

```
*****
;SBTTL TEST 31 - USYRT RECEIVER MSG TEST - BIT MODE, CRC
;
;* THE LINE UNIT IS INITIALIZED AND A MESSAGE IS INITIATED (USING STEPLU) WITH
;* LULP (REG 12) SET TO LOOP THE DATA INTERNALLY IN THE USYRT, AND WITH CRC-
;* CCITT-1. TWO FLAGS, 000,125,252,377,000, AND TWO TERMINATING FLAGS ARE THEN
;* SENT. THE PROGRAM MONITORS IACT, RSOM, AND THE RCV'D CHARS ARE READ
;* FROM AX0-15 AND COMPARED TO EXPECTED VALUES. THE PROGRAM THEN CHECKS FOR
;* IACT = 0, SETS IC TO CLEAR THE RECEIVER, AND CHECKS FOR IACT STILL = 0.
*****
```

```
9612 031450 BGNTST
(3) 031450 T31::
9613 031450 012737 032004 002362 MOV #24$,RETADR ;SET TEST EXIT ADDRESS FOR ERRORS
9614 031456 004737 003576 JSR PC,MSTCLR ;ISSUE MASTER CLEAR
9615 031462 004737 010612 JSR PC,SETUP ;PROGRAM THE USYRT
9616 031466 000000 000
9617 031470 000002 IERR
9618 031472 000000 000
9619 031474 000000 000
9620 031476 012737 000012 002400 MOV #12,REGNUM ;SET LU REG NO. = 12
9621 031504 005037 002402 CLR AXNUM ;SET AX BYTE NO. = 0 FOR AX0
9622 031510 112737 000040 002366 MOVB #LULP,WRIBYT
9623 031516 004737 003722 JSR PC,WRITLU ;SET LULP IN REG 12
9624 031522 012701 003220 MOV #MSG1,R1 ;GET POINTER TO MSG DATA TABLE
9625 031526 012137 002422 3$: MOV (R1)+,TXWORD ;GET CHAR TO BE LOADED
9626 031532 004737 005146 JSR PC,LDTXSI ;LOAD CHAR INTO TX SILO
```

| | | | | | | | |
|------|--------|--------|--------|--------|-----------------------------------|---------------|--|
| 9627 | 031536 | 020127 | 003242 | | CMP | R1,#MSG1+18. | ;SEE IF ALL MSG CHARS LOADED YET |
| 9628 | 031542 | 103771 | | | BLO | 3\$ | ;BR IF NOT YET |
| 9629 | 031544 | 004737 | 005120 | | JSR | PC,WAIT50 | ;ALLOW DATA TO RIPPLE IN SILO |
| 9630 | 031550 | 004737 | 005226 | | JSR | PC,STPLU | ;CLOCK LU FOR 50 CYCLES (UNTIL FIRST |
| 9631 | 031554 | 000062 | | | 50. | | ; DATA CHAR IS ABOUT TO BE RECEIVED) |
| 9632 | 031556 | 004737 | 006666 | | JSR | PC,IACTIV | ;CHK IACT = 0 |
| 9633 | 031562 | 000000 | | | 0 | | |
| 9634 | 031564 | 004737 | 007054 | | JSR | PC,RSEOM | ;CHK RSOM = 0, REOM = 0 |
| 9635 | 031570 | 000000 | | | 0 | | |
| 9636 | 031572 | 004737 | 005226 | | JSR | PC,STPLU | ;CLOCK LU UNTIL 1ST DATA CHAR IS RCV'D |
| 9637 | 031576 | 000006 | | | 6 | | |
| 9638 | 031600 | 012701 | 003224 | | MOV | #MSG1+4,R1 | |
| 9639 | 031604 | 020127 | 003224 | 5\$: | CMP | R1,#MSG1+4 | ;SEE IF 1ST CHAR RCV'D |
| 9640 | 031610 | 001007 | | | BNE | 6\$ | ;BR IF NO |
| 9641 | 031612 | 004737 | 006666 | | JSR | PC,IACTIV | ;CHK IACT = 1 |
| 9642 | 031616 | 000001 | | | 1 | | |
| 9643 | 031620 | 004737 | 007054 | | JSR | PC,RSEOM | ;CHK RSOM = 1, REOM = 0 |
| 9644 | 031624 | 000001 | | | 1 | | |
| 9645 | 031626 | 000420 | | | BR | 9\$ | |
| 9646 | 031630 | 020127 | 003234 | 6\$: | CMP | R1,#MSG1+12. | ;SEE IF LAST CHAR RCV'D |
| 9647 | 031634 | 001007 | | | BNE | 8\$ | ;BR IF NO |
| 9648 | 031636 | 004737 | 006666 | | JSR | PC,IACTIV | ;CHK FOR IACT = 0 |
| 9649 | 031642 | 000000 | | | 0 | | |
| 9650 | 031644 | 004737 | 007054 | | JSR | PC,RSEOM | ;CHK RSOM = 0, REOM = 0 |
| 9651 | 031650 | 000000 | | | 0 | | |
| 9652 | 031652 | 000406 | | | BR | 9\$ | |
| 9653 | 031654 | 004737 | 006666 | 8\$: | JSR | PC,IACTIV | ;CHK FOR IACT = 1 |
| 9654 | 031660 | 000001 | | | 1 | | |
| 9655 | 031662 | 004737 | 007054 | | JSR | PC,RSEOM | ;CHK RSOM = 0, REOM = 0 |
| 9656 | 031666 | 000000 | | | 0 | | |
| 9657 | 031670 | 004737 | 004076 | 9\$: | JSR | PC,READAX | ;READ AX0 |
| 9658 | 031674 | 023721 | 002370 | | CMP | RAX15,(R1)+ | ;COMPARE RCV'D CHAR TO EXPECTED |
| 9659 | 031700 | 001415 | | | BEQ | 12\$ | ;BR IF RCV'D DATA OK |
| 9660 | 031702 | 016137 | 177776 | 002404 | MOV | -2(R1),GOODAT | ;GET EXPECTED DATA |
| 9661 | 031710 | 013737 | 002370 | 002406 | MOV | RAX15,BADDAT | ;GET ACTUAL DATA |
| 9662 | 031716 | 004737 | 004500 | | JSR | PC,GETALL | ;GET REGS FOR PRINTOUT |
| 9663 | | | | | | | |
| 9664 | 031722 | | | | ;REPORT INCORRECT DATA CHAR RCV'D | | |
| (4) | 031722 | 104455 | | | ERRDF | 26,EM26,ERR3 | |
| (5) | 031724 | 000032 | | | | | TRAP C\$ERDF |
| (5) | 031726 | 013043 | | | | | .WORD 26 |
| (5) | 031730 | 015114 | | | | | .WORD EM26 |
| 9665 | 031732 | 000424 | | | | | .WORD ERR3 |
| 9666 | 031734 | 004737 | 005226 | 12\$: | BR | 24\$ | |
| 9667 | 031740 | 000010 | | | JSR | PC,STPLU | ;CLOCK LU 8 CYCLES |
| 9668 | 031742 | 020127 | 003236 | | 8. | | |
| 9669 | 031746 | 103716 | | | CMP | R1,#MSG1+14. | ;SEE IF ALL DATA CHARS CHECKED YET |
| 9670 | 031750 | 004737 | 006666 | | BLO | 5\$ | ;BR IF NOT YET |
| 9671 | 031754 | 000000 | | | JSR | PC,IACTIV | ;CHK IACT = 0 |
| 9672 | 031756 | 004737 | 007054 | | 0 | | |
| 9673 | 031762 | 000000 | | | JSR | PC,RSEOM | ;CHK RSOM = 0, REOM = 0 |
| 9674 | 031764 | 012737 | 000200 | 002366 | 0 | | |
| 9675 | 031772 | 004737 | 003722 | | MOV | #IC,WRIBYT | |
| 9676 | 031776 | 004737 | 006666 | | JSR | PC,WRITLU | ;SET IC (INPUT CLEAR) IN REG 12 |
| 9677 | 032002 | 000000 | | | JSR | PC,IACTIV | ;CHK IACT = 0 |
| 9678 | 032004 | 004737 | 003576 | 24\$: | 0 | | |
| | | | | | JSR | PC,MSTCLR | ;ISSUE CLEAN-UP MASTER CLEAR |

9679 032010
(3) 032010
(3) 032010 104401

ENDTST

L10063: TRAP C\$ETST

9680
9681
9682
9683
9684
9685
9686
9687
9688
9689
9690
9691
9692
9693
9694

```
*****  
:SBTTL TEST 32 - USYRT RECEIVER MSG TEST - CHAR MODE, NO CRC  
:  
:* THE LINE UNIT IS INITIALIZED AND A MESSAGE IS INITIATED (USING STEPLU) WITH  
:* LULP (REG 12) SET TO LOOP THE DATA INTERNALLY IN THE USYRT, AND WITH NO  
:* ERROR DETECTION. TWO SYNCHS, 000,125,252,377,000, AND TWO SYNCHS ARE  
:* THEN SENT. THE PROGRAM MONITORS IACT, AND THE RECEIVED CHARS ARE READ FROM  
:* AX0-15 AND COMPARED TO EXPECTED VALUES. THE PROGRAM THEN CHECKS FOR IACT  
:* STILL = 0, SETS IC TO CLEAR THE RECEIVER, AND CHECKS FOR IACT = 0.  
:*****
```

9695 032012
(3) 032012

BGNTST

T32::

9696 032012 012737 032254 002362
9697 032020 004737 003576
9698 032024 004737 010612
9699 032030 000226
9700 032032 000313
9701 032034 000000
9702 032036 000000
9703 032040 012737 000012 002400
9704 032046 005037 002402
9705 032052 112737 000040 002366
9706 032060 004737 003722
9707 032064 012701 003220
9708 032070 012137 002422 3\$:
9709 032074 004737 005146
9710 032100 020127 003242
9711 032104 103771
9712 032106 004737 005120
9713 032112 004737 005226
9714 032116 000030
9715 032120 004737 006666
9716 032124 000000
9717 032126 004737 005226
9718 032132 000006
9719 032134 012701 003224
9720 032140 004737 006666 10\$:
9721 032144 000001
9722 032146 004737 004076
9723 032152 023721 002370
9724 032156 001415
9725 032160 016137 177776 002404
9726 032166 013737 002370 002406
9727 032174 004737 004500
9728
9729 032200
(4) 032200 104455
(5) 032202 000032

```
MOV #24$,RETADR ;SET TEST EXIT ADDRESS FOR ERRORS  
JSR PC,MSTCLR ;ISSUE MASTER CLEAR  
JSR PC,SETUP ;PROGRAM THE USYRT  
SYNCH  
CRC2!CRC1!STRIP!IERR!DDCMP  
000  
000  
MOV #12,REGNUM ;SET LU REG NO. = 12  
CLR AXNUM ;SET AX BYTE NO. = 0 FOR AX0  
MOV# #LULP,WRIBYT  
JSR PC,WRITLU ;SET LULP IN REG 12  
MOV #MSG1,R1 ;GET POINTER TO MSG DATA TABLE  
3$: MOV (R1)+,TXWORD ;GET CHAR TO BE LOADED  
JSR PC,LDTXSI ;LOAD CHAR INTO TX SILO  
CMP R1,#MSG1+18. ;SEE IF ALL MSG CHARS LOADED YET  
BLO 3$ ;BR IF NOT YET  
JSR PC,WAIT50 ;ALLOW DATA TO RIPPLE IN SILO  
JSR PC,STPLU ;CLOCK LU FOR 24 CYCLES (UNTIL FIRST  
; DATA CHAR IS ABOUT TO BE RECEIVED)  
JSR PC,IACTIV ;CHK IACT = 0  
0  
6  
JSR PC,STPLU ;CLOCK LU UNTIL 1ST DATA CHAR IS RCV'D  
MOV #MSG1+4,R1  
10$: JSR PC,IACTIV ;CHK IACT = 1  
1  
JSR PC,READAX ;READ AX0  
CMP RAX15,(R1)+ ;COMPARE RCV'D CHAR TO EXPECTED  
BEQ 12$ ;BR IF RCV'D DATA OK  
MOV -2(R1),GOODAT ;GET EXPECTED DATA  
MOV RAX15,BADAT ;GET ACTUAL DATA  
JSR PC,GETALL ;GET REGS FOR PRINTOUT  
;REPORT INCORRECT DATA CHAR RCV'D  
ERRDF 26,EM26,ERR3
```

TRAP C\$ERDF
.WORD 26

```

(5) 032204 013043
(5) 032206 015114
9730 032210 000421
9731 032212 004737 005226
9732 032216 000010
9733 032220 020127 003236
9734 032224 103745
9735 032226 004737 006666
9736 032232 000001
9737 032234 012737 000200 002366
9738 032242 004737 003722
9739 032246 004737 006666
9740 032252 000000
9741 032254 004737 003576
9742 032260
(3) 032260
(3) 032260 104401

```

```

12$: BR 24$
JSR PC,STPLU ;CLOCK LU 8 CYCLES
8.
CMP R1,#MSG1+14. ;SEE IF ALL 5 DATA CHARS RCV'D YET
BLO 10$ ;BR IF NOT YET
JSR PC,IACTIV ;CHK FOR IACT STILL = 1
1
MOV #IC,WRIBYT
JSR PC,WRITLU ;SET IC (INPUT CLEAR) IN REG 12
JSR PC,IACTIV ;CHK IACT = 0
0
24$: JSR PC,MSTCLR ;ISSUE CLEAN-UP MASTER CLEAR
ENDTST

```

L10064: TRAP C\$ETST

9743
9744
9745
9746
9747
9748

```

;*****
;SBTTL TEST 33 - USYRT RECEIVER MSG TEST - BIT MODE, NO CRC
;
;* THE LINE UNIT IS INITIALIZED AND A MESSAGE IS INITIATED (USING STEPLU) WITH
;* LULP (REG 12) SET TO LOOP THE DATA INTERNALLY IN THE USYRT, AND WITH ERROR
;* DETECTION INHIBITED. TWO FLAGS, 000,125,252,377,000, AND TWO TERMINATING FLAGS
;* ARE THEN SENT. THE PROGRAM MONITORS IACT, RSOM, AND THE RCV'D CHARS ARE
;* READ FROM AX0-15 AND COMPARED TO EXPECTED VALUES. THE PROGRAM THEN CHECKS FOR
;* IACT = 0, SETS IC TO CLEAR THE RECEIVER, AND CHECKS FOR IACT STILL = 0.
;*****
BGNTST

```

```

9758 032262
(3) 032262
9759 032262 012737 032616 002362
9760 032270 004737 003576
9761 032274 004737 010612
9762 032300 000000
9763 032302 000302
9764 032304 000000
9765 032306 000000
9766 032310 012737 000012 002400
9767 032316 005037 002402
9768 032322 112737 000040 002366
9769 032330 004737 003722
9770 032334 012701 003220
9771 032340 012137 002422
9772 032344 004737 005146
9773 032350 020127 003242
9774 032354 103771
9775 032356 004737 005120
9776 032362 004737 005226
9777 032366 000041
9778 032370 004737 006666
9779 032374 000000
9780 032376 004737 007054

```

```

T33::
MOV #24$,RETADR
JSR PC,MSTCLR ;ISSUE MASTER CLEAR
JSR PC,SETUP ;PROGRAM THE USYRT
000
CRC2!CRC1!IFRR
000
000
MOV #12,REGNUM ;SET LU REG NO. = 12
CLR AXNUM ;SET AX BYTE NO. = 0 FOR AX0
MOVB #LULP,WRIBYT
JSR PC,WRITLU ;SET LULP IN REG 12
MOV #MSG1,R1 ;GET POINTER TO MSG DATA TABLE
3$: MOV (R1)+,TXWORD ;GET CHAR TO BE LOADED
JSR PC,LDTXSI ;LOAD CHAR INTO TX SILO
CMP R1,#MSG1+18. ;SEE IF ALL MSG CHARS LOADED YET
BLO 3$ ;BR IF NOT YET
JSR PC,WAIT50 ;ALLOW DATA TO RIPPLE IN SILO
JSR PC,STPLU ;CLOCK LU FOR 33 CYCLES (UNTIL FIRST
; DATA CHAR IS ABOUT TO BE RECEIVED)
33.
JSR PC,IACTI/ ;CHK IACT = 0
0
JSR PC,RSEOM ;CHK RSOM = 0, REGM = 0

```



```

9781 032402 000000 0
9782 032404 004737 005226 JSR PC,STPLU ;CLOCK LU UNTIL 1ST DATA CHAR IS RCV'D
9783 032410 000006 6
9784 032412 012701 003224 MOV #MSG1+4,R1
9785 032416 020127 003224 5$: CMP R1,#MSG1+4 ;SEE IF 1ST CHAR RCV'D
9786 032422 001007 BNE 6$ ;BR IF NO
9787 032424 004737 006666 JSR PC,IACTIV ;CHK IACT = 1
9788 032430 000001 1
9789 032432 004737 007054 JSR PC,RSEOM ;CHK RSOM = 1, REOM - 0
9790 032436 000001 1
9791 032440 000420 BR 9$
9792 032442 020127 003234 6$: CMP R1,#MSG1+12. ;SEE IF LAST CHAR RCV'D
9793 032446 001007 BNE 8$ ;BR IF NO
9794 032450 004737 006666 JSR PC,IACTIV ;CHK FOR IACT = 0
9795 032454 000000 0
9796 032456 004737 007054 JSR PC,RSEOM ;CHK RSOM = 0, REOM - 0
9797 032462 000000 0
9798 032464 000406 BR 9$
9799 032466 004737 006666 8$: JSR PC,IACTIV ;CHK FOR IACT = 1
9800 032472 000001 1
9801 032474 004737 007054 JSR PC,RSEOM ;CHK RSOM = 0, REOM = 0
9802 032500 000000 0
9803 032502 004737 004076 9$: JSR PC,READAX ;READ AX0
9804 032506 023721 002370 CMP RAX15,(R1)+ ;COMPARE RCV'D CHAR TO EXPECTED
9805 032512 001415 BEQ 12$ ;BR IF RCV'D DATA OK
9806 032514 016137 177776 002404 MOV -2(R1),GOODAT ;GET EXPECTED DATA
9807 032522 013737 002370 002406 MOV RAX15,BADAT ;GET ACTUAL DATA
9808 032530 004737 004500 JSR PC,GETALL ;GET REGS FOR PRINTOUT
9809 ;REPORT INCORRECT DATA CHAR RCV'D
9810 ERRDF 26,EM26,ERR3
(4) 032534 104455 TRAP C$ERDF
(5) 032536 000032 .WORD 26
(5) 032540 013043 .WORD EM26
(5) 032542 015114 .WORD ERR3
9811 032544 000424 BR 24$
9812 032546 004737 005226 12$: JSR PC,STPLU ;CLOCK LU 8 CYCLES
9813 032552 000010 8.
9814 032554 020127 003236 CMP R1,#MSG1+14. ;SEE IF ALL DATA CHARS CHECKED YET
9815 032560 103716 BLO 5$ ;BR IF NOT YET
9816 032562 004737 006666 JSR PC,IACTIV ;CHK IACT = 0
9817 032566 000000 0
9818 032570 004737 007054 JSR PC,RSEOM ;CHK RSOM - 0, REOM - 0
9819 032574 000000 0
9820 032576 012737 000200 002366 MOV #IC,WRIBYT
9821 032604 004737 003722 JSR PC,WRITLU ;SET IC (INPUT CLEAR) IN REG 12
9822 032610 004737 006666 JSR PC,IACTIV ;CHK IACT = 0
9823 032614 000000 0
9824 032616 004737 003576 24$: JSR PC,MSTCLR ;ISSUE CLEAN-UP MASTER CLEAR
9825 032622 ENDTST
(3) 032622 L10065: TRAP C$ETST
(3) 032622 104401
9826
9827
9828
9829
9830

```

9831
9832
9833
9834
9835
9836
9837
9838 032624
(3) 032624
9839 032624 004737 003576
9840 032630 012737 000014 002400
9841 032636 012737 000040 002366
9842 032644 004737 003722
9843 032650 012737 000040 002426
9844 032656 012737 000125 002422
9845 032664 004737 005146
9846 032670 012737 000002 002402
9847 032676 004737 004076
9848 032702 123727 002370 000000
9849 032710 001414
9850 032712 012737 000000 002404
9851 032720 013737 002370 002406
9852 032726 004737 004500
9853
9854 032732
(4) 032732 104455
(5) 032734 000003
(5) 032736 012174
(5) 032740 015114
9855 032742 005037 002426
9856 032746 004737 003576
9857 032752
(3) 032752
(3) 032752 104401
9858
9859
9860
9861
9862
9863
9864
9865
9866
9867
9868
9869
9870
9871
9872
9873
9874 032754
(3) 032754
9875 032754 012737 033654 002362
9876 032762 004737 005512
9877 032766 000000
9878 032770 000000

```
*****
:SBTTL      TEST 34 - SILO-DISABLED TRANSMITTER LOAD TEST
:*
:* THIS TEST DISABLES THE SILOS, LOADS A 125 CHARACTER INTO THE TX SILO, AND
:* READS AX1-15 AND CHECKS THAT THE DATA DID NOT GET LOADED INTO THE USYRT TX
:* BUFFER.
*****
BGNTST
T34::
JSR      PC,MSTCLR      ;ISSUE MASTER CLEAR
MOV      #14,REGNUM     ;SET REG NO. = 14
MOV      #DISSI,WRIBYT  ;SET DISSI BIT
JSR      PC,WRITLU
MOV      #DISSI,DISILO  ;SET DISABLE SILO FLAG
MOV      #125,TXWORD    ;LOAD 125 INTO TX SILO
JSR      PC,LDTXSI
MOV      #2,AXNUM       ;SET REG NO. FOR AX1
JSR      PC,READAX      ;READ AX1-15, AX1-16
CMPB     RAX15,#000     ;CHECK FOR AX1-15 UNCHANGED
BEQ      3$             ;BR IF UNCHANGED
MOV      #000,GOODAT    ;GET EXPECTED DATA
MOV      RAX15,BADDAT   ;GET ACTUAL DATA
JSR      PC,GETALL      ;GET REGS FOR PRINTOUT
:REPORT   REG MISCMPARE
ERRDF    3,EM3,ERR3
TRAP     C$ERDF
        .WORD 3
        .WORD EM3
        .WORD ERR3
3$:      CLR      DISILO ;CLEAR DISABLE SILO FLAG
JSR      PC,MSTCLR     ;ISSUE MASTER CLEAR TO CLEAN UP
ENDTST
L10066:  TRAP     C$ETST
*****
:SBTTL      TEST 35 - SILO-DISABLED MESSAGE TEST - BIT MODE, NO CRC
:*
:* THE LINE UNIT IS INITIALIZED AND A MESSAGE IS INITIATED (USING STEPLU)
:* WITH LULP (REG 12) SET TO LOOP THE DATA INTERNALLY IN THE USYRT, WITH SILO
:* DISABLE SET, AND WITH NO ERROR DETECTION. TWO FLAGS, 000,125,252, AND
:* TERMINATING FLAGS ARE THEN SENT BY LOADING THE TRANSMITTED CHARS INTO
:* REG AX1. THE PROGRAM MONITORS OACT, IACT, RSOM, REOM, ORDY, OCOR, ICIR,
:* IRDY, AND THE RECEIVED CHARS ARE READ FROM AX0 AND COMPARED TO EXPECTED
:* VALUES.
*****
BGNTST
T35::
MOV      #18$,RETADR    ;SET TEST EXIT ADDRESS FOR ERRORS
JSR      PC,INITRN     ;FIND OUT WHICH USYRT CHIP
0
0
```

```

9879 032772 004737 003576      JSR    PC,MSTCLR      ;ISSUE MASTER CLEAR
9880 032776 004737 010612      JSR    PC,SETUP      ;PROGRAM THE USYRT
9881 033002 000000                000
9882 033004 000302                CRC2!CRC1!IERR
9883 033006 000000                000
9884 033010 000000                000
9885 033012 012737 000014 002400  MOV    #14,REGNUM      ;SET REG NO. = 14
9886 033020 012737 000140 002366  MOV    #TXEN!DISSI,WRIBYT
9887 033026 004737 003722      JSR    PC,WRITLU      ;SET TXEN AND DISSI IN REG 14
9888 033032 012737 000140 002426  MOV    #TXEN!DISSI,DISILO ;SET DISABLE SILO FLAG
9889 033040 012737 000012 002400  MOV    #12,REGNUM      ;SET LU REG NO. = 12
9890 033046 112737 000040 002366  MOVB   #LULP,WRIBYT
9891 033054 004737 003722      JSR    PC,WRITLU      ;SET LULP IN REG 12
9892 033060 012701 003220      MOV    #MSG1,R1       ;GET POINTER TO MSG
9893 033064 004737 004634      JSR    PC,OSIRDY      ;CHK ORDY = 1
9894 033070 000001                1
9895 033072 012737 000002 002402  MOV    #2,AXNUM        ;SET AX BYTE NO. FOR AX1
9896 033100 112137 002374      MOVB   (R1)+,WAX15     ;GET A CHAR
9897 033104 112137 002376      MOVB   (R1)+,WAX16
9898 033110 004737 004264      JSR    PC,WRITAX      ;LOAD CHAR INTO USYRT TX BUFFER
9899 033114 004737 004634      JSR    PC,OSIRDY      ;CHK ORDY = 0
9900 033120 000000                0
9901 033122 004737 005324      JSR    PC,OACTIV      ;CHK OACT = 0
9902 033126 000000                0
9903 033130 004737 005226      JSR    PC,STPLU       ;CLOCK LU FOR 3 CYCLES
9904 033134 000003                3
9905 033136 004737 005324      JSR    PC,OACTIV      ;CHK OACT = 1
9906 033142 000001                1
9907 033144 012703 000004      MOV    #4,R3          ;INIT COUNTER
9908 033150 112137 002374 4$:  MOVB   (R1)+,WAX15     ;GET ANOTHER CHAR
9909 033154 112137 002376      MOVB   (R1)+,WAX16
9910 033160 020327 000001      CMP    R3,#1          ;SEE IF LOADING LAST DATA CHAR YET
9911 033164 001006                BNE    5$             ;BR IF NOT
9912 033166 005737 002430      TST    CHPTYP         ;SEE IF SIG USYRT
9913 033172 001403                BEQ    5$             ;BR IF YES
9914 033174 112737 000002 002376  MOVB   #TEOM,WAX16    ;SET TEOM WITH LAST DATA CHAR
9915 033202 004737 004634 5$:  JSR    PC,OSIRDY      ;CHK ORDY = 1
9916 033206 000001                1
9917 033210 004737 004264      JSR    PC,WRITAX      ;LOAD ANOTHER CHAR INTO USYRT TX BUFFER
9918 033214 004737 004634      JSR    PC,OSIRDY      ;CHK ORDY = 0
9919 033220 000000                0
9920 033222 004737 006666      JSR    PC,IACTIV      ;CHK IACT = 0
9921 033226 000000                0
9922 033230 004737 007054      JSR    PC,RSEOM       ;CHK RSOM = 0, REOM = 0
9923 033234 000000                0
9924 033236 004737 005226      JSR    PC,STPLU       ;CLOCK LU FOR 8 CYCLES
9925 033242 000010                8.
9926 033244 004737 005324      JSR    PC,OACTIV      ;CHK OACT = 1
9927 033250 000001                1
9928 033252 004737 004634      JSR    PC,OSIRDY      ;CHK ORDY = 1
9929 033256 000001                1
9930 033260 005303                DEC    R3             ;DECR COUNTER
9931 033262 001332                BNE    4$             ;BR IF NOT DONE YET
9932 033264 004737 006402      JSR    PC,ISIRDY      ;CHK IRDY = 0
9933 033270 000000                0
9934 033272 005737 002430      TST    CHPTYP         ;SEE IF SIG USYRT

