

DMP-11, DMR-11,  
M8203

M8203

STATIC DIAG#1  
CZDMREO

AH-E232E-MC

FICHE 1 OF 1

OCT 1981  
COPYRIGHT © 79-81  
MADE IN USA



2197  
2198  
2199  
2200  
2201  
2202  
2203  
2204  
2205  
2206  
2207  
2208  
2209  
2210  
2211  
2212  
2213  
2214  
2215  
2216  
2217  
2218  
2219  
2220  
2221  
2222  
2223  
2224  
2225  
2226  
2227  
2228  
2229  
2230  
2231

.REM @

IDENTIFICATION

PRODUCT CODE: AC-E231E-MC  
PRODUCT NAME: CZDMRE0 M8203 STATIC DIAG #1  
PRODUCT DATE: JULY 1981  
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, 1981 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL	PDP	UNIBUS	MASSBUS
DEC	DECUS	DECTAPE	

2233  
2234  
2235  
2236  
2237  
2238  
2239  
2240  
2241  
2242  
2243  
2244  
2245  
2246  
2247  
2248  
2249  
2250  
2251  
2252  
2253  
2254  
2255  
2256  
2257  
2258  
2259  
2260  
2261  
2262  
2263  
2264  
2265  
2266  
2267  
2268  
2269  
2270  
2271  
2272  
2273  
2274  
2275  
2276  
2277  
2278  
2279  
2280  
2281  
2282  
2283  
2284  
2285  
2286  
2287  
2288

## 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

2289  
2290  
2291  
2292  
2293  
2294  
2295  
2296  
2297  
2298  
2299  
2300  
2301  
2302  
2303  
2304  
2305  
2306  
2307  
2308

6.3.7.1 EFFECT OF PRINT COMMAND  
6.3.8 DISPLAY COMMAND  
6.3.8.1 UNITS SWITCH  
6.3.8.2 EFFECT OF DISPLAY COMMAND  
6.3.9 FLAGS COMMAND  
6.3.9.1 EFFECT OF FLAGS COMMAND  
6.3.10 ZFLAGS COMMAND  
6.3.10.1 EFFECT OF ZFLAGS COMMAND  
6.3.11 CONTROL CHARACTERS  
6.3.12 HARDWARE PARAMETERS  
6.3.13 SOFTWARE PARAMETERS  
6.3.14 EXTENDED DISCUSSION OF P-TABLE DIALOGUE

7.0 DEVICE INFORMATION TABLES

8.0 TEST DESCRIPTIONS  
8.1 DATA PATTERNS USED

9.0 ERROR INFORMATION  
9.1 ERROR REPORTING

2310  
2311  
2312  
2313  
2314  
2315  
2316  
2317  
2318  
2319  
2320  
2321  
2322  
2323  
2324  
2325  
2326  
2327  
2328  
2329  
2330  
2331  
2332  
2333  
2334  
2335  
2336  
2337  
2338  
2339  
2340  
2341  
2342  
2343  
2344  
2345  
2346  
2347  
2348  
2349  
2350  
2351  
2352  
2353  
2354  
2355  
2356  
2357  
2358  
2359  
2360  
2361  
2362  
2363  
2364  
2365

## 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. 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  
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 INSTALLED, IT IS DISABLED BY THE PROGRAM.

2366  
2367  
2368  
2369  
2370  
2371  
2372  
2373  
2374  
2375  
2376  
2377  
2378  
2379  
2380  
2381  
2382  
2383  
2384  
2385  
2386  
2387  
2388  
2389  
2390  
2391  
2392  
2393  
2394  
2395  
2396  
2397  
2398  
2399  
2400  
2401  
2402  
2403  
2404  
2405  
2406  
2407  
2408  
2409  
2410  
2411  
2412  
2413  
2414  
2415  
2416  
2417  
2418  
2419  
2420  
2421

2422  
2423  
2424  
2425  
2426  
2427  
2428  
2429  
2430  
2431  
2432  
2433  
2434  
2435  
2436  
2437  
2438  
2439  
2440  
2441  
2442  
2443  
2444  
2445  
2446  
2447  
2448  
2449  
2450  
2451  
2452  
2453  
2454  
2455  
2456  
2457  
2458  
2459  
2460  
2461  
2462  
2463  
2464  
2465  
2466  
2467  
2468  
2469  
2470  
2471  
2472  
2473  
2474  
2475  
2476  
2477

#### 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>
- D) ANSWER HARDWARE AND SOFTWARE QUESTIONS

2478  
2479  
2480  
2481  
2482  
2483  
2484  
2485  
2486  
2487  
2488  
2489  
2490  
2491  
2492  
2493  
2494  
2495  
2496  
2497  
2498  
2499  
2500  
2501  
2502  
2503  
2504  
2505  
2506  
2507  
2508  
2509  
2510  
2511  
2512  
2513  
2514  
2515  
2516  
2517  
2518  
2519  
2520  
2521  
2522  
2523  
2524  
2525  
2526  
2527  
2528  
2529  
2530  
2531  
2532  
2533

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-E-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 SET. THE EXIT IS A RETURN TO COMMAND MODE. SEE EXAMPLE AT



2534  
2535  
2536  
2537  
2538  
2539  
2540  
2541  
2542  
2543  
2544  
2545  
2546  
2547  
2548  
2549  
2550  
2551  
2552  
2553  
2554  
2555  
2556  
2557  
2558  
2559  
2560  
2561  
2562  
2563  
2564  
2565  
2566  
2567  
2568  
2569  
2570  
2571  
2572  
2573  
2574  
2575  
2576  
2577  
2578  
2579  
2580  
2581  
2582  
2583  
2584  
2585  
2586  
2587  
2588  
2589

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. HE MAY DO THIS BY GIVING ONE ANSWER TO EACH QUESTION (IN

2590  
2591  
2592  
2593  
2594  
2595  
2596  
2597  
2598  
2599  
2600  
2601  
2602  
2603  
2604  
2605  
2606  
2607  
2608  
2609  
2610  
2611  
2612  
2613  
2614  
2615  
2616  
2617  
2618  
2619  
2620  
2621  
2622  
2623  
2624  
2625  
2626  
2627  
2628  
2629  
2630  
2631  
2632  
2633  
2634  
2635  
2636  
2637  
2638  
2639  
2640  
2641  
2642  
2643  
2644  
2645

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 DIAGLOGUE. THE UNITS WHICH ARE SELECTED MUST NOT HAVE BEEN DROPPED BY THE DROP COMMAND. SEE THE DISCUSSION OF ADD AND DROP COMMANDS BELOW. DEFAULT IS TO TEST ALL UNITS WHICH HAVE NOT BEEN DROPPED BY A DROP

2646  
2647  
2648  
2649  
2650  
2651  
2652  
2653  
2654  
2655  
2656  
2657  
2658  
2659  
2660  
2661  
2662  
2663  
2664  
2665  
2666  
2667  
2668  
2669  
2670  
2671  
2672  
2673  
2674  
2675  
2676  
2677  
2678  
2679  
2680  
2681  
2682  
2683  
2684  
2685  
2686  
2687  
2688  
2689  
2690  
2691  
2692  
2693  
2694  
2695  
2696  
2697  
2698  
2699  
2700  
2701

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>)

2702  
2703  
2704  
2705  
2706  
2707  
2708  
2709  
2710  
2711  
2712  
2713  
2714  
2715  
2716  
2717  
2718  
2719  
2720  
2721  
2722  
2723  
2724  
2725  
2726  
2727  
2728  
2729  
2730  
2731  
2732  
2733  
2734  
2735  
2736  
2737  
2738  
2739  
2740  
2741  
2742  
2743  
2744  
2745  
2746  
2747  
2748  
2749  
2750  
2751  
2752  
2753  
2754  
2755  
2756  
2757

<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.

2758  
2759  
2760  
2761  
2762  
2763  
2764  
2765  
2766  
2767  
2768  
2769  
2770  
2771  
2772  
2773  
2774  
2775  
2776  
2777  
2778  
2779  
2780  
2781  
2782  
2783  
2784  
2785  
2786  
2787  
2788  
2789  
2790  
2791  
2792  
2793  
2794  
2795  
2796  
2797  
2798  
2799  
2800  
2801  
2802  
2803  
2804  
2805  
2806  
2807  
2808  
2809  
2810  
2811  
2812  
2813

### 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

ALL FLAGS ARE CLEARED.

2814  
2815  
2816  
2817  
2818  
2819  
2820  
2821  
2822  
2823  
2824  
2825  
2826  
2827  
2828  
2829  
2830  
2831  
2832  
2833  
2834  
2835  
2836  
2837  
2838  
2839  
2840  
2841  
2842  
2843  
2844  
2845  
2846  
2847  
2848  
2849  
2850  
2851  
2852  
2853  
2854  
2855  
2856  
2857  
2858  
2859  
2860  
2861  
2862  
2863  
2864  
2865  
2866  
2867  
2868  
2869

### 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 SUPPRESSED 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 000-770 (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. M8207 RUN SWITCH (E28 SW7) - 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.

2870  
2871  
2872  
2873  
2874  
2875  
2876  
2877  
2878  
2879  
2880  
2881  
2882  
2883  
2884  
2885  
2886  
2887  
2888  
2889  
2890  
2891  
2892  
2893  
2894  
2895  
2896  
2897  
2898  
2899  
2900  
2901  
2902  
2903  
2904  
2905  
2906  
2907  
2908  
2909  
2910  
2911  
2912  
2913  
2914  
2915  
2916  
2917  
2918  
2919  
2920  
2921  
2922  
2923  
2924  
2925

#### 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  
<QUESTION 1> ? 75

2926  
2927  
2928  
2929  
2930  
2931  
2932  
2933  
2934  
2935  
2936  
2937  
2938  
2939  
2940  
2941  
2942  
2943  
2944  
2945  
2946  
2947  
2948  
2949  
2950  
2951  
2952  
2953

<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).



2955  
2956  
2957  
2958  
2959  
2960  
2961  
2962  
2963  
2964  
2965  
2966  
2967  
2968  
2969  
2970  
2971  
2972  
2973  
2974  
2975  
2976  
2977  
2978  
2979  
2980  
2981  
2982  
2983  
2984  
2985  
2986  
2987  
2988  
2989  
2990  
2991  
2992  
2993  
2994  
2995  
2996  
2997  
2998  
2999  
3000  
3001  
3002  
3003  
3004  
3005  
3006  
3007  
3008  
3009  
3010

7.0 DEVICE INFORMATION TABLES

\*\*\*\*\*  
\* MAINTENANCE REGISTER - BSEL1

\*\*\*\*\*  
RUN = BIT7  
MCLR = BIT6  
STEPLU = BIT4  
LULoop = 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

3011  
3012  
3013  
3014  
3015  
3016  
3017  
3018  
3019  
3020  
3021  
3022  
3023  
3024  
3025  
3026  
3027  
3028  
3029  
3030  
3031  
3032  
3033  
3034  
3035  
3036  
3037  
3038  
3039  
3040  
3041  
3042  
3043  
3044  
3045  
3046  
3047  
3048  
3049  
3050  
3051  
3052  
3053  
3054  
3055  
3056  
3057  
3058  
3059  
3060  
3061  
3062  
3063  
3064  
3065  
3066

```
*****  
* 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  
OVR       = BIT3  
RAB       = BIT2
```

3067  
3068  
3069  
3070  
3071  
3072  
3073  
3074  
3075  
3076  
3077  
3078  
3079  
3080  
3081  
3082  
3083  
3084  
3085  
3086  
3087  
3088  
3089  
3090  
3091  
3092  
3093  
3094  
3095  
3096  
3097  
3098  
3099  
3100  
3101  
3102  
3103  
3104  
3105  
3106  
3107  
3108  
3109  
3110  
3111  
3112  
3113  
3114  
3115  
3116  
3117  
3118  
3119  
3120  
3121  
3122

EBLK = BIT1  
BCC = BIT0

\*\*\*\*\*  
\* IBUS REG 13  
\*\*\*\*\*

RING = BIT7  
DTR = BIT6  
RTS = BIT5  
HDX = BIT4  
MODR = BIT3  
CS = BIT2  
STBY = BIT1  
CARR = BIT0

\*\*\*\*\*  
\* IBUS REG 14  
\*\*\*\*\*

READY = BIT7  
TXEN = BIT6  
DISSI = BIT5  
RDAX = BIT4  
WAX = BIT3  
ENAX = BIT2  
AX2 = BIT1  
AX1 = BIT0

\*\*\*\*\*  
\* IBUS REG 17  
\*\*\*\*\*

SIGR = BIT7  
SIGQ = BIT6  
TXDATA = BIT5  
OCOR = BIT4  
ICIR = BIT3  
TESTMD = BIT2  
MCLK = BIT1  
DDCMP = BIT0

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

RX7 = BIT7  
RX6 = BIT6  
RX5 = BIT5  
RX4 = BIT4  
RX3 = BIT3  
RX2 = BIT2  
RX1 = BIT1  
RX0 = BIT0

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

RERR = BIT7  
ASBC2 = BIT6

3123  
3124  
3125  
3126  
3127  
3128  
3129  
3130  
3131  
3132  
3133  
3134  
3135  
3136  
3137  
3138  
3139  
3140  
3141  
3142  
3143  
3144  
3145  
3146  
3147  
3148  
3149  
3150  
3151  
3152  
3153  
3154  
3155  
3156  
3157  
3158  
3159  
3160  
3161  
3162  
3163  
3164  
3165  
3166  
3167  
3168  
3169  
3170  
3171  
3172  
3173  
3174  
3175  
3176  
3177  
3178

ASBC1 = BIT5  
ASBC0 = BIT4  
ROR = BIT3  
RABT = BIT2  
REOM = BIT1  
RSOM = BIT0

.....  
\* AX1-15 - USYRT REG 2  
.....

TX7 = BIT7  
TX6 = BIT6  
TX5 = BIT5  
TX4 = BIT4  
TX3 = BIT3  
TX2 = BIT2  
TX1 = BIT1  
TX0 = BIT0

.....  
\* 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  
.....

3179  
3180  
3181  
3182  
3183  
3184  
3185  
3186  
3187  
3188  
3189  
3190  
3191  
3192  
3193  
3194  
3195  
3196  
3197  
3198  
3199  
3200  
3201  
3202

I422 = BIT7  
XYZ = BIT6  
C32BCC = BIT5  
V35 = BIT4  
INTGRL = BIT3  
C32ENB = BIT2  
OP = BIT1  
TEST = BIT0  
AX315U = I422!XYZ!C32BCC!V35!INTGRL!OP

\*\*\*\*\*  
\* AX3-16 - USYRT REG 7  
\*\*\*\*\*

TXLEN2 = BIT7  
TXLEN1 = BIT6  
TXLEN0 = BIT5  
RXLEN2 = BIT2  
RXLEN1 = BIT1  
RXLEN0 = BIT0

3204  
3205  
3206  
3207  
3208  
3209  
3210  
3211  
3212  
3213  
3214  
3215  
3216  
3217  
3218  
3219  
3220  
3221  
3222  
3223  
3224  
3225  
3226  
3227  
3228  
3229  
3230  
3231  
3232  
3233  
3234  
3235  
3236  
3237  
3238  
3239  
3240  
3241  
3242  
3243  
3244  
3245  
3246  
3247  
3248  
3249  
3250  
3251  
3252  
3253  
3254  
3255  
3256  
3257  
3258  
3259

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.  
:*****
```

3260  
3261  
3262  
3263  
3264  
3265  
3266  
3267  
3268  
3269  
3270  
3271  
3272  
3273  
3274  
3275  
3276  
3277  
3278  
3279  
3280  
3281  
3282  
3283  
3284  
3285  
3286  
3287  
3288  
3289  
3290  
3291  
3292  
3293  
3294  
3295  
3296  
3297  
3298  
3299  
3300  
3301  
3302  
3303  
3304  
3305  
3306  
3307  
3308  
3309  
3310  
3311  
3312  
3313  
3314  
3315

\*\*\*\*\*  
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  
\*\*\*\*\*

3316  
3317  
3318  
3319  
3320  
3321  
3322  
3323  
3324  
3325  
3326  
3327  
3328  
3329  
3330  
3331  
3332  
3333  
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

```
*****  
: 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-
```



3372  
3373  
3374  
3375  
3376  
3377  
3378  
3379  
3380  
3381  
3382  
3383  
3384  
3385  
3386  
3387  
3388  
3389  
3390  
3391  
3392  
3393  
3394  
3395  
3396  
3397  
3398  
3399  
3400  
3401  
3402  
3403  
3404  
3405  
3406  
3407  
3408  
3409  
3410  
3411  
3412  
3413  
3414  
3415  
3416  
3417  
3418  
3419  
3420  
3421  
3422  
3423  
3424  
3425  
3426  
3427

\* 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 M8207 RUN SWITCH (E28 SW7)  
\* 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,  
:\*\*\*\*\*

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  
3467  
3468  
3469  
3470  
3471  
3472  
3473  
3474  
3475  
3476  
3477  
3478  
3479  
3480  
3481  
3482  
3483

\* 004,010,020,040,100,200,377,377,377,377,377,377,377,  
\* 376,375,373,367,357,337,277,177.  
:\*\*\*\*\*

:\*\*\*\*\*  
\* TEST 17 - AX0-15,AX0-16 READ/WRITE BIT TEST  
\*  
\* IN THIS TEST, A MASTER CLEAR IS DONE, AND THEN A WRITE, READ, AND COMPARE  
\* ARE PERFORMED IN REGS AX0-15,AX0-16 USING EACH WORD OF PATTERN L.  
\* ANY BITS IN AX0-15,AX0-16 WHICH ARE NOT READ/WRITE ARE MASKED OFF (TO 0)  
\* IN THE EXPECTED VALUE BEFORE COMPARISON.  
\* PATTERN L =  
\* FOR REG 15: 000,377,000  
\* FOR REG 16: 000,377,000.  
:\*\*\*\*\*

:\*\*\*\*\*  
\* TEST 18 - AX1-15,AX1-16 READ/WRITE BIT TEST  
\*  
\* IN THIS TEST, A MASTER CLEAR IS DONE, AND THEN A WRITE, READ, AND COMPARE  
\* ARE PERFORMED IN REGS AX1-15,AX1-16 USING EACH WORD OF PATTERN K.  
\* ANY BITS IN AX1-15,AX1-16 WHICH ARE NOT READ/WRITE ARE MASKED OFF (TO 0)  
\* IN THE EXPECTED VALUE BEFORE COMPARISON.  
:\*\*\*\*\*

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

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  
3523  
3524  
3525  
3526  
3527  
3528  
3529  
3530  
3531  
3532  
3533  
3534  
3535  
3536  
3537  
3538  
3539

```
*****  
; 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.  
*****
```

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  
3579  
3580  
3581  
3582  
3583  
3584  
3585  
3586  
3587  
3588  
3589  
3590  
3591  
3592  
3593  
3594  
3595

\*\*\*\*\*  
: 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.

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  
3635  
3636  
3637  
3638  
3639  
3640  
3641  
3642  
3643  
3644  
3645  
3646  
3647  
3648  
3649  
3650  
3651

\*\*\*\*\*

\*\*\*\*\*

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, CRC

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  
3691  
3692  
3693  
3694  
3695  
3696  
3697  
3698  
3699  
3700  
3701  
3702  
3703  
3704  
3705  
3706  
3707

\*  
\* 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  
\* STILL SET AFTER THE MESSAGE.  
:\*\*\*\*\*

:\*\*\*\*\*  
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.  
:\*\*\*\*\*

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  
3747  
3748  
3749  
3750  
3751  
3752  
3753  
3754  
3755  
3756  
3757  
3758  
3759  
3760  
3761  
3762  
3763

\*\*\*\*\*  
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

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  
3803  
3804  
3805  
3806  
3807  
3808  
3809  
3810  
3811  
3812  
3813  
3814  
3815  
3816  
3817  
3818  
3819

\* 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.



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  
3859  
3860  
3861  
3862  
3863  
3864  
3865  
3866  
3867  
3868  
3869  
3870  
3871  
3872  
3873  
3874  
3875

\*\*\*\*\*  
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

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  
3915  
3916  
3917  
3918  
3919  
3920  
3921  
3922  
3923  
3924  
3925  
3926  
3927  
3928  
3929  
3930  
3931

\* 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

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  
3971  
3972  
3973  
3974  
3975  
3976  
3977  
3978  
3979  
3980  
3981  
3982  
3983  
3984  
3985  
3986  
3987

\*\*\*\*\* 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

3988	.BYTE	103
3989	.BYTE	001
3990	.BYTE	100

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

3991		
3992		
3993	PATK: .BYTE	000
3994	.BYTE	000
3995	.BYTE	377
3996	.BYTE	377
3997	.BYTE	125
3998	.BYTE	125
3999	.BYTE	252
4000	.BYTE	252
4001	.BYTE	001
4002	.BYTE	000
4003	.BYTE	002
4004	.BYTE	000
4005	.BYTE	004
4006	.BYTE	000
4007	.BYTE	010
4008	.BYTE	000
4009	.BYTE	020
4010	.BYTE	000
4011	.BYTE	040
4012	.BYTE	000
4013	.BYTE	100
4014	.BYTE	000
4015	.BYTE	200
4016	.BYTE	000
4017	.BYTE	000
4018	.BYTE	001
4019	.BYTE	000
4020	.BYTE	002
4021	.BYTE	000
4022	.BYTE	004
4023	.BYTE	000
4024	.BYTE	010
4025	.BYTE	000
4026	.BYTE	020
4027	.BYTE	000
4028	.BYTE	040
4029	.BYTE	000
4030	.BYTE	100
4031	.BYTE	000
4032	.BYTE	200
4033	.BYTE	376
4034	.BYTE	377
4035	.BYTE	375
4036	.BYTE	377
4037	.BYTE	373
4038	.BYTE	377
4039	.BYTE	367
4040	.BYTE	377
4041	.BYTE	357
4042	.BYTE	377
4043	.BYTE	337

4044	.BYTE	377
4045	.BYTE	277
4046	.BYTE	377
4047	.BYTE	177
4048	.BYTE	377
4049	.BYTE	377
4050	.BYTE	376
4051	.BYTE	377
4052	.BYTE	375
4053	.BYTE	377
4054	.BYTE	373
4055	.BYTE	377
4056	.BYTE	367
4057	.BYTE	377
4058	.BYTE	357
4059	.BYTE	377
4060	.BYTE	337
4061	.BYTE	377
4062	.BYTE	277
4063	.BYTE	377
4064	.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

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 .BYTE 017  
4101 .BYTE 000

\*\*\*\*\* DATA PATTERN O \*\*\*\*\*

PATO:  
4104 .BYTE 000  
4105 .BYTE 041  
4106 .BYTE 004  
4107 .BYTE 010  
4108 .BYTE 020  
4109 .BYTE 040  
4110 .BYTE 100  
4111 .BYTE 101  
4112 .BYTE 200  
4113 .BYTE 201  
4114 .BYTE 300  
4115 .BYTE 111  
4116 .BYTE 301  
4117 .BYTE 375  
4118

\*\*\*\*\* DATA PATTERN P \*\*\*\*\*

PATP:  
4119 .BYTE 000  
4120 .BYTE 113  
4121 .BYTE 200  
4122 .BYTE 040  
4123 .BYTE 020  
4124 .BYTE 010  
4125 .BYTE 001  
4126 .BYTE 104  
4127 .BYTE 007  
4128 .BYTE 105  
4129 .BYTE 007  
4130 .BYTE 144  
4131 .BYTE 107  
4132 .BYTE 157  
4133

\*\*\*\*\* DATA PATTERN U \*\*\*\*\*

PATU:  
4134 .BYTE 000  
4135 .BYTE 000  
4136 .BYTE 001  
4137 .BYTE 000  
4138 .BYTE 013  
4139 .BYTE 000  
4140 .BYTE 011  
4141 .BYTE 000  
4142 .BYTE 021  
4143 .BYTE 000  
4144 .BYTE 101  
4145 .BYTE 000  
4146 .BYTE 301  
4147 .BYTE 000  
4148 .BYTE 000  
4149 .BYTE 001  
4150 .BYTE 000  
4151 .BYTE 001  
4152 .BYTE 000  
4153 .BYTE 000  
4154 .BYTE 002  
4155

4156	.BYTE	000
4157	.BYTE	004
4158	.BYTE	000
4159	.BYTE	040
4160	.BYTE	000
4161	.BYTE	100
4162	.BYTE	000
4163	.BYTE	200
4164	.BYTE	000
4165	.BYTE	346
4166	.BYTE	000
4167	.BYTE	345
4168	.BYTE	000
4169	.BYTE	343
4170	.BYTE	000
4171	.BYTE	307
4172	.BYTE	000
4173	.BYTE	247
4174	.BYTE	000
4175	.BYTE	147

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

4176		
4177		
4178	PATV: .BYTE	000
4179	.BYTE	000
4180	.BYTE	333
4181	.BYTE	000
4182	.BYTE	331
4183	.BYTE	000
4184	.BYTE	323
4185	.BYTE	000
4186	.BYTE	313
4187	.BYTE	000
4188	.BYTE	233
4189	.BYTE	000
4190	.BYTE	133
4191	.BYTE	000
4192	.BYTE	000
4193	.BYTE	001
4194	.BYTE	000
4195	.BYTE	002
4196	.BYTE	000
4197	.BYTE	004
4198	.BYTE	000
4199	.BYTE	040
4200	.BYTE	000
4201	.BYTE	100
4202	.BYTE	000
4203	.BYTE	200
4204	.BYTE	000
4205	.BYTE	346
4206	.BYTE	000
4207	.BYTE	345
4208	.BYTE	000
4209	.BYTE	343
4210	.BYTE	000
4211	.BYTE	307

CZDMRE M8203 STATIC DIAG #1  
CZDMRE.P11 08-JUN-81 13:27

MACY11 30A(1052) 15-JUN-81 15:34 N 3  
PROGRAM DOCUMENT PAGE 6-18

SEQ 0039

4212  
4213  
4214  
4215  
4216  
4217  
4218  
4219  
4220  
4221

.BYTE 000  
.BYTE 247  
.BYTE 000  
.BYTE 147



4223  
4224  
4225  
4226  
4227  
4228  
4229  
4230  
4231  
4232  
4233  
4234  
4235  
4236  
4237  
4238  
4239  
4240  
4241  
4242  
4243  
4244  
4245  
4246  
4247  
4248  
4249  
4250  
4251  
4252  
4253  
4254  
4255  
4256  
4257  
4258  
4259  
4260  
4261  
4262  
4263  
4264  
4265  
4266  
4267  
4268  
4269  
4270  
4271  
4272  
4273  
4274  
4275  
4276  
4277  
4278

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

4279  
4280  
4281  
4282  
4283  
4284  
4285  
4286  
4287

@

4289  
4298  
4299  
4300  
4301  
4302  
4303  
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

002000  
  
002000  
  
002000  
  
000001  
000001  
000001  
000001  
000001  
000001  
000001

.TITLE CZDMRE M8203 STATIC DIAG #1  
.=2000

.MCALL SVC  
SVC

; INITIALIZE SUPERVISOR MACROS

BGNMOD LU1MOD

\$LSTIN= 1  
\$LSTTAG= 1  
SVCINS= 1 ; LIST INSTRUCTIONS, SHIFTED RIGHT  
SVCTST= 1 ; LIST TEST TAGS, SHIFTED RIGHT  
SVCSUB= 1 ; LIST SUBTEST TAGS, SHIFTED RIGHT  
SVCGBL= 1 ; LIST GLOBAL TAGS, SHIFTED RIGHT  
SVCTAG= 1 ; LIST OTHER TAGS, SHIFTED RIGHT

; CHANGE THE VALUES OF THE SVC... SYMBOLS TO BE ZERO IF YOU WISH  
; TO ALIGN THE MACRO CALLS AND THEIR EXPANSIONS. CHANGE THE  
; SYMBOLS TO BE MINUS-ONE TO NOT LIST THE EXPANSIONS. YOU MAY  
; CHANGE THE SYMBOLS AT ANY POINT IN YOUR PROGRAM.





4362  
4363  
4364  
4365  
4366  
4367  
4368  
4369  
4370  
4372  
4373  
4374

.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

002122 000052  
(4) 002122 000052  
(3) 002124  
(6) 002124 021634  
(6) 002126 021716  
(6) 002130 022002  
(6) 002132 022126  
(6) 002134 022230  
(6) 002136 022356  
(6) 002140 022476  
(6) 002142 022652  
(6) 002144 023106  
(6) 002146 023224  
(6) 002150 023342  
(6) 002152 023460  
(6) 002154 023576  
(6) 002156 024226  
(6) 002160 024560  
(6) 002162 025050  
(6) 002164 025260  
(6) 002166 025464  
(6) 002170 025670  
(6) 002172 026126  
(6) 002174 026360  
(6) 002176 026632  
(6) 002200 027310  
(6) 002202 027430  
(6) 002204 027540  
(6) 002206 027734  
(6) 002210 030300  
(6) 002212 030510  
(6) 002214 031040  
(6) 002216 031140  
(6) 002220 031510  
(6) 002222 032052  
(6) 002224 032322  
(6) 002226 032664  
(6) 002230 033014  
(6) 002232 033726  
(6) 002234 034210  
(6) 002236 034410  
(6) 002240 034710  
(6) 002242 035000  
(6) 002244 035142  
(6) 002246 035310

.WORD 42  
L\$DISPATCH: :  
.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

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



4383  
4384  
4385  
4386  
4387  
4388  
4389  
4390  
4391  
(3)  
(3)  
(3)  
4392  
4393  
4394  
4395  
4396  
4397  
4398  
4399  
4400  
4401  
4402  
4403  
4404  
4405  
(3)  
4406  
4407  
4408  
4409  
4410

.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 000  
.WORD 000  
.WORD 000  
.WORD 0  
.WORD 4  
.WORD 1

:MICROPROCESSOR TYPE = M8207  
:M8207 CSR UNIBUS ADDRESS  
:M8207 INTERRUPT VECTOR  
:M8207 INTERRUPT PRIORITY LEVEL = 5  
:LINE UNIT = M8203  
:M8203 REG 11 (E121 SW10,9 , E134 SW9,10)  
:M8203 REG 15 (E134 SW1-8)  
:M8203 REG 16 (E121 SW1-8)  
:H3254&H3255 USED  
:BAUD RATE = 56 K  
:M8207 RUN SWITCH (E28 SW7) IS ON

ENDHW

L10000:



4412  
4413  
4414  
4415  
4416  
4417  
4418  
4419  
(3) 002300  
(3) 002302  
(3) 002302  
4420  
4421  
4422  
(3) 002302  
4423  
4424  
4425  
4426  
4427  
4428

.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:

4430  
4431  
4432  
4433  
4434  
4435  
4436  
4437  
4438  
4439  
4440  
4441  
4442  
4443  
4444  
4445  
4446  
4447  
4448  
4449

002302

.SBTTL GLOBAL EQUATES SECTION

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

EQUALS

: BIT DEFINITIONS

BIT15== 100000  
BIT14== 40000  
BIT13== 20000  
BIT12== 10000  
BIT11== 4000  
BIT10== 2000  
BIT9== 1000  
BIT08== 400  
BIT07== 200  
BIT06== 100  
BIT05== 40  
BIT04== 20  
BIT03== 10  
BIT02== 4  
BIT01== 2  
BIT00== 1

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

EF.START== 32. : START COMMAND WAS ISSUED  
EF.RESTART== 31. : RESTART COMMAND WAS ISSUED

(1)  
(1)  
(1)  
(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  
(1)  
(1) 001000  
(1) 000400  
(1) 000200  
(1) 000100  
(1) 000040  
(1) 000020  
(1) 000010  
(1) 000004  
(1) 000002  
(1) 000001  
(1)  
(1)  
(1)  
(1) 000040  
(1) 000037

(1) 000036 EF.CONTINUE== 30. ; CONTINUE COMMAND WAS ISSUED  
(1) 000035 EF.NEW== 29. ; A NEW PASS HAS BEEN STARTED  
(1) 000034 EF.PWR== 28. ; A POWER-FAIL/POWER-UP OCCURRED

.....  
: PRIORITY LEVEL DEFINITIONS  
.....

(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

.....  
: OPERATOR FLAG BITS  
.....

(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 FNT== 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

4450  
4451  
4452  
4453  
4454  
4455 : \*\*\*\*\*  
4456 : \* PROGRAM EVENT FLAG DEFINITIONS  
4457 : \*\*\*\*\*  
4458  
4459  
4460  
4461

4462 : \*\*\*\*\*  
4463 : \* MAINTENANCE REGISTER - BSEL1  
4464 : \*\*\*\*\*

4465 000200 RUN = BIT7  
4466 000100 MCLR = BIT6  
4467 000020 STEPLU = BIT4  
4468 000010 LULOOP = BIT3  
4469 000004 ROMO = BIT2  
4470 000002 ROMI = BIT1  
4471 000001 STEPMP = BIT0

4472 : \*\*\*\*\*  
4473

```
4474      ;* OBUS REG 10 - TRANSMITTER BUFFER
4475      ;*****
4476      000200 TX7      = BIT7
4477      000100 TX6      = BIT6
4478      000040 TX5      = BIT5
4479      000020 TX4      = BIT4
4480      000010 TX3      = BIT3
4481      000004 TX2      = BIT2
4482      000002 TX1      = BIT1
4483      000001 TX0      = BIT0
4484
4485      ;*****
4486      ;* OBUS REG 11
4487      ;*****
4488      000200 OC       = BIT7
4489      000010 GOAH    = BIT3
4490      000004 ABORT   = BIT2
4491      000002 EOM     = BIT1
4492      000001 SOM     = BIT0
4493
4494      ;*****
4495      ;* OBUS REG 12
4496      ;*****
4497      000200 IC       = BIT7
4498      000100 BPOLL   = BIT6
4499      000040 LULP    = BIT5
4500
4501      ;*****
4502      ;* OBUS REG 13
4503      ;*****
4504      000200 POLL    = BIT7
4505      000100 DTR     = BIT6
4506      000040 SELFR   = BIT5
4507      000020 HDX     = BIT4
4508      000010 MAINT1  = BIT3
4509      000004 MAINT2  = BIT2
4510      000002 SELSBY  = BIT1
4511
4512      ;*****
4513      ;* OBUS REG 14
4514      ;*****
4515      000100 TXEN    = BIT6
4516      000040 DISSI   = BIT5
4517      000020 RDAX    = BIT4
4518      000010 WAX     = BIT3
4519      000004 ENAX    = BIT2
4520      000002 AX2     = BIT1
4521      000001 AX1     = BIT0
4522
4523      ;*****
4524      ;* OBUS REG 17
4525      ;*****
4526      000200 CRC2    = BIT7
4527      000100 CRC1    = BIT6
4528      000040 IDLE    = BIT5
4529      000020 SECA    = BIT4
```

4530	000010	STRIP = BIT3
4531	000004	RDALL = BIT2
4532	000002	IERR = BIT1
4533	000001	DDCMP = BIT0
4534		
4535		::*****
4536		::* IBUS REG 10 - RECEIVER BUFFER
4537		::*****
4538	000200	RX7 = BIT7
4539	000100	PX6 = BIT6
4540	000040	RX5 = BIT5
4541	000020	RX4 = BIT4
4542	000010	RX3 = BIT3
4543	000004	RX2 = BIT2
4544	000002	RX1 = BIT1
4545	000001	RX0 = BIT0
4546		
4547		::*****
4548		::* IBUS REG 11
4549		::*****
4550	000200	OC = BIT7
4551	000100	OACT = BIT6
4552	000040	SW3 = BIT5
4553	000020	ORDY = BIT4
4554	000010	SW2 = BIT3
4555	000004	SW1 = BIT2
4556	000002	SW0 = BIT1
4557	000001	UNRR = BIT0
4558		
4559		::*****
4560		::* IBUS REG 12
4561		::*****
4562	000200	IC = BIT7
4563	000100	IACT = BIT6
4564	000040	LULP = BIT5
4565	000020	IRDY = BIT4
4566	000010	OVRR = BIT3
4567	000004	RAB = BIT2
4568	000002	EBLK = BIT1
4569	000001	BCC = BIT0
4570		
4571		::*****
4572		::* IBUS REG 13
4573		::*****
4574	000200	RING = BIT7
4575	000100	DTR = BIT6
4576	000040	RTS = BIT5
4577	000020	HDX = BIT4
4578	000010	MODR = BIT3
4579	000004	CS = BIT2
4580	000002	STBY = BIT1
4581	000001	CARR = BIT0
4582		
4583		::*****
4584		::* IBUS REG 14
4585		::*****

4586	000200	READY	=	BIT7
4587	000100	TXEN	=	BIT6
4588	000040	DISSI	=	BIT5
4589	000020	RDAX	=	BIT4
4590	000010	WAX	=	BIT3
4591	000004	ENAX	=	BIT2
4592	000002	AX2	=	BIT1
4593	000001	AX1	=	BIT0

\*\*\*\*\*  
:\* IBUS REG 17  
\*\*\*\*\*

4594				
4595				
4596				
4597				
4598	000200	SIGR	=	BIT7
4599	000100	SIGQ	=	BIT6
4600	000040	TXDATA	=	BIT5
4601	000020	OCOR	=	BIT4
4602	000010	ICIR	=	BIT3
4603	000004	TESTMD	=	BIT2
4604	000002	MCLK	=	BIT1
4605	000001	DDCMP	=	BIT0

\*\*\*\*\*  
:\* AX0-15 - USYRT REG 0 (READ ONLY)  
\*\*\*\*\*

4606				
4607				
4608				
4609				
4610	000200	RX7	=	BIT7
4611	000100	RX6	=	BIT6
4612	000040	RX5	=	BIT5
4613	000020	RX4	=	BIT4
4614	000010	RX3	=	BIT3
4615	000004	RX2	=	BIT2
4616	000002	RX1	=	BIT1
4617	000001	RX0	=	BIT0

\*\*\*\*\*  
:\* AX0-16 - USYRT REG 1 (READ ONLY)  
\*\*\*\*\*

4618				
4619				
4620				
4621				
4622	000200	RERR	=	BIT7
4623	000100	ASBC2	=	BIT6
4624	000040	ASBC1	=	BIT5
4625	000020	ASBC0	=	BIT4
4626	000010	ROR	=	BIT3
4627	000004	RABT	=	BIT2
4628	000002	REOM	=	BIT1
4629	000001	RSOM	=	BIT0

\*\*\*\*\*  
:\* AX1-15 - USYRT REG 2  
\*\*\*\*\*

4630				
4631				
4632				
4633				
4634	000200	TX7	=	BIT7
4635	000100	TX6	=	BIT6
4636	000040	TX5	=	BIT5
4637	000020	TX4	=	BIT4
4638	000010	TX3	=	BIT3
4639	000004	TX2	=	BIT2
4640	000002	TX1	=	BIT1
4641	000001	TX0	=	BIT0

```
4642  
4643  
4644  
4645  
4646      000200  
4647      000010  
4648      000004  
4649      000002  
4650      000001  
4651  
4652  
4653  
4654  
4655      000200  
4656      000100  
4657      000040  
4658      000020  
4659      000010  
4660      000004  
4661      000002  
4662      000001  
4663      000226  
4664  
4665  
4666  
4667  
4668      000200  
4669      000100  
4670      000040  
4671      000020  
4672      000010  
4673      000004  
4674      000002  
4675      000001  
4676  
4677  
4678  
4679  
4680      000200  
4681      000100  
4682      000040  
4683      000020  
4684      000010  
4685      000004  
4686      000002  
4687      000001  
4688      000372  
4689  
4690  
4691  
4692  
4693      000200  
4694      000100  
4695      000040  
4696      000004  
4697      000002
```

```
*****  
;* 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  
SYNO      = 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  
C32BCC   = BIT5  
V35      = BIT4  
INTGRL   = BIT3  
C32ENB   = BIT2  
OP       = BIT1  
TEST     = BIT0  
AX315U   = I422!XYZ!C32BCC!V35!INTGRL!OP  
  