```

| | | | | | | | | | | | |
|------|--------|--------|--------|--------|-------|---------|---------------------------|--|---|---------|--|
| 9935 | 033276 | 001007 | | | | BNE | 11\$ | | :BR IF NOT | | |
| 9936 | 033300 | 105037 | 002374 | | | CLRB | WAX15 | | | | |
| 9937 | 033304 | 112737 | 000002 | 002376 | | MOVB | #TEOM,WAX16 | | :LOAD EOM CHAR | | |
| 9938 | 033312 | 004737 | 004264 | | | JSR | PC,WRITAX | | :LOAD ANOTHER CHAR INTO USYRT TX BUFFER | | |
| 9939 | 033316 | 004737 | 005226 | | 11\$: | JSR | PC,STPLU | | :CLOCK LU FOR 3 CYCLES | | |
| 9940 | 033322 | 000003 | | | | 3 | | | | | |
| 9941 | 033324 | 004737 | 006402 | | | JSR | PC,ISIRDY | | :CHK IRDY = 1 | | |
| 9942 | 033330 | 000002 | | | | 2 | | | | | |
| 9943 | 033332 | 004737 | 005324 | | | JSR | PC,OACTIV | | :CHK OACT = 1 | | |
| 9944 | 033336 | 000001 | | | | 1 | | | | | |
| 9945 | 033340 | 004737 | 006666 | | | JSR | PC,IACTIV | | :CHK IACT = 1 | | |
| 9946 | 033344 | 000001 | | | | 1 | | | | | |
| 9947 | 033346 | 004737 | 007054 | | | JSR | PC,RSEOM | | :CHK RSOM = 1, REOM = 0 | | |
| 9948 | 033352 | 000001 | | | | 1 | | | | | |
| 9949 | 033354 | 004737 | 006402 | | | JSR | PC,ISIRDY | | :CHK IRDY = 0 | | |
| 9950 | 033360 | 000000 | | | | 0 | | | | | |
| 9951 | 033362 | 012737 | 000000 | 002402 | | MOV | #0,AXNUM | | :SET AX BYTE NO. FOR AX0 | | |
| 9952 | 033370 | 123727 | 002370 | 000000 | | CMPB | RAX15,#000 | | :COMPARE RCV'D CHAR TO 000 | | |
| 9953 | 033376 | 001415 | | | | BEQ | 9\$ | | :BR IF MATCH | | |
| 9954 | 033400 | 012737 | 000000 | 002404 | | MOV | #0,GOODAT | | :SET EXPECTED DATA | | |
| 9955 | 033406 | 012737 | 002370 | 002406 | 6\$: | MOV | #RAX15,BADDAT | | :SET ACTUAL DATA | | |
| 9956 | 033414 | 004737 | 004500 | | | JSR | PC,GETALL | | :GET REGS FOR PRINTOUT | | |
| 9957 | | | | | | | | | | | |
| 9958 | 033420 | | | | | :REPORT | INCORRECT DATA CHAR RCV'D | | | | |
| (4) | 033420 | 104455 | | | | ERRDF | 26,EM26,ERR3 | | | | |
| (5) | 033422 | 000032 | | | | | | | TRAP | C\$ERDF | |
| (5) | 033424 | 013043 | | | | | | | .WORD | 26 | |
| (5) | 033426 | 015114 | | | | | | | .WORD | EM26 | |
| 9959 | 033430 | 000511 | | | | | | | .WORD | ERR3 | |
| 9960 | 033432 | 004737 | 005226 | | 9\$: | BR | 18\$ | | | | |
| 9961 | 033436 | 000010 | | | | JSR | PC,STPLU | | :CLOCK LU FOR 8 CYCLES | | |
| 9962 | 033440 | 004737 | 006402 | | | 8. | | | | | |
| 9963 | 033444 | 000002 | | | | JSR | PC,ISIRDY | | :CHK IRDY = 1 | | |
| 9964 | 033446 | 004737 | 005324 | | | 2 | | | | | |
| 9965 | 033452 | 000001 | | | | JSR | PC,OACTIV | | :CHK OACT = 1 | | |
| 9966 | 033454 | 004737 | 006666 | | | 1 | | | | | |
| 9967 | 033460 | 000001 | | | | JSR | PC,IACTIV | | :CHK IACT = 1 | | |
| 9968 | 033462 | 004737 | 007054 | | | 1 | | | | | |
| 9969 | 033466 | 000000 | | | | JSR | PC,RSEOM | | :CHK RSOM = 1, REOM = 0 | | |
| 9970 | 033470 | 004737 | 006402 | | | 0 | | | | | |
| 9971 | 033474 | 000000 | | | | JSR | PC,ISIRDY | | :CHK IRDY = 0 | | |
| 9972 | 033476 | 123727 | 002370 | 000125 | | 0 | | | | | |
| 9973 | 033504 | 001404 | | | | CMPB | RAX15,#125 | | :COMPARE 2ND RCV'D CHAR TO 125 | | |
| 9974 | 033506 | 012737 | 000125 | 002404 | | BEQ | 12\$ | | :BR IF MATCH | | |
| 9975 | 033514 | 000734 | | | | MOV | #125,GOODAT | | :SET EXPECTED DATA | | |
| 9976 | 033516 | 004737 | 005226 | | 12\$: | BR | 6\$ | | :BR TO REPORT ERROR | | |
| 9977 | 033522 | 000010 | | | | JSR | PC,STPLU | | :CLOCK LU FOR 8 CYCLES | | |
| 9978 | 033524 | 004737 | 006402 | | | 8. | | | | | |
| 9979 | 033530 | 000002 | | | | JSR | PC,ISIRDY | | :CHK IRDY = 1 | | |
| 9980 | 033532 | 004737 | 005324 | | | 2 | | | | | |
| 9981 | 033536 | 000001 | | | | JSR | PC,OACTIV | | :CHK OACT = 1 | | |
| 9982 | 033540 | 004737 | 006666 | | | 1 | | | | | |
| 9983 | 033544 | 000000 | | | | JSR | PC,IACTIV | | :CHK IACT = 0 | | |
| 9984 | 033546 | 004737 | 007054 | | | 0 | | | | | |
| 9985 | 033552 | 000002 | | | | JSR | PC,RSEOM | | :CHK RSOM = 0, REOM = 1 | | |
| 9986 | 033554 | 004737 | 006402 | | | 2 | | | | | |
| | | | | | | JSR | PC,ISIRDY | | :CHK IRDY = 0 | | |

CZDMRA M8203 STATIC DIAG #1
CZDMRA.P11 18-JUL-79 09:44

MACY11 30A(1052) 18-JUL-79 09:53 PAGE 7-122
TEST 35 - SILO-DISABLED MESSAGE TEST - BIT MODE, NO CRC

G 13

SEQ 0162

| | | | | | | | |
|-------|--------|--------|--------|--------|-----------|---------------|----------------------------------|
| 9987 | 033560 | 000000 | | | 0 | | |
| 9988 | 033562 | 123727 | 002370 | 000252 | CMPB | RAX15,#252 | ;COMPARE 3RD RCV'D CHAR TO 252 |
| 9989 | 033570 | 001404 | | | BEQ | 14\$ | ;BR IF MATCH |
| 9990 | 033572 | 012737 | 000252 | 002404 | MOV | #252,GOODAT | ;SET EXPECTED DATA |
| 9991 | 033600 | 000702 | | | BR | 6\$ | ;BR TO REPORT ERROR |
| 9992 | 033602 | 012737 | 000014 | 002400 | 14\$: MOV | #14,REGNUM | ;SET REG NO. = 14 |
| 9993 | 033610 | 012737 | 000040 | 002366 | MOV | #DISSI,WRIBYT | |
| 9994 | 033616 | 004737 | 003722 | | JSR | PC,WRITLU | ;CLEAR TX ENABLE |
| 9995 | 033622 | 012737 | 000011 | 002400 | MOV | #11,REGNUM | ;SET REG NO. = 11 |
| 9996 | 033630 | 012737 | 000200 | 002366 | MOV | #OC,WRIBYT | |
| 9997 | 033636 | 004737 | 003722 | | JSR | PC,WRITLU | ;SET OC TO SHUT DOWN TRANSMITTER |
| 9998 | 033642 | 004737 | 005120 | | JSR | PC,WAIT50 | ;WAIT FOR SHUTDOWN |
| 9999 | 033646 | 004737 | 005324 | | JSR | PC,OACTIV | ;CHK FOR OACT = 0 |
| 10000 | 033652 | 000000 | | | 0 | | |
| 10001 | 033654 | 005037 | 002426 | | 18\$: CLR | DISILO | ;CLEAR DISABLE SILO FLAG |
| 10002 | 033660 | 004737 | 003576 | | JSR | PC,MSTCLR | ;ISSUE MASTER CLEAR TO CLEAN UP |
| 10003 | 033664 | | | | ENDTST | | |
| (3) | 033664 | | | | | | |
| (3) | 033664 | 104401 | | | | | L10067: TRAP C\$ETST |

10004
10005
10006
10007
10008
10009
10010
10011
10012
10013
10014
10015
10016
10017
10018
10019
10020
10021
10022
10023
10024
10025
10026
10027
(3)
10028
10029
10030
10031
10032
10033
10034
10035
10036
10037
10038
10039

| | | | | | | | |
|-------|--------|--------|--------|--------|--|--|--|
| 10027 | 033666 | | | | | | |
| (3) | 033666 | | | | | | |
| 10028 | 033666 | 012737 | 034142 | 002362 | | | |
| 10032 | 033674 | 004737 | 003576 | | | | |
| 10033 | 033700 | 004737 | 006402 | | | | |
| 10034 | 033704 | 00. 01 | | | | | |
| 10039 | 033706 | 004737 | 005512 | | | | |

```
*****  
:SBTTL TEST 36 - RECEIVER BUFFER TEST - CHAR MODE, CRC  
:  
:* FIRST, A MASTER CLEAR IS DONE AND THE PROGRAM CHECKS FOR ICIR = 1 AND IRDY  
:* - 0. THEN, 2 SOM CHARS ARE LOADED AND CLOCKED INTO THE USYRT, AND 64  
:* BYTES OF A 256-BYTE BINARY COUNT DATA PATTERN (000-377) ARE LOADED INTO  
:* THE TX SILO.  
:* THE LINE UNIT IS THEN CLOCKED UNTIL IRDY = 1, AND THE PROGRAM CHECKS FOR  
:* THIS TO OCCUR WITHIN 40-43 CYCLES. THE PROGRAM READS THE RCV SILO, CHECKS THE  
:* CHAR FOR 000, AND CHECKS FOR IRDY = 0 AGAIN.  
:* THE LINE UNIT IS THEN CLOCKED IN GROUPS OF 8 CYCLES, AND AFTER EACH, THE  
:* PROGRAM CHECKS FOR ICIR = 1, IRDY = 1, UNTIL THE 64TH GROUP, AFTER WHICH  
:* IT CHECKS FOR ICIR = 0, IRDY = 1. THE SECOND DATA CHAR IS READ FROM THE  
:* RECEIVER SILO AND COMPARED TO 001. THEN, THE PROGRAM CHECKS FOR ICIR = 1,  
:* IRDY = 1 AGAIN.  
:* THE REST OF THE BINARY COUNT DATA BYTES ARE CYCLED 8 CLOCKS AND READ AND  
:* COMPARED A BYTE AT A TIME.  
:*****  
:BGNTST  
:-----  
: T36::  
: MOV #A10,RETADR ;SET TEST EXIT ADDRESS FOR ERRORS  
:-----  
: DO MASTER CLR, CHK FOR ICIR = 1, IRDY = 0  
:-----  
: JSR PC,MSTCLR ;ISSUE MASTER CLEAR  
: JSR PC,ISIRDY ;CHK ICIR = 1, IRDY = 0  
:-----  
: LOAD AND CLOCK 2 SOM'S, LOAD 64 BYTES OF BINARY COUNT PATTERN INTO TX SILO,  
: CLOCK LINE UNIT, CHK FOR IRDY = 1 WITHIN 40-43 CYCLES  
:-----  
: JSR PC,INITRN ;LOAD 2 SOM'S, CLOCK THEM INTO USYRT
```

```
10040 033712 000226          SYNCH
10041 033714 000011          STRIP!DDCMP
10042 033716 005003          CLR      R3                ;INIT BINARY COUNT DATA FOR WRITING
10043 033720 010337 002422    2$:  MOV      R3,TXWORD
10044 033724 004737 005146    .SR     PC,LDTXSI         ;LOAD A DATA BYTE INTO TX SILO
10045 033730 005203          .INC     R3                ;INCR DATA
10046 033732 020327 000100    CMP     R3,#64.           ;SEE IF 64 BYTES LOADED YET
10047 033736 002770          BLT     2$                ;BR IF NOT YET
10048 033740 004737 007406    JSR     PC,RCV1ST        ;RECEIVE AND TIME FIRST CHARACTER
10049 033744 000050          40.
10050
10051          ;-----
10052          ; READ RCV SILO, COMPARE FIRST CHAR TO 000, CHK FOR ICIR = 1, IRDY = 0
10053          ;-----
10053 033746 005004          9$:  CLR      R4                ;INIT PATTERN FOR READING
10054 033750 004737 007722    JSR     PC,CKDATA        ;READ RCV SILO, COMPARE DATA
10055 033754 000000          0                ;EXPECTED DATA = 000
10056 033756 000000          0                ;DON'T CLOCK LINE UNIT
10057 033760 005204          16$:  INC      R4                ;INCR DATA FOR READING
10058 033762 004737 006402    JSR     PC,ISIRDY        ;CHK FOR ICIR = 1, IRDY = 0
10059 033766 000001          1
10060
10061          ;-----
10062          ; CLOCK 63 CHAR INTO RCV SILO, CHK ICIR = 1, IRDY = 1
10063          ;-----
10063 033770 004737 005226    18$:  JSR     PC,STPLU        ;CLOCK LU FOR 8 CYCLES
10064 033774 000010          8.
10065 033776 010337 002422    MOV     R3,TXWORD
10066 034002 004737 005146    JSR     PC,LDTXSI        ;LOAD ANOTHER WORD INTO TX SILO
10067 034006 005203          INC     R3                ;INCR PATTERN FOR WRITING
10068 034010 004737 006402    JSR     PC,ISIRDY        ;CHK ICIR = 1, IRDY = 1
10069 034014 000003          3
10070 034016 020327 000177    CMP     R3,#127.         ;SEE IF 63 MORE CHARS CLOCKED YET
10071 034022 002762          BLT     18$              ;BR IF NOT YET
10072
10073          ;-----
10074          ; CLOCK 1 MORE CHAR INTO RCV SILO, CHK ICIR = 0, IRDY = 1
10075          ;-----
10075 034024 004737 005226    JSR     PC,STPLU        ;CLOCK LU FOR 8 CYCLES
10076 034030 000010          8.
10077 034032 004737 006402    JSR     PC,ISIRDY        ;CHK ICIR = 0, IRDY = 1
10078 034036 000002          2
10079
10080          ;-----
10081          ; READ, COMPARE, CLOCK REST OF DATA CHARS
10082          ;-----
10082 034040 010437 034050    20$:  MOV     R4,21$         ;SET EXPECTED DATA
10083 034044 004737 007722    JSR     PC,CKDATA        ;READ AND COMPARE DATA
10084 034050 000000          21$:  0                ;EXPECTED SILO ENTRY GOES HERE
10085 034052 000000          0                ;DON'T CLOCK LINE UNIT
10086 034054 005204          22$:  INC     R4                ;INCR DATA PATTERN FOR READS
10087 034056 020427 000400    CMP     R4,#400         ;SEE IF ALL DONE READING YET
10088 034062 001427          BEQ     32$              ;BR IF DONE READING
10089 034064 004737 006402    JSR     PC,ISIRDY        ;CHK ICIR = 1, IRDY = 1
10090 034070 000003          3
10091 034072 004737 005226    JSR     PC,STPLU        ;CLOCK LU FOR 8 CYCLES
10092 034076 000010          8.
10093 034100 020327 000377    CMP     R3,#377         ;SEE IF ALL CHARS LOADED INTO TX SILO YET
10094 034104 003007          BGT     24$              ;BR IF YES
10095 034106 010337 002422    MOV     R3,TXWORD
```

CZDMRA M8203 STATIC DIAG #1
C? MRA.P11 18-JUL-79 09:44

MACY11 30A(1052) 18-JUL-79 09:53 PAGE 7-124
TEST 36 - RECEIVER BUFFER TEST - CHAR MODE, CRC

SEQ 0164

```

10096 034112 004737 005146      JSR    PC,LDTXSI      ;LOAD ANOTHER CHAR INTO TX SILO
10097 034116 005203              INC    R3              ;INCR DATA PATTERN FOR WRITING
10098 034120 000137 034040      JMP    20$
10099 034124 012737 001000 002422 24$:  MOV    #TXEOM,TXWORD
10100 034132 004737 005146      JSR    PC,LDTXSI      ;LOAD EOM INTO TX SILO
10101 034136 000137 034040      JMP    20$
10102 034142              32$:
10103 034142 004737 003576      A10:  JSR    PC,MSTCLR      ;ISSUE MASTER CLEAR TO CLEAN UP
10104 034146              ENDTST
      (3) 034146
      (3) 034146 104401
                                           L10070:
                                           TRAP   C$ETST

```

```

10105
10106
10107
10108
10109
10110
10111
10112
10113
10114
10115
10116
10117
10118
10119
10120
10121

```

```

;*****
;SBTTL      TEST 37 - RECEIVER CHAR LENGTH TIMING TEST - CHAR MODE, NO CRC
;*
;* THE LINE UNIT IS PLACED IN CHAR MODE, WITH NO ERROR DETECTION, AND A MSG IS
;* INITIATED WITH 2 SYNCH CHARS. NEXT, FIFTEEN 000 CHARS ARE LOADED INTO THE
;* TRANSMITTER SILO. THE LINE UNIT IS THEN CLOCKED USING STEPLU WITH LULOOP
;* SET, WHILE THE RECEIVER CHAR LENGTH IS SET TO THE FOLLOWING VALUES : 5,6,7,8.
;* FOR EACH RCV CHAR LENGTH, THE PROGRAM CHECKS TO MAKE SURE THET USYRT RECEIVER
;* FLAGS OCCUR THE PROPER NO. OF CYCLES APART, FOR EACH RCV CHAR LENGTH.
;* (FOR EXAMPLE A 5-BIT CHAR TAKES 5 CLOCK CYCLES TO BE RECEIVED). A MASTER
;* CL AR IS THEN DONE TO TERMINATE THE OPERATION.
;*****
BGNTST

```

```

10122 034150
      (3) 034150
10123 034150 012737 034342 002362      MOV    #24$,RETADR      ;SET TEST EXIT ADRS FOR ERRORS
10124 034156 004737 005512      JSR    PC,INITRN        ;DO MASTER CLR, LOAD 2 SOM'S
10125 034162 000000
10126 034164 000341
10127 034166 012701 000017      CRC2!CRC1!IDLE!DDCMP
10128 034172 005037 002422      MOV    #15.,R1          ;INIT COUNTER
10129 034176 004737 005146      CLR    TXWORD
10130 034202 005301 3$:      JSR    PC,LDTXSI        ;LOAD A 000 CHAR INTO TX SILO
10131 034204 001374      DEC    R1                ;DECR COUNTER
10132 034206 004737 005120      BNE    3$                ;BR IF NOT DONE LOADING YET
10133 034212 012737 000006 002402      JSR    PC,WAIT50        ;WAIT FOR SILO TO RIPPLE
10134 034220 012737 000000 002374      MOV    #6,AXNUM         ;SET BYTE NO. = 6 FOR AX3
10135 034226 012737 000005 002376      MOV    #000,WAX15       ;SET DATA FOR AX3-15 = 0
10136 034234 004737 004264      MOV    #RXLEN2!RXLENO,WAX16 ;SET RCV LEN = 5
10137 034240 004737 005226      JSR    PC,WRITAX        ;LOAD AX3
10138 034244 100012      JSR    PC,STPLU         ;CLK LU UNTIL TX'ING 1ST DATA CHAR
10139 034246 012737 000006 002376      CHPCHK!10.
10140 034254 004737 004264      MOV    #RXLEN2!RXLEN1,WAX16 ;SET RCV LEN = 6
10141 034260 004737 007406      JSR    PC,WRITAX        ;LOAD AX3
10142 034264 000005      JSR    PC,RCV1ST       ;CLOCK 5-BIT DATA CHAR
10143 034266 012737 000007 002376      5
10144 034274 004737 004264      MOV    #RXLEN2!RXLEN1!RXLENO,WAX16 ;SET RCV LEN = 7
10145 034300 004737 011046      JSR    PC,WRITAX        ;LOAD AX3
10146 034304 000006      JSR    PC,RXCHAR       ;RCV 5-BIT DATA CHAR, CLK 6-BIT
10147 034306 012737 000000 002376      6
10148 034314 004737 004264      MOV    #0,WAX16         ;SET RCV LEN = 8
      JSR    PC,WRITAX        ;LOAD AX3

```

CZDMRA M8203 STATIC DIAG #1
CZDMRA.P11 18-JUL-79 09:44

MACY11 30A(1052) 18-JUL-79 09:53 PAGE 7-125
TEST 37 - RECEIVER CHAR LENGTH TIMING TEST - CHAR MODE, NO CRC

SEQ 0165

| | | | |
|-------|--------|--------|--------|
| 10149 | 034320 | 004737 | 011046 |
| 10150 | 034324 | 000007 | |
| 10151 | 034326 | 004737 | 011046 |
| 10152 | 034332 | 000010 | |
| 10153 | 034334 | 004737 | 011046 |
| 10154 | 034340 | 000010 | |
| 10155 | 034342 | 004737 | 003576 |
| 10156 | 034346 | | |
| (3) | 034346 | | |
| (3) | 034346 | 104401 | |

```

JSR PC,RXCHAR ;RCV 6-BIT DATA CHAR, CLK 7-BIT
7
JSR PC,RXCHAR ;RCV 7-BIT DATA CHAR, CLK 8-BIT
8.
JSR PC,RXCHAR ;RCV 8-BIT DATA CHAR
8.
JSR PC,MSTCLR ;ISSUE MASTER CLEAR TO CLEAN UP
24$:
ENDTST
L10071:
TRAP C$ETST