*****  
;* AX3-16 - USYRT REG 7  
*****  
TXLEN2   = BIT7  
TXLEN1   = BIT6  
TXLEN0   = BIT5  
RXLEN2   = BIT2  
RXLEN1   = BIT1
```

4698 000001  
4699  
4700  
4701  
4702  
4703  
4704  
4705  
4706  
4707 004000  
4708 002000  
4709 001000  
4710 000400  
4711  
4712  
4713  
4714  
4715  
4716  
4717  
4718  
4719 004000  
4720 002000  
4721 001000  
4722 000400  
4723  
4724  
4725  
4726  
4727  
4728  
4729  
4730 002302  
4731 002304  
4732 002306  
4733 002310  
4734 002312  
4735 002314  
4736 002316  
4737 002320  
4738 002322  
4739 002324  
4740 002326  
4741 002330  
4742 002332  
4743 002334  
4744 002336  
4745 002340  
4746  
4747  
4748  
4749  
4750  
4751 100000  
4752  
4753 100000

RXLENO = BIT0

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



4754 100000  
4755  
4756  
4757  
4758  
4759  
4760  
4761  
4762  
4763 021000  
4764 122000  
4765 121000  
4766  
4767  
4768  
4769  
4770 000001  
4771 000002  
4772  
4773  
4774  
4775  
4776

CRCCHK = BIT15

.....  
: \* MICROINSTRUCTION DEFINITIONS  
:.....  
MVI0X = 021000 :MOVE IBUS TO OBUS\*  
MVI10 = 122000 :MOVE IBUS\* TO OBUS  
MVI10X = 121000 :MOVE IBUS\* TO OBUS\*

..... ERROR1 BIT FLAG DEFINITIONS .....

RRDYTO = BIT0  
WRDYTO = BIT1

4778  
4779  
4780  
4781  
4782  
4783  
4784  
4785  
4786  
4787  
4788 002302 000020  
4789  
4790  
4791  
4792  
4793 002342 000000  
4794 002344 000000  
4795 002346 000000  
4796 002350 000000  
4797 002352 000000  
4798 002354 000000  
4799  
4800 002356 000000  
4801 002360 000000  
4802 002362 000000  
4803 002364 000000  
4804 002366 000000  
4805 002370 000000  
4806 002372 000000  
4807 002374 000000  
4808 002376 000000  
4809 002400 000000  
4810 002402 000000  
4811 002404 000000  
4812 002406 000000  
4813 002410 000000  
4814 002412 000000  
4815 002414 000000  
4816 002416 000000  
4817 002420 000000  
4818 002422 000000  
4819 002424 000000  
4820 002426 000000  
4821 002430 000000  
4822 002432 000000  
4823 002434 000000  
4824 002436 000000  
4825 002440 000000  
4826 002442 000000  
4827 002444 000000  
4828  
4829  
4830 002446 160170  
4831 002450 160171  
4832 002452 160172  
4833 002454

.SBTTL GLOBAL DATA SECTION

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

\* STORAGE FOR DEVICE REGISTERS

LUREG: .BLKW 16.

\* MISCELLANEOUS STORAGE

SCRACH: .WORD 0 : GEN'L PURPOSE SCRATCH WORD  
LOGDEV: .WORD 0 : LOGICAL DEVICE NUMBER  
PSTACK: .WORD 0 : CONTAINS BASE LEVEL PROGRAM STACK POINTER  
PRIOR: .WORD 0 : CPU PRIORITY FOR PRINTOUT  
SUBRPC: .WORD 0 : PC OF SUBR CALL FOR ERROR REPORTS  
INTFLG: .WORD 0 : INTERRUPT RECEIVED FLAGS  
: BIT 0 FOR TX, BIT 1 FOR RCV  
ERRFLG: .WORD 0 : SUBROUTINE ERROR FLAG  
TIMFLG: .WORD 0 : EVENT TIME-OUT FLAG  
RETADR: .WORD 0 : SUBR ERROR RETURN ADDRESS  
REDBYT: .WORD 0 : LO BYTE CONTAINS BYTE READ FROM LU REG  
WRIBYT: .WORD 0 : LO BYTE CONTAINS BYTE TO LOAD INTO LU REG  
RAX15: .WORD 0 : LO BYTE CONTAINS BYTE READ FROM REG 15  
RAX16: .WORD 0 : LO BYTE CONTAINS BYTE READ FROM REG 16  
WAX15: .WORD 0 : LO BYTE CONTAINS BYTE TO LOAD INTO REG 15  
WAX16: .WORD 0 : LO BYTE CONTAINS BYTE TO LOAD INTO REG 16  
REGNUM: .WORD 0 : NUMBER (10-17) OF LINE UNIT REG BEING TESTED  
AXNUM: .WORD 0 : NUMBER (0-7) OF EXTENDED REG BYTE BEING TESTED  
GOODAT: .WORD 0 : STORAGE FOR EXPECTED DATA  
BADDAT: .WORD 0 : STORAGE FOR ACTUAL DATA  
LOADAT: .WORD 0 : CONTAINS TEST DATA LOADED INTO REG  
FRSTIM: .WORD 0 : FLAG=0 IF PROGRAM JUST LOADED  
SAVE4: .WORD 0 : SAVE LOC 4 HERE (ERROR TRAP VECTOR)  
SAVE6: .WORD 0 : SAVE LOC 6 HERE (ERROR TRAP VECTOR)  
ERROR1: .WORD 0 : SUBR ERROR BIT FLAGS (DEF'D IN GLOBAL EQUATES)  
TXWORD: .WORD 0 : BITS 0-11 CONTAIN DATA TO LOAD INTO TX SILO  
RXWORD: .WORD 0 : BITS 0-11 CONTAIN DATA READ FROM RCV SILO  
DISILO: .WORD 0 : CONTAINS CURRENT STATE OF DISSI IN BIT 5  
CHPTYP: .WORD 0 : USYRT CHIP TYPE, =0 FOR SIG, ELSE =1  
SAVLEN: .WORD 0 : SAVED TX AND RCV CHAR LENGTHS  
DEVMAP: .WORD 0 : BIT MAP OF ACTIVE DEVICES  
DEVPTR: .WORD 0 : DEVICE MAP BIT POINTER  
UNIT: .WORD 0 : CONTAINS UNIT NO. (1 TO N)  
TSTNUM: .WORD 0 : CONTAINS TEST NUMBER FOR SOME TESTS  
STARES: .WORD 0 : FLAG=0 IF FIRST PASS AFTER STA OR RES

\*\*\*\*\* CURRENT DEVICE PARAMETERS \*\*\*\*\*

MPCSR: .WORD 160170 : POINTER TO MICROPROCESSOR CSR'S  
BSEL1: .WORD 160171 : POINTER TO BSEL1  
BSEL2: .WORD 160172 : POINTER TO BSEL2  
BSEL4:

4834 002454 160174  
4835 002456 160176  
4836 002460 000300  
4837 002462 000304  
4838 002464 000240  
4839 002466 000000  
4840 002470 000000  
4841 002472 000000  
4842 002474 000000  
4843 002476 000000  
4844  
4845  
4846 002500 000  
4847 002501 000  
4848 002502 000  
4849 002503 000  
4850 002504 000  
4851 002505 000  
4852 002506 000  
4853 002507 000  
4854  
4855  
4856 002510 000000  
4857 002512 000000  
4858 002514 000000  
4859 002516 000000  
4860 002520 000000  
4861 002522 000000  
4862 002524 000000  
4863 002526 000000  
4864  
4865  
4866 002530 000000  
4867 002532 000000  
4868 002534 000000  
4869 002536 000000  
4870 002540 000000  
4871 002542 000000  
4872 002544 000000  
4873 002546 000000  
4874  
4875  
4876 002550 000  
4877 002550 056  
4878 002551 000  
4879 002552 257  
4880 002553 100  
4881 002554 377  
4882 002555 377  
4883 002556 306  
4884 002557 200  
4885  
4886 002560 200  
4887  
4888  
4889 002561

SEL4: .WORD 160174 ; POINTER TO SEL4  
SEL6: .WORD 160176 ; POINTER TO SEL6  
MPIVEC: .WORD 300 ; MICROPROCESSOR INPUT INTERRUPT VECTOR  
MPOVEC: .WORD 304 ; MICROPROCESSOR OUTPUT INTERRUPT VECTOR  
MPRIOR: .WORD 240 ; MICROPROCESSOR DEVICE PRIORITY  
LUSWI1: .WORD 0 ; LINE UNIT SWITCH PACK #1  
LUSWI2: .WORD 0 ; LINE UNIT SWITCH PACK #2  
LUSWI3: .WORD 0 ; LINE UNIT SWITCH PACK #3  
TSTCON: .WORD 0 ; TEST CONNECTOR INDICATOR  
RUNINH: .WORD 0 ; RUN SWITCH INDICATOR

\*\*\*\*\* STORAGE FOR DATA READ IN ADDRESS TESTS \*\*\*\*\*

REDDAT: .BYTE 0  
.BYTE 0  
.BYTE 0  
.BYTE 0  
.BYTE 0  
.BYTE 0  
.BYTE 0  
.BYTE 0

\*\*\*\*\* GEN'L PURPOSE SCRATCH STORAGE \*\*\*\*\*

REG0: .WORD 0  
REG1: .WORD 0  
REG2: .WORD 0  
REG3: .WORD 0  
REG4: .WORD 0  
REG5: .WORD 0  
REG6: .WORD 0  
REG7: .WORD 0

\*\*\*\*\* SCRATCH STORAGE FOR MESSAGE REPORTING \*\*\*\*\*

TMP0: .WORD 0  
TMP1: .WORD 0  
TMP2: .WORD 0  
TMP3: .WORD 0  
TMP4: .WORD 0  
TMP5: .WORD 0  
TMP6: .WORD 0  
TMP7: .WORD 0

\*\*\*\*\* INBUS LU REG BIT MASKS FOR UNPREDICTABLE BITS \*\*\*\*\*

UPBITS: .BYTE 000 ; MASK FOR REG 10  
.BYTE 056 ; MASK FOR REG 11  
.BYTE 000 ; MASK FOR REG 12  
.BYTE 257 ; MASK FOR REG 13  
.BYTE 100 ; MASK FOR REG 14  
.BYTE 377 ; MASK FOR REG 15  
.BYTE 377 ; MASK FOR REG 16  
.BYTE 306 ; MASK FOR REG 17

R14NRW: .BYTE 200 ; REG 14 NON-R/W BITS

\*\*\*\*\* MASKS FOR EXTENDED REGISTER NON-READ/WRITE BITS \*\*\*\*\*

ANBITS:

4890	002561	377	.BYTE	377	:MASK FOR AX0-15
4891	002562	377	.BYTE	377	:MASK FOR AX0-16
4892	002563	000	.BYTE	000	:MASK FOR AX1-15
4893	002564	360	.BYTE	360	:MASK FOR AX1-16
4894	002565	000	.BYTE	000	:MASK FOR AX2-15
4895	002566	000	.BYTE	000	:MASK FOR AX2-16
4896	002567	004	.BYTE	004	:MASK FOR AX3-15
4897	002570	030	.BYTE	030	:MASK FOR AX3-16

4898  
4899

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

4900	002571		PATA:		
4901	002571	125	.BYTE	125	
4902	002572	252	.BYTE	252	
4903	002573	000	.BYTE	000	
4904	002574	377	.BYTE	377	
4905	002575	001	.BYTE	001	
4906	002576	002	.BYTE	002	
4907	002577	004	.BYTE	004	
4908	002600	010	.BYTE	010	
4909	002601	020	.BYTE	020	
4910	002602	040	.BYTE	040	
4911	002603	100	.BYTE	100	
4912	002604	200	.BYTE	200	
4913	002605	376	.BYTE	376	
4914	002606	375	.BYTE	375	
4915	002607	373	.BYTE	373	
4916	002610	367	.BYTE	367	
4917	002611	357	.BYTE	357	
4918	002612	337	.BYTE	337	
4919	002613	277	.BYTE	277	
4920	002614	177	.BYTE	177	

4921  
4922

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

4923	002615		PATB:		
4924	002615	000	.BYTE	000	
4925	002616	000	.BYTE	000	
4926	002617	040	.BYTE	040	
4927	002620	100	.BYTE	100	
4928	002621	220	.BYTE	220	
4929	002622	000	.BYTE	000	
4930	002623	000	.BYTE	000	
4931	002624	051	.BYTE	051	

4932  
4933

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

4934	002625		PATC:		
4935	002625	020	.BYTE	020	
4936	002626	020	.BYTE	020	
4937	002627	020	.BYTE	020	

4938  
4939

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

4940	002630		PATD:		
4941	002630	000	.BYTE	000	
4942	002631	040	.BYTE	040	
4943	002632	000	.BYTE	000	

4944  
4945

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

4946	002633	
4947	002633	000
4948	002634	120
4949	002635	020
4950	002636	100
4951	002637	120
4952	002640	000
4953		
4954		
4955	002641	
4956	002641	050
4957	002642	051
4958	002643	050
4959		
4960		
4961	002644	
4962	002644	000
4963	002645	000
4964	002646	240
4965	002647	120
4966	002650	177
4967	002651	000
4968	002652	000
4969	002653	001
4970		
4971		
4972	002654	
4973	002654	000
4974	002655	000
4975	002656	377
4976	002657	017
4977	002660	377
4978	002661	377
4979	002662	375
4980	002663	377
4981		
4982		
4983	002664	
4984	002664	000
4985	002665	000
4986	002666	000
4987	002667	000
4988	002670	000
4989	002671	103
4990	002672	000
4991	002673	000
4992		
4993		
4994	002674	
4995	002674	000
4996	002675	000
4997	002676	010
4998	002677	002
4999	002700	004
5000	002701	103
5001	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 375  
.BYTE 377

\*\*\*\*\* DATA PATTERN I \*\*\*\*\*  
PATI:  
.BYTE 000  
.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

5002	002703	100	.BYTE	100
5003				
5004				
5005	002704	000		
5006	002705	000		
5007	002706	377		
5008	002707	377		
5009	002710	125		
5010	002711	125		
5011	002712	252		
5012	002713	252		
5013	002714	001		
5014	002715	000		
5015	002716	002		
5016	002717	000		
5017	002720	004		
5018	002721	000		
5019	002722	010		
5020	002723	000		
5021	002724	020		
5022	002725	000		
5023	002726	040		
5024	002727	000		
5025	002730	100		
5026	002731	000		
5027	002732	200		
5028	002733	000		
5029	002734	000		
5030	002735	001		
5031	002736	000		
5032	002737	002		
5033	002740	000		
5034	002741	004		
5035	002742	000		
5036	002743	010		
5037	002744	000		
5038	002745	020		
5039	002746	000		
5040	002747	040		
5041	002750	000		
5042	002751	100		
5043	002752	000		
5044	002753	200		
5045	002754	376		
5046	002755	377		
5047	002756	375		
5048	002757	377		
5049	002760	373		
5050	002761	377		
5051	002762	367		
5052	002763	377		
5053	002764	357		
5054	002765	377		
5055	002766	337		
5056	002767	377		
5057	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

5058	002771	377	.BYTE	377
5059	002772	177	.BYTE	177
5060	002773	377	.BYTE	377
5061	002774	377	.BYTE	377
5062	002775	376	.BYTE	376
5063	002776	377	.BYTE	377
5064	002777	375	.BYTE	375
5065	003000	377	.BYTE	377
5066	003001	373	.BYTE	373
5067	003002	377	.BYTE	377
5068	003003	367	.BYTE	367
5069	003004	377	.BYTE	377
5070	003005	357	.BYTE	357
5071	003006	377	.BYTE	377
5072	003007	337	.BYTE	337
5073	003010	377	.BYTE	377
5074	003011	277	.BYTE	277
5075	003012	377	.BYTE	377
5076	003013	177	.BYTE	177

\*\*\*\*\* DATA PATTERN L \*\*\*\*\*  
PATL:

5079	003014		.BYTE	000
5080	003014	000	.BYTE	000
5081	003015	000	.BYTE	000
5082	003016	377	.BYTE	377
5083	003017	377	.BYTE	377
5084	003020	000	.BYTE	000
5085	003021	000	.BYTE	000

\*\*\*\*\* DATA PATTERN M \*\*\*\*\*  
PATM:

5088	003022		.BYTE	000
5089	003022	000	.BYTE	020
5090	003023	020	.BYTE	000
5091	003024	000	.BYTE	000
5092	003025	000	.BYTE	200
5093	003026	200	.BYTE	000
5094	003027	000	.BYTE	000
5095	003030	000	.BYTE	051
5096	003031	051	.BYTE	

\*\*\*\*\* DATA PATTERN N \*\*\*\*\*  
PATN:

5098			.BYTE	000
5099	003032		.BYTE	000
5100	003032	000	.BYTE	000
5101	003033	000	.BYTE	125
5102	003034	000	.BYTE	000
5103	003035	125	.BYTE	000
5104	003036	000	.BYTE	252
5105	003037	252	.BYTE	000
5106	003040	000	.BYTE	377
5107	003041	377	.BYTE	005
5108	003042	005	.BYTE	000
5109	003043	000	.BYTE	012
5110	003044	012	.BYTE	000
5111	003045	000	.BYTE	017
5112	003046	017	.BYTE	000
5113	003047	000	.BYTE	

5114  
5115  
5116 003050  
5117 003050 000  
5118 003051 041  
5119 003052 004  
5120 003053 010  
5121 003054 020  
5122 003055 040  
5123 003056 100  
5124 003057 101  
5125 003060 200  
5126 003061 201  
5127 003062 300  
5128 003063 111  
5129 003064 301  
5130 003065 375  
5131  
5132  
5133 003066  
5134 003066 000  
5135 003067 113  
5136 003070 200  
5137 003071 040  
5138 003072 020  
5139 003073 010  
5140 003074 001  
5141 003075 104  
5142 003076 007  
5143 003077 105  
5144 003100 007  
5145 003101 144  
5146 003102 107  
5147 003103 157  
5148  
5149  
5150 003104 000  
5151 003105 000  
5152 003106 001  
5153 003107 000  
5154 003110 013  
5155 003111 000  
5156 003112 011  
5157 003113 000  
5158 003114 021  
5159 003115 000  
5160 003116 101  
5161 003117 000  
5162 003120 301  
5163 003121 000  
5164 003122 000  
5165 003123 001  
5166 003124 000  
5167 003125 002  
5168 003126 000  
5169 003127 004

\*\*\*\*\* 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

\*\*\*\*\* 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

\*\*\*\*\* 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



5170	003130	000	.BYTE	000
5171	003131	040	.BYTE	040
5172	003132	000	.BYTE	000
5173	003133	100	.BYTE	100
5174	003134	000	.BYTE	000
5175	003135	200	.BYTE	200
5176	003136	000	.BYTE	000
5177	003137	346	.BYTE	346
5178	003140	000	.BYTE	000
5179	003141	345	.BYTE	345
5180	003142	000	.BYTE	000
5181	003143	343	.BYTE	343
5182	003144	000	.BYTE	000
5183	003145	307	.BYTE	307
5184	003146	000	.BYTE	000
5185	003147	247	.BYTE	247
5186	003150	000	.BYTE	000
5187	003151	147	.BYTE	147

5188  
5189

\*\*\*\*\* PATTERN V \*\*\*\*\*

5190	003152	000	PATV: .BYTE	000
5191	003153	000	.BYTE	000
5192	003154	333	.BYTE	333
5193	003155	000	.BYTE	000
5194	003156	331	.BYTE	331
5195	003157	000	.BYTE	000
5196	003160	323	.BYTE	323
5197	003161	000	.BYTE	000
5198	003162	313	.BYTE	313
5199	003163	000	.BYTE	000
5200	003164	233	.BYTE	233
5201	003165	000	.BYTE	000
5202	003166	133	.BYTE	133
5203	003167	000	.BYTE	000
5204	003170	000	.BYTE	000
5205	003171	001	.BYTE	001
5206	003172	000	.BYTE	000
5207	003173	002	.BYTE	002
5208	003174	000	.BYTE	000
5209	003175	004	.BYTE	004
5210	003176	000	.BYTE	000
5211	003177	040	.BYTE	040
5212	003200	000	.BYTE	000
5213	003201	100	.BYTE	100
5214	003202	000	.BYTE	000
5215	003203	200	.BYTE	200
5216	003204	000	.BYTE	000
5217	003205	346	.BYTE	346
5218	003206	000	.BYTE	000
5219	003207	345	.BYTE	345
5220	003210	000	.BYTE	000
5221	003211	343	.BYTE	343
5222	003212	000	.BYTE	000
5223	003213	307	.BYTE	307
5224	003214	000	.BYTE	000
5225	003215	247	.BYTE	247

5226 003216 000 .BYTE 000  
5227 003217 147 .BYTE 147

5228  
5229 003220 ENDPAT:  
5230 .EVEN  
5231  
5232

5233  
5234  
5235

5236 ;\*\*\* TEST MESSAGES TO BE TRANSMITTED \*\*\*  
5237

5238 003220 000400 MSG1: TXSOM  
5239 003222 000400 TXSOM  
5240 003224 000000 000  
5241 003226 000125 125  
5242 003230 000252 252  
5243 003232 000377 377  
5244 003234 000000 000  
5245 003236 001000 TXEOM  
5246 003240 001000 TXEOM  
5247 003242 001000 TXEOM  
5248 003244 001000 TXEOM

5249  
5250 003246 000377 MSG4: 377  
5251 003250 000000 000  
5252 003252 000125 125  
5253 003254 000252 252  
5254 003256 001000 TXEOM  
5255 003260 001000 TXEOM

5256  
5257  
5258  
5259

5260 ;\*\*\* RECEIVED DATA BUFFER (64. WORDS) \*\*\*  
5261 RCVBUF: .BLKW 64.  
5262 003262 000100

5263  
5264  
5265  
5266  
5267  
5268  
5269  
5270  
5271  
5272



5309  
5310  
5311  
5312  
5313  
5314  
5315  
5316  
5317  
5318  
5319  
5320  
5321  
5322  
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

.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  
MOV REGNUM,-(SP) ;SAVE LU REG NO.  
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 #13,REGNUM ;SET LU REG NO. = 13  
CLR WRIBYT  
JSR PC,WRITLU ;CLEAR REG 13  
MOV (SP)+,REGNUM ;RESTORE LU REG NO.  
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.  
\*\*\*\*\*

5365 003672  
5366 003672 010146  
5367 003674 013701 002400  
5368 003700 006301  
5369 003702 006301  
5370 003704 006301  
5371 003706 006301  
5372 003710 052701 000004  
5373 003714 052701 021000  
5374 003720 010137 003730  
5375 003724 004737 003540  
5376 003730 000000  
5377 003732 117737 176516 002364  
5378 003740 105037 002365  
5379 003744 012601  
5380 003746 000207

READLU:  
MOV R1,-(SP) ;SAVE R1  
MOV REGNUM,R1 ;GET LINE UNIT REG NUMBER  
ASL R1 ;SHIFT INTO SOURCE BITS 4-7  
ASL R1  
ASL R1  
ASL R1  
BIS #4,R1 ;SET DESTINATION = BSEL4  
BIS #MVIOX,R1 ;SET REST OF MOVE INSTRUCTION  
MOV R1,2\$ ;SET INSTRUCTION AS SUBROUTINE ARGUMENT  
JSR PC,STPCLK ;EXECUTE MOVE INSTRUCTION  
2\$: .WORD 0 ;INSTRUCTION GOES HERE  
MOVB @BSEL4,REDBYT ;GET LU REG CONTENTS INTO REDBYT  
CLRB REDBYT+1 ;CLR HI BYTE OF STORAGE  
MOV -(SP)+,R1 ;RESTORE R1  
RTS PC ;RETURN

\*\*\*\*\*  
;\* 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,  
\*\*\*\*\*

5391 003750  
5392 003750 010146  
5393 003752 013701 002400  
5394 003756 052701 000100  
5395 003762 052701 122000  
5396 003766 010137 004010  
5397 003772 105037 002367  
5398 003776 113777 002366 176450  
5399 004004 004737 003540  
5400 004010 000000  
5401 004012 012601  
5402 004014 000207

WRITLU:  
MOV R1,-(SP) ;SAVE R1  
MOV REGNUM,R1 ;GET LINE UNIT REG NUMBER  
BIS #100,R1 ;SET SOURCE = BSEL4  
BIS #MVIXO,R1 ;SET REST OF MOVE INSTRUCTION  
MOV R1,2\$ ;SET INSTRUCTION AS SUBROUTINE ARGUMENT  
CLRB WRIBYT+1 ;CLR HI BYTE OF STORAGE  
MOVB WRIBYT,@BSEL4 ;LOAD BYTE INTO BSEL4  
JSR PC,STPCLK ;EXECUTE MOVE INSTRUCTION  
2\$: .WORD 0  
MOV (SP)+,R1 ;RESTORE R1  
RTS PC ;RETURN

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

5412 004016 010146  
5413 004020 013746 002400  
5414 004024 012701 002302  
5415 004030 012737 000010 002400  
5416 004036 004737 003672  
5417 004042 113721 002364  
5418 004046 105021  
5419 004050 005237 002400  
5420 004054 023727 002400 000020

GETREG: MOV R1,-(SP) ;SAVE R1  
MOV REGNUM,-(SP) ;SAVE CURRENT REG NO.  
MOV #LUR10,R1 ;INIT POINTER TO REG STORAGE TABLE  
MOV #10,REGNUM ;INIT LU REG NO. TO 10  
3\$: JSR PC,READLU ;READ A LINE UNIT REG  
MOVB REDBYT,(R1)+ ;PUT BYTE READ INTO TABLE  
CLRB (R1)+ ;CLEAR UPPER BYTE OF TABLE ENTRY  
INC REGNUM ;INCREMENT REG NO.  
CMP REGNUM,#20 ;SEE IF ALL REGS READ YET

5421 004062 002765  
 5422 004064 012637 002400  
 5423 004070 012601  
 5424 004072 000207

BLT 3\$ :BR IF NOT  
 MOV (SP)+,REGNUM :RESTORE CURRENT REG NO.  
 MOV (SP)+,R1 :RESTORE R1  
 RTS PC :RETURN

5425  
 5426  
 5427  
 5428  
 5429

5430  
 5431  
 5432  
 5433  
 5434  
 5435  
 5436  
 5437

```

*****
;* 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.
*****

```

5438 004074  
 5439 004074 152777 000006 176346  
 5440 004102 017677 000000 176346  
 5441 004110 152777 000206 176332  
 5442 004116 062716 000002  
 5443 004122 000207

```

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

```

5444  
 5445  
 5446  
 5447  
 5448  
 5449  
 5450

5451  
 5452  
 5453  
 5454  
 5455

```

*****
;* 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.
*****

```

5456 004124 010146  
 5457 004126 013746 002400  
 5458 004132 042737 000001 002420  
 5459 004140 012737 000014 002400  
 5460 004146 113737 002402 002366  
 5461 004154 006237 002366  
 5462 004160 152737 000024 002366  
 5463 004166 053737 002426 002366  
 5464 004174 004737 003750  
 5465 004200 005001  
 5466 004202 004737 003672 6\$:  
 5467 004206 132737 000200 002364  
 5468 004214 001006  
 5469 004216 005201  
 5470 004220 001370  
 5471 004222 052737 000001 002420  
 5472 004230 000421  
 5473 004232 012737 000015 002400 9\$:  
 5474 004240 004737 003672  
 5475 004244 113737 002364 002370  
 5476 004252 105037 002371

```

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
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

```

5477 004256 012737 000016 002400  
5478 004264 004737 003672  
5479 004270 113737 002364 002372  
5480 004276 105037 002373  
5481 004302 012637 002400  
5482 004306 012601  
5483 004310 000207  
5484  
5485  
5486  
5487  
5488  
5489  
5490  
5491  
5492  
5493  
5494 004312 010146  
5495 004314 013746 002400  
5496 004320 042737 000002 002420  
5497 004326 012737 000014 002400  
5498 004334 113737 002402 002366  
5499 004342 006237 002366  
5500 004346 053737 002426 002366  
5501 004354 004737 003750  
5502 004360 012737 000015 002400  
5503 004366 105037 002375  
5504 004372 113737 002374 002366  
5505 004400 004737 003750  
5506 004404 005237 002400  
5507 004410 105037 002377  
5508 004414 113737 002376 002366  
5509 004422 004737 003750  
5510 004426 012737 000014 002400  
5511 004434 113737 002402 002366  
5512 004442 006237 002366  
5513 004446 152737 000014 002366  
5514 004454 053737 002426 002366  
5515 004462 004737 003750  
5516 004466 005001  
5517 004470 004737 003672  
5518 004474 132737 000200 002364  
5519 004502 001005  
5520 004504 005201  
5521 004506 001370  
5522 004510 052737 000002 002420  
5523 004516 012637 002400  
5524 004522 012601  
5525 004524 000207  
5526  
5527  
5528  
5529  
5530  
5531  
5532

```
MOV #16,REGNUM ;SET REG NO. = 16
JSP 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.
MOV (SP)+,R1 ;RESTORE R1
RTS PC ;RETURN

*****
;* WRITAX - THIS SUBROUTINE WRITES THE USYRT REG PAIR WHOSE NUMBER (0-3) IS
;* PASSED IN BITS 1,2 OF AXNUM ON ENTRY, WITH THE DATA FROM WAX15 AND
;* WAX16. IF LINE UNIT DOES NOT RESPOND WITH READY IN REG 14, WRDYTO BIT
;* IS SET IN ERROR1 ON RETURN.
*****
WRITAX: MOV R1,-(SP) ;SAVE R1
MOV REGNUM,-(SP) ;SAVE CURRENT REG NO.
BIC #WRDYTO,ERROR1 ;CLEAR ERROR BIT
MOV #14,REGNUM ;SET LU REG NO. = 14
MOVB AXNUM,WRIBYT ;SET AX REG NO. BITS
ASR WRIBYT
BIS DISILO,WRIBYT ;SET PROPER STATE OF DISSI BIT
JSR PC,WRITLU ;SET AX NO. BITS IN REG 14
MOV #15,REGNUM ;SET REG NO. = 15
CLRB WAX15+1 ;CLR HI BYTE OF STORAGE
MOVB WAX15,WRIBYT ;SET UP BYTE TO WRITE INTO REG 15
JSR PC,WRITLU ;WRITE BYTE INTO REG 15
INC REGNUM ;SET REG NO. = 16
CLRB WAX16+1 ;CLR HI BYTE OF STORAGE
MOVB WAX16,WRIBYT ;SET UP BYTE TO WRITE INTO REG 16
JSR PC,WRITLU ;WRITE BYTE INTO REG 16
MOV #14,REGNUM ;SET REG NO. = 14
MOVB AXNUM,WRIBYT ;SET AX REG NO. BITS
ASR WRIBYT
BISB #ENAX!WAX,WRIBYT ;SET UP BITS TO LOAD INTO REG 14
BIS DISILO,WRIBYT ;SET PROPER STATE OF DISSI BIT
JSR PC,WRITLU ;SET ENAX AND WAX IN REG 14
CLR R1 ;INIT PROGRAM TIMER
6$: JSR PC,READLU ;READ REG 14
BITB #READY,REDBYT ;SEE IF READY BIT SET IN REG 14 YET
BNE 9$ ;BR IF READY SET
INC R1 ;INCR TIMER
BNE 6$ ;BR IF TIMER DIDN'T TIME OUT YET
9$: BIS #WRDYTO,ERROR1 ;SET ERROR FLAG BIT FOR TIME OUT ON WRITE RDY
MOV (SP)+,REGNUM ;RESTORE CURRENT REG NO.
MOV (SP)+,R1 ;RESTORE R1
RTS PC ;RETURN
```

\*\*\*\*\*  
;\* GETALL - THIS SUBROUTINE READS THE LINE UNIT REGS 10-17 AND THE EXTENDED

```

5533          ;*   REGISTERS AX0-AX3 INTO REGISTER STORAGE TABLE (LUREG:).
5534          ;*****
5535 004526 010146          GETALL: MOV    R1,-(SP)          ;SAVE R1
5536 004530 013746 002402  MOV    AXNUM,-(SP)        ;SAVE CURRENT AX REG BYTE NO.
5537 004534 012737 014435 002530  MOV    #DH5,TMPO         ;SET AX LO BYTE NO.
5538 004542 032737 000001 002402  BIT    #BIT0,AXNUM       ;SEE IF LO OR HI BYTE
5539 004550 001403          BEQ    1$                ;BR IF LO BYTE
5540 004552 012737 014440 002530  MOV    #DH6,TMPO         ;SET AX HI BYTE NO.
5541 004560 004737 004016 1$:   JSR    PC,GETREG        ;READ AND STORE REGS 10-17
5542 004564 142777 000010 175656  BICB  #LULoop,@BSEL1    ;CLEAR LULoop
5543 004572 012701 002322          MOV    #AX0.15,R1       ;INIT POINTER TO REG STORAGE TABLE
5544 004576 005037 002402          CLR    AXNUM            ;INIT AX REG BYTE NO. TO 0
5545 004602 004737 004124 3$:   JSR    PC,READAX        ;READ 2 AX REG BYTES
5546 004606 113721 002370          MOVB  RAX15,(R1)+       ;PUT LO BYTE READ INTO TABLE
5547 004612 105021          CLRB  (R1)+             ;CLEAR UPPER BYTE OF TABLE ENTRY
5548 004614 113721 002372          MOVB  RAX16,(R1)+       ;PUT HI BYTE READ INTO TABLE
5549 004620 105021          CLRB  (R1)+             ;CLEAR UPPER BYTE OF TABLE ENTRY
5550 004622 062737 000002 002402  ADD    #2,AXNUM         ;INCR AX REG BYTE NO.
5551 004630 023727 002402 000010  CMP    AXNUM,#10        ;SEE IF ALL REGS READ YET
5552 004636 002761          BLT   3$                ;BR IF NOT
5553 004640 012637 002402          MOV   (SP)+,AXNUM       ;RESTORE CURRENT AX REG BYTE NO.
5554 004644 012601          MOV   (SP)+,R1         ;RESTORE R1
5555 004646 013737 002402 002532  MOV   AXNUM,TMP1
5556 004654 006237 002532          ASR   TMP1              ;GET EXTENDED REG NO. FOR PRINTOUT
5557 004660 000207          RTS    PC               ;RETURN
5558
5559
5560
5561
5562
5563

```

```

5564          ;*****
5565          ;* OSIRDY - THIS SUBROUTINE CHECKS FOR THE PROPER STATES OF ORDY (REG 11)
5566          ;*   AND OCOR (REG 17) AND REPORTS AN ERROR IF EITHER IS NOT PROPERLY SET
5567          ;*   AS PASSED IN BIT 0 (ORDY) AND BIT 1 (OCOR) OF THE WORD FOLLOWING THE
5568          ;*   CALL.
5569          ;*   IF AN ERROR OCCURS, A RETURN IS MADE TO THE TEST, AT THE ADDRESS IN
5570          ;*   RETADR.
5571          ;*****

```

```

5571 004662 013746 002400  OSIRDY: MOV    REGNUM,-(SP)      ;SAVE LU REG NO.
5572 004666 013746 002352  MOV    SUBRPC,-(SP)
5573 004672 005737 002352  TST   SUBRPC
5574 004676 001006          BNE   1$                ;SEE IF THIS IS A NESTED CALL
5575 004700 016637 000004 002352  MOV    4(SP),SUBRPC     ;BR IF YES
5576 004706 162737 000004 002352  SUB    #4,SUBRPC        ;GET PC OF SUBROUTINE CALL
5577 004714 012737 000011 002400 1$:   MOV    #11,REGNUM       ;SET REG NO. TO 11
5578 004722 004737 003672          JSR   PC,READLU        ;READ REG 11
5579 004726 032776 000001 000004  BIT    #BIT0,@4(SP)     ;GET EXPECTED STATE OF ORDY
5580 004734 001413          BEQ   3$                ;BR IF EXPECTED ORDY = 0
5581 004736 132737 000020 002364  BITB  #ORDY,REDBYT     ;SEE IF ORDY = 1
5582 004744 001022          BNE   9$                ;BR IF ORDY = 1
5583 004746 004737 004526          JSR   PC,GETALL       ;GET REGS FOR PRINTOUT
5584          ;REPORT ORDY NOT SET
5585          ERRDF 7,EM7,ERR4

```

```

(4) 004752 104455          TRAP  C$ERDF
(5) 004754 000007          .WORD 7
(5) 004756 012374          .WORD EM7

```



```

(5) 004760 015622 .WORD ERR4
5586 004762 000451
5587 004764 132737 000020 002364 3$: BR 16$ :TAKE ERROR RETURN
5588 004772 001407 BITB #ORDY,REDBYT :SEE IF ORDY = 0
5589 004774 004737 004526 BEQ 9$ :BR IF ORDY = 0
5590 JSR PC,GETALL :GET REGS FOR PRINTOUT
5591 005000 :REPORT ORDY NOT CLEARED
ERRDF 8,EM8,ERR4
(4) 005000 104455 TRAP C$ERDF
(5) 005002 000010 .WORD 8
(5) 005004 012411 .WORD EM8
(5) 005006 015622 .WORD ERR4
5592 005010 000436
5593 005012 012737 000017 002400 9$: BR 16$ :TAKE ERROR RETURN
5594 005020 004737 003672 MOV #17,REGNUM :SET REG NO. = 17
5595 005024 132776 000002 000004 JSR PC,READLU :READ LU REG 17
5596 005032 001413 BITB #BIT1,@4(SP) :GET EXPECTED STATE OF OCOR
5597 005034 132737 000020 002364 BEQ 12$ :BR IF EXPECTED OCOR = 0
5598 005042 001031 BITB #OCOR,REDBYT :SEE IF OCOR = 1
5599 005044 004737 004526 BNE 20$ :BR IF OCOR = 1
5600 JSR PC,GETALL :GET REGS FOR PRINTOUT
5601 005050 :REPORT OCOR NOT SET
ERRDF 9,EM9,ERR4
(4) 005050 104455 TRAP C$ERDF
(5) 005052 000011 .WORD 9
(5) 005054 012432 .WORD EM9
(5) 005056 015622 .WORD ERR4
5602 005060 000412
5603 005062 132737 000020 002364 12$: BR 16$ :TAKE ERROR RETURN
5604 005070 001416 BITB #OCOR,REDBYT :SEE IF OCOR = 0
5605 005072 004737 004526 BEQ 20$ :BR IF OCOR = 0
5606 JSR PC,GETALL :GET REGS FOR PRINTOUT
5607 005076 :REPORT OCOR NOT CLEARED
ERRDF 10,EM10,ERR4
(4) 005076 104455 TRAP C$ERDF
(5) 005100 000012 .WORD 10
(5) 005102 012447 .WORD EM10
(5) 005104 015622 .WORD ERR4
5608 005106 016637 000002 002400 16$: MOV 2(SP),REGNUM :RESTORE LU REG NO.
5609 005114 013706 002346 MOV PSTACK,SP :RESTORE STACK POINTER TO BASE LEVEL
5610 005120 013746 002362 MOV RETADR,-(SP) :FIX ERROR RETURN PC
5611 005124 000407 BR 23$
5612 005126 062766 000002 000004 20$: ADD #2,4(SP) :FIX UP ERROR-FREE RETURN PC
5613 005134 012637 002352 MOV (SP)+,SUBRPC
5614 005140 012637 002400 MOV (SP)+,REGNUM :RESTORE LU REG NO.
5615 005144 000207 23$: RTS PC :RETURN
5616
5617
5618
5619
5620
5621
5622
5623
5624 005146 010146
5625 005150 012701 000310
5626 005154 005301
5627 005156 001376
5628 005160 012601

```

\*\*\*\*\*  
: \* WAIT50 - THIS SUBROUTINE STALLS FOR AT LEAST 50 MICRO-SEC, AND THEN RETURNS.  
\*\*\*\*\*

```

WAIT50: MOV R1,-(SP) :SAVE R1
MOV #200,,R1 :INIT COUNTER
3$: DEC R1 :DECREMENT COUNTER
BNE 3$ :BR IF NOT DONE YET
MOV (SP)+,R1 :RESTORE R1

```

5629 005162 000207

RTS PC ;RETURN

5630  
 5631  
 5632  
 5633  
 5634  
 5635  
 5636  
 5637

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

5638 005164 000240  
 5639 005166 000240  
 5640 005170 000240  
 5641 005172 000207

STALL: NOP  
 NOP  
 NOP  
 RTS PC

5642  
 5643  
 5644  
 5645  
 5646  
 5647

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

5648  
 5649  
 5650  
 5651 005174 013746 002400  
 5652 005200 042737 170000 002422  
 5653 005206 012737 000011 002400  
 5654 005214 113737 002423 002366  
 5655 005222 004737 003750  
 5656 005226 012737 000010 002400  
 5657 005234 113737 002422 002366  
 5658 005242 004737 003750  
 5659 005246 012637 002400  
 5660 005252 000207

LDTXSI: MOV REGNUM, -(SP) ;SAVE LU REG NO.  
 BIC #170000, TXWORD ;CLEAR UNUSED BITS  
 MOV #11, REGNUM ;SET REG NO. = 11  
 MOVB TXWORD+1, WRIBY ;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, WRIBY ;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

5661  
 5662  
 5663  
 5664  
 5665  
 5666

\*\*\*\*\*  
 \* 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.  
 \*\*\*\*\*

5667  
 5668  
 5669  
 5670  
 5671  
 5672 005254 010146  
 5673 005256 017601 000002  
 5674 005262 001426  
 5675 005264 100006  
 5676 005266 042701 100000  
 5677 005272 005737 002430  
 5678 005276 001401  
 5679 005300 005301  
 5680 005302 152777 000010 175140 2\$:  
 5681 005310 152777 000020 175132 3\$:  
 5682 005316 004737 005164  
 5683 005322 142777 000020 175120  
 5684 005330 004737 005164

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

```
5685 005334 005301          DEC    R1          ;DECREMENT CYCLE COUNTER
5686 005336 001364          BNE    3$          ;BR IF NOT DONE YET
5687 005340 062766 000002 000002 6$:  ADD    #2,2(SP)    ;FIX UP RETURN PC
5688 005346 012601          MOV    (SP)+,R1    ;RESTORE R1
5689 005350 000207          RTS    PC          ;RETURN
```

5690  
5691  
5692  
5693  
5694  
5695

```
*****
;* 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.
*****
```

```
5700 005352 013746 002400  OACTIV: MOV    REGNUM,-(SP)  ;SAVE LU REG NO.
5701 005356 013746 002352  MOV    SUBRPC,-(SP)
5702 005362 005737 002352  TST    SUBRPC      ;SEE IF THIS IS A NESTED CALL
5703 005366 001006          BNE    1$          ;BR IF YES
5704 005370 016637 000004 002352  MOV    4(SP),SUBRPC
5705 005376 162737 000004 002352  SUB    #4,SUBRPC   ;GET PC OF SUBROUTINE CALL
5706 005404 012737 000011 002400 1$:  MOV    #11,REGNUM  ;SET REG NO. = 11-
5707 005412 004737 003672  JSR    PC,READLU   ;READ REG 11
5708 005416 032776 000001 000004  BIT    #BIT0,@4(SP) ;GET EXPECTED STATE OF OACT
5709 005424 001413          BEQ    3$          ;BR IF EXPECTED OACT = 0
5710 005426 132737 000100 002364  BITB   #OACT,REDBYT ;SEE IF OACT = 1
5711 005434 001031          BNE    9$          ;BR IF OACT = 1
5712 005436 004737 004526  JSR    PC,GETALL   ;GET REGS FOR PRINTOUT
```

```
5713 ;REPORT OACT NOT SET
5714 005442 ERRDF 11,EM11,ERR4
(4) 005442 104455 TRAP C$ERDF
(5) 005444 000013 .WORD 11
(5) 005446 012470 .WORD EM11
(5) 005450 015622 .WORD ERR4
```

```
5715 005452 000412          BR     6$          ;TAKE ERROR RETURN
5716 005454 132737 000100 002364 3$:  BITB   #OACT,REDBYT ;SEE IF OACT = 0
5717 005462 001416          BEQ    9$          ;BR IF OACT = 0
5718 005464 004737 004526  JSR    PC,GETALL   ;GET REGS FOR PRINTOUT
5719 ;REPORT OACT NOT CLEARED
```

```
5720 005470 ERRDF 12,EM12,ERR4
(4) 005470 104455 TRAP C$ERDF
(5) 005472 000014 .WORD 12
(5) 005474 012505 .WORD EM12
(5) 005476 015622 .WORD ERR4
```

```
5721 005500 016637 000002 002400 6$:  MOV    2(SP),REGNUM ;RESTORE LU REG NO.
5722 005506 013706 002346  MOV    PSTACK,SP   ;RESTORE PROGRAM STACK TO BASE LEVEL
5723 005512 013746 002362  MOV    RETADR,-(SP) ;FIX UP ERROR RETURN PC
5724 005516 000407          BR     12$         ;
5725 005520 062766 000002 000004 9$:  ADD    #2,4(SP)    ;FIX UP ERROR-FREE RETURN PC
5726 005526 012637 002352  MOV    (SP)+,SUBRPC
5727 005532 012637 002400  MOV    (SP)+,REGNUM ;RESTORE LU REG NO.
5728 005536 000207          RTS    PC          ;RETURN
```

5729  
5730  
5731  
5732

5733  
 5734  
 5735  
 5736  
 5737  
 5738  
 5739  
 5740  
 5741  
 5742  
 5743  
 5744  
 5745  
 5746  
 5747  
 5748  
 5749  
 5750  
 5751  
 5752  
 5753  
 5754  
 5755  
 5756  
 5757  
 5758  
 5759  
 5760  
 5761  
 5762  
 5763  
 5764  
 5765  
 5766  
 5767  
 5768  
 5769  
 5770  
 5771  
 5772  
 5773  
 5774  
 5775  
 5776  
 5777  
 5778  
 5779  
 5780  
 5781  
 5782  
 5783  
 5784  
 5785  
 5786  
 5787  
 5788

005540 010146  
 005542 013746 002400  
 005546 013746 002402  
 005552 016637 000006 002352  
 005560 162737 000004 002352  
 005566 004737 003576  
 005572 004737 004662  
 005576 000001  
 005600 004737 005352  
 005604 000000  
 005606 012737 000004 002402  
 005614 117637 000006 002374  
 005622 012737 000400 002422  
 005630 113737 002374 002422  
 005636 005037 002376  
 005642 004737 004312  
 005646 012737 000017 002400  
 005654 062766 000002 000006  
 005662 117637 000006 002366  
 005670 004737 003750  
 005674 004737 005174  
 005700 004737 005174  
 005704 004737 005146  
 005710 004737 004662  
 005714 000003  
 005716 004737 005352  
 005722 000000  
 005724 005001  
 005726 012737 000011 002400  
 005734 152777 000010 174506 6\$:  
 005742 152777 000020 174500  
 005750 004737 005164  
 005754 004737 003672  
 005760 132737 000100 002364  
 005766 001014  
 005770 142777 000020 174452  
 005776 004737 005164  
 006002 005201  
 006004 020127 000003  
 006010 002751  
 006012 004737 005352  
 006016 000001  
 006020 012737 000017 002400 9\$:  
 006026 005037 002430  
 006032 004737 003672

```

*****
* 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
* CLOCKING THE LINE UNIT UNTIL THE FIRST SYNCH OR FLAG HAS BEEN SERIALIZED
* IN THE USYRT. THE PROGRAM MONITORS ORDY,OCOR, AND OACT FOR VALID STATES,
* THROUGHOUT THE PROCESS.
* IF THE SUBROUTINE DETECTS AN ERROR, A RETURN IS MADE TO THE TEST, AT THE
* ADDRESS CONTAINED IN RETADR.
*****
INITRN: MOV R1,-(SP) ;SAVE R1
        MOV REGNUM,-(SP) ;SAVE LU REG NO.
        MOV AXNUM,-(SP) ;SAVE AX BYTE NO.
        MOV 6(SP),SUBRPC
        SUB #4,SUBRPC ;GET PC OF SUBR CALL
        JSR PC,MSTCLR ;ISSUE A MASTER CLEAR
        JSR PC,OSIRDY ;CHECK ORDY=1, OCOR=0
        1
        JSR PC,OACTIV ;CHK OACT=0
        0
        MOV #4,AXNUM ;SET AX BYTE NO. = 4 FOR AX2
        MOVB @6(SP),WAX15 ;SET DATA BYTE TO LOAD INTO AX2-15
        MOV #TXSOM,TXWORD ;SET TSOM BIT
        MOVB WAX15,TXWORD ;SET SYNCH CHAR
        CLR WAX16
        JSR PC,WRITAX ;LOAD AX2
        MOV #17,REGNUM ;SET REG NO. = 17
        ADD #2,6(SP) ;INCR POINTER TO NEXT DATA BYTE
        MOVB @6(SP),WRIBYT ;SET DATA BYTE TO LOAD INTO REG 17
        JSR PC,WRITLU ;LOAD REG 17
        JSR PC,LDTXSI ;LOAD THE SILO WITH SOM CHAR
        JSR PC,LDTXSI ;LOAD ANOTHER SOM INTO SILO
        JSR PC,WAIT50 ;WAIT FOR DATA TO RIPPLE
        JSR PC,OSIRDY ;CHK ORDY=1, OCOR=1
        3
        JSR PC,OACTIV ;CHK FOR OACT = 0
        0
        CLR R1 ;INIT CYCLE COUNTER
        MOV #11,REGNUM ;SET LU REG NO. = 11
        BISB #LULOOP,@BSEL1 ;SET LINE UNIT LOOP BIT
        BISB #STEPLU,@BSEL1 ;SET CLOCK BIT
        JSR PC,STALL ;STALL FOR MICRO-SEC
        JSR PC,READLU ;READ REG 11
        BITB #OACT,REDBYT ;SEE IF OACT = 1 YET
        BNE 9$ ;BR IF OACT = 1
        BICB #STEPLU,@BSEL1 ;CLEAR CLOCK BIT
        JSR PC,STALL ;STALL FOR A MICRO-SEC
        INC R1 ;INCR CYCLE COUNT
        CMP R1,#3 ;SEE IF 3 CYCLES DONE YET
        BLT 6$ ;BR IF NOT
        JSR PC,OACTIV ;CHK FOR OACT = 1
        1
        MOV #17,REGNUM ;SET REG NO. = 17
        CLR CHPTYP ;CLEAR USYRT CHIP INDICATOR
        JSR PC,READLU ;READ REG 17
  
```

```

5789 006036 132737 000020 002364 BITB #OCOR,REDBYT ;CHK FOR OCOR CLEARED YET
5790 006044 001403 BEQ 12$ ;BR IF YES - IT IS SIG CHIP
5791 006046 012737 000001 002430 MOV #1,CHPTYP ;SET INDICATOR FOR OTHER CHIP TYPE
5792 006054 142777 000020 174366 12$: BICB #STEPLU,@BSEL1 ;CLEAR CLOCK BIT
5793 006062 004737 005164 JSR PC,STALL ;STALL FOR MICRO-SEC
5794 006066 004737 004662 JSR PC,OSIRDY ;CHK FOR ORDY = 1, OCOR = 0
5795 006072 000001 1
5796 006074 062766 000002 000006 ADD #2,6(SP) ;FIX UP RETURN PC
5797 006102 012637 002402 MOV (SP)+,AXNUM ;RESTORE AX BYTE NO.
5798 006106 012637 002400 MOV (SP)+,REGNUM ;RESTORE LU REG NO.
5799 006112 012601 MOV (SP)+,R1 ;RESTORE R1
5800 006114 005037 002352 CLR SUBRPC ;CLEAR SUBR CALL PC
5801 006120 000207 RTS PC ;RETURN
5802
5803
5804
5805
5806
5807

```

```

*****
* 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.
*****

```

```

5817
5818 006122 010146 TXCHAR: MOV R1,-(SP) ;SAVE R1
5819 006124 010246 MOV R2,-(SP) ;SAVE R2
5820 006126 016637 000004 002352 MOV 4(SP),SUBRPC
5821 006134 162737 000004 002352 SUB #4,SUBRPC ;GET PC OF SUBR CALL
5822 006142 017637 000004 002422 MOV @4(SP),TXWORD ;GET DATA TO BE TRANSMITTED
5823 006150 004737 005174 JSR PC,LDTXSI ;LOAD THE TX SILO WITH THE DATA
5824 006154 004737 005146 JSR PC,WAIT50 ;WAIT FOR DATA TO RIPPLE DOWN SILO
5825 006160 062766 000002 000004 ADD #2,4(SP) ;INCR POINTER
5826 006166 005001 CLR R1 ;INIT CYCLE COUNT
5827 006170 017602 000004 MOV @4(SP),R2 ;GET DESIRED NO. OF CYCLES
5828 006174 005702 TST R2 ;SEE IF CHIP TYPE CHK SHOULD BE MADE
5829 006176 100006 BPL 9$ ;BR IF NOT
5830 006200 042702 100000 BIC #BIT15,R2 ;CLEAR FLAG BIT
5831 006204 005737 002430 TST CHPTYP ;SEE IF SIG USYRT
5832 006210 001401 BEQ 9$ ;BR IF YES
5833 006212 005302 DEC R2 ;DECREMENT NO. OF CYCLES
5834 006214 004737 005352 9$: JSR PC,OACTIV ;CHK OACT = 1
5835 006220 000001 1
5836 006222 020102 CMP R1,R2 ;SEE IF REQUIRED CYCLES DONE YET
5837 006224 001410 BEQ 12$ ;BR IF YES
5838 006226 004737 004662 JSR PC,OSIRDY ;CHK ORDY=1, OCOR=1
5839 006232 000003 3
5840 006234 004737 005254 JSR PC,STPLU ;STEP LU ONE CYCLE
5841 006240 000001 1
5842 006242 005201 INC R1 ;INCR CYCLE COUNT
5843 006244 000763 BR 9$
5844 006246 004737 004662 12$: JSR PC,OSIRDY ;CHK ORDY=1, OCOR=0

```

5845 006252 000001  
5846 006254 062766 000002 000004  
5847 006262 005037 002352  
5848 006266 012602  
5849 006270 012601  
5850 006272 000207

1  
ADD #2,4(SP) ;FIX UP RETURN PC  
CLR SUBRPC ;CLEAR SUBR CALL PC  
MOV (SP)+,R2 ;RESTORE R2  
MOV (SP)+,R1 ;RESTORE R1  
RTS PC ;RETURN

5851  
5852  
5853  
5854  
5855  
5856

\*\*\*\*\*  
\* 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.  
\*\*\*\*\*

5860 006274 010146  
5861 006276 013746 002400  
5862 006302 016637 000004 002352  
5863 006310 162737 000004 002352  
5864 006316 012737 000011 002400  
5865 006324 012737 000200 002366  
5866 006332 004737 003750  
5867 006336 004737 005146  
5868 006342 004737 004662  
5869 006346 000001  
5870 006350 004737 005352  
5871 006354 000000  
5872 006356 012737 000013 002400  
5873 006364 004737 003672  
5874 006370 032737 000040 002364  
5875 006376 001406  
5876 006400 004737 004526

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  
;REPORT RTS NOT CLEARED  
ERRDF 65,EM65,ERR4

5877  
5878 006404  
(4) 006404 104455  
(5) 006406 000101  
(5) 006410 014244  
(5) 006412 015622  
5879 006414 005037 002352  
5880 006420 012637 002400  
5881 006424 012601  
5882 006426 000207

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

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

5883  
5884  
5885  
5886  
5887  
5888

\*\*\*\*\*  
\* 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.  
\*\*\*\*\*

5896 006430 013746 002400

ISIRDY: MOV REGNUM,-(SP) ;SAVE LU REG NO.

```

5897 006434 013746 002352      MOV      SUBRPC,-(SP)
5898 006440 005737 002352      TST      SUBRPC          ;SEE IF THIS IS A NESTED CALL
5899 006444 001006                BNE      1$              ;BR IF YES
5900 006446 016637 000004 002352      MOV      4(SP),SUBRPC
5901 006454 162737 000004 002352      SUB      #4,SUBRPC      ;GET PC OF SUBR CALL
5902 006462 012737 000012 002400 1$:      MOV      #12,REGNUM     ;SET REG NO. TO 12
5903 006470 004737 003672                JSR      PC,READLU      ;READ REG 12
5904 006474 032776 000002 000004      BIT      #BIT1,@4(SP)   ;GET EXPECTED STATE OF IRDY
5905 006502 001413                BEQ      3$              ;BR IF EXPECTED IRDY = 0
5906 006504 132737 000020 002364      BITB     #IRDY,REDBYT   ;SEE IF IRDY = 1
5907 006512 001022                BNE      9$              ;BR IF IRDY = 1
5908 006514 004737 004526                JSR      PC,GETALL      ;GET REGS FOR PRINTOUT
5909                                ;REPORT IRDY NOT SET
5910                                ERRDF    17,EM17,ERR4
5911 006520                (4) 006520 104455
5912 006522                (5) 006522 000021
5913 006524                (5) 006524 012643
5914 006526                (5) 006526 015622
5915 006530                TRAP    C$ERDF
5916 006532                .WORD  17
5917 006534                (5) 006534 012643
5918 006536                (5) 006536 015622
5919 006538                .WORD  EM17
5920 006540                (5) 006540 015622
5921 006542                .WORD  ERR4
5922 006544                BR      16$              ;TAKE ERROR EXIT
5923 006546 132737 000020 002364 3$:      BITB     #IRDY,REDBYT   ;SEE IF IRDY = 0
5924 006548 001407                BEQ      9$              ;BR IF IRDY = 0
5925 006550 004737 004526                JSR      PC,GETALL      ;GET REGS FOR PRINTOUT
5926 006552                ;REPORT IRDY NOT CLEARED
5927 006554                ERRDF    18,EM18,ERR4
5928 006556                (4) 006556 104455
5929 006558                (5) 006558 000022
5930 006560                (5) 006560 012660
5931 006562                (5) 006562 015622
5932 006564                TRAP    C$ERDF
5933 006566                .WORD  18
5934 006568                (5) 006568 015622
5935 006570                .WORD  EM18
5936 006572                .WORD  ERR4
5937 006574                BR      16$              ;TAKE ERROR RETURN
5938 006576 012737 000017 002400 9$:      MOV      #17,REGNUM     ;SET REG NO. = 17
5939 006578 004737 003672                JSR      PC,READLU      ;READ REG 17
5940 006580 132776 000001 000004      BITB     #BIT0,@4(SP)   ;GET EXPECTED STATE OF ICIR
5941 006582 001413                BEQ      12$             ;BR IF EXPECTED ICIR = 0
5942 006584 132737 000010 002364      BITB     #ICIR,REDBYT   ;SEE IF ICIR = 1
5943 006586 001031                BNE      20$             ;BR IF ICIR = 1
5944 006588 004737 004526                JSR      PC,GETALL      ;GET REGS FOR PRINTOUT
5945 006590                ;REPORT ICIR NOT SET
5946 006592                ERRDF    19,EM19,ERR4
5947 006594                (4) 006594 104455
5948 006596                (5) 006596 000023
5949 006598                (5) 006600 012701
5950 006600                (5) 006602 015622
5951 006602                TRAP    C$ERDF
5952 006604                .WORD  19
5953 006606                (5) 006606 012701
5954 006608                (5) 006610 015622
5955 006610                .WORD  EM19
5956 006612                .WORD  ERR4
5957 006614                BR      16$              ;TAKE ERROR RETURN
5958 006616 132737 000010 002364 12$:     BITB     #ICIR,REDBYT   ;SEE IF ICIR = 0
5959 006618 001416                BEQ      20$             ;BR IF ICIR = 0
5960 006620 004737 004526                JSR      PC,GETALL      ;GET REGS FOR PRINTOUT
5961 006622                ;REPORT ICIR NOT CLEARED
5962 006624                ERRDF    20,EM20,ERR4
5963 006626                (4) 006626 104455
5964 006628                (5) 006628 000024
5965 006630                (5) 006632 012716
5966 006632                (5) 006634 015622
5967 006634                TRAP    C$ERDF
5968 006636                .WORD  20
5969 006638                (5) 006638 015622
5970 006640                .WORD  EM20
5971 006642                .WORD  ERR4
5972 006644                MOV      2(SP),REGNUM   ;RESTORE LU REG NO.
5973 006646 016637 000002 002400 16$:     MOV      PSTACK,SP      ;RESTORE STACK POINTER TO BASE LEVEL
5974 006648 013706 002346                MOV      RETADR,-(SP)   ;FIX ERROR RETURN PC
5975 006650 013746 002362                BR      23$
5976 006652
5977 006654
5978 006656
5979 006658
5980 006660
5981 006662
5982 006664
5983 006666
5984 006668
5985 006670
5986 006672

```

5937 006674 062766 000002 000004 20\$: ADD #2,4(SP) ;FIX UP ERROR-FREE RETURN PC  
5938 006702 012637 002352 MOV (SP)+,SUBRPC  
5939 006706 012637 002400 MOV (SP)+,REGNUM ;RESTORE LU REG NO.  
5940 006712 000207 23\$: RTS PC ;RETURN

5941  
5942  
5943  
5944  
5945  
5946

5947  
5948  
5949  
5950  
5951  
5952

\*\*\*\*\*  
;\* 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.  
\*\*\*\*\*

5953 006714 013746 002400 IACTIV: MOV REGNUM,-(SP) ;SAVE LU REG NO.  
5954 006720 013746 002352 MOV SUBRPC,-(SP)  
5955 006724 005737 002352 TST SUBRPC ;SEE IF THIS IS A NESTED CALL  
5956 006730 001006 BNE 1\$ ;BR IF YES  
5957 006732 016637 000004 002352 MOV 4(SP),SUBRPC  
5958 006740 162737 000004 002352 SUB #4,SUBRPC ;GET PC OF SUBR CALL  
5959 006746 012737 000012 002400 1\$: MOV #12,REGNUM ;SET REG NO. = 12  
5960 006754 004737 003672 JSR PC,READLU ;READ REG 12  
5961 006760 032776 000001 000004 BIT #BIT0,04(SP) ;GET EXPECTED STATE OF IACT  
5962 006766 001413 BEQ 3\$ ;BR IF EXPECTED IACT = 0  
5963 006770 132737 000100 002364 BITB #IACT,REDBYT ;SEE IF IACT = 1  
5964 006776 001031 BNE 9\$ ;BR IF IACT = 1  
5965 007000 004737 004526 JSR PC,GETALL ;GET REGS FOR PRINTOUT  
5966 ;REPORT IACT NOT SET  
5967 ERRDF 21,EM21,ERR4

(4)  
(5)  
(5)  
(5)

5968  
5969  
5970  
5971  
5972

5973  
(4)  
(5)  
(5)  
(5)

5974  
5975  
5976  
5977  
5978  
5979  
5980  
5981  
5982  
5983  
5984

5968 007014 000412 BR 6\$ ;TAKE ERROR EXIT  
5969 007016 132737 000100 002364 3\$: BITB #IACT,REDBYT ;SEE IF IACT = 0  
5970 007024 001416 BEQ 9\$ ;BR IF IACT = 0  
5971 007026 004737 004526 JSR PC,GETALL ;GET REGS FOR PRINTOUT  
5972 ;REPORT IACT NOT CLEARED  
5973 ERRDF 22,EM22,ERR4

TRAP C\$ERDF  
.WORD 21  
.WORD EM21  
.WORD ERR4

TRAP C\$ERDF  
.WORD 22  
.WORD EM22  
.WORD ERR4

5974 007042 016637 000002 002400 6\$: MOV 2(SP),REGNUM ;RESTORE LU REG NO.  
5975 007050 013706 002346 MOV PSTACK,SP ;RESTORE PROGRAM STACK TO BASE LEVEL  
5976 007054 013746 002362 MOV RETADR,-(SP) ;FIX UP ERROR RETURN PC  
5977 007060 000407 BR 12\$  
5978 007062 062766 000002 000004 9\$: ADD #2,4(SP) ;FIX UP ERROR-FREE RETURN PC  
5979 007070 012637 002352 MOV (SP)+,SUBRPC  
5980 007074 012637 002400 MOV (SP)+,REGNUM ;RESTORE LU REG NO.  
5981 007100 000207 12\$: RTS PC ;RETURN



5985  
 5986  
 5987  
 5988  
 5989  
 5990  
 5991  
 5992  
 5993  
 5994  
 5995  
 5996  
 5997  
 5998  
 5999  
 6000  
 6001  
 6002  
 6003  
 6004  
 6005  
 6006  
 6007  
 (4)  
 (5)  
 (5)  
 (5)  
 6008  
 6009  
 6010  
 6011  
 6012  
 6013  
 (4)  
 (5)  
 (5)  
 (5)  
 6014  
 6015  
 6016  
 6017  
 6018  
 6019  
 6020  
 6021  
 (4)  
 (5)  
 (5)  
 (5)  
 6022  
 6023  
 6024  
 6025  
 6026  
 6027  
 (4)

007102 013746 002402  
 007106 013746 002352  
 007112 005737 002352  
 007116 001006  
 007120 016637 000004 002352  
 007126 162737 000004 002352  
 007134 012737 000001 002402  
 007142 004737 004124  
 007146 032776 000001 000004  
 007154 001413  
 007156 132737 000001 002372  
 007164 001022  
 007166 004737 004526  
 007172  
 007172 104455  
 007174 000035  
 007176 013173  
 007200 017012  
 007202 000444  
 007204 132737 000001 002372  
 007212 001407  
 007214 004737 004526  
 007220  
 007220 104455  
 007222 000034  
 007224 013152  
 007226 017012  
 007230 000431  
 007232 132776 000002 000004  
 007240 001413  
 007242 132737 000002 002372  
 007250 001031  
 007252 004737 004526  
 007256  
 007256 104455  
 007260 000037  
 007262 013231  
 007264 017012  
 007266 000412  
 007270 132737 000002 002372  
 007276 001416  
 007300 004737 004526  
 007304  
 007304 104455

```

*****
;* 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
;* 0,1, RESPECTIVELY, OF THE WORD FOLLOWING THE CALL.
;* IF AN ERROR OCCURS, A RETURN IS MADE TO THE TEST AT THE ADDRESS IN RETADR.
*****
RSEOM: MOV     AXNUM,-(SP)      ;SAVE AX BYTE 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$:     MOV     #1,AXNUM      ;SET AX BYTE NO. FOR AX0-16
        JSR     PC,READAX    ;READ AX0
        BIT     #BIT0,@4(SP) ;GET EXPECTED STATE OF RSOM
        BEQ     3$           ;BR IF EXPECTED RSOM = 0
        BITB   #RSOM,RAX16   ;SEE IF RSOM = 1
        BNE     9$           ;BR IF RSOM = 1
        JSR     PC,GETALL    ;GET REGS FOR PRINTOUT
        ;REPORT RSOM NOT SET
        ERRDF  29,EM29,ERR6

        BR     16$          ;TAKE ERROR EXIT
        BITB   #RSOM,RAX16  ;SEE IF RSOM = 0
        BEQ     9$           ;BR IF RSOM = 0
        JSR     PC,GETALL    ;GET REGS FOR PRINTOUT
        ;REPORT RSOM NOT CLEARED
        ERRDF  28,EM28,ERR6

        BR     16$          ;TAKE ERROR RETURN
        BITB   #BIT1,@4(SP) ;GET EXPECTED STATE OF REOM
        BEQ     12$          ;BR IF EXPECTED REOM = 0
        BITB   #REOM,RAX16  ;SEE IF REOM = 1
        BNE     20$          ;BR IF REOM = 1
        JSR     PC,GETALL    ;GET REGS FOR PRINTOUT
        ;REPORT REOM NOT SET
        ERRDF  31,EM31,ERR6

        BR     16$          ;TAKE ERROR RETURN
        BITB   #REOM,RAX16  ;SEE IF REOM = 0
        BEQ     20$          ;BR IF REOM = 0
        JSR     PC,GETALL    ;GET REGS FOR PRINTOUT
        ;REPORT REOM NOT CLEARED
        ERRDF  30,EM30,ERR6
  
```

TRAP C\$ERDF  
 .WORD 29  
 .WORD EM29  
 .WORD ERR6  
 TRAP C\$ERDF  
 .WORD 28  
 .WORD EM28  
 .WORD ERR6  
 TRAP C\$ERDF  
 .WORD 31  
 .WORD EM31  
 .WORD ERR6  
 TRAP C\$ERDF

(5)	007306	000036							.WORD	30
(5)	007310	013210							.WORD	EM30
(5)	007312	017012							.WORD	ERR6
6028	007314	016637	000002	002402	16\$:	MOV	2(SP),AXNUM	:	RESTORE AX BYTE NO.	
6029	007322	013706	002346			MOV	PSTACK,SP	:	RESTORE STACK POINTER TO BASE LEVEL	
6030	007326	013746	002362			MOV	RETADR,-(SP)	:	FIX ERROR RETURN PC	
6031	007332	000407				BR	23\$			
6032	007334	062766	000002	000004	20\$:	ADD	#2,4(SP)	:	FIX UP ERROR-FREE RETURN PC	
6033	007342	012637	002352			MOV	(SP)+,SUBRPC			
6034	007346	012637	002402			MOV	(SP)+,AXNUM	:	RESTORE AX BYTE NO.	
6035	007352	000207			23\$:	RTS	PC	:	RETURN	

6036  
 6037  
 6038  
 6039  
 6040  
 6041

```

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

6045	007354	013746	002400			RDRXSI: MOV	REGNUM,-(SP)	:	SAVE LU REG NO.	
6046	007360	012737	000012	002400		MOV	#12,REGNUM	:	SET REG NO. = 12	
6047	007366	004737	003672			JSR	PC,READLU	:	READ LU REG 12	
6048	007372	113737	002364	002425		MOVB	REDBYT,RXWORD+1	:	GET HI BITS OF SILO ENTRY	
6049	007400	042737	170000	002424		BIC	#170000,RXWORD	:	CLEAR UNUSED BITS	
6050	007406	012737	000010	002400		MOV	#10,REGNUM	:	SET REG NO. = 10	
6051	007414	004737	003672			JSR	PC,READLU	:	READ REG 10	
6052	007420	113737	002364	002424		MOVB	REDBYT,RXWORD	:	GET LOW BITS OF SILO ENTRY	
6053	007426	012637	002400			MOV	(SP)+,REGNUM	:	RESTORE LU REG NO.	
6054	007432	000207				RTS	PC	:	RETURN	

6055  
 6056  
 6057  
 6058  
 6059  
 6060

```

:*****
:* 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.
:*****
  
```

6069	007434	010146				RCV1ST: MOV	R1,-(SP)	:	SAVE R1	
6070	007436	010346				MOV	R3,-(SP)	:	SAVE R3	
6071	007440	013746	002400			MOV	REGNUM,-(SP)	:	SAVE LU REG NO.	
6072	007444	016637	000006	002352		MOV	6(SP),SUBRPC			
6073	007452	162737	000004	002352		SUB	#4,SUBRPC	:	GET PC OF SUBROUTINE CALL	
6074	007460	012737	000012	002400		MOV	#12,REGNUM	:	SET LU REG NO. = 12	
6075	007466	005001				CLR	R1	:	INIT CYCLE COUNT TO 0	
6076	007470	017603	000006			MOV	@6(SP),R3	:	GET CYCLE COUNT LIMIT	
6077	007474	062703	000003			ADD	#3,R3			
6078	007500	005776	000006			TST	@6(SP)	:	SEE IF DESIRED CYCLES = 0	
6079	007504	001414				BEQ	8\$	:	BR IF YES	
6080	007506	004737	006714			JSR	PC,IACTIV	:	CHK FOR IACT = 0	

6081	007512	000000		0			
6082	007514	004737	006430	JSR	PC,ISIRDY	:CHK FOR ICIR = 1, IRDY = 0	
6083	007520	000001		1			
6084	007522	004737	007102	JSR	PC,RSEOM	:CHK RSOM = 0, REOM = 0 IN AX0-16	
6085	007526	000000		0			
6086	007530	004737	005254	6\$: JSR	PC,STPLU	:CLOCK LU FOR 1 CYCLE	
6087	007534	000001		1			
6088	007536	004737	005146	8\$: JSR	PC,WAIT50	:ALLOW SILO DATA TO RIPPLE	
6089	007542	005201		INC	R1	:INCREMENT CYCLE COUNT	
6090	007544	004737	003672	JSR	PC,READLU	:READ REG 12	
6091	007550	132737	000020	002364	BITB	#IRDY,REDBYT	:SEE IF IRDY = 1 YET
6092	007556	001005		BNE	9\$	:BR IF IRDY = 1	
6093	007560	020103		CMP	R1,R3	:SEE IF LIMIT EXCEEDED	
6094	007562	002762		BLT	6\$	:BR IF NOT YET	
6095	007564	004737	006430	JSR	PC,ISIRDY	:CHK FOR ICIR = 1, IRDY = 1	
6096	007570	000003		3			
6097	007572	020176	000006	9\$: CMP	R1,@6(SP)	:SEE IF LESS THAN REQUIRED CYCLES	
6098	007576	002003		BGE	12\$	:BR IF NOT	
6099	007600	004737	006430	JSR	PC,ISIRDY	:CHK FOR ICIR = 1, IRDY = 0	
6100	007604	000001		1			
6101	007606	004737	006714	12\$: JSR	PC,IACTIV	:CHK FOR IACT = 1	
6102	007612	000001		1			
6103	007614	004737	006430	JSR	PC,ISIRDY	:CHK FOR ICIR = 1, IRDY = 1	
6104	007620	000003		3			
6105	007622	062766	000002	000006	ADD	#2,6(SP)	:FIX UP RETURN PC
6106	007630	012637	002400	MOV	(SP)+,REGNUM	:RESTORE LU REG NO.	
6107	007634	012603		MOV	(SP)+,R3	:RESTORE R3	
6108	007636	012601		MOV	(SP)+,R1	:RESTORE R1	
6109	007640	005037	002352	CLR	SUBRPC	:CLEAR SUBR CALL PC	
6110	007644	000207		RTS	PC	:RETURN	

6111  
 6112  
 6113  
 6114  
 6115  
 6116  
 6117  
 6118  
 6119  
 6120  
 6121  
 6122  
 6123  
 6124  
 6125  
 6126  
 6127  
 6128  
 6129  
 6130  
 6131  
 6132  
 6133  
 6134  
 6135  
 6136

```

:*****
:* 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.
:*****

```

```

STPERR: MOV    REGNUM,-(SP)    :SAVE LU REG NO.
        MOV    #17,REGNUM    :SET LU REG NO. = 17
        MOV    @2(SP),WRIBYT
        BISB   #IERR,WRIBYT
        JSR    PC,WRITLU     :SET IERR BIT IN REG 17
        ADD    #2,2(SP)      :INCREMENT SUBR ARGUMENT POINTER
        MOV    @2(SP),3$     :GET DESIRED NO. OF CYCLES
        JSR    PC,STPLU     :CLOCK LU FOR DESIRED NO. OF CYCLES
        .WORD  0             :NO. OF CYCLES GOES HERE
3$:     BICB   #IERR,WRIBYT
        JSR    PC,WRITLU     :CLEAR IERR BIT IN REG 17
        ADD    #2,2(SP)      :FIX UP RETURN PC
        MOV    (SP)+,REGNUM  :RESTORE LU REG NO.
        RTS    PC           :RETURN

```

6137  
 6138  
 6139  
 6140  
 6141  
 6142  
 6143  
 6144  
 6145  
 6146  
 6147  
 6148  
 6149  
 6150  
 6151  
 6152  
 6153  
 6154  
 6155  
 6156  
 6157  
 6158  
 6159  
 6160  
 6161  
 6162  
 6163  
 6164  
 6165  
 6166  
 6167  
 6168  
 6169  
 6170  
 6171  
 6172  
 6173  
 (4)  
 (5)  
 (5)  
 (5)  
 6174  
 6175  
 6176  
 6177  
 6178  
 (4)  
 (5)  
 (5)  
 (5)  
 6179  
 6180  
 6181  
 6182  
 6183  
 6184

007750 010146  
 007752 013746 002400  
 007756 016637 000004 002352  
 007764 162737 000004 002352  
 007772 017601 000004  
 007776 042701 170000  
 010002 004737 007354  
 010006 023727 002432 000347  
 010014 001005  
 010016 042701 000200  
 010022 042737 000200 002424  
 010030 120137 002424  
 010034 001445  
 010036 005037 002404  
 010042 110137 002404  
 010046 005037 002406  
 010052 113737 002424 002406  
 010060 012737 000011 002400  
 010066 004737 003672  
 010072 132737 000001 002364  
 010100 001410  
 010102 004737 004526  
 010106  
 010106 104455  
 010110 000066  
 010112 014206  
 010114 015622  
 010116 000137 010600  
 010122 012737 000010 002400  
 010130 004737 004526  
 010134  
 010134 104455  
 010136 000042  
 010140 013320  
 010142 020122  
 010144 000137 010600  
 010150 000301  
 010152 012737 000012 002400  
 010160 120137 002425  
 010164 001002  
 010166 000137 010554

```

*****
* CKDATA - THIS SUBROUTINE READS THE RCV SILO AND COMPARES THE SILO ENTRY
* TO BITS 0-11 OF THE FIRST WORD FOLLOWING THE CALL. IF THERE IS A
* MISMATCH, THE ERROR IS REPORTED AND A RETURN IS MADE TO THE TEST AT THE
* ADDRESS CONTAINED IN RETADR. IF BIT 15 = 0 IN THE FIRST WORD
* FOLLOWING THE CALL, THE SUBROUTINE WILL NOT CHECK THE BCC BIT (SILO
* BIT 8). IF THERE ARE NO ERRORS, THE LINE UNIT IS CLOCKED FOR THE
* NUMBER OF CYCLES PASSED IN THE SECOND WORD FOLLOWING THE CALL.
*****
CKDATA: MOV R1, -(SP) ;SAVE R1
MOV REGNUM, -(SP) ;SAVE LU REG NO.
MOV 4(SP), SUBRPC
SUB #4, SUBRPC ;GET PC OF SUBR CALL
MOV @4(SP), R1 ;GET EXPECTED SILO ENTRY
BIC #170000, R1 ;CLEAR UNUSED BITS FOR COMPARE
JSR PC, RDRXSI ;READ RCV SILO
CMP SAVLEN, #TXLEN2!TXLEN1!TXLENO!RXLEN2!RXLEN1!RXLENO
BNE 4$ ;BR IF CHAR LENGTH NOT = 7
BIC #BIT7, R1 ;MASK OFF BIT 8TH BIT
BIC #BIT7, RXWORD
4$: CMPB R1, RXWORD ;COMPARE EXPECTED BITS 0-7 TO ACTUAL
BEQ 6$ ;BR IF MATCH
CLR GOODAT
MOVB R1, GOODAT ;GET EXPECTED DATA
CLR BADDAT
MOVB RXWORD, BADDAT ;GET ACTUAL DATA
MOV #11, REGNUM ;SET REG NO. = 11
JSR PC, READLU ;READ REG 11
BITB #UNRR, REDBYT ;SEE IF TX UNDERRUN ERROR
BEQ 5$ ;BR IF NOT
JSR PC, GETALL ;GET REGS FOR PRINTOUT
;REPORT TX UNDERRUN ERROR
ERRDF 54, EM54, ERR4
TRAP C$ERDF
.WORD 54
.WORD EM54
.WORD ERR4

JMP 36$ ;TAKE ERROR EXIT
5$: MOV #10, REGNUM ;SET REG NO. = 10
JSR PC, GETALL ;GET REGS FOR PRINTOUT
;REPORT RCV'D DATA MISCOMPARE
ERRDF 34, EM34, ERR8
TRAP C$ERDF
.WORD 34
.WORD EM34
.WORD ERR8

JMP 36$ ;TAKE ERROR EXIT
6$: SWAB R1
MOV #12, REGNUM ;SET LU REG NO. FOR ERROR REPORTS
CMPB R1, RXWORD+1 ;COMPARE EXPECTED SILO BITS 8-11 TO ACTUAL
BNE 7$ ;BR IF MISMATCH
JMP 22$ ;CONTINUE

```

```

6185 010172 005037 002404      7$: CLR GOODAT
6186 010176 110137 002404      MOV#B R1,GOODAT ;SET EXPECTED DATA
6187 010202 005037 002406      CLR BADDAT
6188 010206 113737 002425 002406 MOV#B RXWORD+1,BADDAT ;SET ACTUAL DATA
6189 010214 032776 100000 000004 BIT #BCCCHK,@4(SP) ;SEE IF BCC SHOULD BE IGNORED
6190 010222 001433 BEQ 10$ ;BR IF YES
6191 010224 132701 000001 BIT#B #BCC,R1 ;SEE IF EXPECTED BIT = 1
6192 010230 001014 BNE 8$ ;BR IF YES
6193 010232 132737 000001 002425 BIT#B #BCC,RXWORD+1 ;SEE IF ACTUAL BIT = 0
6194 010240 001424 BEQ 10$ ;BR IF YES
6195 010242 004737 004526 JSR PC,GETALL ;GET REGS FOR PRINTOUT
6196 ;REPORT BCC NOT CLEARED
6197 ERRDF 35,EM35,ERR8
(4) 010246 104455 TRAP C$ERDF
(5) 010250 000043 .WORD 35
(5) 010252 013346 .WORD EM35
(5) 010254 020122 .WORD ERR8
6198 010256 000137 010600 JMP 36$ ;TAKE ERROR EXIT
6199 010262 132737 000001 002425 8$: BIT#B #BCC,RXWORD+1 ;SEE IF ACTUAL BIT = 1
6200 010270 001010 BNE 10$ ;BR IF YES
6201 010272 004737 004526 JSR PC,GETALL ;GET REGS FOR PRINTOUT
6202 ;REPORT BCC NOT SET
6203 ERRDF 36,EM36,ERR8
(4) 010276 104455 TRAP C$ERDF
(5) 010300 000044 .WORD 36
(5) 010302 013366 .WORD EM36
(5) 010304 020122 .WORD ERR8
6204 010306 000137 010600 JMP 36$ ;TAKE ERROR EXIT
6205 010312 10$: BIT#B #EBLK,R1 ;SEE IF EXPECTED BIT = 1
6206 010312 132701 000002 BNE 12$ ;BR IF YES
6207 010316 001014 BIT#B #EBLK,RXWORD+1 ;SEE IF ACTUAL BIT = 0
6208 010320 132737 000002 002425 BEQ 14$ ;BR IF YES
6209 010326 001424 JSR PC,GETALL ;GET REGS FOR PRINTOUT
6210 010330 004737 004526 ;REPORT EBLK NOT CLEARED
6211 ERRDF 37,EM37,ERR8
(4) 010334 104455 TRAP C$ERDF
(5) 010336 000045 .WORD 37
(5) 010340 013402 .WORD EM37
(5) 010342 020122 .WORD ERR8
6213 010344 000137 010600 JMP 36$ ;TAKE ERROR EXIT
6214 010350 132737 000002 002425 12$: BIT#B #EBLK,RXWORD+1 ;SEE IF ACTUAL BIT = 1
6215 010356 001010 BNE 14$ ;BR IF YES
6216 010360 004737 004526 JSR PC,GETALL ;GET REGS FOR PRINTOUT
6217 ;REPORT EBLK NOT SET
6218 ERRDF 38,EM38,ERR8
(4) 010364 104455 TRAP C$ERDF
(5) 010366 000046 .WORD 38
(5) 010370 013423 .WORD EM38
(5) 010372 020122 .WORD ERR8
6219 010374 000137 010600 JMP 36$ ;TAKE ERROR EXIT
6220 010400 14$: BIT#B #RAB,R1 ;SEE IF EXPECTED BIT = 1
6221 010400 132701 000004 BNE 16$ ;BR IF YES
6222 010404 001014 BIT#B #RAB,RXWORD+1 ;SEE IF ACTUAL BIT = 0
6223 010406 132737 000004 002425 BEQ 18$ ;BR IF YES
6224 010414 001424
  
```

6225	010416	004737	004526			JSR	PC,GETALL	:GET REGS FOR PRINTOUT		
6226						:REPORT	RAB NOT CLEARED			
6227	010422					ERRDF	39,EM39,ERR8			
(4)	010422	104455							TRAP	C\$ERDF
(5)	010424	000047							.WORD	39
(5)	010426	013440							.WORD	EM39
(5)	010430	020122							.WORD	ERR8
6228	010432	000137	010600			JMP	36\$	:TAKE ERROR EXIT		
6229	010436	132737	000004	002425	15\$:	BITB	#RAB,RXWORD+1	:SEE IF ACTUAL BIT = 1		
6230	010444	001010				BNE	18\$	:BR IF YES		
6231	010446	004737	004526			JSR	PC,GETALL	:GET REGS FOR PRINTOUT		
6232						:REPORT	RAB NOT SET			
6233	010452					ERRDF	40,EM40,ERR8			
(4)	010452	104455							TRAP	C\$ERDF
(5)	010454	000050							.WORD	40
(5)	010456	013460							.WORD	EM40
(5)	010460	020122							.WORD	ERR8
6234	010462	000137	010600			JMP	36\$	:TAKE ERROR EXIT		
6235	010466				18\$:					
6236	010466	132701	000010			BITB	#OVRR,R1	:SEE IF EXPECTED BIT = 1		
6237	010472	001014				BNE	20\$	:BR IF YES		
6238	010474	132737	000010	002425		BITB	#OVRR,RXWORD+1	:SEE IF ACTUAL BIT = 0		
6239	010502	001424				BEQ	22\$	:BR IF YES		
6240	010504	004737	004526			JSR	PC,GETALL	:GET REGS FOR PRINTOUT		
6241						:REPORT	OVRR NOT CLEARED			
6242	010510					ERRDF	41,EM41,ERR8			
(4)	010510	104455							TRAP	C\$ERDF
(5)	010512	000051							.WORD	41
(5)	010514	013474							.WORD	EM41
(5)	010516	020122							.WORD	ERR8
6243	010520	000137	010600			JMP	36\$	:TAKE ERROR EXIT		
6244	010524	132737	000010	002425	20\$:	BITB	#OVRR,RXWORD+1	:SEE IF ACTUAL BIT = 1		
6245	010532	001010				BNE	22\$	:BR IF YES		
6246	010534	004737	004526			JSR	PC,GETALL	:GET REGS FOR PRINTOUT		
6247						:REPORT	OVRR NOT SET			
6248	010540					ERRDF	42,EM42,ERR8			
(4)	010540	104455							TRAP	C\$ERDF
(5)	010542	000052							.WORD	42
(5)	010544	013515							.WORD	EM42
(5)	010546	020122							.WORD	ERR8
6249	010550	000137	010600			JMP	36\$	:TAKE ERROR EXIT		
6250	010554				22\$:					
6251	010554	062766	000002	000004		ADD	#2,4(SP)	:INCR SUBROUTINE ARGUMENT POINTER		
6252	010562	017637	000004	010574		MOV	@4(SP),24\$	:GET DESIRED CYCLE COUNT		
6253	010570	004737	005254			JSR	PC,STPLU	:CLOCK LU FOR DESIRED CYCLES		
6254	010574	000000			24\$:	.WORD	0			
6255	010576	000407				BR	38\$	:TAKE ERROR-FREE EXIT		
6256	010600	011637	002400		36\$:	MOV	(SP),REGNUM	:RESTORE LU REG NO.		
6257	010604	013706	002346			MOV	PSTACK,SP	:RESTORE PROGRAM STACK TO BASE LEVEL		
6258	010610	013746	002362			MOV	RETADR,-(SP)	:FIX UP ERROR RETURN PC		
6259	010614	000406				BR	40\$			
6260	010616	062766	000002	000004	38\$:	ADD	#2,4(SP)	:FIX UP ERROR-FREE RETURN PC		
6261	010624	012637	002400			MOV	(SP)+,REGNUM	:RESTORE LU REG NO.		
6262	010630	012601				MOV	(SP)+,R1	:RESTORE R1		
6263	010632	005037	002352		40\$:	CLR	SUBRPC	:CLEAR SUBROUTINE PC		
6264	010636	000207				RTS	PC	:RETURN		

6265  
 6266  
 6267  
 6268  
 6269  
 6270  
 6271  
 6272  
 6273  
 6274  
 6275 010640 013746 002402  
 6276 010644 013746 002400  
 6277 010650 012737 000004 002402  
 6278 010656 017637 000004 002374  
 6279 010664 005037 002376  
 6280 010670 004737 004312  
 6281 010674 012737 000017 002400  
 6282 010702 062766 000002 000004  
 6283 010710 017637 000004 002366  
 6284 010716 004737 003750  
 6285 010722 012737 000006 002402  
 6286 010730 062766 000002 000004  
 6287 010736 017637 000004 002374  
 6288 010744 062766 000002 000004  
 6289 010752 017637 000004 002376  
 6290 010760 013737 002376 002432  
 6291 010766 004737 004312  
 6292 010772 062766 000002 000004  
 6293 011000 012637 002400  
 6294 011004 012637 002402  
 6295 011010 005037 002352  
 6296 011014 000207  
 6297  
 6298  
 6299  
 6300  
 6301  
 6302  
 6303  
 6304  
 6305  
 6306  
 6307 011016 010146  
 6308 011020 010246  
 6309 011022 017601 000004  
 6310 011026 062766 000002 000004  
 6311 011034 017602 000004  
 6312 011040 062766 000002 000004  
 6313 011046 012137 002422  
 6314 011052 004737 005174  
 6315 011056 005302  
 6316 011060 001372  
 6317 011062 004737 005146  
 6318 011066 012602  
 6319 011070 012601  
 6320 011072 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
      MOV WAX16,SAVLEN ;STORE TX AND RCV CHAR LENGTH
      JSR PC,WRITAX ;LOAD AX3-15, AX3-16
      ADD #2,4(SP) ;FIX RETURN PC
      MOV (SP)+,REGNUM ;RESTORE LU REG NO.
      MOV (SP)+,AXNUM ;RESTORE AX BYTE NO.
      CLR SUBRPC ;CLEAR SUBROUTINE PC STORAGE
      RTS PC ;RETURN
  
```

```

:*****
:* 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$: MOV (R1)+,TXWORD ;GET NEXT MSG WORD
    JSR PC,LDTXSI ;LOAD A WORD INTO TX SILO
    DEC R2 ;DECR COUNT
    BNE 6$ ;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
  
```

6321  
 6322  
 6323  
 6324  
 6325  
 6326  
 6327  
 6328  
 6329  
 6330  
 6331  
 6332  
 6333  
 6334  
 6335  
 6336  
 6337  
 6338  
 6339  
 6340  
 6341  
 6342  
 6343  
 6344  
 6345  
 6346  
 6347  
 6348  
 6349  
 6350  
 6351  
 6352  
 6353  
 6354  
 6355  
 6356  
 6357  
 6358

011074 010146  
 011076 016637 000002 002352  
 011104 162737 000004 002352  
 011112 004737 007354  
 011116 004737 005146  
 011122 005001  
 011124 020176 000002  
 011130 001410  
 011132 004737 006430  
 011136 000001  
 011140 004737 005254  
 011144 000001  
 011146 005201  
 011150 000765  
 011152 004737 006430  
 011156 000003  
 011160 062766 000002 000002  
 011166 005037 002352  
 011172 012601  
 011174 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
        JSR PC,STPLU ;CLK LU 1 CYCLE
        INC R1 ;INCR CYCLE COUNT
        BR 9$
12$:   JSR PC,ISIRDY ;CHK ICIR = 1 IRDY = 1
        ADD #2,2(SP) ;FIX RETURN PC
        CLR SUBRPC ;CLEAR SUBR CALL PC
        MOV (SP)+,R1 ;RESTORE R1
        RTS PC ;RETURN
  
```

6359  
 6360  
 6361  
 6362  
 6363  
 6364  
 6365  
 6366  
 6367  
 6368  
 6369  
 6370  
 6371  
 6372  
 6373  
 6374  
 6375  
 6376

011176 013746 002400  
 011202 010146  
 011204 016637 000004 002352  
 011212 162737 000004 002352  
 011220 012701 000001  
 011224 012737 000017 002400  
 011232 004737 005254  
 011236 000001  
 011240 004737 003672  
 011244 030176 000004  
 011250 001013  
 011252 132737 000040 002364

```

*****
* 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$:     JSR PC,STPLU ;CLOCK LINE UNIT 1 CYCLE
        JSR PC,READLU ;READ REG 17
        BIT R1,@4(SP) ;SEE IF EXPECTED BIT = 1
        BNE 9$ ;BR IF YES
        BITB #TXDATA,REDBYT ;SEE IF ACTUAL BIT = 0
  