```

| | | | |
|-------|--|--|--|
| 10157 | | | |
| 10158 | | | |
| 10159 | | | |
| 10160 | | | |
| 10161 | | | |
| 10162 | | | |

```

*****
:SBTTL TEST 38 - RECEIVER CHAR LENGTH TIMING TEST - BIT MODE, NO CRC
:*
:* THE LINE UNIT IS PLACED IN BIT MODE WITH NO ERROR DETECTION, AND A MESSAGE IS
:* INITIATED WITH 2 FLAG CHARS. NEXT, FIFTEEN 000 CHARS ARE LOADED INTO THE
:* TRANSMITTER SILO. THE LINE UNIT IS THEN CLOCKED USING STEPLU WITH LULOOP
:* SET, WHILE THE RCV CHAR LENGTH IS SET TO THE FOLLOWING VALUES : 8,8,8,7,6,5,
:* 4,3,2,1. FOR EACH RCV CHAR LENGTH, THE PROGRAM CHECKS TO MAKE SURE THAT THE
:* USYRT RECEIVER FLAGS OCCUR THE PROPER NO. OF CYCLES APART, FOR EACH RCV
:* CHAR LENGTH. (FOR EXAMPLE, A 5 BIT CHAR TAKES 5 CLOCK CYCLES TO BE RECEIVED).
:* A MASTER CLEAR IS THEN DONE TO TERMINATE THE OPERATION.
*****
BGNTST

```

| | | | |
|-------|--------|--------|---------------|
| 10174 | 034350 | | |
| (3) | 034350 | | |
| 10175 | 034350 | 012737 | 034642 002362 |
| 10176 | 034356 | 004737 | 005512 |
| 10177 | 034362 | 000000 | |
| 10178 | 034364 | 000300 | |
| 10179 | 034366 | 012701 | 000017 |
| 10180 | 034372 | 005037 | 002422 |
| 10181 | 034376 | 004737 | 005146 |
| 10182 | 034402 | 005301 | |
| 10183 | 034404 | 001374 | |
| 10184 | 034406 | 004737 | 005120 |
| 10185 | 034412 | 012737 | 000006 002402 |
| 10186 | 034420 | 012737 | 000000 002374 |
| 10187 | 034426 | 004737 | 007406 |
| 10188 | 034432 | 000040 | |
| 10189 | 034434 | 012737 | 000000 002376 |
| 10190 | 034442 | 004737 | 004264 |
| 10191 | 034446 | 004737 | 011046 |
| 10192 | 034452 | 000010 | |
| 10193 | 034454 | 012737 | 000007 002376 |
| 10194 | 034462 | 004737 | 004264 |
| 10195 | 034466 | 004737 | 011046 |
| 10196 | 034472 | 000010 | |
| 10197 | 034474 | 012737 | 000006 002376 |
| 10198 | 034502 | 004737 | 004264 |
| 10199 | 034506 | 004737 | 011046 |
| 10200 | 034512 | 000007 | |
| 10201 | 034514 | 012737 | 000005 002376 |

```

T38::
MOV #24$,RETADR ;SET TEST EXIT ADRS FOR ERRORS
JSR PC,INITRN ;DO MASTER CLR, LOAD 2 SOM'S
000
CRC2.CRC1
MOV #15.,R1 ;INIT COUNTER
CLR TXWORD
3$:
JSR PC,LDTXSI ;LOAD A 000 CHAR INTO TX SILO
DEC R1 ;DECR COUNTER
BNE 3$ ;BR IF NOT DONE LOADING YET
JSR PC,WAIT50 ;WAIT FOR SILO TO RIPPLE
MOV #6,AXNUM ;SET BYTE NO. = 6 FOR AX3
MOV #000,WAX15 ;SET DATA FOR AX3-15 - 0
JSR PC,RCV1ST ;CLOCK FIRST 8-BIT DATA CHAR
32.
MOV #0,WAX16 ;SET RCV LEN = 8
JSR PC,WRITAX ;LOAD AX3
JSR PC,RXCHAR ;RCV FIRST 8-BIT DATA CHAR, CLK SECOND 8-BIT
8.
MOV #RXLEN2!RXLEN1,RXLEN0,WAX16 ;SET RCV LEN = 7
JSR PC,WRITAX ;LOAD AX3
JSR PC,RXCHAR ;RCV SECOND 8-BIT DATA CHAR, CLK 3RD 8-BIT
8.
MOV #RXLEN2!RXLEN1,WAX16 ;SET RCV LEN = 6
JSR PC,WRITAX ;LOAD AX3
JSR PC,RXCHAR ;RCV 3RD 8-BIT DATA CHAR, CLK 7-BIT
7
MOV #RXLEN2!RXLEN0,WAX16 ;SET RCV LEN = 5

```


CZDMRA M8203 STATIC DIAG #1
CZDMRA.P11 18-JUL-79 09:44

MACY11 30A(1052) 18-JUL-79 09:53 PAGE 7-126

TEST 38 - RECEIVER CHAR LENGTH TIMING TEST - BIT MODE, NO CRC

SEQ 0166

```

10202 034522 004737 004264 JSR PC,WRITAX ;LOAD AX3
10203 034526 004737 011046 JSR PC,RXCHAR ;RCV 7-BIT DATA CHAR, CLK 6-BIT
10204 034532 000006 6
10205 034534 012737 000004 002376 MOV #RXLEN2,WAX16 ;SET RCV LEN = 4
10206 034542 004737 004264 JSR PC,WRITAX ;LOAD AX3
10207 034546 004737 011046 JSR PC,RXCHAR ;RCV 6-BIT DATA CHAR, CLK 5-BIT
10208 034552 000005 5
10209 034554 012737 000003 002376 MOV #RXLEN1!RXLENO,WAX16 ;SET RCV LEN = 3
10210 034562 004737 004264 JSR PC,WRITAX ;LOAD AX3
10211 034566 004737 011046 JSR PC,RXCHAR ;RCV 5-BIT DATA CHAR, CLR 4-BIT
10212 034572 000004 4
10213 034574 012737 000002 002376 MOV #RXLEN1,WAX16 ;SET RCV LEN = 2
10214 034602 004737 004264 JSR PC,WRITAX ;LOAD AX3
10215 034606 004737 011046 JSR PC,RXCHAR ;RCV 4-BIT DATA CHAR, CLK 3-BIT
10216 034612 000003 3
10217 034614 012737 000001 002376 MOV #RXLENO,WAX16 ;SET RCV LEN = 1
10218 034622 004737 004264 JSR PC,WRITAX ;LOAD AX3
10219 034626 004737 011046 JSR PC,RXCHAR ;RCV 3-BIT DATA CHAR, CLK 2-BIT
10220 034632 000002 2
10221 034634 004737 011046 JSR PC,RXCHAR ;RCV 2-BIT DATA CHAR, CLK 1-BIT
10222 034640 000001 1
10223 034642 004737 003576 24$: JSR PC,MSTCLR ;ISSUE MASTER CLEAR TO CLEAN UP
10224 034646 ENDTST
(3) 034646 L10072: TRAP C$ETST
(3) 034646 104401

```

10225
10226
10227
10228
10229
10230
10231
10232
10233
10234
10235
10236
10237
10238

```

:*****
:SBTTL TEST 39 - TRANSMITTER UNDERRUN ERROR, IDLE MARKING, CHAR MODE,NO CRC
:*
:* THE LINE UNIT IS PLACED IN CHAR MODE, AND THE IDLE BIT IS SET. THEN, A
:* MSG IS INITIATED, A 000 CHAR IS SENT, AND THE TX BUFFER IS NOT SERVICED
:* IN RESPONSE TO THE USYRT TX BUFFER EMPTY FLAG, WHICH CAUSES A TX UNDERRUN
:* ERROR. THEN, THE RECEIVER IS CLOCKED AND CHECKED FOR TWO 377 CHARS TO BE
:* RECEIVED (LINE MARKING) BEFORE SHUTTING DOWN WITH A MASTER CLEAR.
:*****
BGNTST

```

```

10239 034650 T39::
(3) 034650 MOV #24$,RETADR ;SET TEST EXIT ADDRESS FOR ERRORS
10240 034650 012737 034732 002362 JSR PC,INITRN ;MST CLR, LOAD 2 SOM'S
10241 034656 004737 005512 SYNCH
10242 034662 000226 CRC2!CRC1!IDLE!DDCMP
10243 034664 000341
10244 034666 012737 000000 002422 MOV #000,TXWORD
10245 034674 004737 005146 JSR PC,LDTXSI ;LOAD 000 CHAR INTO TX SILO
10246 034700 004737 005226 JSR PC,STPLU ;CLOCK LINE UNIT UNTIL LINE GOES MARKING
10247 034704 000063 51.
10248 034706 004737 007326 JSR PC,RDRXSI ;READ 000 CHAR
10249 034712 004737 007722 JSR PC,CKDATA ;READ AND CHECK FOR MARK CHAR (377)
10250 034716 000377 377
10251 034720 000000 000
10252 034722 004737 007722 JSR PC,CKDATA ;READ AND CHECK FOR ANOTHER MARK CHAR
10253 034726 000377 377
10254 034730 000000 000

```

CZDMRA M8203 STATIC DIAG #1
CZDMRA.P11 18-JUL-79 09:44

MACY11 30A(1052) 18-JUL-79 09:53 PAGE 7-127
TEST 39 - TRANSMITTER UNDERRUN ERROR, IDLE MARKING, CHAR MODE,NO CRC

SEQ 0167

10255 034732 004737 003576
10256 034736
(3) 034736
(3) 034736 104401

24\$: JSR PC,MSTCLR ;ISSUE MASTER CLEAR TO CLEAN UP
ENDTST

L10073: TRAP C\$ETST

10257
10258
10259
10260
10261
10262
10263
10264
10265
10266
10267
10268
10269
10270
10271
10272
10273
10274
10275
10276

:SBTTL TEST 40 - MSG TERMINATION WITH GA CHARS - BIT MODE,NO CRC
:
:* THE DEVICE IS ENABLED FOR TRANSMIT AND RECEIVE, AND A MESSAGE IS
:* INITIATED IN BIT MODE.
:* 2 FLAG CHARACTERS ARE SENT, FOLLOWED BY
:* THE FOLLOWING DATA CHARACTERS : 000, 125, 252, 377, 000. THEN THE LOOP
:* MODE BIT (STRIP) IS SET AND 2 TERMINATING GO-AHEAD CHARACTERS ARE
:* SENT. EACH USYRT RCV FLAG IS TIMED AS IT IS RECEIVED, AND THE 5 DATA
:* WORDS ARE READ AND COMPARED TO EXPECTED VALUES.
:* ALSO, THE FIRST GA CHAR IS CHECKED BY SCANNING THE TXDATA BIT AS THE GA
:* IS BEING TRANSMITTED (GA = 376 OCTAL).
:* THE TEST ALSO CHECKS FOR SETTING OF RAB AND EBLK
:* IN LOOP MODE.
:*****

10277 034740
(3) 034740
10278 034740 012737 035074 002362
10279 034746 004737 005512
10280 034752 000000
10281 034754 000310
10282 034756 004737 010770
10283 034762 003224
10284 034764 000005
10285 034766 012737 005000 002422
10286 034774 004737 005146
10287 035000 004737 005146
10288 035004 004737 005120
10289 035010 004737 005226
10290 035014 100071
10291 035016 004737 011150
10292 035022 000376
10293 035024 004737 007722
10294 035030 000000
10295 035032 000000
10296 035034 004737 007722
10297 035040 000125
10298 035042 000000
10299 035044 004737 007722
10300 035050 000252
10301 035052 000000
10302 035054 004737 007722
10303 035060 000377
10304 035062 000010
10305 035064 004737 007722
10306 035070 003000
10307 035072 000010

BGNTST
T40::
MOV #24\$,RETADR ;SET TEST EXIT ADRS FOR ERRORS
JSR PC,INTRN ;MST CLR, LOAD 2 SOM'S
000
CRC2!CRC1!STRIP
JSR PC,LODMMSG ;LOAD DATA CHARS INTO TX SILO
MSG1+4
5
MOV #TXGOA!TXEOM,TXWORD
JSR PC,LDTXSI ;LOAD A GA CHAR INTO TX SILO
JSR PC,LDTXSI ;LOAD ANOTHER GA
JSR PC,WAIT50 ;ALLOW SILO TO RIPPLE
JSR PC,STPLU ;CLOCK LU UNTIL GA CHAR IS TX'ING
CHPCHK!57.
JSR PC,CKTBIT ;SCAN TXDATA BIT FOR GA CHAR
376
JSR PC,CKDATA ;RCV 000 CHAR, CLK 125
000
0
JSR PC,CKDATA ;RCV 125 CHAR, CLK 252
125
0
JSR PC,CKDATA ;RCV 252 CHAR, CLK 377
252
0
JSR PC,CKDATA ;RCV 377 CHAR
377
8.
JSR PC,CKDATA ;RCV 000 CHAR, CHK RAB - 1, EBLK = 1
3000
8.

```

10308 035074 004737 003576      24$: JSR      PC,MSTCLR      ;ISSUE MASTER CLEAR
10309 035100      ENDTST
      (3) 035100
      (3) 035100 104401      L10074: TRAP C$ETST
10310
10311
10312
10313
10314
10315

```

```

10316 :*****
10317 :SBITL      TEST 41 - IDLE SYNCHS TEST - CHAR MODE
10318 :*
10319 :* THE DEVICE IS ENABLED FOR TRANSMIT AND RECEIVE, AND A MESSAGE IS
10320 :* INITIATED IN CHAR MODE. 24(DEC) SYNCHS ARE SENT.
10321 :* EACH SYNCH IS TIMED AS IT IS RECEIVED, AND THE BITS ARE CHECKED
10322 :* FOR A VALID SYNCH CHAR FOR EACH OF THE 22 SYNCHS WHICH FOLLOW
10323 :* THE FIRST TWO (THESE PERFORM SYNCHRONIZATION, AND ARE NOT READ).
10324 :* WHILE THE LAST SYNCH IS BEING TRANSMITTED, OC IS SET, AND THE
10325 :* NEXT CHAR RCV'D AFTER THE SYNCH IS CHECKED TO BE 377 (LINE MARKING).
10326 :* THEN, A MASTER CLEAR IS ISSUED.
10327 :* THE SYNCH CHAR USED IS 226 (OCTAL).
10328 :*****

```

```

10328 035102      BGNST
      (3) 035102
10329 035102 012737 035242 002362      MOV      #24$,RETADR      ;SET TEST EXIT ADRS FOR ERRORS
10330 035110 004737 005512      JSR      PC,INITRN      ;MST CLR, LOAD 2 SOM'S
10331 035114 000226      SYNCH
10332 035116 000341      CRC2!CRC1!IDLE!DDCMP
10333 035120 012701 000026      MOV      #22.,R1      ;INIT COUNTER
10334 035124 012737 000226 002422      MOV      #SYNCH,TXWORD
10335 035132 004737 005146      6$: JSR      PC,LDTXSI      ;LOAD AN SOM INTO TX SILO
10336 035136 005301      DEC      R1      ;DECR COUNTER
10337 035140 001374      BNE     6$      ;BR IF MORE TO LOAD
10338 035142 004737 005120      JSR      PC,WAIT50      ;ALLOW SILO TO RIPPLE
10339 035146 004737 007406      JSR      PC,RCV1ST      ;CLK LU UNTIL 3RD SYNCH RCV'D (RCVR IS ACTIVE)
10340 035152 000030      24.
10341 035154 012701 000025      MOV      #21.,R1      ;INIT COUNTER
10342 035160 004737 007722      8$: JSR      PC,CKDATA      ;READ A SYNCH, CLK NEXT ONE
10343 035164 000226      SYNCH
10344 035166 000010      8.
10345 035170 005301      DEC      R1      ;DECR COUNTER
10346 035172 001372      BNE     8$      ;BR IF NOT ALL CHECKED
10347 035174 004737 007722      JSR      PC,CKDATA      ;CHECK LAST SYNCH
10348 035200 000226      SYNCH
10349 035202 000004      4
10350 035204 012737 000011 002400      MOV      #11,REGNUM      ;SET REG NO. = 11
10351 035212 112737 000200 002366      MOVB     #OC,WRIBYT
10352 035220 004737 003722      JSR      PC,WRITLU      ;SET OC IN REG 11
10353 035224 004737 005226      JSR      PC,STPLU      ;FINISH CLOCKING CHAR, AND THEN SOME
10354 035230 000014      12.
10355 035232 004737 007722      JSR      PC,CKDATA      ;RCV A MARK CHAR, CHK IT
10356 035236 000377      377
10357 035240 000000      0
10358 035242 004737 003576      24$: JSR      PC,MSTCLR      ;ISSUE MASTER CLEAR TO CLEAN UP
10359 035246      ENDTST
      (3) 035246

```

L10075:

(3) 035246 104401

TRAP CSETST

10360
10361
10362
10363
10364
10365
10366
10367
10368
10369
10370
10371
10372
10373
10374
10375
10376
10377
10378
10379
10380
10381

```
*****~*****  
:SBTTL TEST 42 - STRIP SYNCH TEST  
:  
:* THE DEVICE IS ENABLED FOR TRANSMIT AND RECEIVE, AND A MESSAGE IS  
:* INITIATED IN CHAR MODE AND WITH THE STRIP SYNCH  
:* BIT SET. THEN 24 (DEC) SYNCHS ARE SENT  
:* FOLLOWED BY THE FOLLOWING DATA CHARACTERS : 377, 000, 125, 252,  
:* AND 2 TERMINATING SYNCHS.  
:* EACH OF THE 23 SYNCHS AFTER THE FIRST ARE CHECKED AT THE TRANSMITTER OUTPUT,  
:* BY SCANNING THE TXDATA BIT.  
:* EACH USYRT RCV FLAG IS TIMED AS IT IS RECEIVED, AND THE 4 DATA WORDS  
:* ARE READ AND COMPARED TO EXPECTED VALUES.  
:* FINALLY, THE LINE UNIT IS CLOCKED FOR SEVERAL CHAR TIMES, AND A CHECK  
:* IS MADE FOR OACT = 0 (TEOM SHOULD CAUSE TX ENABLE TO DROP).  
:* THE ABOVE TEST IS REPEATED FOR EACH OF THE FOLLOWING SYNCH CHAR DATA  
:* PATTERNS : 226,000,125,252,376,177.  
:*****~*****
```

10382 035250

BGNTST

(3) 035250

10383 035250 012737 035470 002362
10384 035256 012701 035476
10385 035262 011137 035272
10386 035266 004737 005512
10387 035272 000000
10388 035274 000311
10389 035276 012737 000400 002422
10390 035304 012702 000026
10391 035310 004737 005146
10392 035314 005302
10393 035316 001374
10394 035320 004737 010770
10395 035324 003246
10396 035326 000006
10397 035330 012737 001000 002422
10398 035336 004737 005146
10399 035342 004737 005146
10400 035346 004737 005120
10401 035352 011137 035374
10402 035356 012702 000027
10403 035362 004737 005226
10404 035366 100010
10405 035370 004737 011150
10406 035374 000000
10407 035376 005302
10408 035400 001373
10409 035402 004737 007406
10410 035406 000010
10411 035410 004737 007722
10412 035414 000377
10413 035416 000010

```
MOV #24$,RETADR ;SET TEST EXIT ADRS FOR ERRORS  
MOV #SYNPAT,R1 ;GET POINTER TO DATA  
2$: MOV (R1),3$ ;GET A SYNCH PATTERN  
JSR PC,INITRN ;MST CLR, LOAD 2 SOM'S  
3$: .WORD 0 ;SYNCH PATTERN GOES HERE  
CRC2!CRC1!STRIP!DDCMP  
MOV #TXSOM, TXWORD  
6$: MOV #22.,R2 ;LOAD 22 SOM'S INTO TX SILO  
JSR PC,LDTXSI  
DEC R2  
BNE 6$  
JSR PC,LODMSG ;LOAD DATA CHARS INTO TX SILO  
MSG4  
6  
MOV #TXEOM, TXWORD  
JSR PC,LDTXSI ;LOAD A TEOM  
JSR PC,LDTXSI ;LOAD ANOTHER TEOM  
JSR PC,WAIT50 ;ALLOW SILO TO RIPPLE  
MOV (R1),16$ ;GET CURRENT SYNCH PATTERN  
MOV #23.,R2 ;INIT COUNTER  
JSR PC,STPLU ;CLOCK OUT FIRST SYNCH  
CHPCHK!8.  
14$: JSR PC,CKTBIT ;CHECK TX'D SYNCH  
16$: .WORD 0 ;SYNCH PATTERN GOES HERE  
DEC R2 ;DECR COUNTER  
BNE 14$ ;BR IF NOT DONE CHECKING LAST 23 SYNCHS  
JSR PC,RCV1ST ;CLOCK UNTIL 000 CHAR RCV'D  
8.  
JSR PC,CKDATA ;RCV 377, CLOCK 000  
377  
8.
```

| | | | | | | |
|-------|--------|--------|--------|-----|----------------|----------------------------------|
| 10414 | 035420 | 004737 | 007722 | JSR | PC,CKDATA | ;RCV 000, CLK 125 |
| 10415 | 035424 | 000000 | | 000 | | |
| 10416 | 035426 | 000010 | | 8. | | |
| 10417 | 035430 | 004737 | 007722 | JSR | PC,CKDATA | ;RCV 125, CLK 252 |
| 10418 | 035434 | 000125 | | 125 | | |
| 10419 | 035436 | 000010 | | 8. | | |
| 10420 | 035440 | 004737 | 007722 | JSR | PC,CKDATA | ;RCV 252, CLK END OF MSG |
| 10421 | 035444 | 000252 | | 252 | | |
| 10422 | 035446 | 000040 | | 32. | | |
| 10423 | 035450 | 004737 | 005324 | JSR | PC,OACTIV | ;CHK FOR OACT = 0 |
| 10424 | 035454 | 000000 | | 0 | | |
| 10425 | 035456 | 062701 | 000002 | ADD | #2,R1 | ;INIT SYNCH PATTERN POINTER |
| 10426 | 035462 | 020127 | 035512 | CMP | R1,#SYNPAT+12. | ;SEE IF ALL PATTERNS CHECKED YET |
| 10427 | 035466 | 103675 | | BLO | 2\$ | ;BR IF NOT YET |
| 10428 | 035470 | 004737 | 003576 | JSR | PC,MSTCLR | ;ISSUE MASTER CLEAR TO CLEAN UP |
| 10429 | 035474 | | | | | |
| (3) | 035474 | | | | | |
| (3) | 035474 | 104401 | | | | L10076: TRAP C\$ETST |
| 10430 | | | | | | |
| 10431 | 035476 | 000226 | | | | |
| 10432 | 035500 | 000000 | | | | |
| 10433 | 035502 | 000125 | | | | |
| 10434 | 035504 | 000252 | | | | |
| 10435 | 035506 | 000376 | | | | |
| 10436 | 035510 | 000177 | | | | |
| 10437 | | | | | | |
| 10438 | | | | | | |
| 10439 | | | | | | |
| 10440 | | | | | | |
| 10441 | | | | | | |

24\$:
ENDTST

SYNPAT: 226
000
125
252
376
177

```

10443         .SBTTL  HARDWARE PARAMETER CODING SECTION
10444
10445
10446
10447
10448   ://////////////////
10449   :/ THE HARDWARE PARAMETER CODING SECTION CONTAINS MACROS
10450   :/ THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES. THE
10451   :/ MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
10452   :/ INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES. THE
10453   :/ MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
10454   :/ WITH THE OPERATOR.
10455   ://////////////////
10456 035512           BGNHRD
10457 (3) 035512 000022
10458 (3) 035514
10459           .WORD L10077-L$HARD/2
10460           L$HARD::
10461
10462 035514           GPRMA  ADDRES,2,0,160000,,7776,YES
10463 (4) 035514 001031
10464 (4) 035516 035560
10465 (4) 035520 160000
10466 (4) 035522 177776
10467           .WORD  T$CODE
10468           .WORD  ADDRES
10469           .WORD  T$LLOLIM
10470           .WORD  T$HILIM
10471
10472 035524           GPRMA  VECTOR,4,0,0,674,YES
10473 (4) 035524 002031
10474 (4) 035526 035606
10475 (4) 035530 000000
10476 (4) 035532 000674
10477           .WORD  T$CODE
10478           .WORD  VECTOR
10479           .WORD  T$LLOLIM
10480           .WORD  T$HILIM
10481
10482 035534           GPRMD  PRIRTY,6,0,7000,4,7,YES
10483 (4) 035534 003032
10484 (4) 035536 035637
10485 (4) 035540 007000
10486 (4) 035542 000004
10487 (4) 035544 000007
10488           .WORD  T$CODE
10489           .WORD  PRIRTY
10490           .WORD  7000
10491           .WORD  T$LLOLIM
10492           .WORD  T$HILIM
10493
10494 035546           GPRMD  ISRUN,24,0,7,0,7,YES
10495 (4) 035546 012032
10496 (4) 035550 035670
10497 (4) 035552 000007
10498 (4) 035554 000000
10499 (4) 035556 000007
10500           .WORD  T$CODE
10501           .WORD  ISRUN
10502           .WORD  7
10503           .WORD  T$LLOLIM
10504           .WORD  T$HILIM
10505
10506 035560           ENDHRD
10507 (2)
10508 (3) 035560
10509           .EVEN
10510 L10077:
10511
10512 035560 042504 044526 042503  ADDRESS: .ASCIZ /DEVICE CSR ADDRESS : /
10513 035566 041440 051123 040440
10514 035574 042104 042522 051523
10515 035602 035040 000040
10516 035606 042504 044526 042503  VECTOR: .ASCIZ /DEVICE VECTOR ADDRESS : /
10517 035614 053040 041505 047524
10518 035622 020122 042101 051104
10519 035630 051505 020123 020072
10520 035636 000
10521 035637 004 053105 041511  PRIRTY: .ASCIZ /DEVICE PRIORITY LEVEL : /
10522 035644 020105 051120 047511
10523 035652 044522 054524 046040
  