```



```

6377 011260 001422          BEQ      12$          ;BR IF YES
6378 011262 004737 004526   JSR      PC,GETALL    ;GET REGS FOR PRINTOUT
6379          ;REPORT TXDATA NOT CLEARED
6380 011266          ERRDF  32,EM32,ERR4
(4) 011266 104455          TRAP    C$ERDF
(5) 011270 000040          .WORD  32
(5) 011272 013246          .WORD  EM32
(5) 011274 015622          .WORD  ERR4
6381 011276 000420          BR       20$          ;TAKE ERROR RETURN
6382 011300 132737 000040 002364 9$: BITB    #TXDATA,REDBYT ;SEE IF ACTUAL BIT = 1
6383 011306 001007          BNE     12$          ;BR IF YES
6384 011310 004737 004526   JSR      PC,GETALL    ;GET REGS FOR PRINTOUT
6385          ;REPORT TXDATA BIT NOT SET
6386 011314          ERRDF  33,EM33,ERR4
(4) 011314 104455          TRAP    C$ERDF
(5) 011316 000041          .WORD  33
(5) 011320 013275          .WORD  EM33
(5) 011322 015622          .WORD  ERR4
6387 011324 000405          BR       20$          ;TAKE ERROR EXIT
6388 011326 006301          12$:  ASL     R1          ;SHIFT BIT POINTER
6389 011330 020127 000400   CMP     R1,#400       ;SEE IF 8 BITS SCANNED YET
6390 011334 001336          BNE     4$            ;BR IF NO
6391 011336 000405          BR       22$
6392 011340 013706 002346   20$:  MOV     PSTACK,SP  ;RESTORE PROGRAM STACK POINTER TO BASE LEVEL
6393 011344 013746 002362   MOV     RETADR,-(SP)  ;FIX UP RETURN PC
6394 011350 000406          BR       40$
6395 011352 062766 000002 000004 22$:  ADD     #2,4(SP)      ;FIX UP ERROR-FREE RETURN PC
6396 011360 012601          MOV     (SP)+,R1      ;RESTORE R1
6397 011362 012637 002400   MOV     (SP)+,REGNUM  ;RESTORE REG NO.
6398 011366 005037 002352   40$:  CLR     SUBRPC      ;CLEAR SUBR CALL PC
6399 011372 000207          RTS     PC            ;RETURN

```

6400  
6401  
6402  
6403  
6404  
6405

```

:*****
;* 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.
:*****

```

```

6410 011374 010146          LDMSG1: MOV    R1,-(SP)    ;SAVE R1
6411 011376 010246          MOV    R2,-(SP)    ;SAVE R2
6412 011400 004737 011016   JSR    PC,LODMSG   ;LOAD MSG1 INTO TX SILO
6413 011404 003220          MSG1
6414 011406 000011          9.
6415 011410 012701 003224   MOV    #MSG1+4,R1   ;GET POINTER TO MSG1
6416 011414 012702 003262   MOV    #RCVBUF,R2  ;GET POINTER TO MSG BUF
6417 011420 012122          3$:  MOV    (R1)+,(R2)+ ;LOAD A CHAR INTO MSG-BUF
6418 011422 020127 003236   CMP    R1,#MSG1+14. ;SEE IF DID LAST DATA CHAR YET
6419 011426 103774          BLO    3$          ;BR IF NOT
6420 011430 052762 100400 177776 BIS    #CRCCHK!RXBCC,-2(R2) ;SET EXPECTED BCC
6421 011436 012726 000160   MOV    #160,(SP)+  ;LOAD HI CRC BYTE
6422 011442 012726 000034   MOV    #034,(SP)+  ;LOAD LO CRC BYTE
6423 011446 012602          MOV    (SP)+,R2    ;RESTORE R2
6424 011450 012601          MOV    (SP)+,R1    ;RESTORE R1

```

CZDMRE M8203 STATIC DIAG #1  
CZDMRE.P11 08-JUN-81 13:27

MACY11 30A(1052) 15-JUN-81  
GLOBAL SUBROUTINES

L 7  
15:34 PAGE 7-49

SEQ 0089

6425 011452 000207  
6426  
6427  
6428  
6429  
6430

RTS PC ;RETURN

.SBTTL GLOBAL ERROR REPORT SECTION

:/ THE GLOBAL ERROR REPORT SECTION CONTAINS ERROR MESSAGES  
:/ THAT ARE USED IN MORE THAN ONE TEST.  
:////

6432  
6433  
6434  
6435  
6436  
6437  
6438  
6439

6440 011454 052045 047445 022466

FMT1: .ASCIZ /%T%06%N/

011462 000116

6441 011464 047045 040445 040506

FMT2: .ASCIZ /%N%AFAILING REG: /

011472 046111 047111 020107

011500 042522 035107 000040

6442 011506 040445 054105 042520

FMT3: .ASCIZ /%AEXPECTED: %03%S5%AACTUAL: %03%N/

011514 052103 042105 020072

011522 047445 022463 032523

011530 040445 041501 052524

011536 046101 020072 047445

011544 022463 000116

6443 011550 047045 052045 047045

FMT4: .ASCIZ /%N%T%N%T%N/

011556 052045 047045 000

6444 011563 045 031517 051445

FMT5: .ASCIZ /%03%S5%03%S5%03%S5%03%N/

011570 022465 031517 051445

011576 022465 031517 051445

011604 022465 031517 047045

011612 000

6445 011613 045 032123 047445

FMT6: .ASCIZ /%S4%03%S5%03%S5%03%S5%03%N/

011620 022463 032523 047445

011626 022463 032523 047445

011634 022463 032523 047445

011642 022463 000116

6446 011646 052045 047445 022462

FMT7: .ASCIZ /%T%02%N/

011654 000116

6447 011656 040445 054105 042524

FMT8: .ASCIZ /%AEXTENDED REG AX%01%A-%T%N/

011664 042116 042105 051040

011672 043505 040440 022530

011700 030517 040445 022455

011706 022524 000116

6448 011712 052045 047045 000

FMT9: .ASCIZ /%T%N/

6449 011717 045 050101 020103

FMT10: .ASCIZ /%APC OF SUBR CALL: %06%N/

011724 043117 051440 041125

011732 020122 040503 046114

011740 020072 047445 022466

011746 000116

6450 011750 040445 042522 020107

FMT11: .ASCIZ /%AREG %02%A LOADED WITH: %03%N/

011756 047445 022462 020101

011764 047514 042101 042105

011772 053440 052111 035110

012000 022440 031517 047045

012006 000

6451 012007 045 022516 052101

FMT19: .ASCIZ /%N%ATEST %D2%A NOT RUN%N/

012014 051505 020124 042045

012022 022462 020101 047516

012030 020124 052522 022516

012036 000116

6452 012040 047045 040445 046120

FMT24: .ASCIZ /%N%PLEASE INSURE M8207 RUN SWITCH (E28 SW7) IS ON%N/

CZDMRE M8203 STATIC DIAG #1  
CZDMRE.P11 08-JUN-81 13:27

MACY11 30A(1052) 15-JUN-81 15:34 PAGE 7-51  
GLOBAL ERROR REPORT SECTION

SEQ 0091

012046	040505	042523	044440
012054	051516	051125	020105
012062	034115	030062	020067
012070	052522	020116	053523
012076	052111	044103	024040
012104	031105	020070	053523
012112	024467	044440	020123
012120	047117	047045	000

6453  
6454  
6455  
6456

012125	103	051123	040440
012132	042104	042522	051523
012140	052040	046511	026505
012146	052517	020124	051450
012154	046105	024460	000
012161	122	043505	047040
012166	052117	044440	044516
012174	044524	046101	055111
012202	042105	041040	020131
012210	051515	020124	046103
012216	000122		

EM1: .ASCIZ /CSR ADDRESS TIME-OUT (SELO)/

6457

012161	122	043505	047040
012166	052117	044440	044516
012174	044524	046101	055111
012202	042105	041040	020131
012210	051515	020124	046103
012216	000122		

EM2: .ASCIZ /REG NOT INITIALIZED BY MST CLR/

6458

012220	042522	020107	044515
012226	041523	046517	040520
012234	042522	000	

EM3: .ASCIZ /REG MISCOMPARE/

6459

012237	122	043505	047040
012244	052117	044440	044516
012252	044524	046101	055111
012260	042105	041040	020131
012266	047125	041111	051525
012274	051040	051505	052105
012302	024040	047111	052111
012310	000051		

EM4: .ASCIZ /REG NOT INITIALIZED BY UNIBUS RESET (INIT)/

6460

012312	040515	047111	020124
012320	046103	020113	044502
012326	020124	052123	041525
012334	020113	052101	030040
012342	000		

EM5: .ASCIZ /MAINT CLK BIT STUCK AT 0/

6461

012343	115	044501	052116
012350	041440	045514	041040
012356	052111	051440	052524
012364	045503	040440	020124
012372	000061		

EM6: .ASCIZ /MAINT CLK BIT STUCK AT 1/

6462

012374	051117	054504	047040
012402	052117	051440	052105
012410	000		

EM7: .ASCIZ /ORDY NOT SET/

6463

012411	117	042122	020131
012416	047516	020124	046103
012424	040505	042522	000104

EM8: .ASCIZ /ORDY NOT CLEARED/

6464

012432	041517	051117	047040
012440	052117	051440	052105
012446	000		

EM9: .ASCIZ /OCOR NOT SET/

6465

012447	117	047503	020122
012454	047516	020124	046103
012462	040505	042522	000104

EM10: .ASCIZ /OCOR NOT CLEARED/

6466

012470	040517	052103	047040
--------	--------	--------	--------

EM11: .ASCIZ /OACT NOT SET/

	012476	052117	051440	052105		
	012504	000				
6467	012505	117	041501	020124	EM12:	.ASCIZ /OACT NOT CLEARED/
	012512	047516	020124	046103		
	012520	040505	042522	000104		
6468	012526	047125	051122	047040	EM13:	.ASCIZ /UNRR NOT CLEARED BY SOM/
	012534	052117	041440	042514		
	012542	051101	042105	041040		
	012550	020131	047523	000115		
6469	012556	047125	051122	047040	EM14:	.ASCIZ /UNRR NOT SET/
	012564	052117	051440	052105		
	012572	000				
6470	012573	125	051116	020122	EM15:	.ASCIZ /UNRR NOT CLEARED BY OC/
	012600	047516	020124	046103		
	012606	040505	042522	020104		
	012614	054502	047440	000103		
6471	012622	047125	051122	047040	EM16:	.ASCIZ /UNRR NOT CLEARED/
	012630	052117	041440	042514		
	012636	051101	042105	000		
6472	012643	111	042122	020131	EM17:	.ASCIZ /IRDY NOT SET/
	012650	047516	020124	042523		
	012656	000124				
6473	012660	051111	054504	047040	EM18:	.ASCIZ /IRDY NOT CLEARED/
	012666	052117	041440	042514		
	012674	051101	042105	000		
6474	012701	111	044503	020122	EM19:	.ASCIZ /ICIR NOT SET/
	012706	047516	020124	042523		
	012714	000124				
6475	012716	041511	051111	047040	EM20:	.ASCIZ /ICIR NOT CLEARED/
	012724	052117	041440	042514		
	012732	051101	042105	000		
6476	012737	111	041501	020124	EM21:	.ASCIZ /IACT NOT SET/
	012744	047516	020124	042523		
	012752	000124				
6477	012754	040511	052103	047040	EM22:	.ASCIZ /IACT NOT CLEARED/
	012762	052117	041440	042514		
	012770	051101	042105	000		
6478	012775	104	051523	020111	EM23:	.ASCIZ /DSSI NOT CLEARED/
	013002	047516	020124	046103		
	013010	040505	042522	000104		
6479	013016	051504	044523	047040	EM24:	.ASCIZ /DSSI NOT SET/
	013024	052117	051440	052105		
	013032	000				
6480	013033	104	051523	020111	EM25:	.ASCIZ /DSSI NOT CLEARED BY MST CLR/
	013040	047516	020124	046103		
	013046	040505	042522	020104		
	013054	054502	046440	052123		
	013062	041440	051114	000		
6481	013067	111	041516	051117	EM26:	.ASCIZ /INCORRECT DATA CHAR RCV'D/
	013074	042522	052103	042040		
	013102	052101	020101	044103		
	013110	051101	051040	053103		
	013116	042047	000			
6482	013121	111	041516	051117	EM27:	.ASCIZ /INCORRECT CRC BYTE RCV'D/
	013126	042522	052103	041440		
	013134	041522	041040	052131		

	013142	020105	041522	023526			
6483	013150	000104					
	013152	051522	046517	047040	EM28:	.ASCIZ	/RSOM NOT CLEARED/
	013160	052117	041440	042514			
	013166	051101	042105	000			
6484	013173	122	047523	020115	EM29:	.ASCIZ	/RSOM NOT SET/
	013200	047516	020124	042523			
	013206	000124					
6485	013210	042522	046517	047040	EM30:	.ASCIZ	/REOM NOT CLEARED/
	013216	052117	041440	042514			
	013224	051101	042105	000			
6486	013231	122	047505	020115	EM31:	.ASCIZ	/REOM NOT SET/
	013236	047516	020124	042523			
	013244	000124					
6487	013246	054124	040504	040524	EM32:	.ASCIZ	/TXDATA BIT NOT CLEARED/
	013254	041040	052111	047040			
	013262	052117	041440	042514			
	013270	051101	042105	000			
6488	013275	124	042130	052101	EM33:	.ASCIZ	/TXDATA BIT NOT SET/
	013302	020101	044502	020124			
	013310	047516	020124	042523			
	013316	000124					
6489	013320	041522	023526	020104	EM34:	.ASCIZ	/RCV'D DATA MISCOMPARE/
	013326	040504	040524	046440			
	013334	051511	047503	050115			
	013342	051101	000105				
6490	013346	041502	020103	047516	EM35:	.ASCIZ	/BCC NOT CLEARED/
	013354	020124	046103	040505			
	013362	042522	000104				
6491	013366	041502	020103	047516	EM36:	.ASCIZ	/BCC NOT SET/
	013374	020124	042523	000124			
6492	013402	041105	045514	047040	EM37:	.ASCIZ	/EBLK NOT CLEARED/
	013410	052117	041440	042514			
	013416	051101	042105	000			
6493	013423	105	046102	020113	EM38:	.ASCIZ	/EBLK NOT SET/
	013430	047516	020124	042523			
	013436	000124					
6494	013440	040522	020102	047516	EM39:	.ASCIZ	/RAB NOT CLEARED/
	013446	020124	046103	040505			
	013454	042522	000104				
6495	013460	040522	020102	047516	EM40:	.ASCIZ	/RAB NOT SET/
	013466	020124	042523	000124			
6496	013474	053117	051122	047040	EM41:	.ASCIZ	/OVRR NOT CLEARED/
	013502	052117	041440	042514			
	013510	051101	042105	000			
6497	013515	117	051126	020122	EM42:	.ASCIZ	/OVRR NOT SET/
	013522	047516	020124	042523			
	013530	000124					
6498	013532	053523	050040	041501	EM43:	.ASCIZ	/SW PACK #1 INCORRECT/
	013540	020113	030443	044440			
	013546	041516	051117	042522			
	013554	052103	000				
6499	013557	123	020127	040520	EM44:	.ASCIZ	/SW PACK #2 INCORRECT/
	013564	045503	021440	020062			
	013572	047111	047503	051122			
	013600	041505	000124				

6500	013604	053523	050040	041501	EM45:	.ASCIZ /SW PACK #3 INCORRECT/
	013612	020113	031443	044440		
	013620	041516	051117	042522		
	013626	052103	000			
6501	013631	122	053103	051440	EM46:	.ASCIZ /RCV SILO NOT CLEARED BY IC/
	013636	046111	020117	047516		
	013644	020124	046103	040505		
	013652	042522	020104	054502		
	013660	044440	000103			
6502	013664	051501	042523	041115	EM47:	.ASCIZ /ASSEMB BIT COUNT INCORRECT/
	013672	041040	052111	041440		
	013700	052517	052116	044440		
	013706	041516	051117	042522		
	013714	052103	000			
6503	013717	117	042104	053040	EM48:	.ASCIZ /ODD VRC PARITY BIT NOT SET/
	013724	041522	050040	051101		
	013732	052111	020131	044502		
	013740	020124	047516	020124		
	013746	042523	000124			
6504	013752	042117	020104	051126	EM49:	.ASCIZ /ODD VRC PARITY BIT NOT CLEARED/
	013760	020103	040520	044522		
	013766	054524	041040	052111		
	013774	047040	052117	041440		
	014002	042514	051101	042105		
	014010	000				
6505	014011	105	042526	020116	EM50:	.ASCIZ /EVEN VRC PARITY BIT NOT SET/
	014016	051126	020103	040520		
	014024	044522	054524	041040		
	014032	052111	047040	052117		
	014040	051440	052105	000		
6506	014045	105	042526	020116	EM51:	.ASCIZ /EVEN VRC PARITY BIT NOT CLEARED/
	014052	051126	020103	040520		
	014060	044522	054524	041040		
	014066	052111	047040	052117		
	014074	041440	042514	051101		
	014102	042105	000			
6507	014105	122	040505	054504	EM52:	.ASCIZ /READY NOT SET AFTER AX REG WRITE/
	014112	047040	052117	051440		
	014120	052105	040440	052106		
	014126	051105	040440	020130		
	014134	042522	020107	051127		
	014142	052111	000105			
6508	014146	042522	042101	020131	EM53:	.ASCIZ /READY NOT SET AFTER AX REG READ/
	014154	047516	020124	042523		
	014162	020124	043101	042524		
	014170	020122	054101	051040		
	014176	043505	051040	040505		
	014204	000104				
6509	014206	054124	052440	042116	EM54:	.ASCIZ /TX UNDERRUN ERROR/
	014214	051105	052522	020116		
	014222	051105	047522	000122		
6510	014230	052122	020123	047516	EM60:	.ASCIZ /RTS NOT SET/
	014236	020124	042523	000124		
6511	014244	052122	020123	047516	EM65:	.ASCIZ /RTS NOT CLEARED/
	014252	020124	046103	040505		
	014260	042522	000104			

6512  
6513  
6514  
6515 014264 047111 052502 027523 DH1: .ASCIZ &INBUS/OUTBUS REG &  
014272 052517 041124 051525  
014300 051040 043505 000040  
6516 014306 044514 042516 052440 DH2: .ASCIZ /LINE UNIT INBUS REGS ./  
014314 044516 020124 047111  
014322 052502 020123 042522  
014330 051507 035040 000  
6517 014335 122 043505 030061 DH3: .ASCIZ /REG10 REG11 REG12 REG13/  
014342 020040 051040 043505  
014350 030461 020040 051040  
014356 043505 031061 020040  
014364 051040 043505 031461  
014372 000  
6518 014373 040 020040 051040 DH4: .ASCIZ / REG14 REG15 REG16 REG17/  
014400 043505 032061 020040  
014406 051040 043505 032461  
014414 020040 051040 043505  
014422 033061 020040 051040  
014430 043505 033461 000  
6519 014435 061 000065 DH5: .ASCIZ /15/  
6520 014440 033061 000 DH6: .ASCIZ /16/  
6521 014443 114 047111 020105 DH7: .ASCIZ /LINE UNIT EXTENDED REGS :/  
014450 047125 052111 042440  
014456 052130 047105 042504  
014464 020104 042522 051507  
014472 035040 000  
6522 014475 101 030130 030455 DH8: .ASCIZ /AX0-15 AX0-16 AX1-15 AX1-16/  
014502 020065 040440 030130  
014510 030455 020066 040440  
014516 030530 030455 020065  
014524 040440 030530 030455  
014532 000066  
6523 014534 020040 020040 054101 DH9: .ASCIZ / AX2-15 AX2-16 AX3-15 AX3-16/  
014542 026462 032461 020040  
014550 054101 026462 033061  
014556 020040 054101 026463  
014564 032461 020040 054101  
014572 026463 033061 000

6524  
6525 014600 .EVEN  
6526  
6527  
6528  
6529  
6530

6531 014600 BGNMSG ERR1  
(3) 014600  
6532 014600 PRINTB #FMT1,#ADDRES,MPCSR  
(9) 014600 013746 002446  
(8) 014604 012746 035620  
(7) 014610 012746 011454  
(6) 014614 012746 000003  
(3) 014620 010600

ERR1::

MOV MPCSR, -(SP)  
MOV #ADDRES, -(SP)  
MOV #FMT1, -(SP)  
MOV #3, -(SP)  
MOV SP, R0



(4)	014622	104414					TRAP	(SPNTB
(4)	014624	062706	000010				ADD	#10,SP
6533	014630			ENDMSG				
(3)	014630					L10002:	TRAP	(SMSG
(3)	014630	104423						
6534								
6535								
6536								
6537	014632			BGNMSG	ERR2			
(3)	014632					ERR2::		
6538	014632			PRINTB	#FMT1,#ADDRES,MPCSR			
(9)	014632	013746	002446				MOV	MPCSR,-(SP)
(8)	014636	012746	035620				MOV	#ADDRES,-(SP)
(7)	014642	012746	011454				MOV	#FMT1,-(SP)
(6)	014646	012746	000003				MOV	#3,-(SP)
(3)	014652	010600					MOV	SP,R0
(4)	014654	104414					TRAP	(SPNTB
(4)	014656	062706	000010				ADD	#10,SP
6539	014662			PRINTB	#FMT2			
(7)	014662	012746	011464				MOV	#FMT2,-(SP)
(6)	014666	012746	000001				MOV	#1,-(SP)
(3)	014672	010600					MOV	SP,R0
(4)	014674	104414					TRAP	(SPNTB
(4)	014676	062706	000004				ADD	#4,SP
6540	014702			PRINTB	#FMT7,#DH1,REGNUM			
(9)	014702	013746	002400				MOV	REGNUM,-(SP)
(8)	014706	012746	014264				MOV	#DH1,-(SP)
(7)	014712	012746	011646				MOV	#FMT7,-(SP)
(6)	014716	012746	000003				MOV	#3,-(SP)
(3)	014722	010600					MOV	SP,R0
(4)	014724	104414					TRAP	(SPNTB
(4)	014726	062706	000010				ADD	#10,SP
6541	014732			PRINTB	#FMT3,GOODAT,BADDAT			
(9)	014732	013746	002406				MOV	BADDAT,-(SP)
(8)	014736	013746	002404				MOV	GOODAT,-(SP)
(7)	014742	012746	011506				MOV	#FMT3,-(SP)
(6)	014746	012746	000003				MOV	#3,-(SP)
(3)	014752	010600					MOV	SP,R0
(4)	014754	104414					TRAP	(SPNTB
(4)	014756	062706	000010				ADD	#10,SP
6542	014762			PRINTX	#FMT4,#DH2,#DH3			
(9)	014762	012746	014335				MOV	#DH3,-(SP)
(8)	014766	012746	014306				MOV	#DH2,-(SP)
(7)	014772	012746	011550				MOV	#FMT4,-(SP)
(6)	014776	012746	000003				MOV	#3,-(SP)
(3)	015002	010600					MOV	SP,R0
(4)	015004	104415					TRAP	(SPNTX
(4)	015006	062706	000010				ADD	#10,SP
6543	015012			PRINTX	#FMT5,LUR10,LUR11,LUR12,LUR13			
(11)	015012	013746	002310				MOV	LUR13,-(SP)
(10)	015016	013746	002306				MOV	LUR12,-(SP)
(9)	015022	013746	002304				MOV	LUR11,-(SP)
(8)	015026	013746	002302				MOV	LUR10,-(SP)
(7)	015032	012746	011563				MOV	#FMT5,-(SP)
(6)	015036	012746	000005				MOV	#5,-(SP)
(3)	015042	010600					MOV	SP,R0

(4) 015044 104415  
(4) 015046 062706 000014  
6544 015052  
(8) 015052 012746 014373  
(7) 015056 012746 011712  
(6) 015062 012746 000002  
(3) 015066 010600  
(4) 015070 104415  
(4) 015072 062706 000006  
6545 015076  
(11) 015076 013746 002320  
(10) 015102 013746 002316  
(9) 015106 013746 002314  
(8) 015112 013746 002312  
(7) 015116 012746 011613  
(6) 015122 012746 000005  
(3) 015126 010600  
(4) 015130 104415  
(4) 015132 062706 000014  
6546 015136  
(3) 015136  
(3) 015136 104423  
6547  
6548  
6549  
6550  
6551  
6552 015140  
(3) 015140  
6553 015140  
(9) 015140 013746 002446  
(8) 015144 012746 035620  
(7) 015150 012746 011454  
(6) 015154 012746 000003  
(3) 015160 010600  
(4) 015162 104414  
(4) 015164 062706 000010  
6554 015170  
(7) 015170 012746 011464  
(6) 015174 012746 000001  
(3) 015200 010600  
(4) 015202 104414  
(4) 015204 062706 000004  
6555 015210  
(9) 015210 013746 002530  
(8) 015214 013746 002532  
(7) 015220 012746 011656  
(6) 015224 012746 000003  
(3) 015230 010600  
(4) 015232 104414  
(4) 015234 062706 000010  
6556 015240  
(9) 015240 013746 002406  
(8) 015244 013746 002404  
(7) 015250 012746 011506  
(6) 015254 012746 000003

PRINTX #FMT9,#DH4

PRINTX #FMT6,LUR14,LUR15,LUR16,LUR17

ENDMSG

BGNMSG ERR3

PRINTB #FMT1,#ADDRES,MPCSR

PRINTB #FMT2

PRINTB #FMT8,TMP1,TMPO

PRINTB #FMT3,GOODAT,BADDAT

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  
L10003: 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 TMPO,-(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)  
MOV #3,-(SP)

(3) 015260 010600  
(4) 015262 104414  
(4) 015264 062706 000010  
6557 015270  
(9) 015270 012746 014335  
(8) 015274 012746 014306  
(7) 015300 012746 011550  
(6) 015304 012746 000003  
(3) 015310 010600  
(4) 015312 104415  
(4) 015314 062706 000010  
6558 015320  
(11) 015320 013746 002310  
(10) 015324 013746 002306  
(9) 015330 013746 002304  
(8) 015334 013746 002302  
(7) 015340 012746 011563  
(6) 015344 012746 000005  
(3) 015350 010600  
(4) 015352 104415  
(4) 015354 062706 000014  
6559 015360  
(8) 015360 012746 014373  
(7) 015364 012746 011712  
(6) 015370 012746 000002  
(3) 015374 010600  
(4) 015376 104415  
(4) 015400 062706 000006  
6560 015404  
(11) 015404 013746 002320  
(10) 015410 013746 002316  
(9) 015414 013746 002314  
(8) 015420 013746 002312  
(7) 015424 012746 011613  
(6) 015430 012746 000005  
(3) 015434 010600  
(4) 015436 104415  
(4) 015440 062706 000014  
6561 015444  
(9) 015444 012746 014475  
(8) 015450 012746 014443  
(7) 015454 012746 011550  
(6) 015460 012746 000003  
(3) 015464 010600  
(4) 015466 104415  
(4) 015470 062706 000010  
6562 015474  
(11) 015474 013746 002330  
(10) 015500 013746 002326  
(9) 015504 013746 002324  
(8) 015510 013746 002322  
(7) 015514 012746 011563  
(6) 015520 012746 000005  
(3) 015524 010600  
(4) 015526 104415  
(4) 015530 062706 000014

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 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  
ADD #14,SP

6563	015534			PRINTX #FMT9,#DH9			
(8)	015534	012746	014534			MOV	#DH9,-(SP)
(7)	015540	012746	011712			MOV	#FMT9,-(SP)
(6)	015544	012746	000002			MOV	#2,-(SP)
(3)	015550	010600				MOV	SP,R0
(4)	015552	104415				TRAP	C\$PNTX
(4)	015554	062706	000006			ADD	#6,SP
6564	015560			PRINTX #FMT6,AX2.15,AX2.16,AX3.15,AX3.16			
(11)	015560	013746	002340			MOV	AX3.16,-(SP)
(10)	015564	013746	002336			MOV	AX3.15,-(SP)
(9)	015570	013746	002334			MOV	AX2.16,-(SP)
(8)	015574	013746	002332			MOV	AX2.15,-(SP)
(7)	015600	012746	011613			MOV	#FMT6,-(SP)
(6)	015604	012746	000005			MOV	#5,-(SP)
(3)	015610	010600				MOV	SP,R0
(4)	015612	104415				TRAP	C\$PNTX
(4)	015614	062706	000014			ADD	#14,SP
6565	015620			ENDMSG			
(3)	015620						
(3)	015620	104423				L10004:	TRAP C\$MSG
6566							
6567							
6568							
6569							
6570							
6571	015622			BGNMSG ERR4			
(3)	015622						
6572	015622			PRINTB #FMT10,SUBRPC		ERR4::	
(8)	015622	013746	002352			MOV	SUBRPC,-(SP)
(7)	015626	012746	011717			MOV	#FMT10,-(SP)
(6)	015632	012746	000002			MOV	#2,-(SP)
(3)	015636	010600				MOV	SP,R0
(4)	015640	104414				TRAP	C\$PNTB
(4)	015642	062706	000006			ADD	#6,SP
6573	015646			PRINTB #FMT1,#ADDRES,MPCSR			
(9)	015646	013746	002446			MOV	MPCSR,-(SP)
(8)	015652	012746	035620			MOV	#ADDRES,-(SP)
(7)	015656	012746	011454			MOV	#FMT1,-(SP)
(6)	015662	012746	000003			MOV	#3,-(SP)
(3)	015666	010600				MOV	SP,R0
(4)	015670	104414				TRAP	C\$PNTB
(4)	015672	062706	000010			ADD	#10,SP
6574	015676			PRINTB #FMT2			
(7)	015676	012746	011464			MOV	#FMT2,-(SP)
(6)	015702	012746	000001			MOV	#1,-(SP)
(3)	015706	010600				MOV	SP,R0
(4)	015710	104414				TRAP	C\$PNTB
(4)	015712	062706	000004			ADD	#4,SP
6575	015716			PRINTB #FMT7,#DH1,REGNUM			
(9)	015716	013746	002400			MOV	REGNUM,-(SP)
(8)	015722	012746	014264			MOV	#DH1,-(SP)
(7)	015726	012746	011646			MOV	#FMT7,-(SP)
(6)	015732	012746	000003			MOV	#3,-(SP)
(3)	015736	010600				MOV	SP,R0
(4)	015740	104414				TRAP	C\$PNTB
(4)	015742	062706	000010			ADD	#10,SP

6576 015746  
(9) 015746 012746 014335  
(8) 015752 012746 014306  
(7) 015756 012746 011550  
(6) 015762 012746 000003  
(3) 015766 010600  
(4) 015770 104415  
(4) 015772 062706 000010  
6577 015776  
(11) 015776 013746 002310  
(10) 016002 013746 002306  
(9) 016006 013746 002304  
(8) 016012 013746 002302  
(7) 016016 012746 011563  
(6) 016022 012746 000005  
(3) 016026 010600  
(4) 016030 104415  
(4) 016032 062706 000014  
6578 016036  
(8) 016036 012746 014373  
(7) 016042 012746 011712  
(6) 016046 012746 000002  
(3) 016052 010600  
(4) 016054 104415  
(4) 016056 062706 000006  
6579 016062  
(11) 016062 013746 002320  
(10) 016066 013746 002316  
(9) 016072 013746 002314  
(8) 016076 013746 002312  
(7) 016102 012746 011613  
(6) 016106 012746 000005  
(3) 016112 010600  
(4) 016114 104415  
(4) 016116 062706 000014  
6580 016122  
(9) 016122 012746 014475  
(8) 016126 012746 014443  
(7) 016132 012746 011550  
(6) 016136 012746 000003  
(3) 016142 010600  
(4) 016144 104415  
(4) 016146 062706 000010  
6581 016152  
(11) 016152 013746 002330  
(10) 016156 013746 002326  
(9) 016162 013746 002324  
(8) 016166 013746 002322  
(7) 016172 012746 011563  
(6) 016176 012746 000005  
(3) 016202 010600  
(4) 016204 104415  
(4) 016206 062706 000014  
6582 016212  
(8) 016212 012746 014534  
(7) 016216 012746 011712

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

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)

(6)	016222	012746	000002			MOV	#2,-(SP)
(3)	016226	010600				MOV	SP,R0
(4)	016230	104415				TRAP	C\$PNTX
(4)	016232	062706	000006			ADD	#6,SP
6583	016236			PRINTX	#FMT6,AX2.15,AX2.16,AX3.15,AX3.16		
(11)	016236	013746	002340			MOV	AX3.16,-(SP)
(10)	016242	013746	002336			MOV	AX3.15,-(SP)
(9)	016246	013746	002334			MOV	AX2.16,-(SP)
(8)	016252	013746	002332			MOV	AX2.15,-(SP)
(7)	016256	012746	011613			MOV	#FMT6,-(SP)
(6)	016262	012746	000005			MOV	#5,-(SP)
(3)	016266	010600				MOV	SP,R0
(4)	016270	104415				TRAP	C\$PNTX
(4)	016272	062706	000014			ADD	#14,SP
6584	016276			ENDMSG			
(3)	016276						
(3)	016276	104423				L10005:	TRAP C\$MSG
6585							
6586							
6587							
6588							
6589							
6590	016300			BGNMSG	ERR5		
(3)	016300						
6591	016300			PRINTB	#FMT1,#ADDRES,MPCSR	ERR5::	
(9)	016300	013746	002446			MOV	MPCSR,-(SP)
(8)	016304	012746	035620			MOV	#ADDRES,-(SP)
(7)	016310	012746	011454			MOV	#FMT1,-(SP)
(6)	016314	012746	000003			MOV	#3,-(SP)
(3)	016320	010600				MOV	SP,R0
(4)	016322	104414				TRAP	C\$PNTB
(4)	016324	062706	000010			ADD	#10,SP
6592	016330			PRINTB	#FMT11,REGNUM,LOADAT		
(9)	016330	013746	002410			MOV	LOADAT,-(SP)
(8)	016334	013746	002400			MOV	REGNUM,-(SP)
(7)	016340	012746	011750			MOV	#FMT11,-(SP)
(6)	016344	012746	000003			MOV	#3,-(SP)
(3)	016350	010600				MOV	SP,R0
(4)	016352	104414				TRAP	C\$PNTB
(4)	016354	062706	000010			ADD	#10,SP
6593	016360			PRINTB	#FMT2		
(7)	016360	012746	011464			MOV	#FMT2,-(SP)
(6)	016364	012746	000001			MOV	#1,-(SP)
(3)	016370	010600				MOV	SP,R0
(4)	016372	104414				TRAP	C\$PNTB
(4)	016374	062706	000004			ADD	#4,SP
6594	016400			PRINTB	#FMT8,TMP1,TMP0		
(9)	016400	013746	002530			MOV	TMP0,-(SP)
(8)	016404	013746	002532			MOV	TMP1,-(SP)
(7)	016410	012746	011656			MOV	#FMT8,-(SP)
(6)	016414	012746	000003			MOV	#3,-(SP)
(3)	016420	010600				MOV	SP,R0
(4)	016422	104414				TRAP	C\$PNTB
(4)	016424	062706	000010			ADD	#10,SP
6595	016430			PRINTB	#FMT5,GOODAT,BADDAT		
(9)	016430	013746	002406			MOV	BADDAT,-(SP)

(8) 016434 013746 002404  
(7) 016440 012746 011506  
(6) 016444 012746 000003  
(3) 016450 010600  
(4) 016452 104414  
(4) 016454 062706 000010  
6596 016460  
(9) 016460 012746 014335  
(8) 016464 012746 014306  
(7) 016470 012746 011550  
(6) 016474 012746 000003  
(3) 016500 010600  
(4) 016502 104415  
(4) 016504 062706 000010  
6597 016510  
(11) 016510 013746 002310  
(10) 016514 013746 002306  
(9) 016520 013746 002304  
(8) 016524 013746 002302  
(7) 016530 012746 011563  
(6) 016534 012746 000005  
(3) 016540 010600  
(4) 016542 104415  
(4) 016544 062706 000014  
6598 016550  
(8) 016550 012746 014373  
(7) 016554 012746 011712  
(6) 016560 012746 000002  
(3) 016564 010600  
(4) 016566 104415  
(4) 016570 062706 000006  
6599 016574  
(11) 016574 013746 002320  
(10) 016600 013746 002316  
(9) 016604 013746 002314  
(8) 016610 013746 002312  
(7) 016614 012746 011613  
(6) 016620 012746 000005  
(3) 016624 010600  
(4) 016626 104415  
(4) 016630 062706 000014  
6600 016634  
(9) 016634 012746 014475  
(8) 016640 012746 014443  
(7) 016644 012746 011550  
(6) 016650 012746 000003  
(3) 016654 010600  
(4) 016656 104415  
(4) 016660 062706 000010  
6601 016664  
(11) 016664 013746 002330  
(10) 016670 013746 002326  
(9) 016674 013746 002324  
(8) 016700 013746 002322  
(7) 016704 012746 011563  
(6) 016710 012746 000005

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

(3)	016714	010600				MOV	SP,R0
(4)	016716	104415				TRAP	C\$PNTX
(4)	016720	062706	000014			ADD	#14,SP
6602	016724			PRINTX	#FMT9,#DH9		
(8)	016724	012746	014534			MOV	#DH9,-(SP)
(7)	016730	012746	011712			MOV	#FMT9,-(SP)
(6)	016734	012746	000002			MOV	#2,-(SP)
(3)	016740	010600				MOV	SP,R0
(4)	016742	104415				TRAP	C\$PNTX
(4)	016744	062706	000006			ADD	#6,SP
6603	016750			PRINTX	#FMT6,AX2.15,AX2.16,AX3.15,AX3.16		
(11)	016750	013746	002340			MOV	AX3.16,-(SP)
(10)	016754	013746	002336			MOV	AX3.15,-(SP)
(9)	016760	013746	002334			MOV	AX2.16,-(SP)
(8)	016764	013746	002332			MOV	AX2.15,-(SP)
(7)	016770	012746	011613			MOV	#FMT6,-(SP)
(6)	016774	012746	000005			MOV	#5,-(SP)
(3)	017000	010600				MOV	SP,R0
(4)	017002	104415				TRAP	C\$PNTX
(4)	017004	062706	000014			ADD	#14,SP
6604	017010			ENDMSG			
(3)	017010					L10006:	
(3)	017010	104423				TRAP	C\$MSG
6605							
6606							
6607							
6608							
6609							
6610	017012			BGNMSG	ERR6		
(3)	017012					ERR6::	
6611	017012			PRINTB	#FMT10,SUBRPC		
(8)	017012	013746	002352			MOV	SUBRPC,-(SP)
(7)	017016	012746	011717			MOV	#FMT10,-(SP)
(6)	017022	012746	000002			MOV	#2,-(SP)
(3)	017026	010600				MOV	SP,R0
(4)	017030	104414				TRAP	C\$PNTB
(4)	017032	062706	000006			ADD	#6,SP
6612	017036			PRINTB	#FMT1,#ADDRES,MPCSR		
(9)	017036	013746	002446			MOV	MPCSR,-(SP)
(8)	017042	012746	035620			MOV	#ADDRES,-(SP)
(7)	017046	012746	011454			MOV	#FMT1,-(SP)
(6)	017052	012746	000003			MOV	#3,-(SP)
(3)	017056	010600				MOV	SP,R0
(4)	017060	104414				TRAP	C\$PNTB
(4)	017062	062706	000010			ADD	#10,SP
6613	017066			PRINTB	#FMT2		
(7)	017066	012746	011464			MOV	#FMT2,-(SP)
(6)	017072	012746	000001			MOV	#1,-(SP)
(3)	017076	010600				MOV	SP,R0
(4)	017100	104414				TRAP	C\$PNTB
(4)	017102	062706	000004			ADD	#4,SP
6614	017106			PRINTB	#FMT8,TMP1,TMP0		
(9)	017106	013746	002530			MOV	TMP0,-(SP)
(8)	017112	013746	002532			MOV	TMP1,-(SP)
(7)	017116	012746	011656			MOV	#FMT8,-(SP)
(6)	017122	012746	000003			MOV	#3,-(SP)



(3) 017126 010600  
(4) 017130 104414  
(4) 017132 062706 000010  
6615 017136  
(9) 017136 012746 014335  
(8) 017142 012746 014306  
(7) 017146 012746 011550  
(6) 017152 012746 000003  
(3) 017156 010600  
(4) 017160 104415  
(4) 017162 062706 000010  
6616 017166  
(11) 017166 013746 002310  
(10) 017172 013746 002306  
(9) 017176 013746 002304  
(8) 017202 013746 002302  
(7) 017206 012746 011563  
(6) 017212 012746 000005  
(3) 017216 010600  
(4) 017220 104415  
(4) 017222 062706 000014  
6617 017226  
(8) 017226 012746 014373  
(7) 017232 012746 011712  
(6) 017236 012746 000002  
(3) 017242 010600  
(4) 017244 104415  
(4) 017246 062706 000006  
6618 017252  
(11) 017252 013746 002320  
(10) 017256 013746 002316  
(9) 017262 013746 002314  
(8) 017266 013746 002312  
(7) 017272 012746 011613  
(6) 017276 012746 000005  
(3) 017302 010600  
(4) 017304 104415  
(4) 017306 062706 000014  
6619 017312  
(9) 017312 012746 014475  
(8) 017316 012746 014443  
(7) 017322 012746 011550  
(6) 017326 012746 000003  
(3) 017332 010600  
(4) 017334 104415  
(4) 017336 062706 000010  
6620 017342  
(11) 017342 013746 002330  
(10) 017346 013746 002326  
(9) 017352 013746 002324  
(8) 017356 013746 002322  
(7) 017362 012746 011563  
(6) 017366 012746 000005  
(3) 017372 010600  
(4) 017374 104415  
(4) 017376 062706 000014

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 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  
ADD #14,SP

6621	017402			PRINTX #FMT9,#DH9			
(8)	017402	012746	014534			MOV	#DH9,-(SP)
(7)	017406	012746	011712			MOV	#FMT9,-(SP)
(6)	017412	012746	000002			MOV	#2,-(SP)
(3)	017416	010600				MOV	SP,R0
(4)	017420	104415				TRAP	C\$PNTX
(4)	017422	062706	000006			ADD	#6,SP
6622	017426			PRINTX #FMT6,AX2.15,AX2.16,AX3.15,AX3.16			
(11)	017426	013746	002340			MOV	AX3.16,-(SP)
(10)	017432	013746	002336			MOV	AX3.15,-(SP)
(9)	017436	013746	002334			MOV	AX2.16,-(SP)
(8)	017442	013746	002332			MOV	AX2.15,-(SP)
(7)	017446	012746	011613			MOV	#FMT6,-(SP)
(6)	017452	012746	000005			MOV	#5,-(SP)
(3)	017456	010600				MOV	SP,R0
(4)	017460	104415				TRAP	C\$PNTX
(4)	017462	062706	000014			ADD	#14,SP
6623	017466			ENDMSG			
(3)	017466						
(3)	017466	104423				L10007:	TRAP C\$MSG
6624							
6625							
6626							
6627							
6628							
6629	017470			BGNMSG ERR7			
(3)	017470						
6630	017470			PRINTB #FMT1,#ADDRES,MPCSR		ERR7::	
(9)	017470	013746	002446			MOV	MPCSR,-(SP)
(8)	017474	012746	035620			MOV	#ADDRES,-(SP)
(7)	017500	012746	011454			MOV	#FMT1,-(SP)
(6)	017504	012746	000003			MOV	#3,-(SP)
(3)	017510	010600				MOV	SP,R0
(4)	017512	104414				TRAP	C\$PNTB
(4)	017514	062706	000010			ADD	#10,SP
6631	017520			PRINTB #FMT2			
(7)	017520	012746	011464			MOV	#FMT2,-(SP)
(6)	017524	012746	000001			MOV	#1,-(SP)
(3)	017530	010600				MOV	SP,R0
(4)	017532	104414				TRAP	C\$PNTB
(4)	017534	062706	000004			ADD	#4,SP
6632	017540			PRINTB #FMT7,#DH1,REGNUM			
(9)	017540	013746	002400			MOV	REGNUM,-(SP)
(8)	017544	012746	014264			MOV	#DH1,-(SP)
(7)	017550	012746	011646			MOV	#FMT7,-(SP)
(6)	017554	012746	000003			MOV	#3,-(SP)
(3)	017560	010600				MOV	SP,R0
(4)	017562	104414				TRAP	C\$PNTB
(4)	017564	062706	000010			ADD	#10,SP
6633	017570			PRINTX #FMT4,#DH2,#DH3			
(9)	017570	012746	014335			MOV	#DH3,-(SP)
(8)	017574	012746	014306			MOV	#DH2,-(SP)
(7)	017600	012746	011550			MOV	#FMT4,-(SP)
(6)	017604	012746	000003			MOV	#3,-(SP)
(3)	017610	010600				MOV	SP,R0
(4)	017612	104415				TRAP	C\$PNTX

(4)	017614	062706	000010				ADD	#10,SP
6634	017620			PRINTX	#FMT5,LUR10,LUR11,LUR12,LUR13			
(11)	017620	013746	002310				MOV	LUR13,-(SP)
(10)	017624	013746	002306				MOV	LUR12,-(SP)
(9)	017630	013746	002304				MOV	LUR11,-(SP)
(8)	017634	013746	002302				MOV	LUR10,-(SP)
(7)	017640	012746	011563				MOV	#FMT5,-(SP)
(6)	017644	012746	000005				MOV	#5,-(SP)
(3)	017650	010600					MOV	SP,R0
(4)	017652	104415					TRAP	C\$PNTX
(4)	017654	062706	000014				ADD	#14,SP
6635	017660			PRINTX	#FMT9,#DH4			
(8)	017660	012746	014373				MOV	#DH4,-(SP)
(7)	017664	012746	011712				MOV	#FMT9,-(SP)
(6)	017670	012746	000002				MOV	#2,-(SP)
(3)	017674	010600					MOV	SP,R0
(4)	017676	104415					TRAP	C\$PNTX
(4)	017700	062706	000006				ADD	#6,SP
6636	017704			PRINTX	#FMT6,LUR14,LUR15,LUR16,LUR17			
(11)	017704	013746	002320				MOV	LUR17,-(SP)
(10)	017710	013746	002316				MOV	LUR16,-(SP)
(9)	017714	013746	002314				MOV	LUR15,-(SP)
(8)	017720	013746	002312				MOV	LUR14,-(SP)
(7)	017724	012746	011613				MOV	#FMT6,-(SP)
(6)	017730	012746	000005				MOV	#5,-(SP)
(3)	017734	010600					MOV	SP,R0
(4)	017736	104415					TRAP	C\$PNTX
(4)	017740	062706	000014				ADD	#14,SP
6637	017744			PRINTX	#FMT4,#DH7,#DH8			
(9)	017744	012746	014475				MOV	#DH8,-(SP)
(8)	017750	012746	014443				MOV	#DH7,-(SP)
(7)	017754	012746	011550				MOV	#FMT4,-(SP)
(6)	017760	012746	000003				MOV	#3,-(SP)
(3)	017764	010600					MOV	SP,R0
(4)	017766	104415					TRAP	C\$PNTX
(4)	017770	062706	000010				ADD	#10,SP
6638	017774			PRINTX	#FMT5,AX0.15,AX0.16,AX1.15,AX1.16			
(11)	017774	013746	002330				MOV	AX1.16,-(SP)
(10)	020000	013746	002326				MOV	AX1.15,-(SP)
(9)	020004	013746	002324				MOV	AX0.16,-(SP)
(8)	020010	013746	002322				MOV	AX0.15,-(SP)
(7)	020014	012746	011563				MOV	#FMT5,-(SP)
(6)	020020	012746	000005				MOV	#5,-(SP)
(3)	020024	010600					MOV	SP,R0
(4)	020026	104415					TRAP	C\$PNTX
(4)	020030	062706	000014				ADD	#14,SP
6639	020034			PRINTX	#FMT9,#DH9			
(8)	020034	012746	014534				MOV	#DH9,-(SP)
(7)	020040	012746	011712				MOV	#FMT9,-(SP)
(6)	020044	012746	000002				MOV	#2,-(SP)
(3)	020050	010600					MOV	SP,R0
(4)	020052	104415					TRAP	C\$PNTX
(4)	020054	062706	000006				ADD	#6,SP
6640	020060			PRINTX	#FMT6,AX2.15,AX2.16,AX3.15,AX3.16			
(11)	020060	013746	002340				MOV	AX3.16,-(SP)
(10)	020064	013746	002336				MOV	AX3.15,-(SP)

(9)	020070	013746	002334			MOV	AX2.16,-(SP)
(8)	020074	013746	002332			MOV	AX2.15,-(SP)
(7)	020100	012746	011613			MOV	#FMT6,-(SP)
(6)	020104	012746	000005			MOV	#5,-(SP)
(3)	020110	010600				MOV	SP,R0
(4)	020112	104415				TRAP	C\$PNTX
(4)	020114	062706	000014			ADD	#14,SP
6641	020120			ENDMSG			
(3)	020120				L10010:		
(3)	020120	104423				TRAP	C\$MSG
6642							
6643							
6644							
6645							
6646							
6647	020122			BGNMSG ERR8			
(3)	020122				ERR8::		
6648	020122			PRINTB #FMT10,SUBRPC			
(8)	020122	013746	002352			MOV	SUBRPC,-(SP)
(7)	020126	012746	011717			MOV	#FMT10,-(SP)
(6)	020132	012746	000002			MOV	#2,-(SP)
(3)	020136	010600				MOV	SP,R0
(4)	020140	104414				TRAP	C\$PNTB
(4)	020142	062706	000006			ADD	#6,SP
6649	020146			PRINTB #FMT1,#ADDRES,MPCSR			
(9)	020146	013746	002446			MOV	MPCSR,-(SP)
(8)	020152	012746	035620			MOV	#ADDRES,-(SP)
(7)	020156	012746	011454			MOV	#FMT1,-(SP)
(6)	020162	012746	000003			MOV	#3,-(SP)
(3)	020166	010600				MOV	SP,R0
(4)	020170	104414				TRAP	C\$PNTB
(4)	020172	062706	000010			ADD	#10,SP
6650	020176			PRINTB #FMT2			
(7)	020176	012746	011464			MOV	#FMT2,-(SP)
(6)	020202	012746	000001			MOV	#1,-(SP)
(3)	020206	010600				MOV	SP,R0
(4)	020210	104414				TRAP	C\$PNTB
(4)	020212	062706	000004			ADD	#4,SP
6651	020216			PRINTB #FMT7,#DH1,REGNUM			
(9)	020216	013746	002400			MOV	REGNUM,-(SP)
(8)	020222	012746	014264			MOV	#DH1,-(SP)
(7)	020226	012746	011646			MOV	#FMT7,-(SP)
(6)	020232	012746	000003			MOV	#3,-(SP)
(3)	020236	010600				MOV	SP,R0
(4)	020240	104414				TRAP	C\$PNTB
(4)	020242	062706	000010			ADD	#10,SP
6652	020246			PRINTB #FMT3,GOODAT,BADDAT			
(9)	020246	013746	002406			MOV	BADDAT,-(SP)
(8)	020252	013746	002404			MOV	GOODAT,-(SP)
(7)	020256	012746	011506			MOV	#FMT3,-(SP)
(6)	020262	012746	000003			MOV	#3,-(SP)
(3)	020266	010600				MOV	SP,R0
(4)	020270	104414				TRAP	C\$PNTB
(4)	020272	062706	000010			ADD	#10,SP
6653	020276			PRINTX #FMT4,#DH2,#DH3			
(9)	020276	012746	014335			MOV	#DH3,-(SP)

(8)	020302	012746	014306		MOV	#DH2,-(SP)
(7)	020306	012746	011550		MOV	#FMT4,-(SP)
(6)	020312	012746	000003		MOV	#3,-(SP)
(3)	020316	010600			MOV	SP,R0
(4)	020320	104415			TRAP	C\$PNTX
(4)	020322	062706	000010		ADD	#10,SP
6654	020326			PRINTX	#FMT5,LUR10,LUR11,LUR12,LUR13	
(11)	020326	013746	002310		MOV	LUR13,-(SP)
(10)	020332	013746	002306		MOV	LUR12,-(SP)
(9)	020336	013746	002304		MOV	LUR11,-(SP)
(8)	020342	013746	002302		MOV	LUR10,-(SP)
(7)	020346	012746	011563		MOV	#FMT5,-(SP)
(6)	020352	012746	000005		MOV	#5,-(SP)
(3)	020356	010600			MOV	SP,R0
(4)	020360	104415			TRAP	C\$PNTX
(4)	020362	062706	000014		ADD	#14,SP
6655	020366			PRINTX	#FMT9,#DH4	
(8)	020366	012746	014373		MOV	#DH4,-(SP)
(7)	020372	012746	011712		MOV	#FMT9,-(SP)
(6)	020376	012746	000002		MOV	#2,-(SP)
(3)	020402	010600			MOV	SP,R0
(4)	020404	104415			TRAP	C\$PNTX
(4)	020406	062706	000006		ADD	#6,SP
6656	020412			PRINTX	#FMT6,LUR14,LUR15,LUR16,LUR17	
(11)	020412	013746	002320		MOV	LUR17,-(SP)
(10)	020416	013746	002316		MOV	LUR16,-(SP)
(9)	020422	013746	002314		MOV	LUR15,-(SP)
(8)	020426	013746	002312		MOV	LUR14,-(SP)
(7)	020432	012746	011613		MOV	#FMT6,-(SP)
(6)	020436	012746	000005		MOV	#5,-(SP)
(3)	020442	010600			MOV	SP,R0
(4)	020444	104415			TRAP	C\$PNTX
(4)	020446	062706	000014		ADD	#14,SP
6657	020452			PRINTX	#FMT4,#DH7,#DH8	
(9)	020452	012746	014475		MOV	#DH8,-(SP)
(8)	020456	012746	014443		MOV	#DH7,-(SP)
(7)	020462	012746	011550		MOV	#FMT4,-(SP)
(6)	020466	012746	000003		MOV	#3,-(SP)
(3)	020472	010600			MOV	SP,R0
(4)	020474	104415			TRAP	C\$PNTX
(4)	020476	062706	000010		ADD	#10,SP
6658	020502			PRINTX	#FMT5,AX0.15,AX0.16,AX1.15,AX1.16	
(11)	020502	013746	002330		MOV	AX1.16,-(SP)
(10)	020506	013746	002326		MOV	AX1.15,-(SP)
(9)	020512	013746	002324		MOV	AX0.16,-(SP)
(8)	020516	013746	002322		MOV	AX0.15,-(SP)
(7)	020522	012746	011563		MOV	#FMT5,-(SP)
(6)	020526	012746	000005		MOV	#5,-(SP)
(3)	020532	010600			MOV	SP,R0
(4)	020534	104415			TRAP	C\$PNTX
(4)	020536	062706	000014		ADD	#14,SP
6659	020542			PRINTX	#FMT9,#DH9	
(8)	020542	012746	014534		MOV	#DH9,-(SP)
(7)	020546	012746	011712		MOV	#FMT9,-(SP)
(6)	020552	012746	000002		MOV	#2,-(SP)
(3)	020556	010600			MOV	SP,R0

(4)	020560	104415			TRAP	( \$PNTX
(4)	020562	062706	000006		ADD	#6,SP
6660	020566			PRINTX #FMT6,AX2.15,AX2.16,AX3.15,AX3.16		
(11)	020566	013746	002340		MOV	AX3.16,-(SP)
(10)	020572	013746	002336		MOV	AX3.15,-(SP)
(9)	020576	013746	002334		MOV	AX2.16,-(SP)
(8)	020602	013746	002332		MOV	AX2.15,-(SP)
(7)	020606	012746	011613		MOV	#FMT6,-(SP)
(6)	020612	012746	000005		MOV	#5,-(SP)
(3)	020616	010600			MOV	SP,RO
(4)	020620	104415			TRAP	( \$PNTX
(4)	020622	062706	000014		ADD	#14,SP
6661	020626			ENDMSG		
(3)	020626					
(3)	020626	104423			L10011:	TRAP ( \$MSG
6662						
6663						
6664						
6665						
6666						
6667	020630			BGNMSG ERR9		
(3)	020630				ERR9::	
6668	020630			PRINTB #FMT1,#ADDRES,MPCSR		
(9)	020630	013746	002446		MOV	MPCSR,-(SP)
(8)	020634	012746	035620		MOV	#ADDRES,-(SP)
(7)	020640	012746	011454		MOV	#FMT1,-(SP)
(6)	020644	012746	000003		MOV	#3,-(SP)
(3)	020650	010600			MOV	SP,RO
(4)	020652	104414			TRAP	( \$PNTB
(4)	020654	062706	000010		ADD	#10,SP
6669	020660			PRINTB #FMT2		
(7)	020660	012746	011464		MOV	#FMT2,-(SP)
(6)	020664	012746	000001		MOV	#1,-(SP)
(3)	020670	010600			MOV	SP,RO
(4)	020672	104414			TRAP	( \$PNTB
(4)	020674	062706	000004		ADD	#4,SP
6670	020700			PRINTB #FMT7,#DH1,REGNUM		
(9)	020700	013746	002400		MOV	REGNUM,-(SP)
(8)	020704	012746	014264		MOV	#DH1,-(SP)
(7)	020710	012746	011646		MOV	#FMT7,-(SP)
(6)	020714	012746	000003		MOV	#3,-(SP)
(3)	020720	010600			MOV	SP,RO
(4)	020722	104414			TRAP	( \$PNTB
(4)	020724	062706	000010		ADD	#10,SP
6671	020730			PRINTX #FMT4,#DH2,#DH3		
(9)	020730	012746	014335		MOV	#DH3,-(SP)
(8)	020734	012746	014306		MOV	#DH2,-(SP)
(7)	020740	012746	011550		MOV	#FMT4,-(SP)
(6)	020744	012746	000003		MOV	#3,-(SP)
(3)	020750	010600			MOV	SP,RO
(4)	020752	104415			TRAP	( \$PNTX
(4)	020754	062706	000010		ADD	#10,SP
6672	020760			PRINTX #FMT5,LUR10,LUR11,LUR12,LUR13		
(11)	020760	013746	002310		MOV	LUR13,-(SP)
(10)	020764	013746	002306		MOV	LUR12,-(SP)
(9)	020770	013746	002304		MOV	LUR11,-(SP)

(8) 020774 013746 002302  
(7) 021000 012746 011563  
(6) 021004 012746 000005  
(3) 021010 010600  
(4) 021012 104415  
(4) 021014 062706 000014  
6673 021020  
(8) 021020 012746 014373  
(7) 021024 012746 011712  
(6) 021030 012746 000002  
(3) 021034 010600  
(4) 021036 104415  
(4) 021040 062706 000006  
6674 021044  
(11) 021044 013746 002320  
(10) 021050 013746 002316  
(9) 021054 013746 002314  
(8) 021060 013746 002312  
(7) 021064 012746 011613  
(6) 021070 012746 000005  
(3) 021074 010600  
(4) 021076 104415  
(4) 021100 062706 000014  
6675 021104  
(3) 021104  
(3) 021104 104423  
6676  
6677  
6678  
6679  
6680

PRINTX #FMT9,#DH4

PRINTX #FMT6,LUR14,LUR15,LUR16,LUR17

ENDMSG

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

6682  
6683  
6684  
6685  
6686  
6687  
6688  
6689  
6690  
(3)  
6691  
6692  
(3)  
(3)  
6693  
6694  
6695  
6696  
6697

021106  
021106  
021106  
021106  
021106 104425

.SBTTL REPORT CODING SECTION

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

BGNRPT

ENDRPT

LSRPT::

L10013:

TRAP CSRPT



6699  
6700  
6701  
6702  
6703  
6704  
6705  
6706  
6707  
6708  
6709  
6710  
6711  
6712  
6713  
6714  
6715

021110  
(3) 021110  
021110 177777  
021112 177777  
021114 177777  
021116

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

6717  
6718  
6719  
6720  
6721  
6722  
6723  
6724  
(3)  
6725  
6726  
6727  
6728  
6729  
6730  
6731  
6732  
6733  
6734  
6735  
6736  
6737  
6738  
6739  
6740  
6741  
(3)  
(3)  
6742  
(2)  
6743  
6744  
(3)  
(3)  
6745  
(2)  
6746  
6747  
(3)  
(3)  
6748  
(2)  
6749  
6750  
(3)  
(3)  
6751  
(2)  
6752  
6753  
6754  
6755  
6756  
6757  
6758  
6759

021116  
021116  
021116 010637 002346  
021122 005037 002352  
021126 005037 002426  
021132 005037 002430  
021136 005037 002420  
021142 005037 002432  
021146 005737 002412  
021152 001007  
021154 013737 000004 002414  
021162 013737 000006 002416  
021170 000406  
021172 013737 002414 000004 6737  
021200 013737 002416 000006 6738  
021206 012737 000001 002412 6739  
021214  
021214 012700 000040  
021220 104447  
021222  
021222 103415  
021224  
021224 012700 000037  
021230 104447  
021232  
021232 103411  
021234  
021234 012700 000035  
021240 104447  
021242  
021242 103411  
021244  
021244 012700 000036  
021250 104447  
021252  
021252 103504  
021254 000414  
021256  
021256 005037 002444  
021262 005037 002434  
021266  
021266 012737 177777 002344  
021274 005237 002444

```
.SBTTL INITIALIZE SECTION
://////
:/ THE INITIALIZE SECTION CONTAINS THE CODING THAT IS PERFORMED
:/ AT THE BEGINNING OF THE TEST SEQUENCE ON THE NEXT UNIT.
://////

BGNINIT
LSINIT::

MOV SP,PSTACK ;SAVE BASE-LEVEL STACK POINTER
CLR SUBRPC ;CLEAR SUBR CALL PC
CLR DISILO ;CLEAR CURRENT STATE OF DISSI
CLR CHPTYP ;CLEAR USYRT CHIP TYPE INDICATOR
CLR ERROR1 ;CLEAR ERROR FLAG
CLR SAVLEN ;CLEAR CHAR LENGTH FROM SETUP
TST FRSTIM ;SEE IF FIRST TIME THROUGH AFTER LOAD
BNE 6$ ;BR IF NOT
MOV @#4,SAVE4 ;SAVE ERROR TRAP VECTOR
MOV @#6,SAVE6
BR 9$
6$: MOV SAVE4,@#4 ;RESTORE ERROR TRAP VECTOR
MOV SAVE6,@#6
9$: MOV #1,FRSTIM ;MARK FLAG FOR NEXT TIME THROUGH
;SEE IF PROGRAM JUST STARTED, BR IF YES
REDEF #EF.START
MOV #EF.START,R0
TRAP C$REFG
BCS STARST
;SEE IF PROGRAM JUST RESTARTED, BR IF YES
REDEF #EF.RESTART
MOV #EF.RESTART,R0
TRAP C$REFG
BCS STARST
;SEE IF THIS IS A NEW PASS, BR IF YES
REDEF #EF.NEW
MOV #EF.NEW,R0
TRAP C$REFG
BCS NEWST
;SEE IF PROGRAM WAS JUST CONTINUED
REDEF #EF.CONTINUE
MOV #EF.CONTINUE,R0
TRAP C$REFG
BCS ENDIT
BR GETPRM
STARST: CLR STARES ;CLEAR FLAG TO SHOW JUST HAD STA OR RES
;CLEAR DEVICE MAP
CLR DEVMAP
NEWST: MOV #-1,LOGDEV ;RESET LOGICAL DEVICE TO -1
INC STARES ;INCR NO. OF PASSES SINCE STA OR RES
```



6798  
6799  
6800  
6801  
6802  
6803  
6804  
(3)  
6805  
6806  
(3)  
(3)  
6807  
6808  
6809  
6810  
6811  
6812  
6813  
(3)  
(3)  
6814  
6815  
6816  
(3)  
(3)  
6817  
6818  
6819  
6820  
6821

021466  
021466  
021466 012700 000340  
021472 104441  
021474 012737 021516 000004  
021502 012737 000340 000006  
021510 005777 160732  
021514 000405  
021516 062706 000004  
021522  
021522 013700 002344  
021526 104451  
021530 013737 002414 000004  
021536 013737 002416 000006  
021544  
021544 104461

.SBTTL AUTO DROP UNIT SECTION

:/ THE AUTO DROP CODING DETERMINES WHETHER OR NOT THE DEVICE WHOSE P-TABLE  
:/ WAS JUST OBTAINED IS READY FOR TESTING, AND IT IS DROPPED IF NOT READY.

BGNAUTO

L\$AUTO::

: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  
BR 9\$

:ADDRESS SELO  
:TAKE THIS BR IF DEVICE RESPONDS

:COME HERE IF DEVICE CSR IS NON-EXISTENT

6\$: ADD #4,SP  
DODU LOGDEV

:CLEAN UP THE STACK POINTER  
:DROP THIS UNIT FROM TESTING

MOV LOGDEV,R0  
TRAP C\$DODU

9\$: MOV SAVE4,@#4  
MOV SAVE6,@#6  
ENDAUTO

:RESTORE ERROR TRAP VECTOR

L10016:

TRAP C\$AUTO

6823  
6824  
6825  
6826  
6827  
6828  
6829  
6830  
(3)  
6831  
6832  
6833  
(3)  
(3)  
6834  
6835  
6836  
6837  
6838

021546  
021546  
  
021546  
021546  
021546 104412

.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.  
:/

BGNCLN

ENDCLN

L\$CLEAN::

L10017: TRAP C\$CLEAN

6840  
6841  
6842  
6843  
6844  
6845  
6846  
6847  
(3)  
6848  
6849  
(3)  
6850  
6851  
(8)  
(7)  
(6)  
(3)  
(4)  
(4)  
6852  
(3)  
(3)  
6853  
6854  
021606  
021614  
021622  
021630  
6855  
6856  
6857  
6858  
6859  
6860

021550  
021550  
021550  
021550 104433  
021552  
021552 013746 002344  
021556 012746 021600  
021562 012746 000002  
021566 010600  
021570 104417  
021572 062706 000006  
021576  
021576 104453  
021600 047045 040445 047125  
021606 052111 022440 031104  
021614 040445 042040 047522  
021622 050120 042105 047045  
000  
021632

.SBTTL DROP UNIT SECTION

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

BGNDU

;ISSUE UNIBUS RESET TO CLEAN UP  
BRESET

;PRINT 'UNIT XX DROPPED'  
PRINTF #FMT27,LOGDEV

ENDDU

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

.EVEN

L\$DU::

TRAP C\$RESET  
MOV LOGDEV, -(SP)  
MOV #FMT27, -(SP)  
MOV #2, -(SP)  
MOV SP, R0  
TRAP C\$PRINTF  
ADD #6, SP

L↑0020:

TRAP C\$DU

6862  
6863  
6864  
6865  
6866  
6867  
6868  
6869  
6870  
(3)  
6871  
(3)  
(3)  
6872  
6873  
6874  
6875  
6876  
6877

021632  
021632  
021632  
021632  
021632 104452

.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.  
:/

BGNAU

ENDAU

LSAU::

L10021:

TRAP CSAU

.SBTTL HARDWARE TESTS

6879  
6880  
6881  
6882  
6883  
6884  
6885  
6886  
6887  
6888  
6889  
6890  
6891  
6892  
6893  
6894  
6895  
6896  
6897  
6898  
6899  
6900  
6901  
6902  
6903  
6904  
6905  
6906  
6907  
6908  
6909  
6910  
6911

021634  
(3) 021634  
021634 012700 000340  
(3) 021634 104441  
(3) 021640 012737 021664 000004  
(3) 021640 012737 000340 000006  
(3) 021642 012737 021664 000004  
(3) 021650 012737 000340 000006  
(3) 021656 005777 160564  
(3) 021662 000406  
021664 062706 000004  
68: 021670 104455  
(4) 021670 000001  
(5) 021672 012125  
(5) 021674 014600  
(5) 021676 013737 002414 000004  
021700 013737 002416 000006  
021714  
(3) 021714  
(3) 021714 104401

\*\*\*\*\*  
: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$SPRI
                                :SET UP NON-EXISTENT MEMORY ERROR TRAP VECTOR
68:  MOV #6$,@#4
    MOV #PRI07,@#6
    TST @MPCSR
    BR 98
                                :ADDRESS SELO
                                :TAKE THIS BR IF DEVICE RESPONDS
:COME HERE IF DEVICE CSR IS NON-EXISTENT
68:  ADD #4,SP
                                :CLEAN UP THE STACK POINTER
:REPORT CSR ADDRESS TIME-OUT
  ERDF 1,EM1,ERR1
                                TRAP C$ERDF
                                .WORD 1
                                .WORD EM1
                                .WORD ERR1
98:  MOV SAVE4,@#4
    MOV SAVE6,@#6
                                :RESTORE ERROR TRAP VECTOR
ENDTST
                                L1002:
                                TRAP C$ETST
```

\*\*\*\*\*  
:SBTTL 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.  
\*\*\*\*\*

6912  
6913  
6914  
6915  
6916  
6917  
6918  
6919  
6920  
6921  
6922  
6923  
6924

```
BGNTST
T2::
MOV #14,REGNUM
JSR PC,MSTCLR
JSR PC,READLU
CMPB REDBYT,PATM+4
BEQ 6$
CLR GOODAT
MOV# PATM+4,GOODAT
                                :SET LU REG NO. = 14
                                :ISSUE MASTER CLEAR
                                :READ REG 14
                                :CHK FOR INITIALIZED STATE
                                :BR IF YES
                                :SET EXPECTED REG CONTENTS = 000
                                :SET EXPECTED DATA
```



```
6925 021756 013737 002364 002406      MOV   REDBYT,BADDAT   ;SET ACTUAL REG CONTENTS
6926 021764 004737 004016      JSR   PC,GETREG      ;GET REGS FOR PRINTOUT
6927                               ;REPORT REG NOT CLEARED BY MASTER CLEAR
6928                               ERRDF  2,EM2,ERR2
(4) 021770 104455                               TRAP  C$ERDF
(5) 021772 000002                               .WORD 2
(5) 021774 012161                               .WORD EM2
(5) 021776 014632                               .WORD ERR2
6929 022000                               6$:
6930 022000                               ENDTST
(3) 022000                               L10023:
(3) 022000 104401                               TRAP  C$ETST
```

6931  
6932  
6933  
6934  
6935  
6936  
6937  
6938  
6939  
6940  
6941  
6942  
6943  
6944

```
.....
SBTTL      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.
.....
BGNST
```

```
6945 022002                               T3::
(3) 022002                               JSR   PC,MSTCLR      ;ISSUE MASTER CLEAR
6946 022002 004737 003576      MOV   #14,REGNUM    ;SET LU REG NO. = 14
6947 022006 012737 000014 002400      MOV   #PATA,R1      ;GET POINTER TO DATA PAT IN R1
6948 022014 012701 002571      3$:
6949 022020                               BGNSEG
6950 022020                               TRAP  C$BSEG
(3) 022020 104404
6951 022022 111137 002366      MOVB  (R1),WRIBYT   ;GET A BYTE OF PAT A
6952 022026 143737 002560 002366      BICB  R14NRW,WRIBYT ;MASK OFF NON-READ/WRITE BITS
6953 022034 004737 003750      JSR   PC,WRITLU     ;WRITE DATA BYTE INTO REG 14
6954 022040 004737 003672      JSR   PC,READLU     ;READ DATA BYTE FROM REG 14
6955 022044 143737 002560 002364      BICB  R14NRW,REDBYT ;MASK OFF NON-READ/WRITE BITS
6956 022052 123737 002364 002366      CMPB  REDBYT,WRIBYT ;COMPARE BYTE READ TO BYTE WRITTEN
6957 022060 001414      BEQ   6$           ;BR IF BYTES MATCH
6958 022062 013737 002366 002404      MOV   WRIBYT,GOODAT ;SET EXPECTED REG CONTENTS
6959 022070 013737 002364 002406      MOV   REDBYT,BADDAT ;SET ACTUAL REG CONTENTS
6960 022076 004737 004016      JSR   PC,GETREG     ;GET REGS FOR PRINTOUT
6961                               ;REPORT LINE UNIT REG MISCOMPARE
6962                               ERRDF  3,EM3,ERR2
(4) 022102 104455                               TRAP  C$ERDF
(5) 022104 000003                               .WORD 3
(5) 022106 012220                               .WORD EM3
(5) 022110 014632                               .WORD ERR2
6963 022112                               6$:
6964 022112                               ENDSEG
(3) 022112                               10000$:
(3) 022112 104405                               TRAP  C$ESEG
6965 022114 005201      INC   R1           ;INCREMENT DATA PATTERN POINTER
6966 022116 020127 002615      CMP   R1,#PATB    ;SEE IF ALL WORDS OF PATTERN A USED YET
```

6967 022122 103736  
6968 022124  
(3) 022124  
(3) 022124 104401

BLO 38 ;BR IF NOT DONE YET  
ENDTST

L10024: TRAP C\$ETST

6969  
6970  
6971  
6972  
6973  
6974  
6975  
6976  
6977  
6978

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

6979 022126  
(3) 022126  
6980 022126 004737 003576  
6981 022132 012737 000014 002400  
6982 022140 112737 000377 002366  
6983 022146 004737 003750  
6984 022152 004737 003576  
6985 022156 004737 003672  
6986 022162 123737 002364 003026  
6987 022170 001416  
6988 022172 005037 002404  
6989 022176 113737 003026 002404  
6990 022204 013737 002364 002406  
6991 022212 004737 004016

EGNTST  
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 68 ;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

T4::  
TRAP C\$ERDF  
.WORD 2  
.WORD EM2  
.WORD ERR2

6992  
6993 022216  
(4) 022216 104455  
(5) 022220 000002  
(5) 022222 012161  
(5) 022224 014632  
6994 022226  
6995 022226  
(3) 022226  
(3) 022226 104401

68:  
ENDTST

L10025: TRAP C\$ETST

6996  
6997  
6998  
6999  
7000  
7001  
7002  
7003  
7004  
7005

.....  
: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.  
:.....

7006 022230  
(3) 022230  
7007 022230 004737 003576  
7008 022234 012737 000014 002400  
7009 022242 112737 000377 002366  
7010 022250 004737 003750  
7011 022254  
(3) 022254 104433

EGNTST  
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)

T5::  
TRAP C\$RESET

```

7012 022256 142777 000200 160164      BICB  #RUN,@BSEL1      :CLEAR RUN BIT
7013 022264 012701 000024      MOV   #20.,R1          :INIT LOOP COUNTER
7014 022270 000240      2$:  NOP
7015 022272 005301      DEC   R1               :DECR COUNTER
7016 022274 001375      BNE   2$               :BR TO STALL
7017 022276 004737 003672      JSR   PC,READLU       :READ REG 14
7018 022302 142737 000100 002364      BICB  #TXEN,REDBYT     :CLEAR UNPREDICTABLE BIT
7019 022310 123737 002364 003026      CMPB  REDBYT,PATM+4    :CHK FOR INIT'D STATE
7020 022316 001416      BEQ   6$               :BR IF REG GOT CLEARED
7021 022320 005037 002404      CLR   GOODAT          :SET EXPECTED DATA
7022 022324 113737 003026 002404      MOVB  PATM+4,GOODAT
7023 022332 013737 002364 002406      MOV   REDBYT,BADDAT   :SET ACTUAL DATA
7024 022340 004737 004016      JSR   PC,GETREG       :GET REGS FOR PRINTOUT
7025      :REPORT REG NOT CLEARED BY UNIBUS RESET (INIT)
7026 022344      ERRDF 4,EM4,ERR2
(4) 022344 104455      TRAP  C$ERDF
(5) 022346 000004      .WORD 4
(5) 022350 012237      .WORD EM4
(5) 022352 014632      .WORD ERR2
7027 022354      4$:  ENDTST
(3) 022354      L10026: TRAP  C$ETST
(3) 022354 104401
7029
7030
7031
7032
7033
7034
7035
7036
7037
7038
7039
7040
7041
7042
7043
7044 022356
(3) 022356
7045 022356 004737 003576      JSR   PC,MSTCLR       :ISSUE A MASTER CLEAR
7046 022362 012737 000014 002400      MOV   #14,REGNUM      :SET LU REG NO. = 14
7047 022370 013701 002400      MOV   REGNUM,R1       :SET DESTINATION = OBUS* REG 14
7048 022374 052701 000100      BIS   #100,R1         :SET SOURCE = BSEL4
7049 022400 052701 121000      BIS   #MVIXOX,R1     :SET REST OF MOVE INSTRUCTION
7050 022404 010137 022422      MOV   R1,2$          :SET INSTRUCTION AS SUBROUTINE ARGUMENT
7051 022410 112777 000041 160036      MOVB  #041,@BSEL4    :SET DATA BYTE = 041
7052 022416 004737 003540      JSR   PC,STPCLK       :EXECUTE MOVE INSTRUCTION
7053 022422 000000      2$:  .WORD 0           :INSTRUCTION GOES HERE
7054 022424 004737 003672      JSR   PC,READLU       :READ LU REG 14
7055 022430 123737 002364 003026      CMPB  REDBYT,PATM+4   :CHECK FOR LU REG 14 UNCHANGED
7056 022436 001416      BEQ   4$               :BR IF LU REG 14 UNCHANGED
7057 022440 005037 002404      CLR   GOODAT          :SET EXPECTED DATA
7058 022444 113737 003026 002404      MOVB  PATM+4,GOODAT
7059 022452 013737 002364 002406      MOV   REDBYT,BADDAT   :SET ACTUAL DATA
7060 022460 004737 004016      JSR   PC,GETREG       :GET REGS FOR PRINTOUT
  
```

```

.....
: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::
TRAP  C$ETST
  
```

7061  
7062 022464  
(4) 022464 104455  
(5) 022466 000003  
(5) 022470 012220  
(5) 022472 014632  
7063 022474  
7064 022474  
(3) 022474  
(3) 022474 104401  
7065  
7066  
7067  
7068  
7069  
7070  
7071  
7072  
7073  
7074  
7075  
7076  
7077  
7078  
7079  
7080  
7081 022476  
(3) 022476  
7082 022476 004737 003576  
7083 022502 012737 000010 002400  
7084 022510 012701 002644  
7085 022514 112137 002366 28:  
7086 022520 004737 003750  
7087 022524 005237 002400  
7088 022530 020127 002654  
7089 022534 103767  
7090 022536 004737 003576  
7091 022542 012737 000010 002400  
7092 022550 012702 003022  
7093 022554 012701 002550  
7094 022560  
7095 022560 004737 003672  
7096 022564 142137 002364  
7097 022570 123712 002364  
7098 022574 001417  
7099 022576 005037 002404  
7100 022602 111237 002404  
7101 022606 013737 002364 002406  
7102 022614 004737 004016  
7103  
7104 022620  
(4) 022620 104455  
(5) 022622 000002  
(5) 022624 012161  
(5) 022626 014632  
7105 022630

:REPORT REGISTER MISCOMPARE  
ERRDF 3,EM3,ERR2

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

48:  
ENDTST

L10027:  
TRAP C\$ETST

.....  
: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  
:.....

:BONTST

T7::

JSR PC,MSTCLR :ISSUE MASTER CLEAR  
MOV #10,REGNUM :INIT REG NO. TO 10  
MOV #PATG,R1 :INIT DATA PATTERN POINTER  
28: 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  
28: BLO 28 :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  
38: 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 68 :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

```

(3) 022630 104410
(3) 022632 000016
7106 022634 005237 002400
7107 022640 005202
7108 022642 020227 003032
7109 022646 103744
7110 022650
(3) 022650
(3) 022650 104401
7111
7112
7113
7114
7115
7116
7117
7118
7119
7120
7121
7122
7123
7124
7125 022652
(3) 022652
7126 022652 004737 003576
7127 022656 012701 002500
7128 022662 012702 003022
7129 022666 012703 000010
7130 022672 112221
7131 022674 005303
7132 022676 001375
7133 022700 005001
7134 022702 010137 002400
7135 022706 062737 000010 002400
7136 022714 116137 002615 002366
7137 022722 113761 002366 002500
7138 022730 146161 002550 002500
7139 022736 004737 003750
7140 022742 005003
7141 022744 010337 002400
7142 022750 062737 000010 002400
7143 022756 004737 003672
7144 022762 146337 002550 002364
7145 022770 023727 002400 000011
7146 022776 001006
7147 023000 142737 000020 002364
7148 023006 142763 000020 002500
7149 023014 123763 002364 002500
7150 023022 001420
7151 023024 005037 002404
7152 023030 116337 002500 002404
7153 023036 013737 002364 002406
7154 023044 004737 004016
7155
7156 023050
  
```

```

68: INC REGNUM ; INCREMENT REG NO.
      INC R2 ; INCR DATA PATTERN POINTER
      CMP R2,#PATN ; SEE IF ALL DONE YET
      BLO 38 ; BR IF NOT DONE YET
      ENDTST
  
```

L10030: TRAP C\$ETST

```

.....
: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
:
:.....
  
```

```

:BGNTST
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
38: MOV (R2)+,(R1)+ ; LOAD BYTE OF PATRN INTO EXPECTED DATA AREA
      DEC R3 ; DECR COUNTER
      BNE 38 ; BR IF NOT DONE LOADING YET
      CLR R1 ; INIT DATA PATTERN INDEX FOR WRITING
68: MOV R1,REGNUM ; GET REG NO. FOR WRITING
      ADD #10,REGNUM ; SET DATA BYTE TO BE WRITTEN
      MOV PATB(R1),WRIBYT ; SET EXPECTED DATA FOR READ
      MOV WRIBYT,REDDAT(R1) ; MASK OUT UNPREDICTABLE BITS
      BICB UPBITS(R1),REDDAT(R1) ; WRITE DATA BYTE INTO REG
      JSR PC,WRITLU ; INIT DATA PAT INDEX FOR READS
      CLR R3
98: MOV R3,REGNUM ; GET REG NO. FOR READING
      ADD #10,REGNUM ; READ A LINE UNIT REG
      JSR PC,READLU ; MASK OUT UNPREDICTABLE BITS
      BICB UPBITS(R3),REDBYT ; SEE IF READING REG 11
      CMP REGNUM,#11 ; BR IF NOT
      BNE 108 ; MASK ORDY BIT IN ACTUAL BYTE
      BICB #ORDY,REDBYT ; MASK ORDY BIT IN EXPECTED BYTE
      BICB #ORDY,REDDAT(R3) ; COMPARE BYTE READ TO EXPECTED
108: CMPB REDBYT,REDDAT(R3) ; BR IF DATA MATCHES
      BEQ 128
      CLR GOODAT
      MOV REDDAT(R3),GOODAT ; SET EXPECTED DATA
      MOV REDBYT,BADDAT ; SET ACTUAL DATA
      JSR PC,GETREG ; READ AND STORE REGS 10-17 FOR PRINTOUT
:REPORT REG MISCMPARE
      ERRDF 3,EM3,ERR2
  
```

```

(4) 023050 104455 TRAP C$ERDF
(5) 023052 000003 .WORD 3
(5) 023054 012220 .WORD EM3
(5) 023056 014632 .WORD ERR2
7157 023060 ESCAPE TST
(3) 023060 104410 TRAP C$ESCAPE
(3) 023062 000022 .WORD L10031-
7158 023064 005203 12$: INC R3 ;INCREMENT DATA PATTERN INDEX FOR READING
7159 023066 020327 000010 CMP R3,#10 ;SEE IF ALL REGS READ YET
7160 023072 002724 BLT 9$ ;BR IF NOT
7161 023074 005201 INC R1 ;INCREMENT DATA PAT INDEX FOR WRITING
7162 023076 020127 000010 CMP R1,#10 ;SEE IF ALL REGS WRITTEN YET
7163 023102 002677 BLT 6$ ;BR IF NOT
7164 023104
(3) 023104
(3) 023104 104401 L10031: TRAP C$ETST
7165
7166
7167
7168
7169
7170
7171
7172
7173
7174
7175

```

```

.....
:SBTTL 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.
:
.....

```

```

7176 023106 BGNSTST
(3) 023106 T9::
7177 023106 004737 003576 JSR PC,MSTCLR ;ISSUE MASTER CLEAR
7178 023112 012737 000011 002400 MOV #11,REGNUM ;SET LU REG NO. = 11
7179 023120 012701 002625 MOV #PATC,R1 ;GET POINTER TO DATA PAT IN R1
7180 023124 3$:
7181 023124 BGNSEG TRAP C$BSEG
(3) 023124 104404
7182 023126 111137 002366 MOVB (R1),WRIBYT ;GET A BYTE OF PAT C
7183 023132 004737 003750 JSR PC,WRITLU ;WRITE DATA BYTE INTO REG 11
7184 023136 004737 003672 JSR PC,READLU ;READ DATA BYTE FROM REG 11
7185 023142 143737 002551 002364 BICB UPBITS+1,REDBYT ;MASK OUT UNPREDICTABLE BITS
7186 023150 123737 002364 002366 CMPB REDBYT,WRIBYT ;COMPARE BYTE READ TO BYTE WRITTEN
7187 023156 001414 BEQ 6$ ;BR IF BYTES MATCH
7188 023160 013737 002366 002404 MOV WRIBYT,GOODAT ;SET EXPECTED REG CONTENTS
7189 023166 013737 002364 002406 MOV REDBYT,BADDAT ;SET ACTUAL REG CONTENTS
7190 023174 004737 004016 JSR PC,GETREG ;GET REGS FOR PRINTOUT
7191 :REPORT LINE UNIT REG MISCOMPARE
7192 023200 ERRDF 3,EM3,ERR2
(4) 023200 104455 TRAP C$ERDF
(5) 023202 000003 .WORD 3
(5) 023204 012220 .WORD EM3
(5) 023206 014632 .WORD ERR2
7193 023210 6$:
7194 023210 ENDSEG
(3) 023210 10000$: TRAP C$ESEG
(3) 023210 104405
7195 023212 005201 INC R1 ;INCREMENT DATA PATTERN POINTER
7196 023214 020127 002630 CMP R1,#PATD ;SEE IF ALL WORDS OF PATTERN C USED YET

```

7197 023220 103741 BLD 3\$ ;BR IF NOT DONE YET  
7198 023222  
(3) 023222  
(3) 023222 104401 L10032: TRAP C\$ETST  
7199

7200  
7201  
7202  
7203

.....  
: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.  
:.....

7204  
7205  
7206  
7207  
7208  
7209

7210 023224  
(3) 023224

BGNST T10::

7211 023224 004737 003576  
7212 023230 012737 000012 002400  
7213 023236 012701 002630  
7214 023242  
7215 023242  
(3) 023242 104404  
7216 023244 111137 002366  
7217 023250 004737 003750  
7218 023254 004737 003672  
7219 023260 143737 002552 002364  
7220 023266 123737 002364 002366  
7221 023274 001414  
7222 023276 013737 002366 002404  
7223 023304 013737 002364 002406  
7224 023312 004737 004016

JSR PC,MSTCLR ;ISSUE MASTER CLEAR  
MOV #12,REGNUM ;SET LU REG NO. = 12  
MOV #PATD,R1 ;GET POINTER TO DATA PAT IN R1  
3\$:  
BGNSEG TRAP C\$BSEG  
MOVB (R1),WRIBYT ;GET A BYTE OF PAT D  
JSR PC,WRITLU ;WRITE DATA BYTE INTO REG 12  
JSR PC,READLU ;READ DATA BYTE FROM REG 12  
BICB UPBITS+2,REDBYT ;MASK OUT UNPREDICTABLE BITS  
CMPB REDBYT,WRIBYT ;COMPARE BYTE READ TO BYTE WRITTEN  
BEQ 6\$ ;BR IF BYTES MATCH  
MOV WRIBYT,GOODAT ;SET EXPECTED REG CONTENTS  
MOV REDBYT,BADDAT ;SET ACTUAL REG CONTENTS  
JSR PC,GETREG ;GET REGS FOR PRINTOUT

7225  
7226 023316  
(4) 023316 104455  
(5) 023320 000003  
(5) 023322 012220  
(5) 023324 014632

:REPORT LINE UNIT REG MISCOMPARE  
ERRDF 3,EM3,ERR2 TRAP C\$ERDF  
3  
.WORD EM3  
.WORD ERR2

7227 023326  
7228 023326  
(3) 023326

6\$:  
ENDSEG 10000\$: TRAP C\$ESEG

(3) 023326 104405  
7229 023330 005201  
7230 023332 020127 002633  
7231 023336 103741

INC R1 ;INCREMENT DATA PATTERN POINTER  
CMP R1,#PATE ;SEE IF ALL WORDS OF PATTERN D USED YET  
BLD 3\$ ;BR IF NOT DONE YET

7232 023340  
(3) 023340  
(3) 023340 104401

ENDTST L10033: TRAP C\$ETST

7233  
7234  
7235  
7236  
7237

.....  
:SBTTL TEST 11 - REG 13 READ/WRITE BIT TEST  
:.....

7238  
7239  
7240