```


10477
10478
10479
10480
10481
10482
10483
10484
10485
10486
10487
10488
10489
(3)
(3)
10490
10491
10492
(2)
(3)
10493
10494
10495
10496
10497
10498
10499
10500
10501
10502
10503
10504
10505
10506
10507
10508
10509
10510
10511
10512
10513
10514
10515
(2)
(4)
(4)
(3)
10516
10517

035756
035756 000000
035760
035760
035760
035760
035760
036160 036160
036160 000240
036162 000240
036164 000240
036166
036166
036166 000000
036170 000000
036172
000001

```
.SBTTL SOFTWARE PARAMETER CODING SECTION

://////
:/ THE SOFTWARE PARAMETER CODING SECTION CONTAINS MACROS
:/ THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES. THE
:/ MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
:/ INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES. THE
:/ MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
:/ WITH THE OPERATOR.
://////

          BGNSFT
          .WORD L10100-L$SOFT/2
L$SOFT::

          ENDSFT
          .EVEN
L10100:

.EVEN

:***** PATCH AREA FOR DEBUG *****
PATCH:
          .=.+200
          NOP
          NOP
          NOP
:*****

          ENDMOD
          LASTAD
          .EVEN
          .WORD 0
          .WORD 0
L$LAST::

.END
```


| | | | | | | | | | | | | | | |
|------------------|-------|--------|--------|-------|-------|-------|-------|-------|-------|-------|--------|-------|-------|--|
| ABORT = 000004 | 5590# | | | | | | | | | | | | | |
| ADDRS = 035560 | 7625 | 7631 | 7646 | 7666 | 7684 | 7705 | 7723 | 7742 | 7761 | 10458 | 10465# | | | |
| ADR = 000020 G | 5549# | | | | | | | | | | | | | |
| ANBITS = 002561 | 5987# | 8654 | 8655 | 8705 | 8706 | 8752 | 8753 | | | | | | | |
| APA = 000200 | 5768# | | | | | | | | | | | | | |
| ASBC0 = 000020 | 5725# | | | | | | | | | | | | | |
| ASBC1 = 000040 | 5724# | | | | | | | | | | | | | |
| ASBC2 = 000100 | 5723# | | | | | | | | | | | | | |
| ASSEMB = 000010 | 5406 | | | | | | | | | | | | | |
| AXNUM = 002402 | 5908# | 6553 | 6591 | 6604 | 6629 | 6631 | 6637* | 6643* | 6644 | 6646* | 6648 | 6839 | 6847* | |
| | 6890* | 7086 | 7092* | 7121* | 7127* | 7368 | 7370* | 7378* | 6644 | 6646* | 6648 | 6839 | 6847* | |
| | 8549* | 8560* | 8588* | 8595* | 8597 | 8609* | 8651* | 8666* | 6644 | 6646* | 6648 | 6839 | 6847* | |
| | 8826* | 8863* | 8943* | 8954* | 9059* | 9365* | 9431* | 9541* | 6644 | 6646* | 6648 | 6839 | 6847* | |
| | 9951* | 10133* | 10185* | | | | | | 6644 | 6646* | 6648 | 6839 | 6847* | |
| AX0.15 = 002322 | 5836# | 6636 | 7655 | 7674 | 7694 | 7713 | 7731 | 7751 | | | | | | |
| AX0.16 = 002324 | 5837# | 7655 | 7674 | 7694 | 7713 | 7731 | 7751 | | | | | | | |
| AX1 = 000001 | 5621# | 5693# | | | | | | | | | | | | |
| AX1.15 = 002326 | 5838# | 7655 | 7674 | 7694 | 7713 | 7731 | 7751 | | | | | | | |
| AX1.16 = 002330 | 5839# | 7655 | 7674 | 7694 | 7713 | 7731 | 7751 | | | | | | | |
| AX2 = 000002 | 5620# | 5692# | | | | | | | | | | | | |
| AX2.15 = 002332 | 5840# | 7657 | 7676 | 7696 | 7715 | 7733 | 7753 | | | | | | | |
| AX2.16 = 002334 | 5841# | 7657 | 7676 | 7696 | 7715 | 7733 | 7753 | | | | | | | |
| AX3.15 = 002336 | 5842# | 7657 | 7676 | 7696 | 7715 | 7733 | 7753 | | | | | | | |
| AX3.16 = 002340 | 5843# | 7657 | 7676 | 7696 | 7715 | 7733 | 7753 | | | | | | | |
| AX315U = 000332 | 5786# | 8539 | 8599 | 8816 | | | | | | | | | | |
| A1 = 024166 | 8427 | 8453 | 8472 | 8481 | 8491# | | | | | | | | | |
| A10 = 034142 | 10028 | 10103# | | | | | | | | | | | | |
| A2 = 026566 | 8928 | 8967# | | | | | | | | | | | | |
| A3 = 027246 | 8992 | 9096 | 9100# | | | | | | | | | | | |
| A4 = 027362 | 9122 | 9148# | | | | | | | | | | | | |
| A5 = 027472 | 9171 | 9194# | | | | | | | | | | | | |
| A6 = 027666 | 9219 | 9254# | | | | | | | | | | | | |
| A7 = 030232 | 9280 | 9296 | 9305 | 9317 | 9336 | 9340 | 9344# | | | | | | | |
| A8 = 030446 | 9361 | 9400# | | | | | | | | | | | | |
| A9 = 030776 | 9418 | 9480# | | | | | | | | | | | | |
| BAD DAT = 002406 | 5910# | 7258* | 7259* | 7280* | 7281* | 7634 | 7649 | 7688 | 7745 | 8018* | 8052* | 8083* | 8115* | |
| | 8151* | 8193* | 8245* | 8281* | 8315* | 8349* | 8383* | 8447* | 8466* | 8485* | 8544* | 8555* | 8604* | |
| | 8614* | 8661* | 8671* | 8712* | 8721* | 8759* | 8768* | 8821* | 8831* | 8879* | 8902* | 8949* | 8959* | |
| | 9064* | 9087* | 9563* | 9578* | 9661* | 9726* | 9807* | 9851* | 9955* | | | | | |
| BCC = 000001 | 5669# | 7284 | 7286 | 7292 | | | | | | | | | | |
| BCCCHK = 100000 | 5851# | 7282 | | | | | | | | | | | | |
| BIT0 = 000001 G | 5549# | 5571 | 5583 | 5592 | 5621 | 5633 | 5645 | 5657 | 5669 | 5681 | 5693 | 5705 | 5717 | |
| | 5729 | 5741 | 5750 | 5762 | 5775 | 5785 | 5796 | 5868 | 6631 | 6672 | 6801 | 7013 | 7054 | |
| | 7094 | 7462 | 7853 | | | | | | | | | | | |
| BIT00 = 000001 G | 5549# | | | | | | | | | | | | | |
| BIT01 = 000002 G | 5549# | | | | | | | | | | | | | |
| BIT02 = 000004 G | 5549# | | | | | | | | | | | | | |
| BIT03 = 000010 G | 5549# | | | | | | | | | | | | | |
| BIT04 = 000020 G | 5549# | | | | | | | | | | | | | |
| BIT05 = 000040 G | 5549# | | | | | | | | | | | | | |
| BIT06 = 000100 G | 5549# | | | | | | | | | | | | | |
| BIT07 = 000200 G | 5549# | | | | | | | | | | | | | |
| BIT08 = 000400 G | 5549# | | | | | | | | | | | | | |
| BIT09 = 001000 G | 5549# | | | | | | | | | | | | | |
| BIT1 = 000002 G | 5549# | 5570 | 5582 | 5591 | 5610 | 5620 | 5632 | 5644 | 5656 | 5668 | 5680 | 5692 | 5704 | |
| | 5716 | 5728 | 5740 | 5749 | 5761 | 5774 | 5784 | 5795 | 5869 | 6688 | 6997 | 7108 | | |

| | | | | | | | | | | | | | | |
|------------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|--|
| BIT10 = 002000 G | 5549# | 5806 | 5818 | | | | | | | | | | | |
| BIT11 = 004000 G | 5549# | 5805 | 5817 | | | | | | | | | | | |
| BIT12 = 010000 G | 5549# | | | | | | | | | | | | | |
| BIT13 = 020000 G | 5549# | | | | | | | | | | | | | |
| BIT14 = 040000 G | 5549# | | | | | | | | | | | | | |
| BIT15 = 100000 G | 5549# | 5849 | 5851 | 5852 | 6769 | 6923 | | | | | | | | |
| BIT2 = 000004 G | 5549# | 5569 | 5581 | 5590 | 5609 | 5619 | 5631 | 5643 | 5655 | 5667 | 5679 | 5691 | 5703 | |
| | 5715 | 5727 | 5739 | 5748 | 5760 | 5773 | 5794 | | | | | | | |
| BIT3 = 000010 G | 5549# | 5568 | 5580 | 5589 | 5608 | 5618 | 5630 | 5642 | 5654 | 5666 | 5678 | 5690 | 5702 | |
| | 5714 | 5726 | 5738 | 5747 | 5759 | 5772 | 5783 | | | | | | | |
| BIT4 = 000020 G | 5549# | 5567 | 5579 | 5607 | 5617 | 5629 | 5641 | 5653 | 5665 | 5677 | 5689 | 5701 | 5713 | |
| | 5725 | 5737 | 5758 | 5771 | 5782 | | | | | | | | | |
| BIT5 = 000040 G | 5549# | 5578 | 5599 | 5606 | 5616 | 5628 | 5640 | 5652 | 5664 | 5676 | 5688 | 5700 | 5712 | |
| | 5724 | 5736 | 5757 | 5770 | 5793 | | | | | | | | | |
| BIT6 = 000100 G | 5549# | 5566 | 5577 | 5598 | 5605 | 5615 | 5627 | 5639 | 5651 | 5663 | 5675 | 5687 | 5699 | |
| | 5711 | 5723 | 5735 | 5756 | 5769 | 5781 | 5792 | | | | | | | |
| BIT7 = 000200 G | 5549# | 5565 | 5576 | 5588 | 5597 | 5604 | 5626 | 5638 | 5650 | 5662 | 5674 | 5686 | 5698 | |
| | 5710 | 5722 | 5734 | 5746 | 5755 | 5768 | 5780 | 5791 | 7252 | 7253 | | | | |
| BIT8 = 000400 G | 5549# | 5808 | 5820 | | | | | | | | | | | |
| BIT9 = 001000 G | 5549# | 5807 | 5819 | | | | | | | | | | | |
| BOE = 000400 G | 5549# | | | | | | | | | | | | | |
| B POLL = 000100 | 5598# | | | | | | | | | | | | | |
| BSEL1 = 002450 | 5929# | 6422* | 6424* | 6425* | 6438* | 6439* | 6444* | 6532* | 6534* | 6635* | 6773* | 6774* | 6776* | |
| | 6866* | 6867* | 6872* | 6885* | 7868* | 7869* | 7870 | 8105* | 8804* | | | | | |
| BSEL2 = 002452 | 5930# | 7870* | 7871* | 8440 | 8460 | 8479 | | | | | | | | |
| BSEL4 = 002454 | 5931# | 6470 | 6491* | 8143* | | | | | | | | | | |
| CARR = 000001 | 5681# | | | | | | | | | | | | | |
| CHPCHK = 100000 | 5849# | 9128 | 9177 | 9228 | 9286 | 9327 | 9371 | 9424 | 9508 | 10138 | 10290 | 10404 | | |
| CHPTYP = 002430 | 5919# | 6770 | 6880* | 6884* | 6924 | 7822* | 9912 | 9934 | | | | | | |
| CKDATA = 007722 | 7243# | 10054 | 10083 | 10249 | 10252 | 10293 | 10296 | 10299 | 10302 | 10305 | 10342 | 10347 | 10355 | |
| | 10411 | 10414 | 10417 | 10420 | | | | | | | | | | |
| CKTBIT = 011150 | 7458# | 9510 | 10291 | 10405 | | | | | | | | | | |
| CRCCHK = 100000 | 5852# | 7513 | | | | | | | | | | | | |
| CRCTY0 = 000001 | 5775# | | | | | | | | | | | | | |
| CRCTY1 = 000002 | 5774# | | | | | | | | | | | | | |
| CRCTY2 = 000004 | 5773# | | | | | | | | | | | | | |
| CRC1 = 000100 | 5627# | 9222 | 9700 | 9763 | 9882 | 10126 | 10178 | 10243 | 10281 | 10332 | 10388 | | | |
| CRC2 = 000200 | 5626# | 9222 | 9700 | 9763 | 9882 | 10126 | 10178 | 10243 | 10281 | 10332 | 10388 | | | |
| CS = 000004 | 5679# | | | | | | | | | | | | | |
| C\$AU = 000052 | 5406# | 7964 | | | | | | | | | | | | |
| C\$AUTO = 000061 | 5406# | 7909 | | | | | | | | | | | | |
| C\$BRK = 000022 | 5406# | | | | | | | | | | | | | |
| C\$BSEG = 000004 | 5406# | 8043 | 8273 | 8307 | 8341 | 8375 | 8650 | 8701 | 8748 | 8808 | 8869 | | | |
| C\$BSUB = 000002 | 5406# | 8867 | 8893 | 8929 | | | | | | | | | | |
| C\$CEFG = 000045 | 5406# | | | | | | | | | | | | | |
| C\$CLCK = 000062 | 5406# | | | | | | | | | | | | | |
| C\$CLEA = 000012 | 5406# | 7926 | | | | | | | | | | | | |
| C\$CLOS = 000035 | 5406# | | | | | | | | | | | | | |
| C\$CLP1 = 000006 | 5406# | | | | | | | | | | | | | |
| C\$CVEC = 000036 | 5406# | | | | | | | | | | | | | |
| C\$DCLN = 000044 | 5406# | | | | | | | | | | | | | |
| C\$DODU = 000051 | 5406# | 7906 | | | | | | | | | | | | |
| C\$DRPT = 000024 | 5406# | | | | | | | | | | | | | |
| C\$DU = 000053 | 5406# | 7945 | | | | | | | | | | | | |
| C\$EDIT = 000000 | 5406# | 5446 | | | | | | | | | | | | |
| C\$ERDF = 000055 | 5406# | 6678 | 6684 | 6694 | 6700 | 6807 | 6813 | 6971 | 7003 | 7009 | 7019 | 7025 | 7060 | |

CZDMRA M8203 STATIC DIAG #1
CZDMRA.P11 18-JUL-79 09:44

MACY11 30A(1052) 18-JUL-79 09:53 PAGE 8-4
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0178

| | | | | | | | | | | | | | | | | | | |
|----------|----------|-------|-------|-------|-------|-------|-------|------|------|------|-------|------|------|------|--|--|--|--|
| EM41 | 013450 | 7335 | 7589# | | | | | | | | | | | | | | | |
| EM42 | 013471 | 7341 | 7590# | | | | | | | | | | | | | | | |
| EM43 | 013506 | 7591# | | | | | | | | | | | | | | | | |
| EM44 | 013533 | 7592# | | | | | | | | | | | | | | | | |
| EM45 | 013560 | 7593# | | | | | | | | | | | | | | | | |
| EM46 | 013605 | 7594# | | | | | | | | | | | | | | | | |
| EM47 | 013640 | 7595# | | | | | | | | | | | | | | | | |
| EM48 | 013673 | 7596# | | | | | | | | | | | | | | | | |
| EM49 | 013726 | 7597# | | | | | | | | | | | | | | | | |
| EM5 | 012266 | 7553# | 8450 | 8488 | | | | | | | | | | | | | | |
| EM50 | 013765 | 7598# | | | | | | | | | | | | | | | | |
| EM51 | 014021 | 7599# | | | | | | | | | | | | | | | | |
| EM52 | 014061 | 7600# | 8521 | | | | | | | | | | | | | | | |
| EM53 | 014122 | 7601# | 8535 | | | | | | | | | | | | | | | |
| EM54 | 014162 | 7266 | 7602# | | | | | | | | | | | | | | | |
| EM6 | 012317 | 7554# | 8469 | | | | | | | | | | | | | | | |
| EM60 | 014204 | 7603# | 9252 | | | | | | | | | | | | | | | |
| EM65 | 014220 | 6971 | 7604# | | | | | | | | | | | | | | | |
| EM7 | 012350 | 6678 | 7555# | | | | | | | | | | | | | | | |
| EM8 | 012365 | 6684 | 7556# | | | | | | | | | | | | | | | |
| EM9 | 012406 | 6694 | 7557# | | | | | | | | | | | | | | | |
| ENAX = | 000004 | 5619# | 5691# | 6555 | 6606 | | | | | | | | | | | | | |
| ENDIT | 021440 | 7844 | 7882# | | | | | | | | | | | | | | | |
| ENDPAT | 003220 | 6327# | | | | | | | | | | | | | | | | |
| ENDTRN | 006246 | 6953# | 9147 | 9193 | 9253 | 9399 | 9479 | | | | | | | | | | | |
| EOM = | 000002 | 5591# | | | | | | | | | | | | | | | | |
| ERRFLG | 002356 | 5898# | | | | | | | | | | | | | | | | |
| ERROR1 | 002420 | 5915# | 6551* | 6564* | 6589* | 6615* | 7823* | 8516 | 8530 | | | | | | | | | |
| ERR1 | 014554 G | 7624# | 7995 | | | | | | | | | | | | | | | |
| ERR2 | 014606 G | 7630# | 8021 | 8055 | 8086 | 8118 | 8154 | 8196 | 8248 | 8284 | 8318 | 8352 | 8386 | 8450 | | | | |
| | | 8469 | 8488 | | | | | | | | | | | | | | | |
| ERR3 | 015114 G | 7645# | 8547 | 8558 | 8607 | 8617 | 8664 | 8674 | 8715 | 8724 | 8762 | 8771 | 8824 | 8834 | | | | |
| | | 8905 | 8952 | 8962 | 9067 | 9090 | 9566 | 9581 | 9664 | 9729 | 9810 | 9854 | 9958 | | | | | |
| ERR4 | 015576 G | 6678 | 6684 | 6694 | 6700 | 6807 | 6813 | 6971 | 7003 | 7009 | 7019 | 7025 | 7060 | 7066 | | | | |
| | | 7266 | 7473 | 7479 | 7664# | | | | | | | | | | | | | |
| ERR5 | 016254 G | 7683# | 8882 | | | | | | | | | | | | | | | |
| ERR6 | 016766 G | 7100 | 7106 | 7114 | 7120 | 7703# | | | | | | | | | | | | |
| ERR7 | 017444 G | 7722# | 9252 | 9295 | 9304 | 9316 | 9335 | 9343 | | | | | | | | | | |
| ERR8 | 020076 G | 7271 | 7290 | 7296 | 7305 | 7311 | 7320 | 7326 | 7335 | 7341 | 7740# | | | | | | | |
| ERR9 | 020604 G | 7760# | 8521 | 8535 | | | | | | | | | | | | | | |
| EVL = | 000004 G | 5549# | | | | | | | | | | | | | | | | |
| E\$END = | 002100 | 5406# | | | | | | | | | | | | | | | | |
| E\$LOAD= | 000035 | 5406# | 5446 | | | | | | | | | | | | | | | |
| FMT1 | 011426 | 7533# | 7625 | 7631 | 7646 | 7666 | 7684 | 7705 | 7723 | 7742 | 7761 | | | | | | | |
| FMT10 | 011671 | 7542# | 7665 | 7704 | 7741 | | | | | | | | | | | | | |
| FMT11 | 011722 | 7543# | 7685 | | | | | | | | | | | | | | | |
| FMT19 | 011761 | 7544# | 8426 | | | | | | | | | | | | | | | |
| FMT2 | 011436 | 7534# | 7632 | 7647 | 7667 | 7686 | 7706 | 7724 | 7743 | 7762 | | | | | | | | |
| FMT24 | 012012 | 7545# | 8452 | 8471 | 8490 | | | | | | | | | | | | | |
| FMT27 | 021554 | 7944 | 7947# | | | | | | | | | | | | | | | |
| FMT3 | 011460 | 7535# | 7634 | 7649 | 7688 | 7745 | | | | | | | | | | | | |
| FMT4 | 011522 | 7536# | 7635 | 7650 | 7654 | 7669 | 7673 | 7689 | 7693 | 7708 | 7712 | 7726 | 7730 | 7746 | | | | |
| | | 7750 | 7764 | | | | | | | | | | | | | | | |
| FMT5 | 011535 | 7537# | 7636 | 7651 | 7655 | 7670 | 7674 | 7690 | 7694 | 7709 | 7713 | 7727 | 7731 | 7747 | | | | |
| | | 7751 | 7765 | | | | | | | | | | | | | | | |
| FMT6 | 011565 | 7538# | 7638 | 7653 | 7657 | 7672 | 7676 | 7692 | 7696 | 7711 | 7715 | 7729 | 7733 | 7749 | | | | |

CZDMRA M8203 STATIC DIAG #1
CZDMRA.P11 18-JUL-79 09:44

MACY11 30A(1052) 18-JUL-79 09:53 PAGE 8-5
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0179

| | | | | | | | | | | | | | | |
|----------|--------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| FMT7 | 011620 | 7753 | 7767 | | | | | | | | | | | |
| FMT8 | 011630 | 7539# | 7633 | 7668 | 7725 | 7744 | 7763 | | | | | | | |
| FMT9 | 011664 | 7540# | 7648 | 7687 | 7707 | | | | | | | | | |
| | | 7541# | 7637 | 7652 | 7656 | 7671 | 7675 | 7691 | 7695 | 7710 | 7714 | 7728 | 7732 | 7748 |
| | | 7752 | 7766 | | | | | | | | | | | |
| FRSTIM | 002412 | 5912# | 7825 | 7832* | | | | | | | | | | |
| F\$AU = | 000015 | 5406# | 7963 | 7964 | | | | | | | | | | |
| F\$AUTO= | 000020 | 5406# | 7897 | 7909 | | | | | | | | | | |
| F\$BGN = | 000040 | 5406# | 5412 | 7624 | 7630 | 7645 | 7664 | 7683 | 7703 | 7722 | 7740 | 7760 | 7783 | 7799 |
| | | 7817 | 7897 | 7923 | 7940 | 7963 | 7985 | 7998 | 8010 | 8023 | 8038 | 8043 | 8061 | 8072 |
| | | 8088 | 8099 | 8120 | 8136 | 8156 | 8173 | 8197 | 8202 | 8217 | 8249 | 8256 | 8268 | 8273 |
| | | 8290 | 8302 | 8307 | 8324 | 8336 | 8341 | 8358 | 8370 | 8375 | 8392 | 8419 | 8493 | 8509 |
| | | 8522 | 8536 | 8548 | 8559 | 8564 | 8580 | 8608 | 8618 | 8625 | 8646 | 8650 | 8665 | 8680 |
| | | 8697 | 8701 | 8716 | 8730 | 8744 | 8748 | 8763 | 8777 | 8802 | 8808 | 8825 | 8841 | 8858 |
| | | 8867 | 8869 | 8889 | 8893 | 8907 | 8908 | 8926 | 8929 | 8953 | 8963 | 8968 | 8969 | 8991 |
| | | 9068 | 9091 | 9101 | 9121 | 9150 | 9170 | 9196 | 9218 | 9256 | 9276 | 9345 | 9360 | 9401 |
| | | 9417 | 9481 | 9496 | 9516 | 9532 | 9596 | 9612 | 9679 | 9695 | 9742 | 9758 | 9825 | 9838 |
| | | 9857 | 9874 | 10003 | 10027 | 10104 | 10122 | 10156 | 10174 | 10224 | 10239 | 10256 | 10277 | 10309 |
| | | 10328 | 10359 | 10382 | 10429 | 10456 | 10489 | 10513 | | | | | | |
| F\$CLEA= | 000007 | 5406# | 7923 | 7926 | | | | | | | | | | |
| F\$DU = | 000016 | 5406# | 7940 | 7945 | | | | | | | | | | |
| F\$END | 000041 | 5406# | 5412 | 7626 | 7639 | 7658 | 7677 | 7697 | 7716 | 7734 | 7754 | 7768 | 7785 | 7883 |
| | | 7909 | 7926 | 7945 | 7964 | 7985 | 7998 | 8010 | 8023 | 8038 | 8057 | 8061 | 8072 | 8088 |
| | | 8099 | 8120 | 8136 | 8156 | 8173 | 8197 | 8202 | 8217 | 8249 | 8256 | 8268 | 8286 | 8290 |
| | | 8302 | 8320 | 8324 | 8336 | 8354 | 8358 | 8370 | 8388 | 8392 | 8419 | 8493 | 8509 | 8522 |
| | | 8536 | 8548 | 8559 | 8564 | 8580 | 8608 | 8618 | 8625 | 8646 | 8665 | 8676 | 8680 | 8697 |
| | | 8716 | 8726 | 8730 | 8744 | 8763 | 8773 | 8777 | 8802 | 8825 | 8836 | 8841 | 8858 | 8867 |
| | | 8884 | 8889 | 8893 | 8907 | 8908 | 8926 | 8929 | 8953 | 8963 | 8968 | 8969 | 8991 | 9068 |
| | | 9091 | 9101 | 9121 | 9150 | 9170 | 9196 | 9218 | 9256 | 9276 | 9345 | 9360 | 9401 | 9417 |
| | | 9481 | 9496 | 9516 | 9532 | 9596 | 9612 | 9679 | 9695 | 9742 | 9758 | 9825 | 9838 | 9857 |
| | | 9874 | 10003 | 10027 | 10104 | 10122 | 10156 | 10174 | 10224 | 10239 | 10256 | 10277 | 10309 | 10328 |
| | | 10359 | 10382 | 10429 | 10463 | 10492 | 10513 | | | | | | | |
| F\$HARD= | 000004 | 5406# | 10456 | 10463 | | | | | | | | | | |
| F\$HW = | 000013 | 5406# | 5491 | 5505 | | | | | | | | | | |
| F\$INIT= | 000006 | 5406# | 7817 | 7883 | | | | | | | | | | |
| F\$JMP = | 000050 | 5406# | | | | | | | | | | | | |
| F\$MOD = | 000000 | 5406# | 5412 | 10513 | | | | | | | | | | |
| F\$MSG = | 000011 | 5406# | 7624 | 7626 | 7630 | 7639 | 7645 | 7658 | 7664 | 7677 | 7683 | 7697 | 7703 | 7716 |
| | | 7722 | 7734 | 7740 | 7754 | 7760 | 7768 | | | | | | | |
| F\$PROT= | 000021 | 5406# | 7799 | 7803 | | | | | | | | | | |
| F\$PWR = | 000017 | 5406# | | | | | | | | | | | | |
| F\$RPT = | 000012 | 5406# | 7783 | 7785 | | | | | | | | | | |
| F\$SEG = | 000003 | 5406# | 8043 | 8057 | 8273 | 8286 | 8307 | 8320 | 8341 | 8354 | 8375 | 8388 | 8650 | 8676 |
| | | 8701 | 8726 | 8748 | 8773 | 8808 | 8836 | 8869 | 8884 | | | | | |
| F\$SOFT= | 000005 | 5406# | 10489 | 10492 | | | | | | | | | | |
| F\$SRV = | 000010 | 5406# | | | | | | | | | | | | |
| F\$SUB = | 000002 | 5406# | 8867 | 8889 | 8893 | 8907 | 8929 | 8968 | | | | | | |
| F\$SW = | 000014 | 5406# | 5519 | 5522 | | | | | | | | | | |
| F\$TEST= | 000001 | 5406# | 7985 | 7998 | 8010 | 8023 | 8038 | 8061 | 8072 | 8088 | 8099 | 8120 | 8136 | 8156 |
| | | 8173 | 8202 | 8217 | 8256 | 8268 | 8290 | 8302 | 8324 | 8336 | 8358 | 8370 | 8392 | 8419 |
| | | 8493 | 8509 | 8564 | 8580 | 8625 | 8646 | 8680 | 8697 | 8730 | 8744 | 8777 | 8802 | 8841 |
| | | 8858 | 8908 | 8926 | 8969 | 8991 | 9101 | 9121 | 9150 | 9170 | 9196 | 9218 | 9256 | 9276 |
| | | 9345 | 9360 | 9401 | 9417 | 9481 | 9496 | 9516 | 9532 | 9596 | 9612 | 9679 | 9695 | 9742 |
| | | 9758 | 9825 | 9838 | 9857 | 9874 | 10003 | 10027 | 10104 | 10122 | 10156 | 10174 | 10224 | 10239 |
| | | 10256 | 10277 | 10309 | 10328 | 10359 | 10382 | 10429 | | | | | | |
| GETALL | 004500 | 6628# | 6676 | 6682 | 6692 | 6698 | 6805 | 6811 | 6969 | 7001 | 7007 | 7017 | 7023 | 7058 |

CZDMRA M8203 STATIC DIAG #1
CZDMRA.P11 18-JUL-79 09:44

MACY11 30A(1052) 18-JUL-79 09:53 PAGE 8-7
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0181

| | | | | | | | | | | | | | | |
|----------------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| IXE = 004000 G | 5549# | | | | | | | | | | | | | |
| ISAU = 000041 | 5406# | 7963# | 7964# | | | | | | | | | | | |
| ISAUTO= 000041 | 5406# | 7897# | 7909# | | | | | | | | | | | |
| ISCLN = 000041 | 5406# | 7923# | 7926# | | | | | | | | | | | |
| ISDU = 000041 | 5406# | 7940# | 7945# | | | | | | | | | | | |
| ISHRD = 000041 | 10456# | 10463# | | | | | | | | | | | | |
| ISINIT= 000041 | 5406# | 7817# | 7883# | | | | | | | | | | | |
| ISMOD = 000041 | 5406# | 5412# | 10513# | | | | | | | | | | | |
| ISMSG = 000041 | 5406# | 7624# | 7626# | 7630# | 7639# | 7645# | 7658# | 7664# | 7677# | 7683# | 7697# | 7703# | 7716# | |
| | | 7722# | 7734# | 7740# | 7754# | 7760# | 7768# | | | | | | | |
| ISPROT= 000040 | 5406# | 7799# | | | | | | | | | | | | |
| ISPTAB= 000041 | 5406# | | | | | | | | | | | | | |
| ISPRW = 000041 | 5406# | | | | | | | | | | | | | |
| ISRPT = 000041 | 5406# | 7783# | 7785# | | | | | | | | | | | |
| ISSEG = 000041 | 5406# | 7985 | 8010 | 8038 | 8043# | 8057# | 8072 | 8099 | 8136 | 8173 | 8217 | 8268 | 8273# | |
| | | 8286# | 8302 | 8307# | 8320# | 8336 | 8341# | 8354# | 8370 | 8375# | 8388# | 8419 | 8509 | 8580 |
| | | 8646 | 8650# | 8665 | 8676# | 8697 | 8701# | 8716 | 8726# | 8744 | 8748# | 8763 | 8773# | 8802 |
| | | 8808# | 8825 | 8836# | 8858 | 8867 | 8869# | 8884# | 8893 | 8926 | 8929 | 8991 | 9121 | 9170 |
| | | 9218 | 9276 | 9360 | 9417 | 9496 | 9532 | 9612 | 9695 | 9758 | 9838 | 9874 | 10027 | 10122 |
| | 10174 | 10239 | 10277 | 10328 | 10382 | | | | | | | | | |
| ISSETU= 000041 | 5406# | | | | | | | | | | | | | |
| ISSFT = 000041 | 10489# | 10492# | | | | | | | | | | | | |
| ISSRV = 000041 | 5406# | | | | | | | | | | | | | |
| ISSUB = 000041 | 5406# | 7985 | 8010 | 8038 | 8072 | 8099 | 8136 | 8173 | 8217 | 8268 | 8302 | 8336 | 8370 | |
| | | 8419 | 8509 | 8580 | 8646 | 8697 | 8744 | 8802 | 8858 | 8867# | 8889# | 8893# | 8907# | 8926 |
| | | 8929# | 8953 | 8963 | 8968# | 8991 | 9121 | 9170 | 9218 | 9276 | 9360 | 9417 | 9496 | 9532 |
| | | 9612 | 9695 | 9758 | 9838 | 9874 | 10027 | 10122 | 10174 | 10239 | 10277 | 10328 | 10382 | |
| ISTST = 000041 | 5406# | 7985# | 7998# | 8010# | 8023# | 8038# | 8061# | 8072# | 8088# | 8099# | 8120# | 8136# | 8156# | |
| | | 8173# | 8197 | 8202# | 8217# | 8249 | 8256# | 8268# | 8290# | 8302# | 8324# | 8336# | 8358# | 8370# |
| | | 8392# | 8419# | 8493# | 8509# | 8522 | 8536 | 8548 | 8559 | 8564# | 8580# | 8608 | 8618 | 8625# |
| | | 8646# | 8680# | 8697# | 8730# | 8744# | 8777# | 8802# | 8841# | 8858# | 8867 | 8893 | 8908# | 8926# |
| | | 8929 | 8969# | 8991# | 9068 | 9091 | 9101# | 9121# | 9150# | 9170# | 9196# | 9218# | 9256# | 9276# |
| | | 9345# | 9360# | 9401# | 9417# | 9481# | 9496# | 9516# | 9532# | 9596# | 9612# | 9679# | 9695# | 9742# |
| | | 9758# | 9825# | 9838# | 9857# | 9874# | 10003# | 10027# | 10104# | 10122# | 10156# | 10174# | 10224# | 10239# |
| | 10256# | 10277# | 10309# | 10328# | 10359# | 10382# | 10429# | | | | | | | |
| I422 = 000200 | 5780# | 5786 | | | | | | | | | | | | |
| JSJMP = 000167 | 5406# | | | | | | | | | | | | | |
| LDMSG1 011346 | 7503# | | | | | | | | | | | | | |
| LDTXSI 005146 | 6744# | 6857 | 6858 | 6916 | 7407 | 9003 | 9004 | 9028 | 9076 | 9307 | 9503 | 9546 | 9626 | |
| | 9709 | 9772 | 9845 | 10044 | 10066 | 10096 | 10100 | 10129 | 10181 | 10245 | 10286 | 10287 | 10335 | |
| | 10391 | 10398 | 10399 | | | | | | | | | | | |
| LOADAT 002410 | 5911# | 7685 | 8875* | 8876* | | | | | | | | | | |
| LODMSG 010770 | 7400# | 7505 | 10282 | 10394 | | | | | | | | | | |
| LOE = 040000 G | 5549# | | | | | | | | | | | | | |
| LOGDEV 002344 | 5892# | 7851* | 7857* | 7858 | 7860 | 7906 | 7944 | | | | | | | |
| LOOPIN 004046 | 6531# | 8433 | | | | | | | | | | | | |
| LOT = 000010 G | 5549# | | | | | | | | | | | | | |
| LULoop= 000010 | 5568# | 6444 | 6635 | 6773 | 6866 | 8804 | | | | | | | | |
| LULP = 000040 | 5599# | 5664# | 9542 | 9622 | 9705 | 9768 | 9890 | | | | | | | |
| LUREG 002302 | 5828 | 5829 | 5830 | 5831 | 5832 | 5833 | 5834 | 5835 | 5836 | 5837 | 5838 | 5839 | 5840 | |
| | 5841 | 5842 | 5843 | 5886# | | | | | | | | | | |
| LUR10 = 002302 | 5828# | 6507 | 7636 | 7651 | 7670 | 7690 | 7709 | 7727 | 7747 | 7765 | | | | |
| LUR11 = 002304 | 5829# | 7636 | 7651 | 7670 | 7690 | 7709 | 7727 | 7747 | 7765 | | | | | |
| LUR12 = 002306 | 5830# | 7636 | 7651 | 7670 | 7690 | 7709 | 7727 | 7747 | 7765 | | | | | |
| LUR13 = 002310 | 5831# | 7636 | 7651 | 7670 | 7690 | 7709 | 7727 | 7747 | 7765 | | | | | |
| LUR14 = 002312 | 5832# | 7638 | 7653 | 7672 | 7692 | 7711 | 7729 | 7749 | 7767 | | | | | |

| | | | | | | | |
|--------|--------|---|-------|-------|-------|------|-------|
| LSSW | 002302 | G | 5519# | | | | |
| LSTEST | 002114 | G | 5446# | | | | |
| LSTIML | 002014 | G | 5446# | | | | |
| LSUNIT | 002012 | G | 5446# | 7858 | | | |
| L10000 | 002300 | | 5491 | 5505# | | | |
| L10001 | 002302 | | 5519 | 5522# | | | |
| L10002 | 014604 | | 7626# | | | | |
| L10003 | 015112 | | 7639# | | | | |
| L10004 | 015574 | | 7658# | | | | |
| L10005 | 016252 | | 7677# | | | | |
| L10006 | 016764 | | 7697# | | | | |
| L10007 | 017442 | | 7716# | | | | |
| L10010 | 020074 | | 7734# | | | | |
| L10011 | 020602 | | 7754# | | | | |
| L10012 | 021060 | | 7768# | | | | |
| L10013 | 021062 | | 7785# | | | | |
| L10015 | 021440 | | 7883# | | | | |
| L10016 | 021520 | | 7909# | | | | |
| L10017 | 021522 | | 7926# | | | | |
| L10020 | 021552 | | 7945# | | | | |
| L10021 | 021606 | | 7964# | | | | |
| L10022 | 021670 | | 7998# | | | | |
| L10023 | 021754 | | 8023# | | | | |
| L10024 | 022100 | | 8061# | | | | |
| L10025 | 022202 | | 8088# | | | | |
| L10026 | 022322 | | 8120# | | | | |
| L10027 | 022442 | | 8156# | | | | |
| L10030 | 022616 | | 8197 | 8202# | | | |
| L10031 | 023052 | | 8249 | 8256# | | | |
| L10032 | 023170 | | 8290# | | | | |
| L10033 | 023306 | | 8324# | | | | |
| L10034 | 023424 | | 8358# | | | | |
| L10035 | 023542 | | 8392# | | | | |
| L10036 | 024172 | | 8493# | | | | |
| L10037 | 024524 | | 8522 | 8536 | 8548 | 8559 | 8564# |
| L10040 | 025014 | | 8608 | 8618 | 8625# | | |
| L10041 | 025224 | | 8680# | | | | |
| L10042 | 025430 | | 8730# | | | | |
| L10043 | 025634 | | 8777# | | | | |
| L10044 | 026064 | | 8841# | | | | |
| L10045 | 026316 | | 8908# | | | | |
| L10046 | 026222 | | 8889# | | | | |
| L10047 | 026314 | | 8907# | | | | |
| L10050 | 026570 | | 8969# | | | | |
| L10051 | 026566 | | 8953 | 8963 | 8968# | | |
| L10052 | 027246 | | 9068 | 9091 | 9101# | | |
| L10053 | 027366 | | 9150# | | | | |
| L10054 | 027476 | | 9196# | | | | |
| L10055 | 027672 | | 9256# | | | | |
| L10056 | 030236 | | 9345# | | | | |
| L10057 | 030446 | | 9401# | | | | |
| L10060 | 030776 | | 9481# | | | | |
| L10061 | 031076 | | 9516# | | | | |
| L10062 | 031446 | | 9596# | | | | |
| L10063 | 032010 | | 9679# | | | | |
| L10064 | 032260 | | 9742# | | | | |

CZDMRA M8203 STATIC DIAG #1
CZDMRA.P11 18-JUL-79 09:44

MACY11 30A(1052) 18-JUL-79 09:53 PAGE 8-11
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0185

| | | | | | | | | | | | | | | | | | | | |
|---------|--------|--------|--------|-------|-------|-------|-------|-------|------|-------|-------|-------|-------|-------|------|--|--|--|--|
| PATCH | 035760 | 10504# | | | | | | | | | | | | | | | | | |
| PATD | 002630 | 6038# | 8288 | 8305 | | | | | | | | | | | | | | | |
| PATE | 002633 | 6044# | 8322 | 8339 | | | | | | | | | | | | | | | |
| PATF | 002641 | 6053# | 8356 | 8373 | | | | | | | | | | | | | | | |
| PATG | 002644 | 6059# | 8176 | 8390 | | | | | | | | | | | | | | | |
| PATH | 002654 | 6070# | 8180 | 8512 | | | | | | | | | | | | | | | |
| PATI | 002664 | 6081# | 8524 | 8528 | 8582 | 8667 | 8670 | 8898 | 8901 | | | | | | | | | | |
| PATJ | 002674 | 6092# | 8562 | 8585 | 8589 | 8590 | | | | | | | | | | | | | |
| PATK | 002704 | 6103# | 8648 | 8746 | | | | | | | | | | | | | | | |
| PATL | 003014 | 6177# | 8678 | 8699 | 8775 | | | | | | | | | | | | | | |
| PATM | 003022 | 6186# | 8014 | 8017 | 8079 | 8082 | 8111 | 8114 | 8147 | 8150 | 8184 | 8220 | 8728 | | | | | | |
| PATN | 003032 | 6197# | 8200 | 8927 | | | | | | | | | | | | | | | |
| PATO | 003050 | 6214# | 8860 | 8965 | | | | | | | | | | | | | | | |
| PATP | 003066 | 6231# | 8861 | 8887 | | | | | | | | | | | | | | | |
| PATU | 003104 | 6248# | 8806 | | | | | | | | | | | | | | | | |
| PATV | 003152 | 6288# | 8805 | 8839 | | | | | | | | | | | | | | | |
| PNT = | 001000 | G | 5549# | | | | | | | | | | | | | | | | |
| POLL = | 000200 | | 5604# | | | | | | | | | | | | | | | | |
| PRI = | 002000 | G | 5549# | | | | | | | | | | | | | | | | |
| PRIOR | 002350 | | 5894# | | | | | | | | | | | | | | | | |
| PRIRTY | 035637 | 10460 | 10467# | | | | | | | | | | | | | | | | |
| PRI00 = | 000000 | G | 5549# | | | | | | | | | | | | | | | | |
| PRI01 = | 000040 | G | 5549# | | | | | | | | | | | | | | | | |
| PRI02 = | 000100 | G | 5549# | | | | | | | | | | | | | | | | |
| PRI03 = | 000140 | G | 5549# | | | | | | | | | | | | | | | | |
| PRI04 = | 000200 | G | 5549# | | | | | | | | | | | | | | | | |
| PRI05 = | 000240 | G | 5549# | | | | | | | | | | | | | | | | |
| PRI06 = | 000300 | G | 5549# | | | | | | | | | | | | | | | | |
| PRI07 = | 000340 | G | 5549# | 7899 | 7901 | 7987 | 7989 | | | | | | | | | | | | |
| PSTACK | 002346 | | 5893# | 6702 | 6815 | 7027 | 7068 | 7122 | 7350 | 7485 | 7819* | | | | | | | | |
| RAB = | 000004 | | 5667# | 7314 | 7316 | 7322 | | | | | | | | | | | | | |
| RABT = | 000004 | | 5727# | | | | | | | | | | | | | | | | |
| RAX15 | 002370 | | 5903# | 6568* | 6569* | 6639 | 8539* | 8540 | 8544 | 8599* | 8600 | 8604 | 8658 | 8661 | 8709 | | | | |
| | | | 8712 | 8756 | 8759 | 8816* | 8817 | 8821 | 8945 | 8949 | 9061 | 9064 | 9084 | 9087 | 9560 | | | | |
| | | | 9563 | 9575 | 9578 | 9589 | 9658 | 9661 | 9723 | 9726 | 9804 | 9807 | 9848 | 9851 | 9952 | | | | |
| | | | 9955 | 9972 | 9988 | | | | | | | | | | | | | | |
| RAX16 | 002372 | | 5904# | 6572* | 6573* | 6641 | 7096 | 7102 | 7110 | 7116 | 8551 | 8555 | 8610 | 8614 | 8667 | | | | |
| | | | 8671 | 8718 | 8721 | 8765 | 8768 | 8827 | 8831 | 8873 | 8879 | 8898 | 8902 | 8955 | 8959 | | | | |
| RCVBUF | 003262 | | 6360# | 7509 | | | | | | | | | | | | | | | |
| RCV1ST | 007406 | | 7162# | 10048 | 10141 | 10187 | 10339 | 10409 | | | | | | | | | | | |
| RDALL = | 000004 | | 5631# | | | | | | | | | | | | | | | | |
| RDAX = | 000020 | | 5617# | 5689# | 6555 | | | | | | | | | | | | | | |
| RDRXSI | 007326 | | 7138# | 7249 | 7429 | 10248 | | | | | | | | | | | | | |
| READAX | 004076 | | 6549# | 6638 | 7093 | 8529 | 8596 | 8657 | 8708 | 8755 | 8813 | 8872 | 8897 | 8944 | 9060 | | | | |
| | | | 9083 | 9559 | 9574 | 9588 | 9657 | 9722 | 9803 | 9847 | | | | | | | | | |
| READLU | 003644 | | 6458# | 6509 | 6559 | 6567 | 6571 | 6610 | 6671 | 6687 | 6800 | 6869 | 6881 | 6966 | 6996 | | | | |
| | | | 7012 | 7053 | 7140 | 7144 | 7183 | 7261 | 7466 | 8013 | 8047 | 8078 | 8110 | 8146 | 8187 | | | | |
| | | | 8235 | 8276 | 8310 | 8344 | 8378 | 9015 | 9047 | 9247 | 9290 | 9299 | 9311 | 9330 | 9338 | | | | |
| READY - | 000200 | | 5686# | 6560 | 6611 | | | | | | | | | | | | | | |
| REDBYT | 002364 | | 5901# | 6470* | 6471* | 6510 | 6560 | 6568 | 6572 | 6611 | 6674 | 6680 | 6690 | 6696 | 6803 | | | | |
| | | | 6809 | 6870 | 6882 | 6967 | 6999 | 7005 | 7015 | 7021 | 7056 | 7062 | 7141 | 7145 | 7184 | | | | |
| | | | 7262 | 7469 | 7475 | 8014 | 8018 | 8048* | 8049 | 8052 | 8079 | 8083 | 8111 | 8115 | 8147 | | | | |
| | | | 8151 | 8188* | 8189 | 8193 | 8236* | 8239* | 8241 | 8245 | 8277* | 8278 | 8281 | 8311* | 8312 | | | | |
| | | | 8315 | 8345* | 8346 | 8349 | 8379* | 8380 | 8383 | 8440* | 8441 | 8447 | 8460* | 8461 | 8466 | | | | |
| | | | 8479* | 8480 | 8485 | 9016 | 9048 | 9248 | 9291 | 9300 | 9312 | 9331 | 9339 | | | | | | |
| REDDAT | 002500 | | 5944# | 8219 | 8229* | 8230* | 8240* | 8241 | 8244 | 8583 | 8591* | 8592* | 8600 | 8603 | 8610 | | | | |