```
7241      ;* WRITE, READ, AND COMPARE EACH WORD OF DATA PATTERN E INTO REG 13 :
7242      ;* DATA PATTERN E = 000,120,020,100,120,000.
7243      ;*****
7244      BGNTST
(3)
7245      023342 004737 003576      JSR      PC,MSTCLR      ;ISSUE MASTER CLEAR      T11::
7246      023346 012737 000013 002400  MOV      #13,REGNUM    ;SET LU REG NO. = 13
7247      023354 012701 002633      MOV      #PATE,R1      ;GET POINTER TO DATA PAT IN R1
7248      023360      3$:
7249      023360      BGNSEG
(3)
7250      023360 104404      TRAP      C$BSEG
7251      023362 111137 002366      MOVB     (R1),WRIBYT    ;GET A BYTE OF PAT E
7252      023366 004737 003750      JSR      PC,WRITLU     ;WRITE DATA BYTE INTO REG 13
7253      023372 004737 003672      JSR      PC,READLU     ;READ DATA BYTE FROM REG 13
7254      023404 143737 002553 002364  BICB     UPBITS+3,REDBYT ;MASK OUT UNPREDICTABLE BITS
7255      023412 001414 002366 002366  CMPB     REDBYT,WRIBYT  ;COMPARE BYTE READ TO BYTE WRITTEN
7256      023414 013737 002366 002404  BEQ      6$            ;BR IF BYTES MATCH
7257      023422 013737 002364 002406  MOV      WRIBYT,GOODAT  ;SET EXPECTED REG CONTENTS
7258      023430 004737 004016      MOV      REDBYT,BADDAT  ;SET ACTUAL REG CONTENTS
7259      ;REPORT LINE UNIT REG MISCOMPARE
7260      023434      JSR      PC,GETREG     ;GET REGS FOR PRINTOUT
(4)      ERRDF 3,EM3,ERR2
(5)
(5)
(5)
7261      023444      6$:
7262      023444      ENDSEG
(3)
(3)
7263      023444 104405      TRAP      C$ESEG
7264      023446 005201      INC      R1            ;INCREMENT DATA PATTERN POINTER
7265      023450 020127 002641      CMP      R1,#PATF     ;SEE IF ALL WORDS OF PATTERN E USED YET
7266      023454 103741      BLO     3$            ;BR IF NOT DONE YET
(3)
(3)
7267      023456 104401      TRAP      C$ETST
7268
7269
7270
7271
7272      ;*****
7273      ;SBITL TEST 12 - REG 17 READ/WRITE BIT TEST
7274      ;*****
7275      ;* WRITE, READ, AND COMPARE EACH WORD OF DATA PATTERN F INTO REG 17 :
7276      ;* DATA PATTERN F = 050,051,050.
7277      ;*****
7278      BGNTST
(3)
7279      023460 004737 003576      JSR      PC,MSTCLR     ;ISSUE MASTER CLEAR      T12::
7280      023464 012737 000017 002400  MOV      #17,REGNUM    ;SET LU REG NO. = 17
7281      023472 012701 002641      MOV      #PATF,R1      ;GET POINTER TO DATA PAT IN R1
7282      023476      3$:
7283      023476      BGNSEG
(3)
7284      023476 104404      TRAP      C$BSEG
7284      023500 111137 002366      MOVB     (R1),WRIBYT    ;GET A BYTE OF PAT F
```



```
7285 023504 004737 003750 JSR PC,WRITLU ;WRITE DATA BYTE INTO REG 17
7286 023510 004737 003672 JSR PC,READLU ;READ DATA BYTE FROM REG 17
7287 023514 143737 002557 002364 BICB UPBITS+7,REDBYT ;MASK OUT UNPREDICTABLE BITS
7288 023522 123737 002364 002366 CMPB REDBYT,WRIBYT ;COMPARE BYTE READ TO BYTE WRITTEN
7289 023530 001414 BEQ 6$ ;BR IF BYTES MATCH
7290 023532 013737 002366 002404 MOV WRIBYT,GOODAT ;SET EXPECTED REG CONTENTS
7291 023540 013737 002364 002406 MOV REDBYT,BADDAT ;SET ACTUAL REG CONTENTS
7292 023546 004737 004016 JSR PC,GETREG ;GET REGS FOR PRINTOUT
7293 ;REPORT LINE UNIT REG MISCOMPARE
7294 023552 ERRDF 3,EM3,ERR2
(4) 023552 104455 TRAP C$ERDF
(5) 023554 000003 .WORD 3
(5) 023556 012220 .WORD EM3
(5) 023560 014632 .WORD ERR2
7295 023562 6$:
7296 023562 ENDSEG
(3) 023562 10000$: TRAP C$ESEG
(3) 023562 104405
7297 023564 005201 INC R1 ;INCREMENT DATA PATTERN POINTER
7298 023566 020127 002644 CMP R1,#PATG ;SEE IF ALL WORDS OF PATTERN F USED YET
7299 023572 103741 BLO 3$ ;BR IF NOT DONE YET
7300 023574 ENDTST
(3) 023574 L10035: TRAP C$ETST
(3) 023574 104401
```

```
.....
: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 M8207 RUN SWITCH (E28 SW7)
:* IS OFF, THE TEST WILL BE SKIPPED.
:.....
```

```
7327 023576 BGNST
(3) 023576 T13::
7328 023576 004737 003576 JSR PC,MSTCLR ;PERFORM MASTER CLEAR
7329 023602 012737 000015 002442 MOV #13, TSTNUM ;SET TEST NO.
7330 023610 005737 002476 TST RUNINH ;SEE IF RUN SWITCH IS SET
7331 023614 001020 BNE 1$ ;BR IF YES, TO RUN TEST
```

```

7332 023616 023727 002444 000001      CMP      STARES,#1      ;SEE IF THIS IS FIRST PASS SINCE STA OR RES
7333 023624 001012                      BNE      40$           ;BR IF NOT, TO SKIP PRINTING
7334 023626                      PRINTF   #FMT19,TSTNUM ;PRINT MSG TO SAY TEST NOT RUN
(8) 023626 013746 002442                      MOV      TSTNUM,-(SP)
(7) 023632 012746 012007                      MOV      #FMT19,-(SP)
(6) 023636 012746 000002                      MOV      #2,-(SP)
(3) 023642 010600                      MOV      SP,R0
(4) 023644 104417                      TRAP    C$PNTF
(4) 023646 062706 000006                      ADD     #6,SP
7335 023652 000137 024220      40$:    JMP      A1           ;GO TO SKIP TEST
7336 023656 012737 000017 002400 1$:    MOV     #17,REGNUM ;SET REG NO. = 17
7337 023664 012701 000360      MOV     #360,R1    ;SET INSTRUCTION SOURCE = INBUS REG 17
7338 023670 052701 000002      BIS     #2,R1     ;SET INSTRUCTION DESTINATION = BSEL2
7339 023674 052701 021000      BIS     #MVIOX,R1 ;SET REST OF MOVE INSTRUCTION
7340 023700 010137 023710      MOV     R1,2$    ;SET INSTRUCTION AS SUBROUTINE ARGUMENT
7341 023704 004737 004074      JSR    PC,LOOPIN ;GET MICROPROCESSOR LOOPING ON MOVE INSTRUCTION
7342 023710 000000      2$:    .WORD 0       ;INSTRUCTION GOES HERE
7343
7344
7345      :-----:
7346      : WAIT FOR MCLK BIT TO BE SET TO 1
7347      :-----:
7347 023712 005037 002510      CLR     REGO      ;INIT PROGRAM TIMER
7348 023716 117737 156530 002364 3$:    MOVB   @BSEL2,REDBYT ;GET REG 17 INTO REDBYT
7349 023724 132737 000002 002364      BITB   #MCLK,REDBYT ;SEE IF MCLK BIT = 1 YET
7350 023732 001031                      BNE     6$        ;BR IF MCLK = 1
7351 023734 005237 002510      INC     REGO      ;INCREMENT TIMER
7352 023740 001366                      BNE     3$        ;BR IF PROGRAM TIMER DID NOT TIME OUT YET
7353                      ; (TIME OUT = SEVERAL HUNDRED MILLI-SEC)
7354 023742 012737 000002 002404      MOV     #MCLK,GOODAT ;SET EXPECTED REG CONTENTS
7355 023750 013737 002364 002406      MOV     REDBYT,BADDAT ;SET ACTUAL REG CONTENTS
7356 023756 004737 004016      JSR    PC,GETREG  ;GET REGS FOR PRINTOUT
7357      ;REPORT MCLK BIT STUCK AT 0
7358 023762                      ERRDF  5,EM5,ERR2
(4) 023762 104455                      TRAP    C$ERDF
(5) 023764 000005                      .WORD  5
(5) 023766 012312                      .WORD  EM5
(5) 023770 014632                      .WORD  ERR2
7359      ;TYPE 'PLEASE INSURE M8207 RUN SWITCH (E28 SW7) IS ON'
7360 023772                      PRINTF  #FMT24
(7) 023772 012746 012040                      MOV     #FMT24,-(SP)
(6) 023776 012746 000001                      MOV     #1,-(SP)
(3) 024002 010600                      MOV     SP,R0
(4) 024004 104417                      TRAP    C$PNTF
(4) 024006 062706 000004                      ADD     #4,SP
7361 024012 000137 024220      JMP     A1           ;ESCAPE TO END OF TEST
7362
7363      :-----:
7364      : WAIT FOR MCLK BIT TO BE CLEARED TO 0
7365      :-----:
7366 024016      6$:
7367 024016 005037 002510      CLR     REGO      ;INIT PROGRAM TIMER
7368 024022 117737 156424 002364 8$:    MOVB   @BSEL2,REDBYT ;GET REG 17 INTO REDBYT
7369 024030 132737 000002 002364      BITB   #MCLK,REDBYT ;SEE IF MCLK BIT = 0 YET
7370 024036 001430                      BEQ    10$       ;BR IF MCLK = 0
7371 024040 005237 002510      INC     REGO      ;INCREMENT TIMER
7372 024044 001366                      BNE     8$        ;BR IF TIMER DID NOT TIME OUT YET

```

```

7373 024046 005037 002404          CLR    GOODAT      ;SET EXPECTED REG CONTENTS
7374 024052 013737 002364 002406  MOV    REDBYT,BADAT ;SET ACTUAL REG CONTENTS
7375 024060 004737 004016          JSR    PC,GETREG   ;GET REGS FOR PRINTOUT
7376          ;REPORT MCLK BIT STUCK AT 1
7377 024064          ERRDF  6,EM6,ERR2
(4) 024064 104455          TRAP   C$ERDF
(5) 024066 000006          .WORD 6
(5) 024070 012343          .WORD EM6
(5) 024072 014632          .WORD ERR2
7378          ;TYPE 'PLEASE INSURE M8207 RUN SWITCH (E28 SW7) IS ON'
7379 024074          PRINTF #FMT24
(7) 024074 012746 012040          MOV    #FMT24,-(SP)
(6) 024100 012746 000001          MOV    #1,-(SP)
(3) 024104 010600          MOV    SP,RO
(4) 024106 104417          TRAP  C$PNTF
(4) 024110 062706 000004          ADD   #4,SP
7380 024114 000137 024220          JMP   A1           ;ESCAPE TO END OF TEST
7381
7382 -----
7383 ; WAIT FOR MCLK BIT TO BE SET TO 1 AGAIN
7384 -----
7385 10$:
7386 024120 005037 002510          CLR    REGO        ;INIT PROGRAM TIMER
7387 024124 117737 156322 002364 12$: MOVB  @BSEL2,REDBYT ;GET REG 17 INTO REDBYT
7388 024132 132737 000002 002364  BITB  #MCLK,REDBYT ;SEE IF MCLK BIT = 1 YET
7389 024140 001027          BNE   A1           ;BE IF MCLK = 1
7390 024142 005237 002510          INC   REGO        ;INCREMENT TIMER
7391 024146 001366          BNE   12$         ;BR IF TIMER DID NOT TIME OUT YET
7392 024150 012737 000002 002404  MOV    #MCLK,GOODAT ;SET EXPECTED REG CONTENTS
7393 024156 013737 002364 002406  MOV    REDBYT,BADAT ;SET ACTUAL REG CONTENTS
7394 024164 004737 004016          JSR    PC,GETREG   ;GET REGS FOR PRINTOUT
7395          ;REPORT MCLK BIT STUCK AT 0
7396 024170          ERRDF  5,EM5,ERR2
(4) 024170 104455          TRAP   C$ERDF
(5) 024172 000005          .WORD 5
(5) 024174 012312          .WORD EM5
(5) 024176 014632          .WORD ERR2
7397          ;TYPE 'PLEASE INSURE M8207 RUN SWITCH (E28 SW7) IS ON'
7398 024200          PRINTF #FMT24
(7) 024200 012746 012040          MOV    #FMT24,-(SP)
(6) 024204 012746 000001          MOV    #1,-(SP)
(3) 024210 010600          MOV    SP,RO
(4) 024212 104417          TRAP  C$PNTF
(4) 024214 062706 000004          ADD   #4,SP
7399 024220          A1:
7400 024220 004737 003576          JSR   PC,MSTCLR   ;ISSUE MASTER CLEAR
7401 024224          ENDTST
(3) 024224          L10036:
(3) 024224 104401          TRAP  C$ETST
7402
7403
7404
7405
7406
7407
7408 *****
.SBTTL TEST 14 - EXTENDED REGISTER MASTER CLEAR TEST
  
```

```
7409
7410
7411
7412
7413
7414
7415
7416
7417 024226
(3) 024226
7418 024226 004737 003576
7419 024232 005037 002402
7420 024236 012701 002654
7421 024242 112137 002374
7422 024246 112137 002376
7423 024252 004737 004312
7424 024256 032737 000002 002420
7425 024264 001413
7426 024266 012737 000014 002400
7427 024274 004737 004016
7428
7429 024300
(4) 024300 104455
(5) 024302 000064
(5) 024304 014105
(5) 024306 020630
7430 024310
(3) 024310 104410
(3) 024312 000244
7431 024314 062737 000002 002402 8$:
7432 024322 020127 002664
7433 024326 103745
7434 024330 004737 003576
7435 024334 005037 002402
7436 024340 012701 002664
7437 024344 004737 004124
7438 024350 032737 000001 002420 2$:
7439 024356 001413
7440 024360 012737 000014 002400
7441 024366 004737 004016
7442
7443 024372
(4) 024372 104455
(5) 024374 000065
(5) 024376 014146
(5) 024400 020630
7444 024402
(3) 024402 104410
(3) 024404 000152
7445 024406 023727 002402 000006 10$:
7446 024414 001003
7447 024416 142737 000372 002370
7448 024424 123711 002370 3$:
7449 024430 001417
7450 024432 005037 002404
7451 024436 111137 002404
```

\*\*\*\*\*  
BGNTST  
T14::  
JSR PC,MSTCLR ;ISSUE AN INITIAL MASTER CLEAR  
CLR AXNUM ;INIT AX REG BYTE NO. TO 0  
MOV #PATH,R1 ;INIT DATA PATTERN POINTER  
1\$: MOV (R1)+,WAX15 ;SET BITS TO LOAD INTO LO BYTE  
MOVB (R1)+,WAX16 ;SET BITS TO LOAD INTO HI BYTE  
JSR PC,WRITAX ;WRITE EXTENDED REG  
7424: BIT #WRDYTO,ERROR1 ;SEE IF READY FAILED TO SET  
BEQ 8\$ ;BR IF NOT  
7426: MOV #14,REGNUM ;SET LU REG NO = 14  
JSR PC,GETREG ;GET REGS FOR PRINTOUT  
;REPORT READY NOT SET AFTER AX REG WRITE  
ERRDF 52,EM52,ERR9  
TRAP C\$ERDF  
.WORD 52  
.WORD EM52  
.WORD ERR9  
ESCAPE TST  
TRAP C\$ESCAPE  
.WORD L10037-  
8\$: ADD #2,AXNUM ;INCR REG BYTE NO.  
CMP R1,#PATI ;SEE IF ALL REGS WRITTEN YET  
BLO 1\$ ;BR IF NOT DONE WRITING YET  
JSR PC,MSTCLR ;ISSUE MASTER CLEAR  
CLR AXNUM ;INIT EXTENDED REG BYTE NO. TO 0  
MOV #PATI,R1 ;INIT DATA PAT POINTER FOR READS  
2\$: JSR PC,READAX ;READ AN EXTENDED REG  
BIT #RRDYTO,ERROR1 ;SEE IF READY FAILED TO SET  
BEQ 10\$ ;BR IF NOT  
7440: MOV #14,REGNUM ;SET LU REG NO. = 14  
JSR PC,GETREG ;GET REGS FOR PRINTOUT  
;REPORT READY NOT SET AFTER AX REG READ  
ERRDF 53,EM53,ERR9  
TRAP C\$ERDF  
.WORD 53  
.WORD EM53  
.WORD ERR9  
ESCAPE TST  
TRAP C\$ESCAPE  
.WORD L10037-  
10\$: CMP AXNUM,#6 ;SEE IF AX3-15  
BNE 3\$ ;BR IF NOT  
BICB #AX315U,RAX15 ;MASK OFF UNPREDICTABLE BITS  
3\$: CMPB RAX15,(R1) ;COMPARE LO BYTE TO EXPECTED VALUE  
BEQ 4\$ ;BR IF DATA MATCHES  
CLR GOODAT  
MOV (R1),GOODAT ;GET EXPECTED DATA BYTE

```
7452 024442 013737 002370 002406      MOV      RAX15,BADDAT      ;GET ACTUAL DATA BYTE
7453 024450 004737 004526      JSR      PC,GETALL        ;GET REGS FOR PRINTOUT
7454      ;REPORT REG NOT INITIALIZED BY MASTER CLEAR
7455      ERRDF      2,EM2,ERR3
(4) 024454 104455
(5) 024456 000002
(5) 024460 012161
(5) 024462 015140
7456 024464      ESCAPE TST
(3) 024464 104410
(3) 024466 000070
7457 024470 005237 002402      4$: INC      AXNUM      ;INCREMENT AX BYTE NO.
7458 024474 005201      INC      R1              ;INCREMENT PAT POINTER
7459 024476 123711 002372      CMPB    RAX16,(R1)      ;COMPARE HI BYTE TO EXPECTED VALUE
7460 024502 001417      BEQ     6$              ;BR IF DATA MATCHES
7461 024504 005037 002404      CLR     GOODAT
7462 024510 111137 002404      MOVB   (R1),GOODAT     ;GET EXPECTED DATA BYTE
7463 024514 013737 002372 002406      MOV     RAX16,BADDAT   ;GET ACTUAL DATA BYTE
7464 024522 004737 004526      JSR     PC,GETALL      ;GET REGS FOR PRINTOUT
7465      ;REPORT REG NOT INITIALIZED BY MASTER CLEAR
7466      ERRDF      2,EM2,ERR3
(4) 024526 104455
(5) 024530 000002
(5) 024532 012161
(5) 024534 015140
7467 024536      ESCAPE TST
(3) 024536 104410
(3) 024540 000016
7468 024542 005237 002402      6$: INC      AXNUM      ;INCR AX BYTE NO.
7469 024546 005201      INC     R1              ;INCR PAT POINTER
7470 024550 020127 002674      CMP     R1,#PATJ      ;SEE IF ALL REGS READ YET
7471 024554 103673      BLO    2$              ;BR IF NOT DONE READING YET
7472 024556      ENDTST
(3) 024556
(3) 024556 104401
7473
7474
7475
7476
7477
7478
7479      ;*****
7480      ;SBTTL      TEST 15 - EXTENDED REGISTER ADDRESSING TEST
7481      ;*
7482      ;* FIRST, ISSUE A MASTER CLEAR TO PUT REGS INTO INITIALIZED STATES SHOWN IN
7483      ;* PATTERN I. THEN, WRITE A DIFFERENT WORD OF PATTERN J INTO EACH EXTENDED (AX)
7484      ;* REG, AND AFTER EACH WRITE, READ AND COMPARE ALL EXTENDED REGS TO EXPECTED
7485      ;* VALUES.
7486      ;* PATTERN I = 000,000,000,000,000,103,000,000
7487      ;* PATTERN J = 000,000,010,002,004,103,001,100
7488      ;*****
7488 024560      BGNST
(3) 024560
7489 024560 004737 003576      JSR     PC,MSTCLR     ;ISSUE MASTER CLEAR
7490 024564 012701 002664      MOV     #PATI,R1     ;INIT POINTER TO PAT I
7491 024570 012702 002500      MOV     #REDDAT,R2   ;INIT POINTER TO EXPECTED DATA AREA
7492 024574 112122      3$: MOVB   (R1)+,(R2)+ ;MOVE PAT I INTO REDDAT TABLE
```

TRAP C\$ERDF  
.WORD 2  
.WORD EM2  
.WORD ERR3

TRAP C\$ESCAPE  
.WORD L10037-

TRAP C\$ERDF  
.WORD 2  
.WORD EM2  
.WORD ERR3

TRAP C\$ESCAPE  
.WORD L10037-

L10037: TRAP C\$ETST

T15::



7535  
7536  
7537  
7538  
7539  
7540  
7541  
7542  
7543  
7544  
7545  
7546  
7547  
7548  
7549  
7550  
7551  
7552  
7553  
7554  
(3)  
7555  
7556  
7557  
7558  
(3)  
7559  
7560  
7561  
7562  
7563  
7564  
7565  
7566  
7567  
7568  
7569  
7570  
7571  
7572  
(4)  
(5)  
(5)  
(5)  
7573  
(3)  
(3)  
7574  
7575  
7576  
7577  
7578  
7579  
7580  
7581  
7582

025050  
025050  
025050 004737 003576  
025054 012701 002704  
025060  
025060 104404  
025062 012737 000004 002402  
025070 111137 002374  
025074 116137 000001 002376  
025102 143737 002565 002374  
025110 143737 002566 002376  
025116 004737 004312  
025122 004737 004124  
025126 123737 002370 002374  
025134 001416  
025136 013737 002374 002404  
025144 013737 002370 002406  
025152 004737 004526  
025156  
(4) 025156 104455  
(5) 025160 000003  
(5) 025162 012220  
(5) 025164 015140  
025166  
(3) 025166 104410  
(3) 025170 000052  
025172 005237 002402  
025176 123737 002372 002671  
025204 001416  
025206 005037 002404  
025212 113737 002671 002404  
025220 013737 002372 002406  
025226 004737 004526  
025232

```
*****  
: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,  
:* 000,000,000,000,000,000,000,376,375,373,367,357,337,277,177,  
:* 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$:  
BGNSEG  
TRAP C$BSEG  
MOV #4,AXNUM :SET BYTE NO. = 4  
MOV (R1),WAX15 :SET DATA TO WRITE INTO LO BYTE  
MOV 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 MISCMPARE  
ERRDF 3,EM3,ERR3  
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 MISCMPARE  
ERRDF 3,EM3,ERR3
```

(4) 025232 104455  
(5) 025234 000003  
(5) 025236 012220  
(5) 025240 015140  
7583 025242  
7584 025242  
(3) 025242  
(3) 025242 104405  
7585 025244 062701 000002  
7586 025250 020127 003014  
7587 025254 103701  
7588 025256  
(3) 025256  
(3) 025256 104401  
7589  
7590  
7591  
7592  
7593  
7594  
7595  
7596  
7597  
7598  
7599  
7600  
7601  
7602  
7603  
7604  
7605 025260  
(3) 025260  
7606 025260 004737 003576  
7607 025264 012701 003014  
7608 025270  
7609 025270  
(3) 025270 104404  
7610 025272 012737 000000 002402  
7611 025300 111137 002374  
7612 025304 116137 000001 002376  
7613 025312 143737 002561 002374  
7614 025320 143737 002562 002376  
7615 025326 004737 004312  
7616 025332 004737 004124  
7617 025336 123737 002370 002374  
7618 025344 001416  
7619 025346 013737 002374 002404  
7620 025354 013737 002370 002406  
7621 025362 004737 004526  
7622  
7623 025366  
(4) 025366 104455  
(5) 025370 000003  
(5) 025372 012220  
(5) 025374 015140  
7624 025376

9\$:

ENDSEG

ENDTST

ADD #2,R1  
CMP R1,#PATL  
BLO 3\$

:INCR PATTERN POINTER  
:SEE IF ALL DATA WRITTEN YET  
:BR IF NOT DONE YET

10000\$:

TRAP C\$ERDF  
.WORD 3  
.WORD EM3  
.WORD ERR3

TRAP C\$ESEG

L10041:

TRAP C\$ETST

\*\*\*\*\*  
:SBTTL TEST 17 - AX0-15,AX0-16 READ/WRITE BIT TEST  
:  
:\* IN THIS TEST, A MASTER CLEAR IS DONE, AND THEN A WRITE, READ, AND COMPARE  
:\* ARE PERFORMED IN REGS AX0-15,AX0-16 USING EACH WORD OF PATTERN L.  
:\* ANY BITS IN AX0-15,AX0-16 WHICH ARE NOT READ/WRITE ARE MASKED OFF (TO 0)  
:\* IN THE EXPECTED VALUE BEFORE COMPARISON.  
:\* PATTERN L =  
:\* FOR REG 15: 000,377,000  
:\* FOR REG 16: 000,377,000.  
\*\*\*\*\*

BGNTST

3\$:

BGNSEG

JSR PC,MSTCLR  
MOV #PATL,R1  
MOV #0,AXNUM  
MOVB (R1),WAX15  
MOVB 1(R1),WAX16  
BICB ANBITS+0,WAX15  
BICB ANBITS+1,WAX16  
JSR PC,WRITAX  
JSR PC,READAX  
CMPB RAX15,WAX15  
BEQ 6\$  
MOV WAX15,GOODAT  
MOV RAX15,BADDAT  
JSR PC,GETALL  
:REPORT REG MISC  
ERRDF 3,EM3,ERR3

:ISSUE MASTER CLEAR  
:INIT DATA PATTERN POINTER  
:SET BYTE NO. = 0  
:SET DATA TO WRITE INTO LO BYTE  
:SET DATA TO WRITE INTO HI BYTE  
:MASK OFF NON-READ/WRITE BITS IN LO BYTE  
:MASK OFF NON-READ/WRITE BITS IN HI BYTE  
:LOAD DATA INTO AX0-15,AX0-16  
:READ AX0-15 AND AX0-16  
:COMPARE LO BYTE DATA READ  
:BR IF DATA MATCHES  
:SET EXPECTED DATA  
:SET ACTUAL DATA  
:GET REGS FOR PRINTOUT

T17::

TRAP C\$BSEG

TRAP C\$ERDF  
.WORD 3  
.WORD EM3  
.WORD ERR3

ESCAPE SEG



```
(3) 025376 104410 TRAP C$ESCAPE
(3) 025400 000046 .WORD 10000$-
7625 025402 005237 002402 6$: INC AXNUM ;SET AX BYTE NO. = 1
7626 025406 123737 002372 002376 CMPB RAX16,WAX16 ;COMPARE HI BYTE DATA READ
7627 025414 001414 BEQ 9$ ;BR IF DATA MATCHES
7628 025416 013737 002376 002404 MOV WAX16,GOODAT ;SET EXPECTED DATA
7629 025424 013737 002372 002406 MOV RAX16,BADDAT ;SET ACTUAL DATA
7630 025432 004737 004526 JSR PC,GETALL ;GET REGS FOR PRINTOUT
7631 ;REPORT REG MISC COMPARE
7632 025436 ERRDF 3,EM3,ERR3
(4) 025436 104455 TRAP C$ERDF
(5) 025440 000003 .WORD 3
(5) 025442 012220 .WORD EM3
(5) 025444 015140 .WORD ERR3
7633 025446 9$:
7634 025446 ENDSEG
(3) 025446 10000$: TRAP C$ESEG
7635 025450 062701 000002 ADD #2,R1 ;INCR PATTERN POINTER
7636 025454 020127 003022 CMP R1,#PATM ;SEE IF ALL DATA WRITTEN YET
7637 025460 103703 BLO 3$ ;BR IF NOT DONE YET
7638 025462 ENDTST
(3) 025462 L10042: TRAP C$ETST
(3) 025462 104401
7639
7640
7641
7642
7643
7644
7645 :*****
7646 :SBTTL TEST 18 - AX1-15,AX1-16 READ/WRITE BIT TEST
7647 :*
7648 :* IN THIS TEST, A MASTER CLEAR IS DONE, AND THEN A WRITE, READ, AND COMPARE
7649 :* ARE PERFORMED IN REGS AX1-15,AX1-16 USING EACH WORD OF PATTERN K.
7650 :* ANY BITS IN AX1-15,AX1-16 WHICH ARE NOT READ/WRITE ARE MASKED OFF (TO 0)
7651 :* IN THE EXPECTED VALUE BEFORE COMPARISON.
7652 :*****
7653 025464 004737 003576 JSR PC,MSTCLR ;ISSUE MASTER CLEAR
(3) 025464 012701 002704 MOV #PATK,R1 ;INIT DATA PATTERN POINTER
7654 025470
7655 025474 3$:
7656 025474 BGNSEG TRAP C$BSEG
(3) 025474 104404
7657 025476 012737 000002 002402 MOV #2,AXNUM ;SET BYTE NO. = 2
7658 025504 111137 002374 MOVB (R1),WAX15 ;SET DATA TO WRITE INTO LO BYTE
7659 025510 116137 000001 002376 MOVB 1(R1),WAX16 ;SET DATA TO WRITE INTO HI BYTE
7660 025516 143737 002563 002374 BICB ANBITS+2,WAX15 ;MASK OFF NON-READ/WRITE BITS IN LO BYTE
7661 025524 143737 002564 002376 BICB ANBITS+3,WAX16 ;MASK OFF NON-READ/WRITE BITS IN HI BYTE
7662 025532 004737 004312 JSR PC,WRITAX ;LOAD DATA INTO AX1-15,AX1-16
7663 025536 004737 004124 JSR PC,READAX ;READ AX1-15 AND AX1-16
7664 025542 123737 002370 002374 CMPB RAX15,WAX15 ;COMPARE LO BYTE DATA READ
7665 025550 001416 BEQ 6$ ;BR IF DATA MATCHES
7666 025552 013737 002374 002404 MOV WAX15,GOODAT ;SET EXPECTED DATA
7667 025560 013737 002370 002406 MOV RAX15,BADDAT ;SET ACTUAL DATA
7668 025566 004737 004526 JSR PC,GETALL ;GET REGS FOR PRINTOUT
```

7669  
7670 025572  
(4) 025572 104455  
(5) 025574 000003  
(5) 025576 012220  
(5) 025600 015140  
7671 025602  
(3) 025602 104410  
(3) 025604 000046  
7672 025606 005237 002402  
7673 025612 123737 002372 002376  
7674 025620 001414  
7675 025622 013737 002376 002404  
7676 025630 013737 002372 002406  
7677 025636 004737 004526  
7678  
7679 025642  
(4) 025642 104455  
(5) 025644 000003  
(5) 025646 012220  
(5) 025650 015140  
7680 025652  
7681 025652  
(3) 025652  
(3) 025652 104405  
7682 025654 062701 000002  
7683 025660 020127 003014  
7684 025664 103703  
7685 025666  
(3) 025666  
(3) 025666 104401  
7686  
7687  
7688  
7689  
7690  
7691  
7692  
7693  
7694  
7695  
7696  
7697  
7698  
7699  
7700  
7701  
7702  
7703  
7704  
7705  
7706  
7707  
7708  
7709  
7710 025670

```
;REPORT REG MISCOMPARE  
ERRDF 3,EM3,ERR3  
  
ESCAPE SEG  
  
6$: INC AXNUM ;SET AX BYTE NO. = 3  
CMPB RAX16,WAX16 ;COMPARE HI BYTE DATA READ  
BEQ 9$ ;BR IF DATA MATCHES  
MOV WAX16,GOODAT ;SET EXPECTED DATA  
MOV RAX16,BADDAT ;SET ACTUAL DATA  
JSR PC,GETALL ;GET REGS FOR PRINTOUT  
  
;REPORT REG MISCOMPARE  
ERRDF 3,EM3,ERR3  
  
9$:  
ENDSEG  
  
ADD #2,R1 ;INCR PATTERN POINTER  
CMP R1,#PATL ;SEE IF ALL DATA WRITTEN YET  
BLO 3$ ;BR IF NOT DONE YET  
  
ENDTST
```

TRAP C\$ERDF  
.WORD 3  
.WORD EM3  
.WORD ERR3  
  
TRAP C\$ESCAPE  
.WORD 10000\$-.  
  
TRAP C\$ERDF  
.WORD 3  
.WORD EM3  
.WORD ERR3  
  
10000\$: TRAP C\$ESEG  
  
L10043: TRAP C\$ETST

```
.....  
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  
.....  
BGNST
```

```

(3) 025670
7711 025670 004737 003576          JSR      PC,MSTCLR      ;ISSUE MASTER CLEAR      119::
7712 025674 142777 000010 154546  BICB    #LULOP,@BSEL1  ;CLEAR LULOP
7713 025702 012702 003152          MOV     #PATV,R2      ;INIT PATTERN V POINTER
7714 025706 012701 003104          MOV     #PATU,R1      ;INIT PATTERN U POINTER
7715 025712
7716 025712          3$:      BGNSEG
(3) 025712 104404          TRAP    C$BSEG
7717 025714 012737 000006 002402  MOV     #6,AXNUM      ;SET BYTE NO. = 6
7718 025722 111237 002374          MOVB   (R2),WAX15     ;SET DATA TO WRITE INTO LO BYTE
7719 025726 116237 000001 002376  MOVB   1(R2),WAX16    ;SET DATA TO WRITE INTO HI BYTE
7720 025734 004737 004312          JSR    PC,WRITAX     ;LOAD DATA INTO AX3-15,AX3-16
7721 025740 004737 004124          JSR    PC,READAX    ;READ AX3-15 AND AX3-16
7722 025744 132737 000001 002374  BITB   #TEST,WAX15   ;SEE IF AN INTERFACE IS SELECTED
7723 025752 001003          BNE    4$           ;BR IF YES
7724 025754 142737 000372 002370  BICB   #AX315U,RAX15  ;MASK OFF UNPREDICTABLE BITS
7725 025762 142737 000040 002370 4$:      BICB   #C32BCC,RAX15  ;CLEAR CRC32 BCC BIT
7726 025770 123711 002370          CMPB   RAX15,(R1)    ;COMPARE LO BYTE DATA READ
7727 025774 001417          BEQ    6$           ;BR IF DATA MATCHES
7728 025776 005037 002404          CLR    GOODAT
7729 026002 111137 002404          MOVB   (R1),GOODAT   ;SET EXPECTED DATA
7730 026006 013737 002370 002406  MOV     RAX15,BADDAT  ;SET ACTUAL DATA
7731 026014 004737 004526          JSR    PC,GETALL    ;GET REGS FOR PRINTOUT
7732          ;REPORT REG MISCMPARE
7733 026020          ERRDF  3,EM3,ERR3
(4) 026020 104455          TRAP    C$ERDF
(5) 026022 000003          .WORD  3
(5) 026024 012220          .WORD  EM3
(5) 026026 015140          .WORD  ERR3
7734 026030          ESCAPE  SEG
(3) 026030 104410          TRAP    C$ESCAPE
(3) 026032 000052          .WORD  10000$-.
7735 026034 005237 002402 6$:      INC     AXNUM         ;SET AX BYTE NO. = 7
7736 026040 123761 002372 000001  CMPB   RAX16,1(R1)   ;COMPARE HI BYTE DATA READ
7737 026046 001416          BEQ    9$           ;BR IF DATA MATCHES
7738 026050 005037 002404          CLR    GOODAT
7739 026054 116137 000001 002404  MOVB   1(R1),GOODAT  ;SET EXPECTED DATA
7740 026062 013737 002372 002406  MOV     RAX16,BADDAT  ;SET ACTUAL DATA
7741 026070 004737 004526          JSR    PC,GETALL    ;GET REGS FOR PRINTOUT
7742          ;REPORT REG MISCMPARE
7743 026074          ERRDF  3,EM3,ERR3
(4) 026074 104455          TRAP    C$ERDF
(5) 026076 000003          .WORD  3
(5) 026100 012220          .WORD  EM3
(5) 026102 015140          .WORD  ERR3
7744 026104          9$:
7745 026104          ENDSEG
(3) 026104          10000$:
(3) 026104 104405          TRAP    C$ESEG
7746 026106 062702 000002          ADD     #2,R2        ;INCR PATTERN V POINTER
7747 026112 062701 000002          ADD     #2,R1        ;INCR PATTERN U POINTER
7748 026116 020127 003152          CMP     R1,#PATV     ;SEE IF ALL DATA WRITTEN YET
7749 026122 103673          BLO    3$           ;BR IF NOT DONE YET
7750 026124          ENDTST
(3) 026124          L10044:
(3) 026124 104401          TRAP    C$ETST
  
```

7751  
7752  
7753  
7754  
7755  
7756  
7757  
7758  
7759  
7760  
7761  
7762  
7763  
7764  
7765  
7766  
7767  
7768  
7769  
7770  
7771  
7772  
7773  
7774  
7775  
7776  
7777  
7778  
7779  
7780  
7781  
7782  
7783  
7784  
7785  
7786  
7787  
7788  
7789  
7790  
7791  
7792  
7793  
7794  
7795  
7796

026126  
(3) 026126  
026126 004737 003576  
026132 012701 003050  
026136 012702 003066  
026142 012737 000017 002400  
026150 012737 000005 002402  
  
026156  
(3) 026156  
(3) 026156 104402  
026160  
026160 104404  
026162 111137 002366  
026166 004737 003750  
026172 004737 004124  
026176 123712 002372  
026202 001421  
026204 005037 002410  
026210 111137 002410  
026214 005037 002404  
026220 111237 002404  
026224 013737 002372 002406  
026232 004737 004526  
  
026236  
(4) 026236 104455  
(5) 026240 000003  
(5) 026242 012220  
(5) 026244 016300  
026246  
026246 104405  
026250 005201  
026252 005202  
026254 020127 003066

```
*****  
:SBTTL 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).  
:*****  
BGNTST  
T20::  
JSR PC,MSTCLR ;ISSUE MASTER CLEAR  
MOV #PATO,R1 ;INIT PAT O 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$BSEG  
MOVB (R1),WRIBY ;SET BYTE TO BE WRITTEN  
JSR PC,WRITLU ;WRITE DATA BYTE INTO REG 17  
JSR PC,READAX ;READ AX2  
CMFB RAX16,(R2) ;COMPARE AX2-16 TO EXPECTED DATA  
BEQ 6$ ;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 MISCOMPARE  
ERRDF 3,EM3,ERR5  
6$: TRAP C$ERDF  
TRAP C$ESEG  
10000$:  
INC R1 ;INCR PAT O POINTER  
INC R2 ;INCR PAT P POINTER  
CMP R1,#PATP ;SEE IF ALL BYTES LOADED YET
```

```
7797 026260 103737 BLO 3$ ;BR IF NOT
7798 026262 ENDSUB
(3) 026262 L10046: TRAP C$ESUB
(3) 026262 104403
7799
7800 :-----:
7801 : LOAD REG 17, DO MASTER CLEAR, READ AND COMPARE AX2-16 :-----:
7802 026264 BGNSUB
(3) 026264 T20.2: TRAP C$BSUB
(3) 026264 104402
7803 026266 112737 000375 002366 MOVB #375,WRIBY ;SET DATA TO BE LOADED
7804 026274 004737 003750 JSR PC,WRITLU ;LOAD DATA INTO REG 17
7805 026300 004737 003576 JSR PC,MSTCLR ;PERFORM MASTER CLEAR
7806 026304 004737 004124 JSR PC,READAX ;READ AX2-15,AX2-16
7807 026310 123737 002372 002671 CMPB RAX16,PATI+5 ;SEE IF AX2-16 WAS INIT'D CORRECTLY
7808 026316 001416 BEQ 6$ ;BR IF YES
7809 026320 005037 002404 CLR GOODAT
7810 026324 113737 002671 002404 MOVB PATI+5,GOODAT ;SET EXPECTED DATA
7811 026332 013737 002372 002406 MOV RAX16,BADDAT ;SET ACTUAL DATA
7812 026340 004737 004526 JSR PC,GETALL ;GET REGS FOR PRINTOUT
7813 :REPORT REG NOT INITIALIZED BY MASTER CLEAR
7814 026344 ERRDF 2,EM2,ERR3 TRAP C$ERDF
(4) 026344 104455 .WORD 2
(5) 026346 000002 .WORD EM2
(5) 026350 012161 .WORD ERR3
(5) 026352 015140
7815 026354 6$: ENDSUB
7816 026354
(3) 026354 L10047: TRAP C$ESUB
(3) 026354 104403
7817 026356 ENDTST
(3) 026356 L10045: TRAP C$ETST
(3) 026356 104401
7818
7819
7820
7821
7822
7823 :*****:
7824 :SBTTL TEST 21 - TRANSMITTER BUFFER DATA TEST:
7825 :* A MASTER CLEAR IS DONE FIRST, AND THEN A BYTE OF PATTERN N IS LOADED INTO:
7826 :* REG 11 AND THE NEXT BYTE IS LOADED TWICE INTO REG 10. THE PROGRAM THEN WAITS:
7827 :* AT LEAST 50 MICRO-SEC, AND THEN IT READS AND COMPARES AX1-15 TO THE BYTE:
7828 :* WHICH WAS LOADED INTO REG 10, AND IT READS AND COMPARES AX1-16 TO THE BYTE:
7829 :* WHICH WAS LOADED INTO REG 11. THIS PROCESS IS REPEATED (INCLUDING THE MASTER:
7830 :* CLEAR) FOR EACH PAIR OF BYTES IN PATTERN N.
7831 :* PATTERN N =
7832 :* FOR REG 10: 000,125,252,377,000,000,000
7833 :* FOR REG 11: 000,000,000,000,005,012,017:
7834 :*****:
7835 026360 BGNTST
(3) 026360 T21.:
7836 026360 012701 003032 MOV #PATN,R1 ;INIT PATTERN POINTER
7837 026364 012737 026626 002362 MOV #A2,RETADR ;SET SUBROUTINE ERROR RETURN ADDRESS
7838 026372
(3) 026372 T21.1:
```

```

(3) 026372 104402
7839 026374 004737 003576
7840 026400 004737 004662
7841 026404 000001
7842 026406 012737 000011 002400
7843 026414 111137 002366
7844 026420 004737 003750
7845 026424 012737 000010 002400
7846 026432 116137 000001 002366
7847 026440 004737 003750
7848 026444 004737 003750
7849 026450 004737 005146
7850 026454 004737 004662
7851 026460 000003
7852 026462 012737 000002 002402
7853 026470 004737 004124
7854 026474 123761 002370 000001
7855 026502 001420
7856 026504 005037 002404
7857 026510 116137 000001 002404
7858 026516 013737 002370 002406
7859 026524 004737 004526
7860
7861 026530
(4) 026530 104455
(5) 026532 000003
(5) 026534 012220
(5) 026536 015140
7862 026540
(3) 026540 104410
(3) 026542 000064
7863 026544 005237 002402
7864 026550 123711 002372
7865 026554 001417
7866 026556 005037 002404
7867 026562 111137 002404
7868 026566 013737 002372 002406
7869 026574 004737 004526
7870
7871 026600
(4) 026600 104455
(5) 026602 000003
(5) 026604 012220
(5) 026606 015140
7872 026610
(3) 026610 104410
(3) 026612 000014
7873 026614 062701 000002
7874 026620 020127 003050
7875 026624 103663
7876 026626
7877 026626
(3) 026626
(3) 026626 104403
7878 026630
(3) 026630

3$: JSR PC,MSTCLR ;ISSUE MASTER CLEAR TRAP C$SUB
     JSR PC,OSIRDY ;CHECK ORDY AND OCOR FOR EXPECTED STATES
     1
     MOV #11,REGNUM ;SET LU REG NO. = 11
     MOVB (R1),WRIBYT ;SET DATA BYTE TO BE WRITTEN
     JSR PC,WRITLU ;WRITE BYTE INTO REG 11
     MOV #10,REGNUM ;SET LU REG NO. = 10
     MOVB 1(R1),WRIBYT ;SET DATA BYTE TO BE WRITTEN
     JSR PC,WRITLU ;WRITE BYTE INTO REG 10
     JSR PC,WRITLU ;WRITE IT AGAIN (SO 2 ENTRIES ARE IN SILO)
     JSR PC,WAIT50 ;WAIT FOR SILO DATA TO RIPPLE
     JSR PC,OSIRDY ;CHECK ORDY AND OCOR FOR EXPECTED STATES
     3
     MOV #2,AXNUM ;SET BYTE NO. FOR AX1-15
     JSR PC,READAX ;READ AX1-15, AX1-16
     CMPB RAX15,1(R1) ;COMPARE AX1-15 TO EXPECTED
     BEQ 6$ ;BR IF MATCH
     CLR GOODAT
     MOVB 1(R1),GOODAT ;SET EXPECTED DATA
     MOV RAX15,BADDAT ;SET ACTUAL DATA
     JSR PC,GETALL ;GET REGS FOR PRINTOUT
;REPORT REG MISCMPARE
ERRDF 3,EM3,ERR3

(4) 026530 104455 TRAP C$ERDF
(5) 026532 000003 .WORD 3
(5) 026534 012220 .WORD EM3
(5) 026536 015140 .WORD ERR3

7862 026540 ESCAPE SUB TRAP C$ESCAPE
(3) 026540 104410 .WORD L10051-
(3) 026542 000064

6$: INC AXNUM ;INCR AX BYTE NO.
     CMPB RAX16,(R1) ;COMPARE AX1-16 TO EXPECTED
     BEQ 9$ ;BR IF MATCH
     CLR GOODAT
     MOVB (R1),GOODAT ;SET EXPECTED DATA
     MOV RAX16,BADDAT ;SET ACTUAL DATA
     JSR PC,GETALL ;GET REGS FOR PRINTOUT
;REPORT REG MISCMPARE
ERRDF 3,EM3,ERR3

(4) 026600 104455 TRAP C$ERDF
(5) 026602 000003 .WORD 3
(5) 026604 012220 .WORD EM3
(5) 026606 015140 .WORD ERR3

7872 026610 ESCAPE SUB TRAP C$ESCAPE
(3) 026610 104410 .WORD L10051-
(3) 026612 000014

9$: ADD #2,R1 ;INCR DATA PATTERN POINTER
     CMP R1,#PATO ;SEE IF ALL DATA BYTES WRITTEN YET
     BLO 3$ ;BR IF NOT DONE YET

A2: ENDSUB

(3) 026626 L10051:
(3) 026626 104403 TRAP C$SUB
7878 026630
(3) 026630 L10050:
  
```

TRAP (SETST

(3) 026630 104401  
7879  
7880  
7881  
7882  
7883  
7884  
7885  
7886  
7887  
7888  
7889  
7890  
7891  
7892  
7893  
7894  
7895  
7896  
7897  
7898  
7899  
7900 026632  
(3) 026632  
7901 026632 012737 027306 002362  
7902  
7903  
7904  
7905 026640 004737 003576  
7906 026644 004737 004662  
7907 026650 000001  
7908  
7909  
7910  
7911 026652 012737 000400 002422  
7912 026660 004737 005174  
7913 026664 004737 005174  
7914 026670 004737 005146  
7915 026674 004737 004662  
7916 026700 000003  
7917  
7918  
7919  
7920 026702 005001  
7921 026704 012737 000017 002400  
7922 026712 004737 005254  
7923 026716 000001  
7924 026720 004737 003672  
7925 026724 132737 000020 002364  
7926 026732 001404  
7927 026734 005201  
7928 026736 020127 000003  
7929 026742 002763  
7930 026744 004737 004662  
7931 026750 000001  
7932

```
*****  
: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$: JSR PC,STPLU ;STEP LU 1 CYCLE  
1  
JSR PC,READLU ;READ REG 17  
BITB #OCOR,REDBYT ;SEE IF OCOR = 0 YET  
BEQ 6$ ;BR IF OCOR = 0  
INC R1 ;INCR CYCLE COUNT  
7928: CMP R1,#3 ;SEE IF 3 CYCLES DONE YET  
7929: BLT 3$ ;BR IF NO  
6$: JSR PC,OSIRDY ;CHK ORDY=1, OCOR=0  
1  
-----
```

```

7933      ; LOAD 64 BINARY COUNT CHARS INTO SILO, CHK ORDY=0
7934      -----
7935 026752 005003      CLR      R3      ;INIT PATTERN FOR WRITING
7936 026754 010337 002422 8$:  MOV      R3,TXWORD ;SET DATA TO BE WRITTEN
7937 026760 004737 005174      JSR      PC,LDTXSI ;LOAD DATA CHAR INTO TX SILO
7938 026764 004737 005146      JSR      PC,WAIT50 ;WAIT FOR DATA TO RIPPLE IN SILO
7939 026770 020327 000077      CMP      R3,#63.  ;SEE IF 64TH CHAR JUST LOADED
7940 026774 001004      BNE      9$      ;BR IF NO
7941 026776 012737 000002 027020  MOV      #2,14$   ;SET UP TO CHK ORDY=0,OCOR=1
7942 027004 000403      BR       12$     ;
7943 027006 012737 000003 027020 9$:  MOV      #3,14$   ;SET UP TO CHK ORDY=1,OCOR=1
7944 027014 004737 004662 12$:  JSR      PC,OSIRDY ;CHK ORDY, OCOR
7945 027020 000000 14$:  .WORD   0
7946 027022 005203      INC      R3      ;INCR PATTERN FOR WRITES
7947 027024 020327 000100      CMP      R3,#64.  ;SEE IF 64 CHARS LOADED YET
7948 027030 002751      BLT     8$      ;BR IF NO
7949      -----
7950      ; CLOCK LINE UNIT, CHECK ORDY = 1 WITHIN 8 CYCLES
7951      -----
7952 027032 012737 000011 002400  MOV      #11,REGNUM ;SET REG NO. = 11
7953 027040 005001      CLR      R1      ;INIT CYCLE COUNT
7954 027042 004737 005254 16$:  JSR      PC,STPLU  ;CLOCK LU FOR 1 CYCLE
7955 027046 000001      1
7956 027050 004737 003672      JSR      PC,READLU ;READ REG 11
7957 027054 132737 000020 002364  BITB     #ORDY,REDBYT ;SEE IF ORDY = 1 YET
7958 027062 001004      BNE     19$     ;BR IF YES
7959 027064 005201      INC      R1      ;INCR CYCLE COUNT
7960 027066 020127 000010      CMP      R1,#8.  ;SEE IF 8 CYCLES YET
7961 027072 002763      BLT     16$     ;BR IF NOT YET
7962 027074 004737 004662 19$:  JSR      PC,OSIRDY ;CHK ORDY = 1, OCOR = 1
7963 027100 000003      3
7964      -----
7965      ; READ AND COMPARE FIRST CHARACTER IN AX1-15
7966      -----
7967 027102 005004      CLR      R4      ;INIT PATTERN FOR READING
7968 027104 012737 000002 002402  MOV      #2,AXNUM  ;SET AX BYTE NO. FOR AX1-15
7969 027112 004737 004124      JSR      PC,READAX ;READ AX1-15
7970 027116 123704 002370      CMPB     RAX15,R4 ;COMPARE AX1-15 TO EXPECTED
7971 027122 001415      BEQ     20$     ;BR IF MATCH
7972 027124 010437 002404      MOV      R4,GOODAT ;SET EXPECTED DATA
7973 027130 013737 002370 002406  MOV      RAX15,BADDAT ;SET ACTUAL DATA
7974 027136 004737 004526      JSR      PC,GETALL ;GET REGS FOR PRINTOUT
7975      ;REPORT REG MISCOMPARE
7976 027142      ERRDF   3,EM3,ERR3
7977 (4) 027142 104455      TRAP    C$ERDF
7978 (5) 027144 000003      .WORD   3
7979 (5) 027146 012220      .WORD   EM3
7980 (5) 027150 015140      .WORD   ERR3
7981 027152      ESCAPE  TST
7982 (3) 027152 104410      TRAP    C$ESCAPE
7983 (3) 027154 000132      .WORD   L10052-.
7984 027156      20$:
7985      ; LOAD AND COMPARE REST OF CHARS, MONITOR ORDY, OCOR
7986      -----
7987 027156 020327 000377 24$:  CMP      R3,#255. ;SEE IF ALL CHARS LOADED YET
  
```



```
7983 027162 003010          BGT      26$          ;BR IF YES
7984 027164 010337 002422    MOV      R3,TXWORD    ;SET DATA TO BE WRITTEN
7985 027170 004737 005174    JSR      PC,LDTXSI    ;LOAD DATA CHAR INTO TX SILO
7986 027174 005203          INC      R3           ;INCR DATA TO BE WRITTEN
7987 027176 004737 004662    JSR      PC,OSIRDY    ;CHK ORDY=0, OCOR=1
7988 027202 000002          2        ;
7989 027204 005204          26$: INC      R4           ;INCR DAT FOR READING
7990 027206 004737 005254    JSR      PC,STPLU     ;CLOCK LINE UNIT FOR 8 CYCLES
7991 027212 000010          8        ;
7992 027214 004737 004124    JSR      PC,READAX    ;READ AX1-15
7993 027220 123704 002370    CMPB    RAX15,R4     ;COMPARE AX1-15 TO EXPECTED
7994 027224 001415          BEQ      27$         ;BR IF MATCH
7995 027226 010437 002404    MOV      R4,GOODAT   ;SET EXPECTED DATA
7996 027232 013737 002370 002406 MOV      RAX15,BADDAT ;SET ACTUAL DATA
7997 027240 004737 004526    JSR      PC,GETALL    ;GET REGS FOR PRINTOUT
7998          ;REPORT REG MISCMPARE
7999          ERRDF 3,EM3,ERR3
(4) 027244 104455          TRAP    C$ERDF
(5) 027246 000003          .WORD  3
(5) 027250 012220          .WORD  EM3
(5) 027252 015140          .WORD  ERR3
8000 027254          ESCAPE  TST          TRAP    C$ESCAPE
(3) 027254 104410          (3) 027256 000030          .WORD  L10052-
8001 027260 020427 000377    27$: CMP      R4,#255. ;SEE IF WE READ LAST CHAR YET
8002 027264 001004          BNE     29$         ;BR IF NOT
8003 027266 004737 004662    JSR      PC,OSIRDY    ;CHK ORDY=1, OCOR=0
8004 027272 000001          1
8005 027274 000404          BR      A3
8006 027276 004737 004662    29$: JSR      PC,OSIRDY ;CHK ORDY=1, OCOR=1
8007 027302 000003          3
8008 027304 000724          BR      24$         ;CONTINUE
8009 027306          A3:
8010 027306          ENDTST
(3) 027306          L10052:
(3) 027306 104401          TRAP    C$ETST
8011
8012
8013
8014
8015
8016          ;*****
8017          .SBTTL  TEST 23 - TX MSG TIMING TEST, CHAR MODE, WITH CRC
8018          ;*
8019          ;* IN THIS TEST, AN ENTIRE MESSAGE IS TRANSMITTED (USING STEPLU AND LULOOP)
8020          ;* AND THE PROGRAM MONITORS THE OCCURRENCE OF USYRT TX BUFFER EMPTY FLAGS
8021          ;* (BY SCANNING ORDY AND OCOR) AND OACT, AT EACH STEP. THE TEST IS DONE IN
8022          ;* CHARACTER ORIENTED PROTOCOL MODE, USING 8-BIT CHARS AND CRC-16.
8023          ;* THE FOLLOWING STEPS ARE DONE:
8024          ;* A MASTER CLEAR IS DONE, AND THE LINE UNIT IS PLACED IN CHAR MODE.
8025          ;* SOM IS SET TWICE TO SEND 2 SYNCH CHARS. THEN, 2 000 CHARS ARE SENT, AND
8026          ;* THEN 2 TERMINATING SYNCHS ARE SENT.
8027          ;* THE TRANSMITTER IS THEN DISABLED, USING OC, AND OACT IS MONITORED FOR THE
8028          ;* CLEARED STATE AFTER THE 3RD SYNCH COMPLETES.
8029          ;*****
8030 027310          BGNST
```

```
(3) 027310
8031 027310 012737 027422 002362      MOV      #A4,RETADR      ;SET TEST EXIT ADDRESS FOR ERRORS      T23::
8032 027316 004737 005540              JSR      PC,INITRN      ;DO MASTER CLR, LOAD 2 SOM*2
8033 027322 000226                      SYNCH
8034 027324 000011                      STRIP!DDCMP
8035 027326 004737 006122              JSR      PC,TXCHAR      ;LOAD A 000 CHAR, TX FIRST SYNCH (226)
8036 027332 000000                      000
8037 027334 100010                      CHPCHK!8.
8038 027336 004737 006122              JSR      PC,TXCHAR      ;LOAD 2ND 000 CHAR, TX 2ND SYNCH
8039 027342 000000                      000
8040 027344 000010                      8.
8041 027346 004737 006122              JSR      PC,TXCHAR      ;LOAD EOM CHAR, TX FIRST 000 CHAR
8042 027352 001000                      TXEOM
8043 027354 000010                      8.
8044 027356 004737 006122              JSR      PC,TXCHAR      ;LOAD EOM CHAR, TX 2ND 000 CHAR
8045 027362 001000                      TXEOM
8046 027364 000010                      8.
8047 027366 004737 006122              JSR      PC,TXCHAR      ;LOAD EOM, TX CRC-16 CHAR
8048 027372 001000                      TXEOM
8049 027374 000020                      16.
8050 027376 004737 006122              JSR      PC,TXCHAR      ;LOAD EOM, TX FIRST TERMINATING SYNCH
8051 027402 001000                      TXEOM
8052 027404 000010                      8.
8053 027406 004737 006122              JSR      PC,TXCHAR      ;LOAD EOM, TX 2ND TERMINATING SYNCH
8054 027412 001000                      TXEOM
8055 027414 000010                      8.
8056 027416 004737 006274              JSR      PC,ENDTRN      ;SET OC, MONITOR OCOR
8057 027422                                A4:
8058 027422 004737 003576              JSR      PC,MSTCLR      ;ISSUE MASTER CLEAR TO CLEAN UP
8059 027426                                ENDTST
(3) 027426                                L10053:
(3) 027426 104401                                TRAP      C$ETST
```

8060  
8061  
8062  
8063  
8064  
8065  
8066  
8067  
8068  
8069  
8070  
8071  
8072  
8073  
8074  
8075  
8076  
8077  
8078

```
*****
: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
```

```
(3) 027430
8080 027430 012737 027532 002362      MOV      #A5,RETADR      ;SET TEST EXIT ADDRESS FOR ERRORS      T24::
8081 027436 004737 005540              JSR      PC,INITRN      ;DO MASTER CLR, LOAD 2 SOM*2
8082 027442 000000                      000
```

8083	027444	000000		000		
8084	027446	004737	006122	JSR	PC, TXCHAR	;LOAD A 000 CHAR, TX FIRST FLAG
8085	027452	000000		000		
8086	027454	100010		CHPCHK!8.		
8087	027456	004737	006122	JSR	PC, TXCHAR	;LOAD 2ND 000 CHAR, TX 2ND FLAG
8088	027462	000000		000		
8089	027464	000010		8.		
8090	027466	004737	006122	JSR	PC, TXCHAR	;LOAD EOM CHAR, TX FIRST 000 CHAR
8091	027472	001000		TXEOM		
8092	027474	000010		8.		
8093	027476	004737	006122	JSR	PC, TXCHAR	;LOAD EOM, TX 2ND 000 CHAR AND CRC-CCITT-1 CHAR
8094	027502	001000		TXEOM		
8095	027504	000030		24.		
8096	027506	004737	006122	JSR	PC, TXCHAR	;LOAD EOM, TX FIRST TERMINATING FLAG
8097	027512	001000		TXEOM		
8098	027514	000010		8.		
8099	027516	004737	006122	JSR	PC, TXCHAR	;LOAD EOM, TX 2ND TERMINATING FLAG
8100	027522	001000		TXEOM		
8101	027524	000010		8.		
8102	027526	004737	006274	JSR	PC, ENDTRN	;SET OC, MONITOR OCOR
8103	027532					
8104	027532	004737	003576	JSR	PC, MSTCLR	;ISSUE MASTER CLEAR TO CLEAN UP
8105	027536					
(3)	027536					
(3)	027536	104401				L10054: TRAP C\$ETST

A5:  
ENDTST

8106  
8107  
8108  
8109  
8110  
8111  
8112  
8113  
8114  
8115  
8116  
8117  
8118  
8119  
8120  
8121  
8122  
8123  
8124  
8125  
8126

```
*****  
: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
```

8127	027540					
(3)	027540					T25::
8128	027540	012737	027726	002362	MOV #A6, RETADR	;SET TEST EXIT ADDRESS FOR ERRORS
8129	027546	004737	005540		JSR PC, INITRN	;DO MASTER CLR, LOAD 2 SOM*2
8130	027552	000226			SYNCH	
8131	027554	000311			CRC2!CRC1!STRIP!DDCMP	
8132	027556	012737	000014	002400	MOV #14, REGNUM	;SET REG NO. = 14
8133	027564	012737	000100	002366	MOV #TXEN, WRIBYT	;SET TXEN BIT
8134	027572	004737	003750		JSR PC, WRITLU	;LOAD TXEN BIT IN REG 14
8135	027576	004737	006122		JSR PC, TXCHAR	;LOAD A 000 CHAR, TX FIRST SYNCH (226)

```
8136 027602 000000 000  
8137 027604 100010 CHPCHK!8.  
8138 027606 004737 006122 JSR PC,TXCHAR ;LOAD 2ND 000 CHAR, TX 2ND SYNCH  
8139 027612 000000 000  
8140 027614 000010 8.  
8141 027616 004737 006122 JSR PC,TXCHAR ;LOAD EOM CHAR, TX FIRST 000 CHAR  
8142 027622 001000 TXEOM  
8143 027624 000010 8.  
8144 027626 004737 006122 JSR PC,TXCHAR ;LOAD EOM CHAR, TX 2ND 000 CHAR  
8145 027632 001000 TXEOM  
8146 027634 000010 8.  
8147 027636 004737 006122 JSR PC,TXCHAR ;LOAD EOM, TX FIRST TERMINATING SYNCH  
8148 027642 001000 TXEOM  
8149 027644 000010 8.  
8150 027646 004737 006122 JSR PC,TXCHAR ;LOAD EOM, TX 2ND TERMINATING SYNCH  
8151 027652 001000 TXEOM  
8152 027654 000010 8.  
8153 027656 004737 005254 JSR PC,STPLU ;CLK PAST END OF MSG  
8154 027662 000030 24.  
8155 027664 012737 000013 002400 MOV #13,REGNUM ;SET REG NO. = 13  
8156 027672 004737 003672 JSR PC,READLU ;READ REG 13  
8157 027676 032737 000040 002364 BIT #RTS,REDBYT ;CHK FOR RTS STILL SET  
8158 027704 001006 BNE 3$ ;BR IF RTS SET  
8159 027706 004737 004526 JSR PC,GETALL ;GET REGS FOR PRINTOUT  
8160 ;REPORT RTS NOT SET  
8161 027712 ERRDF 60,EM60,ERR7  
8162 (4) 027712 104455 TRAP C$ERDF  
8163 (5) 027714 000074 .WORD 60  
8164 (5) 027716 014230 .WORD EM60  
8165 (5) 027720 017470 .WORD ERR7  
8166 027722 004737 006274 3$: JSR PC,ENDTRN ;SET OC, MONITOR OCOR  
8167 027726 A6: JSR PC,MSTCLR ;ISSUE MASTER CLEAR TO CLEAN UP  
8168 027726 004737 003576  
8169 027732 ENDTST  
8170 (3) 027732 (3) 104401 L10055: TRAP C$ETST  
8171  
8172  
8173  
8174  
8175  
8176  
8177  
8178  
8179  
8180  
8181  
8182  
8183  
8184  
8185 027734
```

```
*****  
:SBTTL 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.  
:*****  
:BGNTST
```

T26::

```
(3) 027734
8186
8187 -----
8188 : CAUSE TX UNDERRUN, CHK UNRR = 1; SET SOM, CHK UNRR = 0
8189 027734 012737 030272 002362 MOV #A7,RETADR ;SET TEST EXIT ADDRESS FOR ERRORS
8190 027742 004737 005540 JSR PC,INITRN ;DO MASTER CLEAR, LOAD 2 SOM'S
8191 027746 000226 SYNCH
8192 027750 000011 STRIP!DDCMP
8193 027752 004737 006122 JSR PC,TXCHAR ;LOAD A 000 CHAR, TX FIRST SYNCH
8194 027756 000000 000
8195 027760 100010 CHPCHK!8.
8196 027762 004737 005254 JSR PC,STPLU ;CLOCK THE TRANSMITTER UNTIL 7 BITS OF
8197 027766 000016 14. ; THE 000 CHAR HAVE BEEN TRANSMITTED
8198 027770 012737 000011 002400 MOV #11,REGNUM ;SET LU REG NO. = 11
8199 027776 004737 003672 JSR PC,READLU ;READ REG 11
8200 030002 132737 000001 002364 BITB #UNRR,REDBYT ;CHK FOR UNRR = 0
8201 030010 001410 BEQ 6$ ;BR IF UNRR = 0
8202 030012 004737 004526 JSR PC,GETALL ;GET REGS FOR PRINTOUT
8203 ;REPORT UNRR NOT CLEARED
8204 030016 ERRDF 16,EM16,ERR7
(4) 030016 104455 TRAP C$ERDF
(5) 030020 000020 .WORD 16
(5) 030022 012622 .WORD EM16
(5) 030024 017470 .WORD ERR7
8205 030026 000137 030272 JMP A7 ;SKIP TO END OF TEST
8206 030032 004737 005254 6$: JSR PC,STPLU ;CLOCK LAST BIT OF 000 CHAR
8207 030036 000003 3
8208 030040 004737 003672 JSR PC,READLU ;READ REG 11
8209 030044 132737 000001 002364 BITB #UNRR,REDBYT ;CHK FOR UNRR = 1
8210 030052 001010 BNE 9$ ;BR IF UNRR = 1
8211 030054 004737 004526 JSR PC,GETALL ;GET REGS FOR PRINTOUT
8212 ;REPORT UNRR NOT SET
8213 030060 ERRDF 14,EM14,ERR7
(4) 030060 104455 TRAP C$ERDF
(5) 030062 000016 .WORD 14
(5) 030064 012556 .WORD EM14
(5) 030066 017470 .WORD ERR7
8214 030070 000137 030272 JMP A7 ;SKIP TO END OF TEST
8215 030074 012737 000400 002422 9$: MOV #TXSOM,TXWORD ;SET SOM CHAR TO BE WRITTEN
8216 030102 004737 005174 JSR PC,LDTXSI ;LOAD SOM CHAR INTO TX SILO
8217 030106 004737 005146 JSR PC,WAIT50 ;WAIT FOR SILO DATA TO RIPPLE
8218 030112 004737 005254 JSR PC,STPLU ;CLOCK LU FOR 2 CYCLES
8219 030116 000002 2
8220 030120 004737 003672 JSR PC,READLU ;READ REG 11
8221 030124 132737 000001 002364 BITB #UNRR,REDBYT ;CHK FOR UNRR = 0
8222 030132 001410 BEQ 12$ ;BR IF UNRR = 0
8223 030134 004737 004526 JSR PC,GETALL ;GET REGS FOR PRINTOUT
8224 ;REPORT UNRR NOT CLEARED BY SOM
8225 030140 ERRDF 13,EM13,ERR7
(4) 030140 104455 TRAP C$ERDF
(5) 030142 000015 .WORD 13
(5) 030144 012526 .WORD EM13
(5) 030146 017470 .WORD ERR7
8226 030150 000137 030272 JMP A7 ;SKIP TO END OF TEST
8227 030154 12$:
8228 -----
```

```
8229                                     :CAUSE TX UNDERRUN, CHK UNRR = 1; DO MASTER CLR, CHK UNRR = 0
8230 -----
8231 030154 004737 005540                JSR    PC,INITRN      ;DO MASTER CLEAR, LOAD 2 SOM'S
8232 030160 000226                        SYNCH
8233 030162 000051                        IDLE!STRIP!DDCMP
8234 030164 004737 006122                JSR    PC,TXCHAR     ;LOAD A 000 CHAR, TX FIRST SYNCH
8235 030170 000000                        000
8236 030172 100010                        CHPCHK!8.
8237 030174 004737 005254                JSR    PC,STPLU     ;STEP THE LU UNTIL 000 HAS BEN TRANSMITTED
8238 030200 000021                        17.
8239 030202 004737 003672                JSR    PC,READLU    ;READ REG 11
8240 030206 132737 000001 002364        BITB   #UNRR,REDBYT ;CHK FOR UNR = 1
8241 030214 001010                        BNE    16$          ;BR IF UNRR = 1
8242 030216 004737 004526                JSR    PC,GETALL    ;GET REGS FOR PRINTOUT
8243                                     :REPORT UNRR NOT SET
8244 030222                                ERRDF  14,EM14,ERR7
      (4) 030222 104455
      (5) 030224 000016                                TRAP   C$ERDF
      (5) 030226 012556                                .WORD 14
      (5) 030230 017470                                .WORD EM14
      (5) 030230 017470                                .WORD ERR7
8245 030232 000137 030272                JMP    A7           ;SKIP TO END OF TEST
8246 030236 004737 003576                16$: JSR    PC,MSTCLR ;ISSUE MASTER CLEAR
8247 030242 004737 003672                JSR    PC,READLU    ;READ REG 11
8248 030246 132737 000001 002364        BITB   #UNRR,REDBYT ;CHK FOR UNRR = 0
8249 030254 001406                        SEQ    A7           ;BR IF UNRR = 0
8250 030256 004737 004526                JSR    PC,GETALL    ;GET REGS FOR PRINTOUT
8251                                     :REPORT UNRR NOT CLEARED BY OC
8252 030262                                ERRDF  15,EM15,ERR7
      (4) 030262 104455                                TRAP   C$ERDF
      (5) 030264 000017                                .WORD 15
      (5) 030266 012573                                .WORD EM15
      (5) 030270 017470                                .WORD ERR7
8253 030272 004737 003576                A7:  JSR    PC,MSTCLR ;ISSUE CLEAN-UP MASTER CLEAR
8254 030276                                ENDTST
      (3) 030276                                L10056: TRAP   C$ETST
      (3) 030276 104401
8255
8256
8257
8258
8259
8260                                     :*****
8261 .SBTTL  TEST 27 - TRANSMIT CHAR LENGTH TIMING TEST - CHAR MODE, CRC
8262 :*
8263 :* THE LINE UNIT IS PLACED IN CHAR MODE (DDCMP) AND A MESSAGE IS INITIATED
8264 :* WITH AN 8-BIT SYNCH AND A 5-BIT SYNCH CHAR. NEXT, A 000 CHAR IS SENT WITH
8265 :* EACH OF THE FOLLOWING TX CHAR LENGTHS : 5 BITS, 6 BITS, 7 BITS, 8 BITS.
8266 :* (FOR EXAMPLE, A 5-BIT CHAR REQUIRES 5 CLOCK CYCLES TO BE TRANSMITTED). TWO
8267 :* TERMINATING SYNCHS ARE SENT AFTER THE DATA.
8268 :*****
8269 BGNTST
      (3) 030300
8270 030300 012737 030506 002362        MOV    #A8,RETADR   ;SET TEST EXIT ADDRESS FOR ERRORS
8271 030306 004737 005540                JSR    PC,INITRN   ;DO MASTER CLR, LOAD 2 SOM'S
8272 030312 000000                        000
8273 030314 000041                        IDLE!DDCMP
      T27::
```

8274	030316	012737	000006	002402	MOV	#6,AXNUM	;SET BYTE NO. = 6 FOR AX3
8275	030324	012737	000000	002374	MOV	#000,WAX15	;SET DATA FOR AX3-15 = 0
8276	030332	012737	000240	002376	MOV	#TXLEN2!TXLENO,WAX16	;SET TX LENGTH = 5 FOR AX3-16
8277	030340	004737	004312		JSR	PC,WRITAX	;LOAD AX3-15,AX3-16
8278	030344	004737	006122		JSR	PC,TXCHAR	;LOAD 5-BIT 000 CHAR, TX 8-BIT SYNCH
8279	030350	000000			000		
8280	030352	100010			CHPCHK!8.		
8281	030354	004737	006122		JSR	PC,TXCHAR	;LOAD 6-BIT 000 CHAR, TX 5-BIT SYNCH
8282	030360	000000			000		
8283	030362	000005			5		
8284	030364	012737	000300	002376	MOV	#TXLEN2!TXLEN1,WAX16	
8285	030372	004737	004312		JSR	PC,WRITAX	;SET TX CHAR LENGTH = 6
8286	030376	004737	006122		JSR	PC,TXCHAR	;LOAD 7-BIT 000 CHAR, TX 5-BIT 000 CHAR
8287	030402	000000			000		
8288	030404	000005			5		
8289	030406	012737	000340	002376	MOV	#TXLEN2!TXLEN1!TXLENO,WAX16	
8290	030414	004737	004312		JSR	PC,WRITAX	;SET TX CHAR LENGTH = 7
8291	030420	004737	006122		JSR	PC,TXCHAR	;LOAD 8-BIT 000 CHAR, TX 6-BIT 000 CHAR
8292	030424	000000			000		
8293	030426	000006			6		
8294	030430	012737	000000	002376	MOV	#000,WAX16	
8295	030436	004737	004312		JSR	PC,WRITAX	;SET TX CHAR LENGTH = 8
8296	030442	004737	006122		JSR	PC,TXCHAR	;LOAD EOM, TX 7-BIT 000 CHAR
8297	030446	001000			TXEOM		
8298	030450	000007			7		
8299	030452	004737	006122		JSR	PC,TXCHAR	;LOAD EOM, TX 8-BIT 000 CHAR
8300	030456	001000			TXEOM		
8301	030460	000010			8.		
8302	030462	004737	006122		JSR	PC,TXCHAR	;LOAD EOM, TX CRC-16 CHAR
8303	030466	001000			TXEOM		
8304	030470	000020			16.		
8305	030472	004737	006122		JSR	PC,TXCHAR	;LOAD EOM, TX FIRST TERMINATING SYNCH
8306	030476	001000			TXEOM		
8307	030500	000010			8.		
8308	030502	004737	006274		JSR	PC,ENDTRN	;CLEAR TRANSMITTER
8309	030506						
8310	030506						
(3)	030506						
(3)	030506	104401					

A8:  
ENDTST

L10057:  
TRAP C\$ETST

8311  
8312  
8313  
8314  
8315  
8316  
8317  
8318  
8319  
8320  
8321  
8322  
8323  
8324  
8325  
8326 030510  
(3) 030510

\*\*\*\*\*  
: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 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.  
:\*\*\*\*\*  
:BGNTST

T28::

8327	030510	012737	031036	002362	MOV	#A9,RETADR	:SET TEST EXIT ADDRESS FOR ERRORS
8328	030516	004737	005540		JSR	PC,INITRN	:DO MASTER CLR, LOAD 2 SOM'S
8329	030522	000000			000		
8330	030524	000000			000		
8331	030526	004737	006122		JSR	PC, TXCHAR	:LOAD FIRST 8-BIT 000 CHAR, TX FIRST FLAG
8332	030532	000000			000		
8333	030534	100010			CHPCHK!8.		
8334	030536	004737	006122		JSR	PC, TXCHAR	:LOAD 2ND 8-BIT 000 CHAR, TX 2ND FLAG
8335	030542	000000			000		
8336	030544	000010			8.		
8337	030546	004737	006122		JSR	PC, TXCHAR	:LOAD 1-BIT 000 CHAR, TX FIRST 8-BIT 000 CHAR
8338	030552	000000			000		
8339	030554	000010			8.		
8340	030556	012737	000006	002402	MOV	#6,AXNUM	:SET BYTE NO. = 6 FOR AX3
8341	030564	012737	000000	002374	MOV	#000,WAX15	:SET DATA FOR AX3-15 = 0
8342	030572	012737	000040	002376	MOV	#TXLENO,WAX16	:SET TX CHAR LENGTH = 1 FOR AX3-16
8343	030600	004737	004312		JSR	PC,WRITAX	:LOAD AX3-15,AX3-16
8344	030604	004737	006122		JSR	PC, TXCHAR	:LOAD 2-BIT 000 CHAR, TX 2ND 8-BIT 000 CHAR
8345	030610	000000			000		
8346	030612	000010			8.		
8347	030614	012737	000100	002376	MOV	#TXLEN1,WAX16	
8348	030622	004737	004312		JSR	PC,WRITAX	:SET TX CHAR LENGTH = 2
8349	030626	004737	006122		JSR	PC, TXCHAR	:LOAD 3-BIT 000 CHAR, TX 1-BIT 000 CHAR
8350	030632	000000			000		
8351	030634	000001			1		
8352	030636	012737	000140	002376	MOV	#TXLEN1!TXLENO,WAX16	
8353	030644	004737	004312		JSR	PC,WRITAX	:SET TX CHAR LENGTH = 3
8354	030650	004737	006122		JSR	PC, TXCHAR	:LOAD 4-BIT 000 CHAR, TX 2-BIT 000 CHAR
8355	030654	000000			000		
8356	030656	000002			2		
8357	030660	012737	000200	002376	MOV	#TXLEN2,WAX16	
8358	030666	004737	004312		JSR	PC,WRITAX	:SET TX CHAR LENGTH = 4
8359	030672	004737	006122		JSR	PC, TXCHAR	:LOAD 5-BIT 000 CHAR, TX 3-BIT 000 CHAR
8360	030676	000000			000		
8361	030700	000003			3		
8362	030702	012737	000240	002376	MOV	#TXLEN2!TXLENO,WAX16	
8363	030710	004737	004312		JSR	PC,WRITAX	:SET TX CHAR LENGTH = 5
8364	030714	004737	006122		JSR	PC, TXCHAR	:LOAD 6-BIT 000 CHAR, TX 4-BIT 000 CHAR
8365	030720	000000			000		
8366	030722	000004			4		
8367	030724	012737	000300	002376	MOV	#TXLEN2!TXLEN1,WAX16	
8368	030732	004737	004312		JSR	PC,WRITAX	:SET TX CHAR LENGTH = 6
8369	030736	004737	006122		JSR	PC, TXCHAR	:LOAD 7-BIT 000 CHAR, TX 5-BIT 000 CHAR
8370	030742	000000			000		
8371	030744	000005			5		
8372	030746	012737	000340	002376	MOV	#TXLEN2!TXLEN1!TXLENO,WAX16	
8373	030754	004737	004312		JSR	PC,WRITAX	:SET TX CHAR LENGTH = 7
8374	030760	004737	006122		JSR	PC, TXCHAR	:LOAD 8-BIT 000 CHAR, TX 6-BIT 000 CHAR
8375	030764	000000			000		
8376	030766	000006			6		
8377	030770	012737	000000	002376	MOV	#000,WAX16	
8378	030776	004737	004312		JSR	PC,WRITAX	:SET TX CHAR LENGTH = 8
8379	031002	004737	006122		JSR	PC, TXCHAR	:LOAD EOM, TX 7-BIT 000 CHAR
8380	031006	001000			TXEOM		
8381	031010	000007			7		
8382	031012	004737	006122		JSR	PC, TXCHAR	:LOAD EOM, TX 8-BIT 000 CHAR, CRC-CCITT-1 CHAR



8383 031016 001000  
8384 031020 000031  
8385 031022 004737 006122  
8386 031026 001000  
8387 031030 000010  
8388 031032 004737 006274  
8389 031036  
8390 031036  
(3) 031036  
(3) 031036 104401

TXEOM  
25.  
JSR PC, TXCHAR ;LOAD EOM, TX FIRST TERMINATING FLAG  
TXEOM  
8.  
JSR PC, ENDTRN ;CLEAR TRANSMITTER  
A9:  
ENDTST

L10060: TRAP C\$ETST

8391  
8392  
8393  
8394  
8395  
8396  
8397  
8398  
8399  
8400  
8401  
8402  
8403  
8404

\*\*\*\*\*  
:SBTTL TEST 29 - TXDATA BIT TEST - CHAR MODE, 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.  
:\*\*\*\*\*

8405 031040  
(3) 031040  
8406 031040 004737 005540  
8407 031044 000226  
8408 031046 000011  
8409 031050 012701 003224  
8410 031054 010103  
8411 031056 012137 002422  
8412 031062 004737 005174  
8413 031066 020127 003242  
8414 031072 103771  
8415 031074 004737 005146  
8416 031100 004737 005254  
8417 031104 100020  
8418 031106 011337 031116  
8419 031112 004737 011176  
8420 031116 000000  
8421 031120 062703 000002  
8422 031124 020327 003236  
8423 031130 103766  
8424 031132 004737 003576  
8425 031136  
(3) 031136  
(3) 031136 104401

BGNTST  
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 C\$ETST

8426  
8427  
8428  
8429  
8430  
8431  
8432  
8433

\*\*\*\*\*  
:SBTTL TEST 30 - USYRT RECEIVER MSG TEST - CHAR MODE, CRC  
:\*

8434  
 8435  
 8436  
 8437  
 8438  
 8439  
 8440  
 8441  
 (3)  
 8442  
 8443  
 8444  
 8445  
 8446  
 8447  
 8448  
 8449  
 8450  
 8451  
 8452  
 8453  
 8454  
 8455  
 8456  
 8457  
 8458  
 8459  
 8460  
 8461  
 8462  
 8463  
 8464  
 8465  
 8466  
 8467  
 8468  
 8469  
 8470  
 8471  
 8472  
 8473  
 8474  
 8475  
 (4)  
 (5)  
 (5)  
 (5)  
 8476  
 8477  
 8478  
 8479  
 8480  
 8481  
 8482  
 8483  
 8484

031140  
 031140  
 031140 012737 031502 002362  
 031146 004737 003576  
 031152 004737 010640  
 031156 000226  
 031160 000013  
 031162 000000  
 031164 000000  
 031166 012737 000012 002400  
 031174 005037 002402  
 031200 112737 000040 002366  
 031206 004737 003750  
 031212 012701 003220  
 031216 012137 002422  
 031222 004737 005174  
 031226 020127 003246  
 031232 103771  
 031234 004737 005146  
 031240 004737 005254  
 031244 000050  
 031246 004737 006714  
 031252 000000  
 031254 004737 005254  
 031260 000006  
 031262 012701 003224  
 031266 004737 006714  
 031272 000001  
 031274 004737 004124  
 031300 023721 002370  
 031304 001415  
 031306 016137 177776 002404  
 031314 013737 002370 002406  
 031322 004737 004526  
 031326  
 031326 104455  
 031330 000032  
 031332 013067  
 031334 015140  
 031336 000461  
 031340 004737 005254  
 031344 000010  
 031346 020127 003236  
 031352 103745  
 031354 004737 006714  
 031360 000001  
 031362 004737 004124  
 031366 123727 002370 000160

```

;* 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
;* STILL SET AFTER THE MESSAGE.
:*****
BGNTST
T30::
MOV #24$,RETADR ;SET TEST EXIT ADDRESS FOR ERRORS
JSR PC,MSTCLR ;ISSUE MASTER CLEAR
JSR PC,SETUP ;PROGRAM THE USYRT
SYNCH
STRIP! IERR!DDCMP
000
000
MOV #12,REGNUM ;SET LU REG NO. = 12
CLR AXNUM ;SET AX BYTE NO. = 0 FOR AX0
MOVB #LULP,WRIBYT
JSR PC,WRITLU ;SET LULP IN REG 12
MOV #MSG1,R1 ;GET POINTER TO MSG DATA TABLE
3$: MOV (R1)+,TXWORD ;GET CHAR TO BE LOADED
JSR PC,LDTXSI ;LOAD CHAR INTO TX SILO
CMP R1,#MSG1+22. ;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 40 CYCLES (UNTIL FIRST
; DATA CHAR IS ABOUT TO BE RECEIVED)
JSR PC,IACTIV ;CHK IACT = 0
0
JSR PC,STPLU ;CLOCK LU UNTIL 1ST DATA CHAR IS RCV'D
6
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,BADDAT ;GET ACTUAL DATA
JSR PC,GETALL ;GET REGS FOR PRINTOUT
;REPORT INCORRECT DATA CHAR RCV'D
ERRDF 26,EM26,ERR3
TRAP C$ERDF
.WORD 26
.WORD EM26
.WORD ERR3
12$: BR 24$
JSR PC,STPLU ;CLOCK LU 8 CYCLES
8.
CMP R1,#MSG1+14. ;SEE IF CHECKING HI CRC BYTE YET
BLO 10$ ;BR IF NOT YET
JSR PC,IACTIV ;CHK IACT = 1
1
JSR PC,READAX ;READ AX0
CMPB RAX15,#160 ;CMP RCV'D CHAR TO EXPECTED HI CRC BYTE
  
```

```

8485 031374 001415 BEQ 16$ ;BR IF HI CRC BYTE RCV'D OK
8486 031376 012737 000160 002404 MOV #160,GOODAT ;GET EXPECTED DATA
8487 031404 013737 002370 002406 14$: MOV RAX15,BADDAT ;SET ACTUAL DATA
8488 031412 004737 004526 JSR PC,GETALL ;GET REGS FOR PRINTOUT
8489 ;REPORT INCORRECT CRC BYTE RCV'D
8490 ERRDF 27,EM27,ERR3
(4) 031416 104455 TRAP C$ERRDF
(5) 031420 000033 .WORD 27
(5) 031422 013121 .WORD EM27
(5) 031424 015140 .WORD ERR3
8491 031426 000425 BR 24$
8492 031430 004737 005254 16$: JSR PC,STPLU ;CLOCK LU FOR 8 CYCLES
8493 031434 000010 8.
8494 031436 004737 006714 JSR PC,IACTIV ;CHK IACT = 1
8495 031442 000001 1
8496 031444 012737 000034 002404 MOV #034,GOODAT ;GET EXPECTED LO CRC BYTE
8497 031452 004737 004124 JSR PC,READAX ;READ AX0
8498 031456 123727 002370 000034 CMPB RAX15,#034 ;CMP RCV'D CHAR TO EXPECTED LO CRC BYTE
8499 031464 001347 BNE 14$ ;BR IF LO CRC INCORRECT
8500 031466 004737 005254 JSR PC,STPLU ;CLOCK LU 8 CYCLES
8501 031472 000010 8.
8502 031474 004737 006714 JSR PC,IACTIV ;CHK IACT STILL = 1
8503 031500 000001 1
8504 031502 004737 003576 24$: JSR PC,MSTCLR ;ISSUE CLEAN-UP MASTER CLEAR
8505 031506 ENDTST
(3) 031506 L10062: TRAP C$SETST
(3) 031506 104401

```

8506  
8507  
8508  
8509  
8510