| | | | | | | | | | | | | | | |
|----------|--------|-------|--------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| SAVE6 | 002416 | 5914# | 7828* | 7831 | 7908 | 7997 | | | | | | | | |
| SAVLEN | 002432 | 5920# | 6446* | 7250 | 7383* | 7824* | | | | | | | | |
| SCRACH | 002342 | 5891# | | | | | | | | | | | | |
| SEC = | 000020 | 5771# | | | | | | | | | | | | |
| SECA = | 000020 | 5629# | | | | | | | | | | | | |
| SELFR = | 000040 | 5606# | | | | | | | | | | | | |
| SELSBY = | 000002 | 5610# | | | | | | | | | | | | |
| SEL4 | 002454 | 5932# | 7872* | 7873* | | | | | | | | | | |
| SEL6 | 002456 | 5933# | 6423* | 6533* | 7874* | 7875* | | | | | | | | |
| SETUP | 010612 | 7368# | 9535 | 9615 | 9698 | 9761 | 9880 | | | | | | | |
| SFPTBL | 002302 | 5519# | | | | | | | | | | | | |
| SIGQ = | 000100 | 5699# | | | | | | | | | | | | |
| STGR = | 000200 | 5698# | | | | | | | | | | | | |
| SOM = | 000001 | 5592# | | | | | | | | | | | | |
| STALL | 005136 | 6731# | 6775 | 6777 | 6868 | 6873 | 6886 | | | | | | | |
| STARES | 002444 | 5925# | 7847* | 7852* | 8424 | | | | | | | | | |
| STARST | 021232 | 7835 | 7838 | 7846# | | | | | | | | | | |
| STBY = | 000002 | 5680# | | | | | | | | | | | | |
| STEPLU = | 000020 | 5567# | 6774 | 6776 | 6867 | 6872 | 6885 | | | | | | | |
| STEPMP = | 000001 | 5571# | 6424 | 6425 | | | | | | | | | | |
| STPCLK | U03540 | 6421# | 6468 | 6492 | 8144 | | | | | | | | | |
| STPERR | 007620 | 7215# | | | | | | | | | | | | |
| STPLU | 005226 | 6765# | 6933 | 7179 | 7222 | 7346 | 7436 | 7464 | 9013 | 9045 | 9081 | 9244 | 9287 | 9297 |
| | | 9309 | 9328 | 9507 | 9550 | 9554 | 9568 | 9583 | 9591 | 9630 | 9636 | 9666 | 9713 | 9717 |
| | | 9731 | 9776 | 9782 | 9812 | 9903 | 9924 | 9939 | 9960 | 9976 | 10063 | 10075 | 10091 | 10137 |
| | | 10246 | 10289 | 10353 | 10403 | | | | | | | | | |
| STR = | 000040 | 5770# | | | | | | | | | | | | |
| STRIP = | 000010 | 5630# | 9125 | 9222 | 9283 | 9324 | 9499 | 9537 | 9700 | 10041 | 10281 | 10388 | | |
| SUBRPC | 002352 | 5895# | 6665 | 6666 | 6668* | 6669* | 6706* | 6794 | 6795 | 6797* | 6798* | 6819* | 6840* | 6841* |
| | | 6893* | 6913* | 6914* | 6940* | 6955* | 6956* | 6972* | 6990 | 6991 | 6993* | 6994* | 7031* | 7047 |
| | | 7048 | 7050* | 7051* | 7072* | 7087 | 7088 | 7090* | 7091* | 7126* | 7165* | 7166* | 7202* | 7245* |
| | | 7246* | 7356* | 7388* | 7427* | 7428* | 7443* | 7460* | 7461* | 7491* | 7665 | 7704 | 7741 | 7820* |
| SVCGBL - | 000000 | 5406# | 5412 | 5420# | 5446 | 5469 | 5491 | 5519 | 6383 | 6388 | 7624 | 7630 | 7645 | 7664 |
| | | 7683 | 7703 | 7722 | 7740 | 7760 | 7783 | 7799 | 7817 | 7897 | 7923 | 7940 | 7963 | 10456 |
| | | 10489 | 10515# | | | | | | | | | | | |
| SVCINS = | 000001 | 5406# | 5417# | 5446 | 5469 | 5491 | 5519 | 6383 | 6388 | 6678 | 6684 | 6694 | 6700 | 6807 |
| | | 6813 | 6971 | 7003 | 7009 | 7019 | 7025 | 7060 | 7066 | 7100 | 7106 | 7114 | 7120 | 7266 |
| | | 7271 | 7290 | 7296 | 7305 | 7311 | 7320 | 7326 | 7335 | 7341 | 7473 | 7479 | 7625 | 7626 |
| | | 7631 | 7632 | 7633 | 7634 | 7635 | 7636 | 7637 | 7638 | 7639 | 7646 | 7647 | 7648 | 7649 |
| | | 7650 | 7651 | 7652 | 7653 | 7654 | 7655 | 7656 | 7657 | 7658 | 7665 | 7666 | 7667 | 7668 |
| | | 7669 | 7670 | 7671 | 7672 | 7673 | 7674 | 7675 | 7676 | 7677 | 7684 | 7685 | 7686 | 7687 |
| | | 7688 | 7689 | 7690 | 7691 | 7692 | 7693 | 7694 | 7695 | 7696 | 7697 | 7704 | 7705 | 7706 |
| | | 7707 | 7708 | 7709 | 7710 | 7711 | 7712 | 7713 | 7714 | 7715 | 7716 | 7723 | 7724 | 7725 |
| | | 7726 | 7727 | 7728 | 7729 | 7730 | 7731 | 7732 | 7733 | 7734 | 7741 | 7742 | 7743 | 7744 |
| | | 7745 | 7746 | 7747 | 7748 | 7749 | 7750 | 7751 | 7752 | 7753 | 7754 | 7761 | 7762 | 7763 |
| | | 7764 | 7765 | 7766 | 7767 | 7768 | 7785 | 7834 | 7835 | 7837 | 7838 | 7840 | 7841 | 7843 |
| | | 7844 | 7860 | 7861 | 7883 | 7899 | 7906 | 7909 | 7926 | 7942 | 7944 | 7945 | 7964 | 7987 |
| | | 7995 | 7998 | 8021 | 8023 | 8043 | 8055 | 8057 | 8061 | 8086 | 8088 | 8104 | 8118 | 8120 |
| | | 8154 | 8156 | 8196 | 8197 | 8202 | 8248 | 8249 | 8256 | 8273 | 8284 | 8286 | 8290 | 8307 |
| | | 8318 | 8320 | 8324 | 8341 | 8352 | 8354 | 8358 | 8375 | 8386 | 8388 | 8392 | 8426 | 8450 |
| | | 8452 | 8469 | 8471 | 8488 | 8490 | 8493 | 8521 | 8522 | 8535 | 8536 | 8547 | 8548 | 8558 |
| | | 8559 | 8564 | 8607 | 8608 | 8617 | 8618 | 8625 | 8650 | 8664 | 8665 | 8674 | 8676 | 8680 |
| | | 8701 | 8715 | 8716 | 8724 | 8726 | 8730 | 8748 | 8762 | 8763 | 8771 | 8773 | 8777 | 8808 |
| | | 8824 | 8825 | 8834 | 8836 | 8841 | 8867 | 8869 | 8882 | 8884 | 8889 | 8893 | 8905 | 8907 |
| | | 8908 | 8929 | 8952 | 8953 | 8962 | 8963 | 8968 | 8969 | 9067 | 9068 | 9090 | 9091 | 9101 |
| | | 9150 | 9196 | 9252 | 9256 | 9295 | 9304 | 9316 | 9335 | 9343 | 9345 | 9401 | 9481 | 9516 |

CZDMRA M8203 STATIC DIAG #1
CZDMRA.P11 18-JUL-79 09:44

MACY11 30A(1052) 18-JUL-79 09:53 PAGE 8-14
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0188

| | | | | | | | | | | | | | |
|----------------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|-------|-------|-------|
| | 9566 | 9581 | 9596 | 9664 | 9679 | 9729 | 9742 | 9810 | 9825 | 9854 | 9857 | 9958 | 10003 |
| | 10104 | 10156 | 10224 | 10256 | 10309 | 10359 | 10429 | 10456 | 10458 | 10459 | 10460 | 10461 | 10463 |
| | 10489 | 10492 | 10515 | | | | | | | | | | |
| SVCSUB= 000001 | 5406# | 5419# | 8867 | 8893 | 8929 | | | | | | | | |
| SVCTAG= 000001 | 5406# | 5421# | 5505 | 5522 | 7626 | 7639 | 7658 | 7677 | 7697 | 7716 | 7734 | 7754 | 7768 |
| | 7785 | 7883 | 7909 | 7926 | 7945 | 7964 | 7998 | 8023 | 8057 | 8061 | 8088 | 8120 | 8156 |
| | 8202 | 8256 | 8286 | 8290 | 8320 | 8324 | 8354 | 8358 | 8388 | 8392 | 8493 | 8564 | 8625 |
| | 8676 | 8680 | 8726 | 8730 | 8773 | 8777 | 8836 | 8841 | 8884 | 8889 | 8907 | 8908 | 8968 |
| | 8969 | 9101 | 9150 | 9196 | 9256 | 9345 | 9401 | 9481 | 9516 | 9596 | 9679 | 9742 | 9825 |
| | 9857 | 10003 | 10104 | 10156 | 10224 | 10256 | 10309 | 10359 | 10429 | 10463 | 10492 | | |
| SVCTST= 000001 | 5406# | 5418# | 7985 | 8010 | 8038 | 8072 | 8099 | 8136 | 8173 | 8217 | 8268 | 8302 | 8336 |
| | 8370 | 8419 | 8509 | 8580 | 8646 | 8697 | 8744 | 8802 | 8858 | 8926 | 8991 | 9121 | 9170 |
| | 9218 | 9276 | 9360 | 9417 | 9496 | 9532 | 9612 | 9695 | 9758 | 9838 | 9874 | 10027 | 10122 |
| | 10174 | 10239 | 10277 | 10328 | 10382 | | | | | | | | |
| SWO = 000002 | 5656# | | | | | | | | | | | | |
| SW1 = 000004 | 5655# | | | | | | | | | | | | |
| SW2 = 000010 | 5654# | | | | | | | | | | | | |
| SW3 = 000040 | 5652# | | | | | | | | | | | | |
| SYNCH = 000226 | 5763# | 9124 | 9221 | 9282 | 9323 | 9498 | 9536 | 9699 | 10040 | 10242 | 10331 | 10334 | 10343 |
| | 10348 | | | | | | | | | | | | |
| SYNPAT 035476 | 10384 | 10426 | 10431# | | | | | | | | | | |
| SYNO = 000001 | 5762# | | | | | | | | | | | | |
| SYN1 = 000002 | 5761# | | | | | | | | | | | | |
| SYN2 = 000004 | 5760# | | | | | | | | | | | | |
| SYN3 = 000010 | 5759# | | | | | | | | | | | | |
| SYN4 = 000020 | 5758# | | | | | | | | | | | | |
| SYN5 = 000040 | 5757# | | | | | | | | | | | | |
| SYN6 = 000100 | 5756# | | | | | | | | | | | | |
| SYN7 = 000200 | 5755# | | | | | | | | | | | | |
| S&LSYM= 010000 | 5406# | 5505# | 5522# | 7626# | 7639# | 7658# | 7677# | 7697# | 7716# | 7734# | 7754# | 7768# | 7785# |
| | 7883# | 7909# | 7926# | 7945# | 7964# | 7998# | 8023# | 8043# | 8061# | 8088# | 8120# | 8156# | 8202# |
| | 8256# | 8273# | 8290# | 8307# | 8324# | 8341# | 8358# | 8375# | 8392# | 8493# | 8564# | 8625# | 8650# |
| | 8680# | 8701# | 8730# | 8748# | 8777# | 8808# | 8841# | 8869# | 8889# | 8907# | 8908# | 8968# | 8969# |
| | 9101# | 9150# | 9196# | 9256# | 9345# | 9401# | 9481# | 9516# | 9596# | 9679# | 9742# | 9825# | 9857# |
| | 10003# | 10104# | 10156# | 10224# | 10256# | 10309# | 10359# | 10429# | 10463# | 10492# | | | |
| | 5749# | 9914 | 9937 | | | | | | | | | | |
| | 5746# | | | | | | | | | | | | |
| TEOM = 000002 | 5785# | 8814 | | | | | | | | | | | |
| TERR = 000200 | 5703# | | | | | | | | | | | | |
| TEST = 000001 | 5899# | | | | | | | | | | | | |
| TESTMD= 000004 | 5964# | 6630* | 6633* | 7648 | 7687 | 7707 | | | | | | | |
| TIMFLG 002360 | 5965# | 6648* | 6649* | 7648 | 7687 | 7707 | | | | | | | |
| TMP0 002530 | 5966# | | | | | | | | | | | | |
| TMP1 002532 | 5967# | | | | | | | | | | | | |
| TMP2 002534 | 5968# | | | | | | | | | | | | |
| TMP3 002536 | 5969# | | | | | | | | | | | | |
| TMP4 002540 | 5970# | | | | | | | | | | | | |
| TMP5 002542 | 5971# | | | | | | | | | | | | |
| TMP6 002544 | 5750# | | | | | | | | | | | | |
| TMP7 002546 | 5940# | | | | | | | | | | | | |
| TSOM = 000001 | 5924# | 8421* | 8426 | | | | | | | | | | |
| TSTCON 002474 | 5748# | | | | | | | | | | | | |
| TSTNUM 002442 | 5806# | | | | | | | | | | | | |
| TXAB - 000004 | 6911# | 9126 | 9129 | 9132 | 9135 | 9138 | 9141 | 9144 | 9175 | 9178 | 9181 | 9184 | 9187 |
| TXABT = 002000 | 9190 | 9226 | 9229 | 9232 | 9235 | 9238 | 9241 | 9284 | 9325 | 9369 | 9372 | 9377 | 9382 |
| TXCHAR 006074 | 9387 | 9390 | 9393 | 9396 | 9422 | 9425 | 9428 | 9435 | 9440 | 9445 | 9450 | 9455 | 9460 |

| | | | | | | | | | | | | | | |
|-----------------|--------|--------|--------|--------|-------|--------|--------|--------|--------|--------|--------|--------|--------|--|
| TXDATA= 000040 | 9465 | 9470 | 9473 | 9476 | | | | | | | | | | |
| TXEN = 000100 | 5700# | 7469 | 7475 | | | | | | | | | | | |
| TXEOM = 001000 | 5615# | 5687# | 9224 | 9886 | 9888 | | | | | | | | | |
| | 5807# | 6343 | 6344 | 6345 | 6346 | 6352 | 6353 | 9133 | 9136 | 9139 | 9142 | 9145 | 9182 | |
| | 9185 | 9188 | 9191 | 9233 | 9236 | 9239 | 9242 | 9388 | 9391 | 9394 | 9397 | 9471 | 9474 | |
| | 9477 | 10099 | 10285 | 10397 | | | | | | | | | | |
| TXGA = 000010 | 5747# | | | | | | | | | | | | | |
| TXGOA = 004000 | 5805# | 10285 | | | | | | | | | | | | |
| TXLENO= 000040 | 5793# | 7250 | 9367 | 9380 | 9433 | 9443 | 9453 | 9463 | | | | | | |
| TXLEN1= 000100 | 5792# | 7250 | 9375 | 9380 | 9438 | 9443 | 9458 | 9463 | | | | | | |
| TXLEN2= 000200 | 5791# | 7250 | 9367 | 9375 | 9380 | 9448 | 9453 | 9458 | 9463 | | | | | |
| TXSOM = 000400 | 5808# | 6336 | 6337 | 6849 | 9002 | 9306 | 10389 | | | | | | | |
| TXWORD 002422 | 5916# | 6745* | 6747 | 6750 | 6849* | 6850* | 6915* | 7406* | 9002* | 9027* | 9075* | 9306* | 9502* | |
| | 9545* | 9625* | 9708* | 9771* | 9844* | 10043* | 10065* | 10095* | 10099* | 10128* | 10180* | 10244* | 10285* | |
| | 10334* | 10389* | 10397* | | | | | | | | | | | |
| TX0 = 000001 | 5583# | 5741# | | | | | | | | | | | | |
| TX1 = 000002 | 5582# | 5740# | | | | | | | | | | | | |
| TX2 = 000004 | 5581# | 5739# | | | | | | | | | | | | |
| TX3 = 000010 | 5580# | 5738# | | | | | | | | | | | | |
| TX4 = 000020 | 5579# | 5737# | | | | | | | | | | | | |
| TX5 = 000040 | 5578# | 5736# | | | | | | | | | | | | |
| TX6 = 000100 | 5577# | 5735# | | | | | | | | | | | | |
| TX7 = 000200 | 5576# | 5734# | | | | | | | | | | | | |
| TSARGC= 000001 | 5446# | 7625# | 7631# | 7632# | 7633# | 7634# | 7635# | 7636# | 7637# | 7638# | 7646# | 7647# | 7648# | |
| | 7649# | 7650# | 7651# | 7652# | 7653# | 7654# | 7655# | 7656# | 7657# | 7665# | 7666# | 7667# | 7668# | |
| | 7669# | 7670# | 7671# | 7672# | 7673# | 7674# | 7675# | 7676# | 7684# | 7685# | 7686# | 7687# | 7688# | |
| | 7689# | 7690# | 7691# | 7692# | 7693# | 7694# | 7695# | 7696# | 7704# | 7705# | 7706# | 7707# | 7708# | |
| | 7709# | 7710# | 7711# | 7712# | 7713# | 7714# | 7715# | 7723# | 7724# | 7725# | 7726# | 7727# | 7728# | |
| | 7729# | 7730# | 7731# | 7732# | 7733# | 7741# | 7742# | 7743# | 7744# | 7745# | 7746# | 7747# | 7748# | |
| | 7749# | 7750# | 7751# | 7752# | 7753# | 7761# | 7762# | 7763# | 7764# | 7765# | 7766# | 7767# | 7944# | |
| | 8426# | 8452# | 8471# | 8490# | | | | | | | | | | |
| TS CODE= 012032 | 10458# | 10459# | 10460# | 10461# | | | | | | | | | | |
| TSERRN= 000032 | 5406# | 6678# | 6684# | 6694# | 6700# | 6807# | 6813# | 6971# | 7003# | 7009# | 7019# | 7025# | 7060# | |
| | 7066# | 7100# | 7106# | 7114# | 7120# | 7266# | 7271# | 7290# | 7296# | 7305# | 7311# | 7320# | 7326# | |
| | 7335# | 7341# | 7473# | 7479# | 7995# | 8021# | 8055# | 8086# | 8118# | 8154# | 8196# | 8248# | 8284# | |
| | 8318# | 8352# | 8386# | 8450# | 8469# | 8488# | 8521# | 8535# | 8547# | 8558# | 8607# | 8617# | 8664# | |
| | 8674# | 8715# | 8724# | 8762# | 8771# | 8824# | 8834# | 8882# | 8905# | 8952# | 8962# | 9067# | 9090# | |
| | 9252# | 9295# | 9304# | 9316# | 9335# | 9343# | 9566# | 9581# | 9664# | 9729# | 9810# | 9854# | 9958# | |
| TSFXCP= 000000 | 10458# | 10459# | 10460# | 10461# | | | | | | | | | | |
| TSFLAG= 000040 | 8197# | 8249# | 8522# | 8536# | 8548# | 8559# | 8608# | 8618# | 8665# | 8716# | 8763# | 8825# | 8953# | |
| | 8963# | 9068# | 9091# | | | | | | | | | | | |
| TSGMAN= 000000 | 5406# | | | | | | | | | | | | | |
| TSHILI= 000007 | 10458# | 10459# | 10460# | 10461# | | | | | | | | | | |
| TSLAST= 000001 | 5406# | 10515# | | | | | | | | | | | | |
| TSLOLI= 000000 | 10458# | 10459# | 10460# | 10461# | | | | | | | | | | |
| TSLSYM= 010000 | 5406# | 5505 | 5522 | 7626 | 7639 | 7658 | 7677 | 7697 | 7716 | 7734 | 7754 | 7768 | 7785 | |
| | 7883 | 7909 | 7926 | 7945 | 7964 | 7998 | 8023 | 8061 | 8088 | 8120 | 8156 | 8202 | 8256 | |
| | 8290 | 8324 | 8358 | 8392 | 8493 | 8564 | 8625 | 8680 | 8730 | 8777 | 8841 | 8889 | 8907 | |
| | 8908 | 8968 | 8969 | 9101 | 9150 | 9196 | 9256 | 9345 | 9401 | 9481 | 9516 | 9596 | 9679 | |
| | 9742 | 9825 | 9857 | 10003 | 10104 | 10156 | 10224 | 10256 | 10309 | 10359 | 10429 | 10463 | 10492 | |
| TSLTNO= 000052 | 10515# | | | | | | | | | | | | | |
| TSNEST= 177777 | 5406# | 5412# | 5491# | 5505# | 5519# | 5522# | 7624# | 7626# | 7630# | 7639# | 7645# | 7658# | 7664# | |
| | 7677# | 7683# | 7697# | 7703# | 7716# | 7722# | 7734# | 7740# | 7754# | 7760# | 7768# | 7783# | 7785# | |
| | 7799# | 7803# | 7817# | 7883# | 7897# | 7909# | 7923# | 7926# | 7940# | 7945# | 7963# | 7964# | 7985# | |
| | 7998# | 8010# | 8023# | 8038# | 8043# | 8057# | 8061# | 8072# | 8088# | 8099# | 8120# | 8136# | 8156# | |
| | 8173# | 8202# | 8217# | 8256# | 8268# | 8273# | 8286# | 8290# | 8302# | 8307# | 8320# | 8324# | 8336# | |

CZDMRA M8203 STATIC DIAG #1
CZDMRA.P11 18-JUL-79 09:44

MACY11 30A(1052) 18-JUL-79 09:53 PAGE 8-16
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0190

| | | | | | | | | | | | | | |
|-----------------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| | 8341# | 8354# | 8358# | 8370# | 8375# | 8388# | 8392# | 8419# | 8493# | 8509# | 8564# | 8580# | 8625# |
| | 8646# | 8650# | 8676# | 8680# | 8697# | 8701# | 8726# | 8730# | 8744# | 8748# | 8773# | 8777# | 8802# |
| | 8808# | 8836# | 8841# | 8858# | 8867# | 8869# | 8884# | 8889# | 8893# | 8907# | 8908# | 8926# | 8929# |
| | 8968# | 8969# | 8991# | 9101# | 9121# | 9150# | 9170# | 9196# | 9218# | 9256# | 9276# | 9345# | 9360# |
| | 9401# | 9417# | 9481# | 9496# | 9516# | 9532# | 9596# | 9612# | 9679# | 9695# | 9742# | 9758# | 9825# |
| | 9838# | 9857# | 9874# | 10003# | 10027# | 10104# | 10122# | 10156# | 10174# | 10224# | 10239# | 10256# | 10277# |
| | 10309# | 10328# | 10359# | 10382# | 10429# | 10456# | 10463# | 10489# | 10492# | 10513# | | | |
| T\$NSO = 000000 | 5412# | 10513 | | | | | | | | | | | |
| T\$NS1 = 000005 | 5491# | 5505 | 5519# | 5522 | 7624# | 7626 | 7630# | 7639 | 7645# | 7658 | 7664# | 7677 | 7683# |
| | 7697 | 7703# | 7716 | 7722# | 7734 | 7740# | 7754 | 7760# | 7768 | 7783# | 7785 | 7799# | 7803 |
| | 7817# | 7883 | 7897# | 7909 | 7923# | 7926 | 7940# | 7945 | 7963# | 7964 | 7985# | 7998 | 8010# |
| | 8023 | 8038# | 8061 | 8072# | 8088 | 8099# | 8120 | 8136# | 8156 | 8173# | 8202 | 8217# | 8256 |
| | 8268# | 8290 | 8302# | 8324 | 8336# | 8358 | 8370# | 8392 | 8419# | 8493 | 8509# | 8564 | 8580# |
| | 8625 | 8646# | 8680 | 8697# | 8730 | 8744# | 8777 | 8802# | 8841 | 8858# | 8908 | 8926# | 8969 |
| | 8991# | 9101 | 9121# | 9150 | 9170# | 9196 | 9218# | 9256 | 9276# | 9345 | 9360# | 9401 | 9417# |
| | 9481 | 9496# | 9516 | 9532# | 9596 | 9612# | 9679 | 9695# | 9742 | 9758# | 9825 | 9838# | 9857 |
| | 9874# | 10003 | 10027# | 10104 | 10122# | 10156 | 10174# | 10224 | 10239# | 10256 | 10277# | 10309 | 10328# |
| | 10359 | 10382# | 10429 | 10456# | 10463 | 10489# | 10492 | | | | | | |
| T\$NS2 = 000002 | 8043# | 8057 | 8273# | 8286 | 8307# | 8320 | 8341# | 8354 | 8375# | 8388 | 8650# | 8676 | 8701# |
| | 8726 | 8748# | 8773 | 8808# | 8836 | 8867# | 8889 | 8893# | 8907 | 8929# | 8968 | | |
| T\$NS3 = 000003 | 8869# | 8884 | | | | | | | | | | | |
| T\$PTNU= 000000 | 5406# | | | | | | | | | | | | |
| T\$SAVL- 177777 | 5406# | | | | | | | | | | | | |
| T\$SEGL= 177777 | 5406# | 8043# | 8057# | 8273# | 8286# | 8307# | 8320# | 8341# | 8354# | 8375# | 8388# | 8650# | 8665 |
| | 8676# | 8701# | 8716 | 8726# | 8748# | 8763 | 8773# | 8808# | 8825 | 8836# | 8869# | 8884# | |
| T\$SEK0= 010000 | 8043# | 8057 | 8273# | 8286 | 8307# | 8320 | 8341# | 8354 | 8375# | 8388 | 8650# | 8665 | 8676 |
| | 8701# | 8716 | 8726 | 8748# | 8763 | 8773 | 8808# | 8825 | 8836 | 8869# | 8884 | | |
| T\$SUBN= 000000 | 5406# | 7985# | 8010# | 8038# | 8072# | 8099# | 8136# | 8173# | 8217# | 8268# | 8302# | 8336# | 8370# |
| | 8419# | 8509# | 8580# | 8646# | 8697# | 8744# | 8802# | 8858# | 8867# | 8893# | 8926# | 8929# | 8991# |
| | 9121# | 9170# | 9218# | 9276# | 9360# | 9417# | 9496# | 9532# | 9612# | 9695# | 9758# | 9838# | 9874# |
| | 10027# | 10122# | 10174# | 10239# | 10277# | 10328# | 10382# | | | | | | |
| T\$TAGL= 177777 | 5406# | | | | | | | | | | | | |
| T\$TAGN= 010101 | 5406# | 5491# | 5519# | 7624# | 7630# | 7645# | 7664# | 7683# | 7703# | 7722# | 7740# | 7760# | 7783# |
| | 7799# | 7817# | 7897# | 7923# | 7940# | 7963# | 7985# | 8010# | 8038# | 8072# | 8099# | 8136# | 8173# |
| | 8217# | 8268# | 8302# | 8336# | 8370# | 8419# | 8509# | 8580# | 8646# | 8697# | 8744# | 8802# | 8858# |
| | 8867# | 8893# | 8926# | 8929# | 8991# | 9121# | 9170# | 9218# | 9276# | 9360# | 9417# | 9496# | 9532# |
| | 9612# | 9695# | 9758# | 9838# | 9874# | 10027# | 10122# | 10174# | 10239# | 10277# | 10328# | 10382# | 10456# |
| | 10489# | | | | | | | | | | | | |
| T\$TEMP= 000000 | 5469# | 5505# | 5522# | 7626# | 7639# | 7658# | 7677# | 7697# | 7716# | 7734# | 7754# | 7768# | 7785# |
| | 7803# | 7883# | 7909# | 7926# | 7945# | 7964# | 7998# | 8023# | 8057# | 8061# | 8088# | 8120# | 8156# |
| | 8197# | 8202# | 8249# | 8256# | 8286# | 8290# | 8320# | 8324# | 8354# | 8358# | 8388# | 8392# | 8493# |
| | 8522# | 8536# | 8548# | 8559# | 8564# | 8608# | 8618# | 8625# | 8665# | 8676# | 8680# | 8716# | 8726# |
| | 8730# | 8763# | 8773# | 8777# | 8825# | 8836# | 8841# | 8884# | 8889# | 8907# | 8908# | 8953# | 8963# |
| | 8968# | 8969# | 9068# | 9091# | 9101# | 9150# | 9196# | 9256# | 9345# | 9401# | 9481# | 9516# | 9596# |
| | 9679# | 9742# | 9825# | 9857# | 10003# | 10104# | 10156# | 10224# | 10256# | 10309# | 10359# | 10429# | 10458# |
| | 10459# | 10460# | 10461# | 10463# | 10492# | 10513# | | | | | | | |
| T\$TEST= 000052 | 5406# | 7985# | 8010# | 8038# | 8072# | 8099# | 8136# | 8173# | 8217# | 8268# | 8302# | 8336# | 8370# |
| | 8419# | 8509# | 8580# | 8646# | 8697# | 8744# | 8802# | 8858# | 8867 | 8893 | 8926# | 8929 | 8991# |
| | 9121# | 9170# | 9218# | 9276# | 9360# | 9417# | 9496# | 9532# | 9612# | 9695# | 9758# | 9838# | 9874# |
| | 10027# | 10122# | 10174# | 10239# | 10277# | 10328# | 10382# | 10515 | | | | | |
| T\$TSTM= 177777 | 5406# | 6678 | 6684 | 6694 | 6700 | 6807 | 6813 | 6971 | 7003 | 7009 | 7019 | 7025 | 7060 |
| | 7066 | 7100 | 7106 | 7114 | 7120 | 7266 | 7271 | 7290 | 7296 | 7305 | 7311 | 7320 | 7326 |
| | 7335 | 7341 | 7473 | 7479 | 7625 | 7626 | 7631 | 7632 | 7633 | 7634 | 7635 | 7636 | 7637 |
| | 7638 | 7639 | 7646 | 7647 | 7648 | 7649 | 7650 | 7651 | 7652 | 7653 | 7654 | 7655 | 7656 |
| | 7657 | 7658 | 7665 | 7666 | 7667 | 7668 | 7669 | 7670 | 7671 | 7672 | 7673 | 7674 | 7675 |
| | 7676 | 7677 | 7684 | 7685 | 7686 | 7687 | 7688 | 7689 | 7690 | 7691 | 7692 | 7693 | 7694 |

| | | | | | | | | | | | | | | | | | | | |
|---------|----------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--|--|--|--|--|
| T20 | 026066 G | 5469 | 8858# | | | | | | | | | | | | | | | | |
| T20.1 | 026116 | 8867# | | | | | | | | | | | | | | | | | |
| T20.2 | 026224 | 8893# | | | | | | | | | | | | | | | | | |
| T21 | 026320 G | 5469 | 8926# | | | | | | | | | | | | | | | | |
| T21.1 | 026332 | 8929# | | | | | | | | | | | | | | | | | |
| T22 | 026572 G | 5469 | 8991# | | | | | | | | | | | | | | | | |
| T23 | 027250 G | 5469 | 9121# | | | | | | | | | | | | | | | | |
| T24 | 027370 G | 5469 | 9170# | | | | | | | | | | | | | | | | |
| T25 | 027500 G | 5469 | 9218# | | | | | | | | | | | | | | | | |
| T26 | 027674 G | 5469 | 9276# | | | | | | | | | | | | | | | | |
| T27 | 030240 G | 5469 | 9360# | | | | | | | | | | | | | | | | |
| T28 | 030450 G | 5469 | 9417# | | | | | | | | | | | | | | | | |
| T29 | 031000 G | 5469 | 9496# | | | | | | | | | | | | | | | | |
| T3 | 021756 G | 5469 | 8038# | | | | | | | | | | | | | | | | |
| T30 | 031100 G | 5469 | 9532# | | | | | | | | | | | | | | | | |
| T31 | 031450 G | 5469 | 9612# | | | | | | | | | | | | | | | | |
| T32 | 032012 G | 5469 | 9695# | | | | | | | | | | | | | | | | |
| T33 | 032262 G | 5469 | 9758# | | | | | | | | | | | | | | | | |
| T34 | 032624 G | 5469 | 9838# | | | | | | | | | | | | | | | | |
| T35 | 032754 G | 5469 | 9874# | | | | | | | | | | | | | | | | |
| T36 | 033666 G | 5469 | 10027# | | | | | | | | | | | | | | | | |
| T37 | 034150 G | 5469 | 10122# | | | | | | | | | | | | | | | | |
| T38 | 034350 G | 5469 | 10174# | | | | | | | | | | | | | | | | |
| T39 | 034650 G | 5469 | 10239# | | | | | | | | | | | | | | | | |
| T4 | 022102 G | 5469 | 8072# | | | | | | | | | | | | | | | | |
| T40 | 034740 G | 5469 | 10277# | | | | | | | | | | | | | | | | |
| T41 | 035102 G | 5469 | 10328# | | | | | | | | | | | | | | | | |
| T42 | 035250 G | 5469 | 10382# | | | | | | | | | | | | | | | | |
| T5 | 022204 G | 5469 | 8099# | | | | | | | | | | | | | | | | |
| T6 | 022324 G | 5469 | 8136# | | | | | | | | | | | | | | | | |
| T7 | 022444 G | 5469 | 8173# | | | | | | | | | | | | | | | | |
| T8 | 022620 G | 5469 | 8217# | | | | | | | | | | | | | | | | |
| T9 | 023054 G | 5469 | 8268# | | | | | | | | | | | | | | | | |
| UAM = | 000200 G | 5549# | | | | | | | | | | | | | | | | | |
| UNIT | 002440 | 5923# | | | | | | | | | | | | | | | | | |
| UNRR = | 000001 | 5657# | 7262 | 9291 | 9300 | 9312 | 9331 | 9339 | | | | | | | | | | | |
| UPBITS | 002550 | 5974# | 8185 | 8230 | 8236 | 8277 | 8311 | 8345 | 8379 | | | | | | | | | | |
| VECTOR | 035606 | 10459 | 10466# | | | | | | | | | | | | | | | | |
| V35 = | 000020 | 5782# | 5786 | | | | | | | | | | | | | | | | |
| WAIT50 | 005120 | 6717# | 6859 | 6917 | 6960 | 7181 | 7410 | 7430 | 8940 | 9005 | 9029 | 9308 | 9506 | 9549 | | | | | |
| | | 9629 | 9712 | 9775 | 9998 | 10132 | 10184 | 10288 | 10338 | 10400 | | | | | | | | | |
| WAX = | 000010 | 5618# | 5690# | 6606 | | | | | | | | | | | | | | | |
| WAX15 | 002374 | 5905# | 6596* | 6597 | 6848* | 6850 | 7371* | 7380* | 8513* | 8589* | 8591 | 8652* | 8654* | 8658 | | | | | |
| | | 8660 | 8703* | 8705* | 8709 | 8711 | 8750* | 8752* | 8756 | 8758 | 8810* | 8814 | 9366* | 9432* | | | | | |
| | | 9896* | 9908* | 9936* | 10134* | 10186* | | | | | | | | | | | | | |
| WAX16 | 002376 | 5906# | 6600* | 6601 | 6851* | 7372* | 7382* | 7383 | 8514* | 8590* | 8592 | 8653* | 8655* | 8704* | | | | | |
| | | 8706* | 8718 | 8720 | 8751* | 8753* | 8765 | 8767 | 8811* | 9367* | 9375* | 9380* | 9385* | 9433* | | | | | |
| | | 9438* | 9443* | 9448* | 9453* | 9458* | 9463* | 9468* | 9897* | 9909* | 9914* | 9937* | 10135* | 10139* | | | | | |
| | | 10143* | 10147* | 10189* | 10193* | 10197* | 10201* | 10205* | 10209* | 10213* | 10217* | | | | | | | | |
| WRDYTO= | 000002 | 5869# | 6589 | 6615 | 8516 | | | | | | | | | | | | | | |
| WRIBYT | 002366 | 5902# | 6490* | 6491 | 6553* | 6554* | 6555* | 6556* | 6591* | 6592* | 6593* | 6597* | 6601* | 6604* | | | | | |
| | | 6605* | 6606* | 6607* | 6747* | 6750* | 6855* | 6958* | 7217* | 7218* | 7224* | 7376* | 8044* | 8045* | | | | | |
| | | 8049 | 8051 | 8075* | 8102* | 8177* | 8228* | 8229 | 8274* | 8278 | 8280 | 8308* | 8312 | 8314 | | | | | |
| | | 8342* | 8346 | 8348 | 8376* | 8380 | 8382 | 8870* | 8894* | 8934* | 8937* | 9224* | 9542* | 9622* | | | | | |
| | | 9674* | 9705* | 9737* | 9768* | 9820* | 9841* | 9886* | 9890* | 9993* | 9996* | 10351* | | | | | | | |
| WRITAX | 004264 | 6587# | 6852 | 7373 | 7384 | 8515 | 8593 | 8656 | 8707 | 8754 | 8812 | 9368 | 9376 | 9381 | | | | | |

CZDMRA M8203 STATIC DIAG #1
CZDMRA.P11 18-JUL-79 09:44

MACY11 30A(1052) 18-JUL-79 09:53 PAGE 8-19
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0193

| | | | | | | | | | | | | | |
|-----------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|--------|-------|------|-------|
| | 9386 | 9434 | 9439 | 9444 | 9449 | 9454 | 9459 | 9464 | 9469 | 9898 | 9917 | 9938 | 10136 |
| WRITLU 003722 | 10140 | 10144 | 10148 | 10190 | 10194 | 10198 | 10202 | 10206 | 10210 | 10214 | 10218 | | |
| | 6484# | 6557 | 6594 | 6598 | 6602 | 6608 | 6748 | 6751 | 6856 | 6959 | 7219 | 7225 | 7377 |
| | 8046 | 8076 | 8103 | 8178 | 8231 | 8275 | 8309 | 8343 | 8377 | 8871 | 8895 | 8935 | 8938 |
| | 8939 | 9225 | 9543 | 9623 | 9675 | 9706 | 9738 | 9769 | 9821 | 9842 | 9887 | 9891 | 9994 |
| | 9997 | 10352 | | | | | | | | | | | |
| XYZ = 000100 | 5781# | 5786 | | | | | | | | | | | |
| X\$ALWA= 000000 | 5406# | | | | | | | | | | | | |
| X\$FALS= 000040 | 5406# | | | | | | | | | | | | |
| X\$OFFS= 000400 | 5406# | | | | | | | | | | | | |
| X\$TRUE= 000020 | 5406# | | | | | | | | | | | | |
| \$LSTIN= 000001 | 5415# | | | | | | | | | | | | |
| \$LSTTA= 000001 | 5416# | | | | | | | | | | | | |
| . = 036172 | 5398# | 5886# | 6360# | 6388# | 7618# | 7948# | 8197 | 8249 | 8522 | 8536 | 8548 | 8559 | 8608 |
| | 8618 | 8665 | 8716 | 8763 | 8825 | 8953 | 8963 | 9068 | 9091 | 10505# | | | |

| | | | | | | | | | | | | | | | |
|--------|-------|-------|--------|--------|--------|--------|-------|--------|--------|--------|--------|--------|--------|--------|--------|
| ENDSRV | 932# | 5406# | | | | | | | | | | | | | |
| ENDSUB | 952# | 5406# | 8889 | 8907 | 8968 | | | | | | | | | | |
| ENDSW | 974# | 5406# | 5522 | | | | | | | | | | | | |
| ENDTST | 988# | 5406# | 7998 | 8023 | 8061 | 8088 | 8120 | 8156 | 8202 | 8256 | 8290 | 8324 | 8358 | 8392 | 8493 |
| | 8564 | 8625 | 8680 | 8730 | 8777 | 8841 | 8908 | 8969 | 9101 | 9150 | 9196 | 9256 | 9345 | 9401 | 9481 |
| | 9516 | 9596 | 9679 | 9742 | 9825 | 9857 | 10003 | 10104 | 10156 | 10224 | 10256 | 10309 | 10359 | 10429 | |
| EQUALS | 1009# | 5406# | 5549 | | | | | | | | | | | | |
| ERRDF | 1087# | 5406# | 6678 | 6684 | 6694 | 6700 | 6807 | 6813 | 6971 | 7003 | 7009 | 7019 | 7025 | 7060 | 7066 |
| | 7100 | 7106 | 7114 | 7120 | 7266 | 7271 | 7290 | 7296 | 7305 | 7311 | 7320 | 7326 | 7335 | 7341 | 7473 |
| | 7479 | 7995 | 8021 | 8055 | 8086 | 8118 | 8154 | 8196 | 8248 | 8284 | 8318 | 8352 | 8386 | 8450 | 8469 |
| | 8488 | 8521 | 8535 | 8547 | 8558 | 8607 | 8617 | 8664 | 8674 | 8715 | 8724 | 8762 | 8771 | 8824 | 8834 |
| | 8882 | 8905 | 8952 | 8962 | 9067 | 9090 | 9252 | 9295 | 9304 | 9316 | 9335 | 9343 | 9566 | 9581 | 9664 |
| | 9729 | 9810 | 9854 | 9958 | | | | | | | | | | | |
| ERRHRD | 1099# | 5406# | | | | | | | | | | | | | |
| ERROR | 1109# | 5406# | | | | | | | | | | | | | |
| ERRSF | 1118# | 5406# | | | | | | | | | | | | | |
| ERRSOF | 1130# | 5406# | | | | | | | | | | | | | |
| ERRTBL | 1140# | 5406# | | | | | | | | | | | | | |
| ESCAPE | 1156# | 5406# | 8197 | 8249 | 8522 | 8536 | 8548 | 8559 | 8608 | 8618 | 8665 | 8716 | 8763 | 8825 | 8953 |
| | 8963 | 9068 | 9091 | | | | | | | | | | | | |
| EXIT | 1186# | 5406# | | | | | | | | | | | | | |
| FEQUAL | 1228# | 5406# | | | | | | | | | | | | | |
| GETBYT | 1246# | 5406# | | | | | | | | | | | | | |
| GETPRI | 1264# | 5406# | | | | | | | | | | | | | |
| GETWOR | 1256# | 5406# | | | | | | | | | | | | | |
| GMANIA | 1286# | 5406# | | | | | | | | | | | | | |
| GMANID | 1299# | 5406# | | | | | | | | | | | | | |
| GMANIL | 1315# | 5406# | | | | | | | | | | | | | |
| GPHARD | 1328# | 5406# | 7860 | | | | | | | | | | | | |
| GPRMA | 1340# | 5406# | 10458 | 10459 | | | | | | | | | | | |
| GPRMD | 1372# | 5406# | 10460 | 10461 | | | | | | | | | | | |
| GPRML | 1407# | 5406# | | | | | | | | | | | | | |
| HEADER | 1432# | 5406# | 5446 | | | | | | | | | | | | |
| INLOOP | 1446# | 5406# | | | | | | | | | | | | | |
| IOSETU | 1453# | 5406# | | | | | | | | | | | | | |
| IOSTAR | 1466# | 5406# | | | | | | | | | | | | | |
| KT11 | 1488# | 5406# | | | | | | | | | | | | | |
| LASTAD | 1659# | 5406# | 10515 | | | | | | | | | | | | |
| MANUAL | 1677# | 5406# | | | | | | | | | | | | | |
| MEMORY | 1685# | 5406# | | | | | | | | | | | | | |
| MSBYTE | 2901# | 5406# | 5446# | | | | | | | | | | | | |
| MSCHEC | 3206# | 5406# | | | | | | | | | | | | | |
| MSCNTO | 3279# | 5406# | 10458# | 10459# | 10460# | 10461# | | | | | | | | | |
| MSCOUN | 3124# | 5406# | 7625# | 7631# | 7632# | 7633# | 7634# | 7635# | 7636# | 7637# | 7638# | 7646# | 7647# | 7648# | 7649# |
| | 7650# | 7651# | 7652# | 7653# | 7654# | 7655# | 7656# | 7657# | 7665# | 7666# | 7667# | 7668# | 7669# | 7670# | 7671# |
| | 7672# | 7673# | 7674# | 7675# | 7676# | 7684# | 7685# | 7686# | 7687# | 7688# | 7689# | 7690# | 7691# | 7692# | 7693# |
| | 7694# | 7695# | 7696# | 7704# | 7705# | 7706# | 7707# | 7708# | 7709# | 7710# | 7711# | 7712# | 7713# | 7714# | 7715# |
| | 7723# | 7724# | 7725# | 7726# | 7727# | 7728# | 7729# | 7730# | 7731# | 7732# | 7733# | 7741# | 7742# | 7743# | 7744# |
| | 7745# | 7746# | 7747# | 7748# | 7749# | 7750# | 7751# | 7752# | 7753# | 7761# | 7762# | 7763# | 7764# | 7765# | 7766# |
| | 7767# | 7944# | 8426# | 8452# | 8471# | 8490# | | | | | | | | | |
| MSDATA | 2614# | 5406# | 5446# | 6383# | 6388# | | | | | | | | | | |
| MSDECR | 3063# | 5406# | 5505# | 5522# | 7626# | 7639# | 7658# | 7677# | 7697# | 7716# | 7734# | 7754# | 7768# | 7785# | 7803# |
| | 7883# | 7909# | 7926# | 7945# | 7964# | 7998# | 8023# | 8057# | 8061# | 8088# | 8120# | 8156# | 8202# | 8256# | 8286# |
| | 8290# | 8320# | 8324# | 8354# | 8358# | 8388# | 8392# | 8493# | 8564# | 8625# | 8676# | 8680# | 8726# | 8730# | 8773# |
| | 8777# | 8836# | 8841# | 8884# | 8889# | 8907# | 8908# | 8968# | 8969# | 9101# | 9150# | 9196# | 9256# | 9345# | 9401# |
| | 9481# | 9516# | 9596# | 9679# | 9742# | 9825# | 9857# | 10003# | 10104# | 10156# | 10224# | 10256# | 10309# | 10359# | 10429# |

| | | | | | | | | | | | | | | | | | | |
|---------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--|--|--|
| | 10463# | 10492# | 10513# | | | | | | | | | | | | | | | |
| MSDEFA | 3263# | 5406# | 10458# | 10459# | 10460# | 10461# | | | | | | | | | | | | |
| MSBENDE | 3145# | 5406# | 5505# | 5522# | 7626# | 7639# | 7658# | 7677# | 7697# | 7716# | 7734# | 7754# | 7768# | 7785# | 7883# | | | |
| | 7909# | 7926# | 7945# | 7964# | 7998# | 8023# | 8057# | 8061# | 8088# | 8120# | 8156# | 8202# | 8256# | 8286# | 8290# | | | |
| | 8320# | 8324# | 8354# | 8358# | 8388# | 8392# | 8493# | 8564# | 8625# | 8676# | 8680# | 8726# | 8730# | 8773# | 8777# | | | |
| | 8836# | 8841# | 8884# | 8889# | 8907# | 8908# | 8968# | 8969# | 9101# | 9150# | 9196# | 9256# | 9345# | 9401# | 9481# | | | |
| | 9516# | 9596# | 9679# | 9742# | 9825# | 9857# | 10003# | 10104# | 10156# | 10224# | 10256# | 10309# | 10359# | 10429# | 10463# | | | |
| MSERRI | 10492# | 10513# | | | | | | | | | | | | | | | | |
| | 2365# | 5406# | 6678# | 6684# | 6694# | 6700# | 6807# | 6813# | 6971# | 7003# | 7009# | 7019# | 7025# | 7060# | 7066# | | | |
| | 7100# | 7106# | 7114# | 7120# | 7266# | 7271# | 7290# | 7296# | 7305# | 7311# | 7320# | 7326# | 7335# | 7341# | 7473# | | | |
| | 7479# | 7995# | 8021# | 8055# | 8086# | 8118# | 8154# | 8196# | 8248# | 8284# | 8318# | 8352# | 8386# | 8450# | 8469# | | | |
| | 8488# | 8521# | 8535# | 8547# | 8558# | 8607# | 8617# | 8664# | 8674# | 8715# | 8724# | 8762# | 8771# | 8824# | 8834# | | | |
| | 8882# | 8905# | 8952# | 8962# | 9067# | 9090# | 9252# | 9295# | 9304# | 9316# | 9335# | 9343# | 9566# | 9581# | 9664# | | | |
| | 9729# | 9810# | 9854# | 9958# | | | | | | | | | | | | | | |
| MSDESCA | 2921# | 5406# | 8197# | 8249# | 8522# | 8536# | 8548# | 8559# | 8608# | 8618# | 8665# | 8716# | 8763# | 8825# | 8953# | | | |
| | 8963# | 9068# | 9091# | | | | | | | | | | | | | | | |
| MSDESCS | 2932# | 5406# | 8197# | 8249# | 8522# | 8536# | 8548# | 8559# | 8608# | 8618# | 8665# | 8716# | 8763# | 8825# | 8953# | | | |
| | 8963# | 9068# | 9091# | | | | | | | | | | | | | | | |
| MSXCP | 3186# | 5406# | 10458# | 10459# | 10460# | 10461# | | | | | | | | | | | | |
| MSXIT | 2943# | 5406# | | | | | | | | | | | | | | | | |
| MSXSE | 2965# | 5406# | | | | | | | | | | | | | | | | |
| MSXTJ | 2954# | 5406# | | | | | | | | | | | | | | | | |
| MSGEN | 3087# | 5406# | 5412# | 5446# | 5469# | 5491# | 5505# | 5519# | 5522# | 6383# | 6388# | 7624# | 7626# | 7630# | 7639# | | | |
| | 7645# | 7658# | 7664# | 7677# | 7683# | 7697# | 7703# | 7716# | 7722# | 7734# | 7740# | 7754# | 7760# | 7768# | 7783# | | | |
| | 7785# | 7799# | 7817# | 7883# | 7897# | 7909# | 7923# | 7926# | 7940# | 7945# | 7963# | 7964# | 7985# | 7998# | 8010# | | | |
| | 8023# | 8038# | 8057# | 8061# | 8072# | 8088# | 8099# | 8120# | 8136# | 8156# | 8173# | 8202# | 8217# | 8256# | 8268# | | | |
| | 8286# | 8290# | 8302# | 8320# | 8324# | 8336# | 8354# | 8358# | 8370# | 8388# | 8392# | 8419# | 8493# | 8509# | 8564# | | | |
| | 8580# | 8625# | 8646# | 8676# | 8680# | 8697# | 8726# | 8730# | 8744# | 8773# | 8777# | 8802# | 8836# | 8841# | 8858# | | | |
| | 8867# | 8884# | 8889# | 8893# | 8907# | 8908# | 8926# | 8929# | 8968# | 8969# | 8991# | 9101# | 9121# | 9150# | 9170# | | | |
| | 9196# | 9218# | 9256# | 9276# | 9345# | 9360# | 9401# | 9417# | 9481# | 9496# | 9516# | 9532# | 9596# | 9612# | 9679# | | | |
| | 9695# | 9742# | 9758# | 9825# | 9838# | 9857# | 9874# | 10003# | 10027# | 10104# | 10122# | 10156# | 10174# | 10224# | 10239# | | | |
| | 10256# | 10277# | 10309# | 10328# | 10359# | 10382# | 10429# | 10456# | 10463# | 10489# | 10492# | 10515# | | | | | | |
| MSGENB | 2764# | 5406# | | | | | | | | | | | | | | | | |
| MSGETS | 3079# | 5406# | 5505# | 5522# | 7626# | 7639# | 7658# | 7677# | 7697# | 7716# | 7734# | 7754# | 7768# | 7785# | 7803# | | | |
| | 7883# | 7909# | 7926# | 7945# | 7964# | 7998# | 8023# | 8057# | 8061# | 8088# | 8120# | 8156# | 8202# | 8256# | 8286# | | | |
| | 8290# | 8320# | 8324# | 8354# | 8358# | 8388# | 8392# | 8493# | 8564# | 8625# | 8665# | 8676# | 8680# | 8716# | 8726# | | | |
| | 8730# | 8763# | 8773# | 8777# | 8825# | 8836# | 8841# | 8884# | 8889# | 8907# | 8908# | 8968# | 8969# | 9101# | 9150# | | | |
| | 9196# | 9256# | 9345# | 9401# | 9481# | 9516# | 9596# | 9679# | 9742# | 9825# | 9857# | 10003# | 10104# | 10156# | 10224# | | | |
| | 10256# | 10309# | 10359# | 10429# | 10463# | 10492# | 10513# | | | | | | | | | | | |
| MSGETT | 2634# | 5406# | 8197# | 8249# | 8522# | 8536# | 8548# | 8559# | 8608# | 8618# | 8665# | 8716# | 8763# | 8825# | 8953# | | | |
| | 8963# | 9068# | 9091# | | | | | | | | | | | | | | | |
| MSGNGB | 2689# | 5406# | 5412# | 5446# | 5469# | 5491# | 5519# | 6383# | 6388# | 7624# | 7630# | 7645# | 7664# | 7683# | 7703# | | | |
| | 7722# | 7740# | 7760# | 7783# | 7799# | 7817# | 7897# | 7923# | 7940# | 7963# | 10456# | 10489# | 10515# | | | | | |
| MSGNIN | 3101# | 5406# | 5446# | 5469# | 5491# | 5519# | 6383# | 6388# | 6678# | 6684# | 6694# | 6700# | 6807# | 6813# | 6971# | | | |
| | 7003# | 7009# | 7019# | 7025# | 7060# | 7066# | 7100# | 7106# | 7114# | 7120# | 7266# | 7271# | 7290# | 7296# | 7305# | | | |
| | 7311# | 7320# | 7326# | 7335# | 7341# | 7473# | 7479# | 7625# | 7626# | 7631# | 7632# | 7633# | 7634# | 7635# | 7636# | | | |
| | 7637# | 7638# | 7639# | 7646# | 7647# | 7648# | 7649# | 7650# | 7651# | 7652# | 7653# | 7654# | 7655# | 7656# | 7657# | | | |
| | 7658# | 7665# | 7666# | 7667# | 7668# | 7669# | 7670# | 7671# | 7672# | 7673# | 7674# | 7675# | 7676# | 7677# | 7684# | | | |
| | 7685# | 7686# | 7687# | 7688# | 7689# | 7690# | 7691# | 7692# | 7693# | 7694# | 7695# | 7696# | 7697# | 7704# | 7705# | | | |
| | 7706# | 7707# | 7708# | 7709# | 7710# | 7711# | 7712# | 7713# | 7714# | 7715# | 7716# | 7723# | 7724# | 7725# | 7726# | | | |
| | 7727# | 7728# | 7729# | 7730# | 7731# | 7732# | 7733# | 7734# | 7741# | 7742# | 7743# | 7744# | 7745# | 7746# | 7747# | | | |
| | 7748# | 7749# | 7750# | 7751# | 7752# | 7753# | 7754# | 7761# | 7762# | 7763# | 7764# | 7765# | 7766# | 7767# | 7768# | | | |
| | 7785# | 7834# | 7835# | 7837# | 7838# | 7840# | 7841# | 7843# | 7844# | 7860# | 7861# | 7883# | 7899# | 7906# | 7909# | | | |
| | 7926# | 7942# | 7944# | 7945# | 7964# | 7987# | 7995# | 7998# | 8021# | 8023# | 8043# | 8055# | 8057# | 8061# | 8086# | | | |
| | 8088# | 8104# | 8118# | 8120# | 8154# | 8156# | 8196# | 8197# | 8202# | 8248# | 8249# | 8256# | 8273# | 8284# | 8286# | | | |
| | 8290# | 8307# | 8318# | 8320# | 8324# | 8341# | 8352# | 8354# | 8358# | 8375# | 8386# | 8388# | 8392# | 8426# | 8450# | | | |

| | | | | | | | | | | | | | | | |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| | 8452# | 8469# | 8471# | 8488# | 8490# | 8493# | 8521# | 8522# | 8535# | 8536# | 8547# | 8548# | 8558# | 8559# | 8564# |
| | 8607# | 8608# | 8617# | 8618# | 8625# | 8650# | 8664# | 8665# | 8674# | 8676# | 8680# | 8701# | 8715# | 8716# | 8724# |
| | 8726# | 8730# | 8748# | 8762# | 8763# | 8771# | 8773# | 8777# | 8808# | 8824# | 8825# | 8834# | 8836# | 8841# | 8867# |
| | 8869# | 8882# | 8884# | 8889# | 8893# | 8905# | 8907# | 8908# | 8929# | 8952# | 8953# | 8962# | 8963# | 8968# | 8969# |
| | 9067# | 9068# | 9090# | 9091# | 9101# | 9150# | 9196# | 9252# | 9256# | 9295# | 9304# | 9316# | 9335# | 9343# | 9345# |
| | 9401# | 9481# | 9516# | 9566# | 9581# | 9596# | 9664# | 9679# | 9729# | 9742# | 9810# | 9825# | 9854# | 9857# | 9958# |
| | 10003# | 10104# | 10156# | 10224# | 10256# | 10309# | 10359# | 10429# | 10456# | 10458# | 10459# | 10460# | 10461# | 10463# | 10489# |
| | 10492# | 10515# | | | | | | | | | | | | | |
| MSGNLS | 2717# | 5406# | 8057# | 8286# | 8320# | 8354# | 8388# | 8676# | 8726# | 8773# | 8836# | 8884# | | | |
| MSGNSU | 2679# | 5406# | 8867# | 8893# | 8929# | | | | | | | | | | |
| MSGNTA | 2659# | 5406# | 5505# | 5522# | 7626# | 7639# | 7658# | 7677# | 7697# | 7716# | 7734# | 7754# | 7768# | 7785# | 7803# |
| | 7909# | 7926# | 7945# | 7964# | 7998# | 8023# | 8061# | 8088# | 8120# | 8156# | 8202# | 8256# | 8290# | 8324# | 8358# |
| | 8392# | 8493# | 8564# | 8625# | 8680# | 8730# | 8777# | 8841# | 8889# | 8907# | 8908# | 8968# | 8969# | 9101# | 9150# |
| | 9196# | 9256# | 9345# | 9401# | 9481# | 9516# | 9596# | 9679# | 9742# | 9825# | 9857# | 10003# | 10104# | 10156# | 10224# |
| MSGNTE | 10256# | 10309# | 10359# | 10429# | 10463# | 10492# | | | | | | | | | |
| | 2669# | 5406# | 7985# | 8010# | 8038# | 8072# | 8099# | 8136# | 8173# | 8217# | 8268# | 8302# | 8336# | 8370# | 8419# |
| | 8509# | 8580# | 8646# | 8697# | 8744# | 8802# | 8858# | 8926# | 8991# | 9121# | 9170# | 9218# | 9276# | 9360# | 9417# |
| | 9496# | 9532# | 9612# | 9695# | 9758# | 9838# | 9874# | 10027# | 10122# | 10174# | 10239# | 10277# | 10328# | 10382# | |
| MSHAPT | 2477# | 5406# | 5446# | | | | | | | | | | | | |
| MSHNAP | 2565# | 5406# | 5446# | | | | | | | | | | | | |
| MSINCR | 3054# | 5406# | 5412# | 5491# | 5519# | 6678# | 6684# | 6694# | 6700# | 6807# | 6813# | 6971# | 7003# | 7009# | 7019# |
| | 7025# | 7060# | 7066# | 7100# | 7106# | 7114# | 7120# | 7266# | 7271# | 7290# | 7296# | 7305# | 7311# | 7320# | 7326# |
| | 7335# | 7341# | 7473# | 7479# | 7624# | 7625# | 7626# | 7630# | 7631# | 7632# | 7633# | 7634# | 7635# | 7636# | 7637# |
| | 7638# | 7639# | 7645# | 7646# | 7647# | 7648# | 7649# | 7650# | 7651# | 7652# | 7653# | 7654# | 7655# | 7656# | 7657# |
| | 7658# | 7664# | 7665# | 7666# | 7667# | 7668# | 7669# | 7670# | 7671# | 7672# | 7673# | 7674# | 7675# | 7676# | 7677# |
| | 7683# | 7684# | 7685# | 7686# | 7687# | 7688# | 7689# | 7690# | 7691# | 7692# | 7693# | 7694# | 7695# | 7696# | 7697# |
| | 7703# | 7704# | 7705# | 7706# | 7707# | 7708# | 7709# | 7710# | 7711# | 7712# | 7713# | 7714# | 7715# | 7716# | 7722# |
| | 7723# | 7724# | 7725# | 7726# | 7727# | 7728# | 7729# | 7730# | 7731# | 7732# | 7733# | 7734# | 7740# | 7741# | 7742# |
| | 7743# | 7744# | 7745# | 7746# | 7747# | 7748# | 7749# | 7750# | 7751# | 7752# | 7753# | 7754# | 7760# | 7761# | 7762# |
| | 7763# | 7764# | 7765# | 7766# | 7767# | 7768# | 7783# | 7785# | 7799# | 7817# | 7834# | 7837# | 7840# | 7843# | 7860# |
| | 7883# | 7897# | 7899# | 7906# | 7909# | 7923# | 7926# | 7940# | 7942# | 7944# | 7945# | 7963# | 7964# | 7985# | 7987# |
| | 7995# | 7998# | 8010# | 8021# | 8023# | 8038# | 8043# | 8055# | 8057# | 8061# | 8072# | 8086# | 8088# | 8099# | 8104# |
| | 8118# | 8120# | 8136# | 8154# | 8156# | 8173# | 8196# | 8197# | 8202# | 8217# | 8248# | 8249# | 8256# | 8268# | 8273# |
| | 8284# | 8286# | 8290# | 8302# | 8307# | 8318# | 8320# | 8324# | 8336# | 8341# | 8352# | 8354# | 8358# | 8370# | 8375# |
| | 8386# | 8388# | 8392# | 8419# | 8426# | 8450# | 8452# | 8469# | 8471# | 8488# | 8490# | 8493# | 8509# | 8521# | 8522# |
| | 8535# | 8536# | 8547# | 8548# | 8558# | 8559# | 8564# | 8580# | 8607# | 8608# | 8617# | 8618# | 8625# | 8646# | 8650# |
| | 8664# | 8665# | 8674# | 8676# | 8680# | 8697# | 8701# | 8715# | 8716# | 8724# | 8726# | 8730# | 8744# | 8748# | 8762# |
| | 8763# | 8771# | 8773# | 8777# | 8802# | 8808# | 8824# | 8825# | 8834# | 8836# | 8841# | 8858# | 8867# | 8869# | 8882# |
| | 8884# | 8889# | 8893# | 8905# | 8907# | 8908# | 8926# | 8929# | 8952# | 8953# | 8962# | 8963# | 8968# | 8969# | 8991# |
| | 9067# | 9068# | 9090# | 9091# | 9101# | 9121# | 9150# | 9170# | 9196# | 9218# | 9252# | 9256# | 9276# | 9295# | 9304# |
| | 9316# | 9335# | 9343# | 9345# | 9360# | 9401# | 9417# | 9481# | 9496# | 9516# | 9532# | 9566# | 9581# | 9596# | 9612# |
| | 9664# | 9679# | 9695# | 9729# | 9742# | 9758# | 9810# | 9825# | 9838# | 9854# | 9857# | 9874# | 9958# | 10003# | 10027# |
| MSIOSE | 10104# | 10122# | 10156# | 10174# | 10224# | 10239# | 10256# | 10277# | 10309# | 10328# | 10359# | 10382# | 10429# | 10456# | 10489# |
| MSLDRO | 2431# | 5406# | | | | | | | | | | | | | |
| MSMASK | 2771# | 5406# | 7834# | 7837# | 7840# | 7843# | 7860# | 7899# | 7906# | 7987# | | | | | |
| MSMCHI | 2390# | 5406# | | | | | | | | | | | | | |
| MSMCLO | 87# | 5406# | | | | | | | | | | | | | |
| MSMSK1 | 2327# | 5406# | | | | | | | | | | | | | |
| MSPOP | 2402# | 5406# | | | | | | | | | | | | | |
| | 2646# | 5406# | 5505# | 5522# | 7626# | 7639# | 7658# | 7677# | 7697# | 7716# | 7734# | 7754# | 7768# | 7785# | 7803# |
| | 7883# | 7909# | 7926# | 7945# | 7964# | 7998# | 8023# | 8057# | 8061# | 8088# | 8120# | 8156# | 8202# | 8256# | 8286# |
| | 8290# | 8320# | 8324# | 8354# | 8358# | 8388# | 8392# | 8493# | 8564# | 8625# | 8676# | 8680# | 8726# | 8730# | 8773# |
| | 8777# | 8836# | 8841# | 8884# | 8889# | 8907# | 8908# | 8968# | 8969# | 9101# | 9150# | 9196# | 9256# | 9345# | 9401# |
| | 9481# | 9516# | 9596# | 9679# | 9742# | 9825# | 9857# | 10003# | 10104# | 10156# | 10224# | 10256# | 10309# | 10359# | 10429# |
| MSPRIN | 10463# | 10492# | 10513# | | | | | | | | | | | | |
| | 2349# | 5406# | 7625# | 7631# | 7632# | 7633# | 7634# | 7635# | 7636# | 7637# | 7638# | 7646# | 7647# | 7648# | 7649# |

| | | | | | | | | | | | | | | | |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| | 7650# | 7651# | 7652# | 7653# | 7654# | 7655# | 7656# | 7657# | 7665# | 7666# | 7667# | 7668# | 7669# | 7670# | 7671# |
| | 7672# | 7673# | 7674# | 7675# | 7676# | 7684# | 7685# | 7686# | 7687# | 7688# | 7689# | 7690# | 7691# | 7692# | 7693# |
| | 7694# | 7695# | 7696# | 7704# | 7705# | 7706# | 7707# | 7708# | 7709# | 7710# | 7711# | 7712# | 7713# | 7714# | 7715# |
| | 7723# | 7724# | 7725# | 7726# | 7727# | 7728# | 7729# | 7730# | 7731# | 7732# | 7733# | 7741# | 7742# | 7743# | 7744# |
| | 7745# | 7746# | 7747# | 7748# | 7749# | 7750# | 7751# | 7752# | 7753# | 7761# | 7762# | 7763# | 7764# | 7765# | 7766# |
| MSPUSH | 2337# | 5406# | 5412# | 5491# | 5519# | 7624# | 7630# | 7645# | 7664# | 7683# | 7703# | 7722# | 7740# | 7760# | 7783# |
| | 7799# | 7817# | 7897# | 7923# | 7940# | 7963# | 7985# | 8010# | 8038# | 8043# | 8072# | 8099# | 8136# | 8173# | 8217# |
| | 8268# | 8273# | 8302# | 8307# | 8336# | 8341# | 8370# | 8375# | 8419# | 8509# | 8580# | 8646# | 8650# | 8697# | 8701# |
| | 8744# | 8748# | 8802# | 8808# | 8858# | 8867# | 8869# | 8893# | 8926# | 8929# | 8991# | 9121# | 9170# | 9218# | 9270# |
| | 9360# | 9417# | 9496# | 9532# | 9612# | 9695# | 9758# | 9838# | 9874# | 10027# | 10122# | 10174# | 10239# | 10277# | 10328# |
| MSPUT | 10382# | 10456# | 10489# | | | | | | | | | | | | |
| | 2819# | 5406# | 7625# | 7631# | 7632# | 7633# | 7634# | 7635# | 7636# | 7637# | 7638# | 7646# | 7647# | 7648# | 7649# |
| | 7650# | 7651# | 7652# | 7653# | 7654# | 7655# | 7656# | 7657# | 7665# | 7666# | 7667# | 7668# | 7669# | 7670# | 7671# |
| | 7672# | 7673# | 7674# | 7675# | 7676# | 7684# | 7685# | 7686# | 7687# | 7688# | 7689# | 7690# | 7691# | 7692# | 7693# |
| | 7694# | 7695# | 7696# | 7704# | 7705# | 7706# | 7707# | 7708# | 7709# | 7710# | 7711# | 7712# | 7713# | 7714# | 7715# |
| | 7723# | 7724# | 7725# | 7726# | 7727# | 7728# | 7729# | 7730# | 7731# | 7732# | 7733# | 7741# | 7742# | 7743# | 7744# |
| | 7745# | 7746# | 7747# | 7748# | 7749# | 7750# | 7751# | 7752# | 7753# | 7761# | 7762# | 7763# | 7764# | 7765# | 7766# |
| MSPUT1 | 7767# | 7944# | 8426# | 8452# | 8471# | 8490# | | | | | | | | | |
| | 2842# | 5406# | 7625# | 7631# | 7632# | 7633# | 7634# | 7635# | 7636# | 7637# | 7638# | 7646# | 7647# | 7648# | 7649# |
| | 7650# | 7651# | 7652# | 7653# | 7654# | 7655# | 7656# | 7657# | 7665# | 7666# | 7667# | 7668# | 7669# | 7670# | 7671# |
| | 7672# | 7673# | 7674# | 7675# | 7676# | 7684# | 7685# | 7686# | 7687# | 7688# | 7689# | 7690# | 7691# | 7692# | 7693# |
| | 7694# | 7695# | 7696# | 7704# | 7705# | 7706# | 7707# | 7708# | 7709# | 7710# | 7711# | 7712# | 7713# | 7714# | 7715# |
| | 7723# | 7724# | 7725# | 7726# | 7727# | 7728# | 7729# | 7730# | 7731# | 7732# | 7733# | 7741# | 7742# | 7743# | 7744# |
| | 7745# | 7746# | 7747# | 7748# | 7749# | 7750# | 7751# | 7752# | 7753# | 7761# | 7762# | 7763# | 7764# | 7765# | 7766# |
| MSRADI | 7767# | 7944# | 8426# | 8452# | 8471# | 8490# | | | | | | | | | |
| MSRBRO | 3151# | 5406# | 10458# | 10459# | 10460# | 10461# | | | | | | | | | |
| MSRNRO | 2787# | 5406# | | | | | | | | | | | | | |
| MSSETS | 2802# | 5406# | 7860# | | | | | | | | | | | | |
| | 3071# | 5406# | 5412# | 5491# | 5519# | 7624# | 7630# | 7645# | 7664# | 7683# | 7703# | 7722# | 7740# | 7760# | 7783# |
| | 7799# | 7817# | 7897# | 7923# | 7940# | 7963# | 7985# | 8010# | 8038# | 8043# | 8072# | 8099# | 8136# | 8173# | 8217# |
| | 8268# | 8273# | 8302# | 8307# | 8336# | 8341# | 8370# | 8375# | 8419# | 8509# | 8580# | 8646# | 8650# | 8697# | 8701# |
| | 8744# | 8748# | 8802# | 8808# | 8858# | 8867# | 8869# | 8893# | 8926# | 8929# | 8991# | 9121# | 9170# | 9218# | 9276# |
| | 9360# | 9417# | 9496# | 9532# | 9612# | 9695# | 9758# | 9838# | 9874# | 10027# | 10122# | 10174# | 10239# | 10277# | 10328# |
| MSSTAR | 10382# | 10456# | 10489# | | | | | | | | | | | | |
| MSSVC | 2468# | 5406# | | | | | | | | | | | | | |
| | 2746# | 5406# | 6678 | 6684 | 6694 | 6700 | 6807 | 6813 | 6971 | 7003 | 7009 | 7019 | 7025 | 7060 | 7066 |
| | 7100 | 7106 | 7114 | 7120 | 7266 | 7271 | 7290 | 7296 | 7305 | 7311 | 7320 | 7326 | 7335 | 7341 | 7473 |
| | 7479 | 7625# | 7626# | 7631# | 7632# | 7633# | 7634# | 7635# | 7636# | 7637# | 7638# | 7639# | 7646# | 7647# | 7648# |
| | 7649# | 7650# | 7651# | 7652# | 7653# | 7654# | 7655# | 7656# | 7657# | 7658# | 7665# | 7666# | 7667# | 7668# | 7669# |
| | 7670# | 7671# | 7672# | 7673# | 7674# | 7675# | 7676# | 7677# | 7684# | 7685# | 7686# | 7687# | 7688# | 7689# | 7690# |
| | 7691# | 7692# | 7693# | 7694# | 7695# | 7696# | 7697# | 7704# | 7705# | 7706# | 7707# | 7708# | 7709# | 7710# | 7711# |
| | 7712# | 7713# | 7714# | 7715# | 7716# | 7723# | 7724# | 7725# | 7726# | 7727# | 7728# | 7729# | 7730# | 7731# | 7732# |
| | 7733# | 7734# | 7741# | 7742# | 7743# | 7744# | 7745# | 7746# | 7747# | 7748# | 7749# | 7750# | 7751# | 7752# | 7753# |
| | 7754# | 7761# | 7762# | 7763# | 7764# | 7765# | 7766# | 7767# | 7768# | 7785# | 7834# | 7837# | 7840# | 7843# | 7860# |
| | 7883# | 7899# | 7906# | 7909# | 7926# | 7942# | 7944# | 7945# | 7964# | 7987# | 7995 | 7998# | 8021 | 8023# | 8043# |
| | 8055 | 8057# | 8061# | 8086 | 8088# | 8104# | 8118 | 8120# | 8154 | 8156# | 8196 | 8197# | 8202# | 8248 | 8249# |
| | 8256# | 8273# | 8284 | 8286# | 8290# | 8307# | 8318 | 8320# | 8324# | 8341# | 8352 | 8354# | 8358# | 8375# | 8386 |
| | 8388# | 8392# | 8426# | 8450 | 8452# | 8469 | 8471# | 8488 | 8490# | 8493# | 8521 | 8522# | 8535 | 8536# | 8547 |
| | 8548# | 8558 | 8559# | 8564# | 8607 | 8608# | 8617 | 8618# | 8625# | 8650# | 8664 | 8665# | 8674 | 8676# | 8680# |
| | 8701# | 8715 | 8716# | 8724 | 8726# | 8730# | 8748# | 8762 | 8763# | 8771 | 8773# | 8777# | 8808# | 8824 | 8825# |
| | 8834 | 8836# | 8841# | 8867# | 8869# | 8882 | 8884# | 8889# | 8893# | 8905 | 8907# | 8908# | 8929# | 8952 | 8953# |
| | 8962 | 8963# | 8968# | 8969# | 9067 | 9068# | 9090 | 9091# | 9101# | 9150# | 9196# | 9252 | 9256# | 9295 | 9304 |
| | 9316 | 9335 | 9343 | 9345# | 9401# | 9481# | 9516# | 9566 | 9581 | 9596# | 9664 | 9679# | 9729 | 9742# | 9810 |
| MSTLAB | 9825# | 9854 | 9857# | 9958 | 10003# | 10104# | 10156# | 10224# | 10256# | 10309# | 10359# | 10429# | | | |
| | 2739# | 5406# | 6678# | 6684# | 6694# | 6700# | 6807# | 6813# | 6971# | 7003# | 7009# | 7019# | 7025# | 7060# | 7066# |

CZDMRA M8203 STATIC DIAG #1 MACY11 30A(1052) 18-JUL-79 09:53 PAGE 9-6
CZDMRA.P11 18-JUL-79 09:44 CROSS REFERENCE TABLE -- MACRO NAMES

F 16

SEQ 0200

| | | | | | | |
|--------|-------|-------|------|------|------|------|
| REDEF | 1949# | 5406# | 7834 | 7837 | 7840 | 7843 |
| RFLAGS | 1967# | 5406# | | | | |
| SETPRI | 1977# | 5406# | 7899 | 7987 | | |
| SETVEC | 1986# | 5406# | | | | |
| SLASH | 1998# | 5406# | | | | |
| STARS | 2015# | 5406# | | | | |
| SVC | 2036# | 5405# | 5406 | | | |
| XFER | 2299# | 5406# | | | | |
| XFERF | 2310# | 5406# | | | | |
| XFERT | 2319# | 5406# | | | | |

. ABS. 036172 000

ERRORS DETECTED: 0

SAIL:CZDMRA,CZDMRA/CRF/NL:TOC=CZDMP.MLB,CZDMRA.P11
RUN-TIME: 145 173 13 SECONDS
RUN-TIME RATIO: 876/332=2.6
CORE USED: 18K (35 PAGES)