```

*****
: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.
*****
BGNST

```

```

8521 031510 T31::
(3) 031510 ;SET TEST EXIT ADDRESS FOR ERRORS
8522 031510 012737 032044 002362 MOV #24$,RETADR ;ISSUE MASTER CLEAR
8523 031516 004737 003576 JSR PC,MSTCLR ;PROGRAM THE USYRT
8524 031522 004737 010640 JSR PC,SETUP
8525 031526 000000 000
8526 031530 000002 IERR
8527 031532 000000 000
8528 031534 000000 000
8529 031536 012737 000012 002400 MOV #12,REGNUM ;SET LU REG NO. = 12
8530 031544 005037 002402 CLR AXNUM ;SET AX BYTE NO. = 0 FOR AX0
8531 031550 112737 000040 002366 MOVB #LULP,WRIBYT
8532 031556 004737 003750 JSR PC,WRITLU ;SET LULP IN REG 12
8533 031562 012701 003220 MOV #MSG1,R1 ;GET POINTER TO MSG DATA TABLE

```

8534	031566	012137	002422	3\$:	MOV	(R1)+,TXWORD	:GET CHAR TO BE LOADED		
8535	031572	004737	005174		JSR	PC,LDTXSI	:LOAD CHAR INTO TX SILO		
8536	031576	020127	003242		CMP	R1,#MSG1+18.	:SEE IF ALL MSG CHARS LOADED YET		
8537	031602	103771			BLO	3\$	:BR IF NOT YET		
8538	031604	004737	005146		JSR	PC,WAIT50	:ALLOW DATA TO RIPPLE IN SILO		
8539	031610	004737	005254		JSR	PC,STPLU	:CLOCK LU FOR 50 CYCLES (UNTIL FIRST		
8540	031614	000062			50.		: DATA CHAR IS ABOUT TO BE RECEIVED)		
8541	031616	004737	006714		JSR	PC,IACTIV	:CHK IACT = 0		
8542	031622	000000			0				
8543	031624	004737	007102		JSR	PC,RSEOM	:CHK RSOM = 0, REOM = 0		
8544	031630	000000			0				
8545	031632	004737	005254		JSR	PC,STPLU	:CLOCK LU UNTIL 1ST DATA CHAR IS RCV'D		
8546	031636	000006			6				
8547	031640	012701	003224		MOV	#MSG1+4,R1			
8548	031644	020127	003224	5\$:	CMP	R1,#MSG1+4	:SEE IF 1ST CHAR RCV'D		
8549	031650	001007			BNE	6\$	:BR IF NO		
8550	031652	004737	006714		JSR	PC,IACTIV	:CHK IACT = 1		
8551	031656	000001			1				
8552	031660	004737	007102		JSR	PC,RSEOM	:CHK RSOM = 1, REOM = 0		
8553	031664	000001			1				
8554	031666	000420			BR	9\$			
8555	031670	020127	003234	6\$:	CMP	R1,#MSG1+12.	:SEE IF LAST CHAR RCV'D		
8556	031674	001007			BNE	8\$	:BR IF NO		
8557	031676	004737	006714		JSR	PC,IACTIV	:CHK FOR IACT = 0		
8558	031702	000000			0				
8559	031704	004737	007102		JSR	PC,RSEOM	:CHK RSOM = 0, REOM = 0		
8560	031710	000000			0				
8561	031712	000406			BR	9\$			
8562	031714	004737	006714	8\$:	JSR	PC,IACTIV	:CHK FOR IACT = 1		
8563	031720	000001			1				
8564	031722	004737	007102		JSR	PC,RSEOM	:CHK RSOM = 0, REOM = 0		
8565	031726	000000			0				
8566	031730	004737	004124	9\$:	JSR	PC,READAX	:READ AXO		
8567	031734	023721	002370		CMP	RAX15,(R1)+	:COMPARE RCV'D CHAR TO EXPECTED		
8568	031740	001415			BEQ	12\$	:BR IF RCV'D DATA OK		
8569	031742	016137	177776	002404	MOV	-2(R1),GOODAT	:GET EXPECTED DATA		
8570	031750	013737	002370	002406	MOV	RAX15,BADDAT	:GET ACTUAL DATA		
8571	031756	004737	004526		JSR	PC,GETALL	:GET REGS FOR PRINTOUT		
8572					:REPORT	INCORRECT DATA CHAR RCV'D			
8573	031762				ERRDF	26,EM26,ERR3			
(4)	031762	104455						TRAP	C\$ERDF
(5)	031764	000032						.WORD	26
(5)	031766	013067						.WORD	EM26
(5)	031770	015140						.WORD	ERR3
8574	031772	000424			BR	24\$			
8575	031774	004737	005254	12\$:	JSR	PC,STPLU	:CLOCK LU 8 CYCLES		
8576	032000	000010			8.				
8577	032002	020127	003236		CMP	R1,#MSG1+14.	:SEE IF ALL DATA CHARS CHECKED YET		
8578	032006	103716			BLO	5\$	:BR IF NOT YET		
8579	032010	004737	006714		JSR	PC,IACTIV	:CHK IACT = 0		
8580	032014	000000			0				
8581	032016	004737	007102		JSR	PC,RSEOM	:CHK RSOM = 0, REOM = 0		
8582	032022	000000			0				
8583	032024	012737	000200	002366	MOV	#IC,WRIBYT			
8584	032032	004737	003750		JSR	PC,WRITLU	:SET IC (INPUT CLEAR) IN REG 12		
8585	032036	004737	006714		JSR	PC,IACTIV	:CHK IACT = 0		

8586 032042 000000  
8587 032044 004737 003576  
8588 032050  
(3) 032050  
(3) 032050 104401

0  
24\$: JSR PC,MSTCLR :ISSUE CLEAN-UP MASTER CLEAR  
ENDTST

L10063: TRAP C\$ETST

8589  
8590  
8591  
8592  
8593  
8594  
8595  
8596  
8597  
8598  
8599  
8600  
8601  
8602  
8603

\*\*\*\*\*  
: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.  
:\*\*\*\*\*  
:BGNTST

8604 032052  
(3) 032052  
8605 032052 012737 032314 002362  
8606 032060 004737 003576  
8607 032064 004737 010640  
8608 032070 000226  
8609 032072 000313  
8610 032074 000000  
8611 032076 000000  
8612 032100 012737 000012 002400  
8613 032106 005037 002402  
8614 032112 112737 000040 002366  
8615 032120 004737 003750  
8616 032124 012701 003220  
8617 032130 012137 002422 3\$:  
8618 032134 004737 005174  
8619 032140 020127 003242  
8620 032144 103771  
8621 032146 004737 005146  
8622 032152 004737 005254  
8623 032156 000030  
8624 032160 004737 006714  
8625 032164 000000  
8626 032166 004737 005254  
8627 032172 000006  
8628 032174 012701 003224  
8629 032200 004737 006714 10\$:  
8630 032204 000001  
8631 032206 004737 004124  
8632 032212 023721 002370  
8633 032216 001415  
8634 032220 016137 177776 002404  
8635 032226 013737 002370 002406  
8636 032234 004737 004526  
8637  
8638 032240

T32::  
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  
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 24 CYCLES (UNTIL FIRST  
: DATA CHAR IS ABOUT TO BE RECEIVED)  
JSR PC,IACTIV :CHK IACT = 0  
0  
JSR PC,STPLU :CLOCK LU UNTIL 1ST DATA CHAR IS RCV'D  
6  
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,BADDAT :GET ACTUAL DATA  
JSR PC,GETALL :GET REGS FOR PRINTOUT  
:REPORT INCORRECT DATA CHAR RCV'D  
ERRDF 26,EM26,ERR3

(4)	032240	104455							TRAP	C\$ERDF
(5)	032242	000032							.WORD	26
(5)	032244	013067							.WORD	EM26
(5)	032246	015140							.WORD	ERR3
8639	032250	000421								
8640	032252	004737	005254	12\$:	BR	24\$				
8641	032256	000010			JSR	PC,STPLU				;CLOCK LU 8 CYCLES
8642	032260	020127	003236		8.					
8643	032264	103745			CMP	R1,#MSG1+14.				;SEE IF ALL 5 DATA CHARS RCV'D YET
8644	032266	004737	006714		BLO	10\$				;BR IF NOT YET
8645	032272	000001			JSR	PC,IACTIV				;CHK FOR IACT STILL = 1
8646	032274	012737	000200	002366	1					
8647	032302	004737	003750		MOV	#IC,WRIBYT				
8648	032306	004737	006714		JSR	PC,WRITLU				;SET IC (INPUT CLEAR) IN REG 12
8649	032312	000000			JSR	PC,IACTIV				;CHK IACT = 0
8650	032314	004737	003576		0					
8651	032320				24\$:	JSR	PC,MSTCLR			;ISSUE CLEAN-UP MASTER CLEAR
(3)	032320				ENDTST					
(3)	032320	104401							L10064:	
8652									TRAP	C\$ETST

```

:*****
: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

```

8667	032322									
(3)	032322									T33::
8668	032322	012737	032656	002362	MOV	#24\$,RETADR				
8669	032330	004737	003576		JSR	PC,MSTCLR				;ISSUE MASTER CLEAR
8670	032334	004737	010640		JSR	PC,SETUP				;PROGRAM THE USYRT
8671	032340	000000			000					
8672	032342	000302			CRC2!CRC1!IERR					
8673	032344	000000			000					
8674	032346	000000			000					
8675	032350	012737	000012	002400	MOV	#12,REGNUM				;SET LU REG NO. = 12
8676	032356	005037	002402		CLR	AXNUM				;SET AX BYTE NO. = 0 FOR AX0
8677	032362	112737	000040	002366	MOV#	#LULP,WRIBYT				
8678	032370	004737	003750		JSR	PC,WRITLU				;SET LULP IN REG 12
8679	032374	012701	003220		MOV	#MSG1,R1				;GET POINTER TO MSG DATA TABLE
8680	032400	012137	002422		3\$:	MOV	(R1)+,TXWORD			;GET CHAR TO BE LOADED
8681	032404	004737	005174		JSR	PC,LDTXSI				;LOAD CHAR INTO TX SILO
8682	032410	020127	003242		CMP	R1,#MSG1+18.				;SEE IF ALL MSG CHARS LOADED YET
8683	032414	103771			BLO	3\$				;BR IF NOT YET
8684	032416	004737	005146		JSR	PC,WAIT50				;ALLOW DATA TO RIPPLE IN SILO
8685	032422	004737	005254		JSR	PC,STPLU				;CLOCK LU FOR 33 CYCLES (UNTIL FIRST
8686	032426	000041			33.					; DATA CHAR IS ABOUT TO BE RECEIVED)
8687	032430	004737	006714		JSR	PC,IACTIV				;CHK IACT = 0



8738  
8739  
8740  
8741  
8742  
8743  
8744  
8745  
8746  
8747  
(3)  
8748  
8749  
8750  
8751  
8752  
8753  
8754  
8755  
8756  
8757  
8758  
8759  
8760  
8761  
8762  
8763  
(4)  
(5)  
(5)  
(5)  
8764  
8765  
8766  
(3)  
(3)  
8767  
8768  
8769  
8770  
8771  
8772  
8773  
8774  
8775  
8776  
8777  
8778  
8779  
8780  
8781  
8782  
8783  
(3)  
8784  
8785

032664  
032664  
032664 004737 003576  
032670 012737 000014 002400  
032676 012737 000040 002366  
032704 004737 003750  
032710 012737 000040 002426  
032716 012737 000125 002422  
032724 004737 005174  
032730 012737 000002 002402  
032736 004737 004124  
032742 123727 002370 000000  
032750 001414  
032752 012737 000000 002404  
032760 013737 002370 002406  
032766 004737 004526  
032772  
(4) 032772 104455  
(5) 032774 000003  
(5) 032776 012220  
(5) 033000 015140  
033002 005037 002426  
033006 004737 003576  
033012  
(3) 033012  
(3) 033012 104401

\*\*\*\*\*  
: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



8786	033026	000000			0		
8787	033030	000000			0		
8788	033032	004737	003576		JSR	PC,MSTCLR	:ISSUE MASTER CLEAR
8789	033036	004737	010640		JSR	PC,SETUP	:PROGRAM THE USYRT
8790	033042	000000			000		
8791	033044	000302			CRC2!CRC1!IERR		
8792	033046	000000			000		
8793	033050	000000			000		
8794	033052	012737	000014	002400	MOV	#14,REGNUM	:SET REG NO. = 14
8795	033060	012737	000140	002366	MOV	#TXEN!DISSI,WRIBYT	
8796	033066	004737	003750		JSR	PC,WRITLU	:SET TXEN AND DISSI IN REG 14
8797	033072	012737	000140	002426	MOV	#TXEN!DISSI,DISILO	:SET DISABLE SILO FLAG
8798	033100	012737	000012	002400	MOV	#12,REGNUM	:SET LU REG NO. = 12
8799	033106	112737	000040	002366	MOVB	#LULP,WRIBYT	
8800	033114	004737	003750		JSR	PC,WRITLU	:SET LULP IN REG 12
8801	033120	012701	003220		MOV	#MSG1,R1	:GET POINTER TO MSG
8802	033124	004737	004662		JSR	PC,OSIRDY	:CHK ORDY = 1
8803	033130	000001			1		
8804	033132	012737	000002	002402	MOV	#2,AXNUM	:SET AX BYTE NO. FOR AX1
8805	033140	112137	002374		MOVB	(R1)+,WAX15	:GET A CHAR
8806	033144	112137	002376		MOVB	(R1)+,WAX16	
8807	033150	004737	004312		JSR	PC,WRITAX	:LOAD CHAR INTO USYRT TX BUFFER
8808	033154	004737	004662		JSR	PC,OSIRDY	:CHK ORDY = 0
8809	033160	000000			0		
8810	033162	004737	005352		JSR	PC,OACTIV	:CHK OACT = 0
8811	033166	000000			0		
8812	033170	004737	005254		JSR	PC,STPLU	:CLOCK LU FOR 3 CYCLES
8813	033174	000003			3		
8814	033176	004737	005352		JSR	PC,OACTIV	:CHK OACT = 1
8815	033202	000001			1		
8816	033204	012703	000004		MOV	#4,R3	:INIT COUNTER
8817	033210	112137	002374	4\$:	MOVB	(R1)+,WAX15	:GET ANOTHER CHAR
8818	033214	112137	002376		MOVB	(R1)+,WAX16	
8819	033220	020327	000001		CMP	R3,#1	:SEE IF LOADING LAST DATA CHAR YET
8820	033224	001006			BNE	5\$	:BR IF NOT
8821	033226	005737	002430		TST	CHPTYP	:SEE IF SIG USYRT
8822	033232	001403			BEQ	5\$	:BR IF YES
8823	033234	112737	000002	002376	MOVB	#TEOM,WAX16	:SET TEOM WITH LAST DATA CHAR
8824	033242	004737	004662	5\$:	JSR	PC,OSIRDY	:CHK ORDY = 1
8825	033246	000001			1		
8826	033250	004737	004312		JSR	PC,WRITAX	:LOAD ANOTHER CHAR INTO USYRT TX BUFFER
8827	033254	004737	004662		JSR	PC,OSIRDY	:CHK ORDY = 0
8828	033260	000000			0		
8829	033262	004737	006714		JSR	PC,IACTIV	:CHK IACT = 0
8830	033266	000000			0		
8831	033270	004737	007102		JSR	PC,RSEOM	:CHK RSOM = 0, REOM = 0
8832	033274	000000			0		
8833	033276	004737	005254		JSR	PC,STPLU	:CLOCK LU FOR 8 CYCLES
8834	033302	000010			8		
8835	033304	004737	005352		JSR	PC,OACTIV	:CHK OACT = 1
8836	033310	000001			1		
8837	033312	004737	004662		JSR	PC,OSIRDY	:CHK ORDY = 1
8838	033316	000001			1		
8839	033320	005303			DEC	R3	:DECR COUNTER
8840	033322	001332			BNE	4\$	:BR IF NOT DONE YET
8841	033324	004737	006430		JSR	PC,ISIRDY	:CHK IRDY = 0

```

8842 033330 000000 0
8843 033332 005737 002430 TST CHPTYP ;SEE IF SIG USYRT
8844 033336 001007 BNE 11$ ;BR IF NOT
8845 033340 105037 002374 CLR B WAX15
8846 033344 112737 000002 002376 MOVB #TEOM,WAX16 ;LOAD EOM CHAR
8847 033352 004737 004312 JSR PC,WRITAX ;LOAD ANOTHER CHAR INTO USYRT TX BUFFER
8848 033356 004737 005254 11$: JSR PC,STPLU ;CLOCK LU FOR 3 CYCLES
8849 033362 000003 3
8850 033364 004737 006430 JSR PC,ISIRDY ;CHK IRDY = 1
8851 033370 000002 2
8852 033372 004737 005352 JSR PC,DACTIV ;CHK OACT = 1
8853 033376 000001 1
8854 033400 004737 006714 JSR PC,IACTIV ;CHK IACT = 1
8855 033404 000001 1
8856 033406 004737 007102 JSR PC,RSEOM ;CHK RSOM = 1, REOM = 0
8857 033412 000001 1
8858 033414 004737 006430 JSR PC,ISIRDY ;CHK IRDY = 0
8859 033420 000000 0
8860 033422 012737 000000 002402 MOV #0,AXNUM ;SET AX BYTE NO. FOR AX0
8861 033430 123727 002370 000000 CMPB RAX15,#000 ;COMPARE RCV'D CHAR TO 000
8862 033436 001415 BEQ 9$ ;BR IF MATCH
8863 033440 012737 000000 002404 MOV #0,GOODAT ;SET EXPECTED DATA
8864 033446 013737 002370 002406 6$: MOV RAX15,BADDAT ;SET ACTUAL DATA
8865 033454 004737 004526 JSR PC,GETALL ;GET REGS FOR PRINTOUT
8866 ;REPORT INCORRECT DATA CHAR RCV'D
8867 033460 ERRDF 26,EM26,ERR3
(4) 033460 104455
(5) 033462 000032
(5) 033464 013067
(5) 033466 015140
8868 033470 000511
8869 033472 004737 005254 9$: JSR PC,STPLU ;CLOCK LU FOR 8 CYCLES
8870 033476 000010 8.
8871 033500 004737 006430 JSR PC,ISIRDY ;CHK IRDY = 1
8872 033504 000002 2
8873 033506 004737 005352 JSR PC,DACTIV ;CHK OACT = 1
8874 033512 000001 1
8875 033514 004737 006714 JSR PC,IACTIV ;CHK IACT = 1
8876 033520 000001 1
8877 033522 004737 007102 JSR PC,RSEOM ;CHK RSOM = 0, REOM = 0
8878 033526 000000 0
8879 033530 004737 006430 JSR PC,ISIRDY ;CHK IRDY = 0
8880 033534 000000 0
8881 033536 123727 002370 000125 CMPB RAX15,#125 ;COMPARE 2ND RCV'D CHAR TO 125
8882 033544 001404 BEQ 12$ ;BR IF MATCH
8883 033546 012737 000125 002404 MOV #125,GOODAT ;SET EXPECTED DATA
8884 033554 000734 BR 6$ ;BR TO REPORT ERROR
8885 033556 004737 005254 12$: JSR PC,STPLU ;CLOCK LU FOR 8 CYCLES
8886 033562 000010 8.
8887 033564 004737 006430 JSR PC,ISIRDY ;CHK IRDY = 1
8888 033570 000002 2
8889 033572 004737 005352 JSR PC,DACTIV ;CHK OACT = 1
8890 033576 000001 1
8891 033600 004737 006714 JSR PC,IACTIV ;CHK IACT = 0
8892 033604 000000 0
8893 033606 004737 007102 JSR PC,RSEOM ;CHK RSOM = 0, REOM = 1

```

```

TRAP C$ERDF
.WORD 26
.WORD EM26
.WORD ERR3

```

```

8894 033612 000002
8895 033614 004737 006430
8896 033620 000000
8897 033622 123727 002370 000252
8898 033630 001404
8899 033632 012737 000252 002404
8900 033640 000702
8901 033642 012737 000014 002400 14$:
8902 033650 012737 000040 002366
8903 033656 004737 003750
8904 033662 012737 000011 002400
8905 033670 012737 000200 002366
8906 033676 004737 003750
8907 033702 004737 005146
8908 033706 004737 005352
8909 033712 000000
8910 033714 005037 002426 18$:
8911 033720 004737 003576
8912 033724
(3) 033724
(3) 033724 104401

```

```

2
JSR PC,ISIRDY ;CHK IRDY = 0
0
CMPB RAX15,#252 ;COMPARE 3RD RCV'D CHAR TO 252
BEQ 14$ ;BR IF MATCH
MOV #252,GOODAT ;SET EXPECTED DATA
BR 6$ ;BR TO REPORT ERROR
MOV #14,REGNUM ;SET REG NO. = 14
MOV #DISSI,WRIBYT
JSR PC,WRITLU ;CLEAR TX ENABLE
MOV #11,REGNUM ;SET REG NO. = 11
MOV #OC,WRIBYT
JSR PC,WRITLU ;SET OC TO SHUT DOWN TRANSMITTER
JSR PC,WAIT50 ;WAIT FOR SHUTDOWN
JSR PC,OACTION ;CHK FOR OACT = 0
0
CLR DISILO ;CLEAR DISABLE SILO FLAG
JSR PC,MSTCLR ;ISSUE MASTER CLEAR TO CLEAN UP

```

ENDTST

L10067: TRAP C\$ETST

```

8913
8914
8915
8916
8917
8918
8919
8920
8921
8922
8923
8924
8925
8926
8927
8928
8929
8930
8931
8932
8933
8934
8935
8936 033726
(3) 033726
8937 033726 012737 034202 002362
8938
8939
8940
8941 033734 004737 003576
8942 033740 004737 006430
8943 033744 000001
8944
8945
8946

```

```

*****
: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
1
-----
: 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

```

```

8947
8948 033746 004737 005540
8949 033752 000226
8950 033754 000011
8951 033756 005003
8952 033760 010337 002422
8953 033764 004737 005174
8954 033770 005203
8955 033772 020327 000100
8956 033776 002770
8957 034000 004737 007434
8958 034004 000050
8959
8960
8961
8962 034006 005004
8963 034010 004737 007750
8964 034014 000000
8965 034016 000000
8966 034020 005204
8967 034022 004737 006430
8968 034026 000001
8969
8970
8971
8972 034030 004737 005254
8973 034034 000010
8974 034036 010337 002422
8975 034042 004737 005174
8976 034046 005203
8977 034050 004737 006430
8978 034054 000003
8979 034056 020327 000177
8980 034062 002762
8981
8982
8983
8984 034064 004737 005254
8985 034070 000010
8986 034072 004737 006430
8987 034076 000002
8988
8989
8990
8991 034100 010437 034110
8992 034104 004737 007750
8993 034110 000000
8994 034112 000000
8995 034114 005204
8996 034116 020427 000400
8997 034122 001427
8998 034124 004737 006430
8999 034130 000003
9000 034132 004737 005254
9001 034136 000010
9002 034140 020327 000377

```

```

-----
: JSR PC,INITRN ;LOAD 2 SOM'S, CLOCK THEM INTO USYRT
SYNCH
STRIP!DDCMP
2$: CLR R3 ;INIT BINARY COUNT DATA FOR WRITING
MOV R2,TXWORD ;LOAD A DATA BYTE INTO TX SILO
JSR PC,LDTXSI ;INCR DATA
INC R3 ;SEE IF 64 BYTES LOADED YET
CMP R3,#64. ;BR IF NOT YET
BLT 2$ ;RECEIVE AND TIME FIRST CHARACTER
JSR PC,RCV1ST
40.
-----
: READ RCV SILO, COMPARE FIRST CHAR TO 000, CHK FOR ICIR = 1, IRDY = 0
-----
9$: CLR R4 ;INIT PATTERN FOR READING
JSR PC,CKDATA ;READ RCV SILO, COMPARE DATA
0 ;EXPECTED DATA = 000
0 ;DON'T CLOCK LINE UNIT
16$: INC R4 ;INCR DATA FOR READING
JSR PC,ISIRDY ;CHK FOR ICIR = 1, IRDY = 0
1
-----
: CLOCK 63 CHARS INTO RCV SILO, CHK ICIR = 1, IRDY = 1
-----
18$: JSR PC,STPLU ;CLOCK LU FOR 8 CYCLES
8.
MOV R3,TXWORD
JSR PC,LDTXSI ;LOAD ANOTHER WORD INTO TX SILO
INC R3 ;INCR PATTERN FOR WRITING
JSR PC,ISIRDY ;CHK ICIR = 1, IRDY = 1
3
CMP R3,#127. ;SEE IF 63 MORE CHARS CLOCKED YET
BLT 18$ ;BR IF NOT YET
-----
: CLOCK 1 MORE CHAR INTO RCV SILO, CHK ICIR = 0, IRDY = 1
-----
JSR PC,STPLU ;CLOCK LU FOR 8 CYCLES
8.
JSR PC,ISIRDY ;CHK ICIR = 0, IRDY = 1
2
-----
: READ, COMPARE, CLOCK REST OF DATA CHARS
-----
20$: MOV R4,21$ ;SET EXPECTED DATA
JSR PC,CKDATA ;READ AND COMPARE DATA
21$: 0 ;EXPECTED SILO ENTRY GOES HERE
0 ;DON'T CLOCK LINE UNIT
22$: INC R4 ;INCR DATA PATTERN FOR READS
CMP R4,#400 ;SEE IF ALL DONE READING YET
BEQ 32$ ;BR IF DONE READING
JSR PC,ISIRDY ;CHK ICIR = 1, IRDY = 1
3
JSR PC,STPLU ;CLOCK LU FOR 8 CYCLES
8.
CMP R3,#377 ;SEE IF ALL CHARS LOADED INTO TX SILO YET

```

```

9003 034144 003007          BGT      24$          ;BR IF YES
9004 034146 010337 002422  MOV      R3,TXWORD
9005 034152 004737 005174  JSR      PC,LDTXSI   ;LOAD ANOTHER CHAR INTO TX SILO
9006 034156 005203          INC      R3          ;INCR DATA PATTERN FOR WRITING
9007 034160 000137 034100  JMP      20$
9008 034164 012737 001000 002422 24$: MOV      #TXEOM,TXWORD
9009 034172 004737 005174  JSR      PC,LDTXSI   ;LOAD COM INTO TX SILO
9010 034176 000137 034100  JMP      20$
9011 034202          32$:
9012 034202 004737 003576  A10: JSR      PC,MSTCLR ;ISSUE MASTER CLEAR TO CLEAN UP
9013 034206          ENDTST
(3) 034206
(3) 034206 104401
9014
9015
9016
9017
9018
9019
9020
9021
9022
9023
9024
9025
9026
9027
9028
9029
9030
9031 034210
(3) 034210
9032 034210 012737 034402 002362  MOV      #24$,RETADR ;SET TEST EXIT ADRS FOR ERRORS
9033 034216 004737 005540  JSR      PC,INITRN   ;DO MASTER CLR, LOAD 2 SOM'S
9034 034222 000000 000
9035 034224 000341 000
9036 034226 012701 000017  MOV      #15,R1      ;INIT COUNTER
9037 034232 005037 002422  CLR      TXWORD
9038 034236 004737 005174  3$: JSR      PC,LDTXSI ;LOAD A 000 CHAR INTO TX SILO
9039 034242 005301 000
9040 034244 001374 000
9041 034246 004737 005146  DEC      R1          ;DECR COUNTER
9042 034252 012737 000006 002402  BNE      3$         ;BR IF NOT DONE LOADING YET
9043 034260 012737 000000 002374  JSR      PC,WAIT50   ;WAIT FOR SILO TO RIPPLE
9044 034266 012737 000005 002376  MOV      #6,AXNUM    ;SET BYTE NO. = 6 FOR AX3
9045 034274 004737 004312  MOV      #000,WAX15  ;SET DATA FOR AX3-15 = 0
9046 034300 004737 005254  MOV      #RXLEN2!RXLENO,WAX16 ;SET RCV LEN = 5
9047 034304 100012 000
9048 034306 012737 000006 002376  JSR      PC,WRITAX   ;LOAD AX3
9049 034314 004737 004312  JSR      PC,STPLU   ;CLK LU UNTIL TX'ING 1ST DATA CHAR
9050 034320 004737 007434  CHPCBK!10.
9051 034324 000005 000
9052 034326 012737 000007 002376  MOV      #RXLEN2!RXLEN1!RXLENO,WAX16 ;SET RCV LEN = 6
9053 034334 004737 004312  JSR      PC,WRITAX   ;LOAD AX3
9054 034340 004737 011074  JSR      PC,RCV1ST  ;CLOCK 5-BIT DATA CHAR
9055 034344 000006 000

```

L10070: TRAP C\$ETST

```

:*****
: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 USVRT 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
T37::
MOV #24$,RETADR ;SET TEST EXIT ADRS FOR ERRORS
JSR PC,INITRN ;DO MASTER CLR, LOAD 2 SOM'S
000
CRC2!CRC1!IDLE!DDCMP
MOV #15,R1 ;INIT COUNTER
CLR TXWORD
3$: 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
MOV #RXLEN2!RXLENO,WAX16 ;SET RCV LEN = 5
JSR PC,WRITAX ;LOAD AX3
JSR PC,STPLU ;CLK LU UNTIL TX'ING 1ST DATA CHAR
CHPCBK!10.
MOV #RXLEN2!RXLEN1!RXLENO,WAX16 ;SET RCV LEN = 6
JSR PC,WRITAX ;LOAD AX3
JSR PC,RCV1ST ;CLOCK 5-BIT DATA CHAR
5
MOV #RXLEN2!RXLEN1!RXLENO,WAX16 ;SET RCV LEN = 7
JSR PC,WRITAX ;LOAD AX3
JSR PC,RXCHAR ;RCV 5-BIT DATA CHAR, CLK 6-BIT
6

```

TEST 37 - RECEIVER CHAR LENGTH TIMING TEST - CHAR MODE, NO CRC

```

9056 034346 012737 000000 002376      MOV      #0,WAX16      ;SET RCV LEN = 8
9057 034354 004737 004312      JSR      PC,WRITAX    ;LOAD AX3
9058 034360 004737 011074      JSR      PC,RXCHAR    ;RCV 6-BIT DATA CHAR, CLK 7-BIT
9059 034364 000007              7
9060 034366 004737 011074      JSR      PC,RXCHAR    ;RCV 7-BIT DATA CHAR, CLK 8-BIT
9061 034372 000010              8.
9062 034374 004737 011074      JSR      PC,RXCHAR    ;RCV 8 BIT DATA CHAR
9063 034400 000010              8.
9064 034402 004737 003576      JSR      PC,MSTCLR    ;ISSUE MASTER CLEAR TO CLEAN UP
9065 034406
(3) 034406
(3) 034406 104401
L10071: TRAP C$ETST

```

```

*****
: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

```

```

9083 034410
(3) 034410
9084 034410 012737 034702 002362      MOV      #24$,RETADR  T38::
9085 034416 004737 005540      JSR      PC,INITRN    ;DO MASTER CLR, LOAD 2 SOM'S
9086 034422 000000
9087 034424 000300      CRC2!CRC1
9088 034426 012701 000017      MOV      #15.,R1      ;INIT COUNTER
9089 034432 005037 002422      CLR      TXWORD
9090 034436 004737 005174      JSR      PC,LDTXSI    ;LOAD A 000 CHAR INTO TX SILO
9091 034442 005301              DEC      R1            ;DECR COUNTER
9092 034444 001374              BNE     3$            ;BR IF NOT DONE LOADING YET
9093 034446 004737 005146      JSR      PC,WAIT50    ;WAIT FOR SILO TO RIPPLE
9094 034452 012737 000006 002402      MOV      #6,AXNUM     ;SET BYTE NO. = 6 FOR AX3
9095 034460 012737 000000 002374      MOV      #000,WAX15   ;SET DATA FOR AX3-15 = 0
9096 034466 004737 007434      JSR      PC,RCV1ST    ;CLOCK FIRST 8-BIT DATA CHAR
9097 034472 000040              32.
9098 034474 012737 000000 002376      MOV      #0,WAX16     ;SET RCV LEN = 8
9099 034502 004737 004312      JSR      PC,WRITAX    ;LOAD AX3
9100 034506 004737 011074      JSR      PC,RXCHAR    ;RCV FIRST 8-BIT DATA CHAR, CLK SECOND 8-BIT
9101 034512 000010              8.
9102 034514 012737 000007 002376      MOV      #RXLEN2!RXLEN1!RXLEN0,WAX16 ;SET RCV LEN = 7
9103 034522 004737 004312      JSR      PC,WRITAX    ;LOAD AX3
9104 034526 004737 011074      JSR      PC,RXCHAR    ;RCV SECOND 8-BIT DATA CHAR, CLK 3RD 8-BIT
9105 034532 000010              8.
9106 034534 012737 000006 002376      MOV      #RXLEN2!RXLEN1,WAX16 ;SET RCV LEN = 6
9107 034542 004737 004312      JSR      PC,WRITAX    ;LOAD AX3
9108 034546 004737 011074      JSR      PC,RXCHAR    ;RCV 3RD 8-BIT DATA CHAR, CLK 7-BIT

```

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

```

9109 034552 000007          7
9110 034554 012737 000005 002376  MOV    #RXLEN2!RXLENO,WAX16    ;SET RCV LEN = 5
9111 034562 004737 004312          JSR    PC,WRITAX                ;LOAD AX3
9112 034566 004737 011074          JSR    PC,RXCHAR                ;RCV 7-BIT DATA CHAR, CLK 6-BIT
9113 034572 000006          6
9114 034574 012737 000004 002376  MOV    #RXLEN2,WAX16           ;SET RCV LEN = 4
9115 034602 004737 004312          JSR    PC,WRITAX                ;LOAD AX3
9116 034606 004737 011074          JSR    PC,RXCHAR                ;RCV 6-BIT DATA CHAR, CLK 5-BIT
9117 034612 000005          5
9118 034614 012737 000003 002376  MOV    #RXLEN1!RXLENO,WAX16    ;SET RCV LEN = 3
9119 034622 004737 004312          JSR    PC,WRITAX                ;LOAD AX3
9120 034626 004737 011074          JSR    PC,RXCHAR                ;RCV 5-BIT DATA CHAR, CLR 4-BIT
9121 034632 000004          4
9122 034634 012737 000002 002376  MOV    #RXLEN1,WAX16           ;SET RCV LEN = 2
9123 034642 004737 004312          JSR    PC,WRITAX                ;LOAD AX3
9124 034646 004737 011074          JSR    PC,RXCHAR                ;RCV 4-BIT DATA CHAR, CLK 3-BIT
9125 034652 000003          3
9126 034654 012737 000001 002376  MOV    #RXLENO,WAX16           ;SET RCV LEN = 1
9127 034662 004737 004312          JSR    PC,WRITAX                ;LOAD AX3
9128 034666 004737 011074          JSR    PC,RXCHAR                ;RCV 3-BIT DATA CHAR, CLK 2-BIT
9129 034672 000002          2
9130 034674 004737 011074          JSR    PC,RXCHAR                ;RCV 2-BIT DATA CHAR, CLK 1-BIT
9131 034700 000001          1
9132 034702 004737 003576          JSR    PC,MSTCLR                ;ISSUE MASTER CLEAR TO CLEAN UP
9133 034706          24$:
(3) 034706          ENDTST
(3) 034706 104401

```

L10072: TRAP C\$ETST

```

:*****
: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

```

```

9148 034710          T39::
(3) 034710
9149 034710 012737 034772 002362  MOV    #24$,RETADR             ;SET TEST EXIT ADDRESS FOR ERRORS
9150 034716 004737 005540          JSR    PC,INITRN               ;MST CLR, LOAD 2 SOM'S
9151 034722 000226          SYNCH
9152 034724 000341          CRC2!CRC1!IDLE!DDCMP
9153 034726 012737 000000 002422  MOV    #000,TXWORD
9154 034734 004737 005174          JSR    PC,LDTXSI               ;LOAD 000 CHAR INTO TX SILO
9155 034740 004737 005254          JSR    PC,STPLU                ;CLOCK LINE UNIT UNTIL LINE GOES MARKING
9156 034744 000063          51.
9157 034746 004737 007354          JSR    PC,RDRXSI               ;READ 000 CHAR
9158 034752 004737 007750          JSR    PC,CKDATA               ;READ AND CHECK FOR MARK CHAR (377)
9159 034756 000377          377
9160 034760 000000          000
9161 034762 004737 007750          JSR    PC,CKDATA               ;READ AND CHECK FOR ANOTHER MARK CHAR

```

CZDMRE M8203 STATIC DIAG #1  
CZDMRE.P11 08-JUN-81 13:27

MACY11 30A(1052) 15-JUN-81 15:34 PAGE 7-127  
TEST 39 - TRANSMITTER UNDERRUN ERROR, IDLE MARKING, CHAR MODE,NO CRC

SEQ 0167

9162 034766 000377  
9163 034770 000000  
9164 034772 004737 003576  
9165 034776  
(3) 034776  
(3) 034776 104401

377  
000  
24\$: JSR PC,MSTCLR ;ISSUE MASTER CLEAR TO CLEAN UP  
ENDTST  
L10073:  
TRAP C\$ETST

9166  
9167  
9168  
9169  
9170  
9171  
9172  
9173  
9174  
9175  
9176  
9177  
9178  
9179  
9180  
9181  
9182  
9183  
9184  
9185

\*\*\*\*\*  
: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.  
:\*\*\*\*\*

9186 035000  
(3) 035000  
9187 035000 012737 035134 002362  
9188 035006 004737 005540  
9189 035012 000000  
9190 035014 000310  
9191 035016 004737 011016  
9192 035022 003224  
9193 035024 000005  
9194 035026 012737 005000 002422  
9195 035034 004737 005174  
9196 035040 004737 005174  
9197 035044 004737 005146  
9198 035050 004737 005254  
9199 035054 100071  
9200 035056 004737 011176  
9201 035062 000376  
9202 035064 004737 007750  
9203 035070 000000  
9204 035072 000000  
9205 035074 004737 007750  
9206 035100 000125  
9207 035102 000000  
9208 035104 004737 007750  
9209 035110 000252  
9210 035112 000000  
9211 035114 004737 007750  
9212 035120 000377  
9213 035122 000010  
9214 035124 004737 007750

BGNTST  
T40::  
MOV #24\$,RETADR ;SET TEST EXIT ADRS FOR ERRORS  
JSR PC,INITRN ;MST CLR, LOAD 2 SOM'S  
000  
CRC2!CRC1!STRIP  
JSR PC,LODMSG ;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



9215 035130 00300C  
9216 035132 000010  
9217 035134 004737 003576  
9218 035140  
(3) 035140  
(3) 035140 104401  
9219  
9220  
9221  
9222  
9223  
9224  
9225  
9226  
9227  
9228  
9229  
9230  
9231  
9232  
9233  
9234  
9235  
9236  
9237 035142  
(3) 035142

3000  
8.  
24\$: JSR PC,MSTCLR ;ISSUE MASTER CLEAR  
ENDTST

L10074:  
TRAP C\$ETST

\*\*\*\*\*  
:SBTTL 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).  
:\*\*\*\*\*  
BGNTST

9238 035142 012737 035302 002362  
9239 035150 004737 005540  
9240 035154 000226  
9241 035156 000341  
9242 035160 012701 000026  
9243 035164 012737 000226 002422  
9244 035172 004737 005174  
9245 035176 005301  
9246 035200 001374  
9247 035202 004737 005146  
9248 035206 004737 007434  
9249 035212 000030  
9250 035214 012701 000025  
9251 035220 004737 007750  
9252 035224 000226  
9253 035226 000010  
9254 035230 005301  
9255 035232 001372  
9256 035234 004737 007750  
9257 035240 000226  
9258 035242 000004  
9259 035244 012737 000011 002400  
9260 035252 112737 000200 002366  
9261 035260 004737 003750  
9262 035264 004737 005254  
9263 035270 000014  
9264 035272 004737 007750  
9265 035276 000377  
9266 035300 000000  
9267 035302 004737 003576

T41::  
MOV #24\$,RETADR ;SET TEST EXIT ADRS FOR ERRORS  
JSR PC,INITRN ;MST CLR, LOAD 2 SOM'S  
SYNCH  
CRC2!CRC1!IDLE!DDCMP  
MOV #22.,R1 ;INIT COUNTER  
MOV #SYNCH,TXWORD  
6\$: JSR PC,LDTXSI ;LOAD AN SOM INTO TX SILO  
DEC R1 ;DECR COUNTER  
BNE 6\$ ;BR IF MORE TO LOAD  
JSR PC,WAIT50 ;ALLOW SILO TO RIPPLE  
JSR PC,RCV1ST ;CLK LU UNTIL 3RD SYNCH RCV'D (RCVR IS ACTIVE)  
24.  
MOV #21.,R1 ;INIT COUNTER  
8\$: JSR PC,CKDATA ;READ A SYNCH, CLK NEXT ONE  
SYNCH  
8.  
DEC R1 ;DECR COUNTER  
BNE 8\$ ;BR IF NOT ALL CHECKED  
JSR PC,CKDATA ;CHECK LAST SYNCH  
SYNCH  
4  
MOV #11,REGNUM ;SET REG NO. = 11  
MOVB #OC,WRIBYT  
JSR PC,WRITLU ;SET OC IN REG 11  
JSR PC,STPLU ;FINISH CLOCKING CHAR, AND THEN SOME  
12.  
JSR PC,CKDATA ;RCV A MARK CHAR, CHK IT  
377  
0  
24\$: JSR PC,MSTCLR ;ISSUE MASTER CLEAR TO CLEAN UP

9268 035306  
(3) 035306  
(3) 035306 104401

ENDTST

L10075: TRAP C\$ETST

9269  
9270  
9271  
9272  
9273  
9274  
9275  
9276  
9277  
9278  
9279  
9280  
9281  
9282  
9283  
9284  
9285  
9286  
9287  
9288  
9289  
9290

```
*****
: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.
*****
```

9291 035310  
(3) 035310  
9292 035310 012737 035530 002362  
9293 035316 012701 035536  
9294 035322 011137 035332  
9295 035326 004737 005540  
9296 035332 000000  
9297 035334 000311  
9298 035336 012737 000400 002422  
9299 035344 012702 000026  
9300 035350 004737 005174  
9301 035354 005302  
9302 035356 001374  
9303 035360 004737 011016  
9304 035364 003246  
9305 035366 000006  
9306 035370 012737 001000 002422  
9307 035376 004737 005174  
9308 035402 004737 005174  
9309 035406 004737 005146  
9310 035412 011137 035434  
9311 035416 012702 000027  
9312 035422 004737 005254  
9313 035426 100010  
9314 035430 004737 011176  
9315 035434 000000  
9316 035436 005302  
9317 035440 001373  
9318 035442 004737 007434  
9319 035446 000010  
9320 035450 004737 007750

EGNTST

```
T42::
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$: .WORD 0 ;SYNCH PATTERN GOES HERE
CRC2!CRC1!STRIP!DDCMP
MOV #TXSOM,TXWORD
MOV #22.,R2 ;LOAD 22 SOM'S INTO TX SILO
6$: 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
```

```

9321 035454 000377 377
9322 035456 000010 8.
9323 035460 004737 007750 JSR PC,CKDATA ;RCV 000, CLK 125
9324 035464 000000 000
9325 035466 000010 8.
9326 035470 004737 007750 JSR PC,CKDATA ;RCV 125, CLK 252
9327 035474 000125 125
9328 035476 000010 8.
9329 035500 004737 007750 JSR PC,CKDATA ;PCV 252, CLK END OF MSG
9330 035504 000252 252
9331 035506 000040 32.
9332 035510 004737 005352 JSR PC,OACTIV ;CHK FOR OACT = 0
9333 035514 000000 0
9334 035516 062701 000002 ADD #2,R1 ;INIT SYNCH PATTERN POINTER
9335 035522 020127 035552 CMP R1,#SYNPAT+12. ;SEE IF ALL PATTERNS CHECKED YET
9336 035526 103675 BLO 2$ ;BR IF NOT YET
9337 035530 004737 003576 JSR PC,MSTCLR ;ISSUE MASTER CLEAR TO CLEAN UP
9338 035534 24$: ENDTST
(3) 035534 L10076:
(3) 035534 104401 TRAP C$ETST
9339
9340 035536 000226 SYNPAT: 226
9341 035540 000000 000
9342 035542 000125 125
9343 035544 000252 252
9344 035546 000376 376
9345 035550 000177 177
9346
9347
9348
9349
9350

```

.SBTTL HARDWARE PARAMETER CODING SECTION

:/ THE HARDWARE 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.

```
9352
9353
9354
9355
9356
9357
9358
9359
9360
9361
9362
9363
9364
9365 035552          BGNHRD
      (3) 035552      000022
      (3) 035554
                                     .WORD L10077-L$HARD/2
                                     L$HARD::
9366
9367 035554          GPRMA  ADDRES,2,0,160000,177776,YES
      (4) 035554      001031
      (4) 035556      035620
      (4) 035560      160000
      (4) 035562      177776
                                     .WORD  T$CODE
                                     .WORD  ADDRES
                                     .WORD  T$LLOLIM
                                     .WORD  T$HILIM
9368 035564          GPRMA  VECTOR,4,0,0,770,YES
      (4) 035564      002031
      (4) 035566      035646
      (4) 035570      000000
      (4) 035572      000770
                                     .WORD  T$CODE
                                     .WORD  VECTOR
                                     .WORD  T$LLOLIM
                                     .WORD  T$HILIM
9369 035574          GPRMD  PRIRTY,6,0,7000,4,7,YES
      (4) 035574      003032
      (4) 035576      035677
      (4) 035600      007000
      (4) 035602      000004
      (4) 035604      000007
                                     .WORD  T$CODE
                                     .WORD  PRIRTY
                                     .WORD  7000
                                     .WORD  T$LLOLIM
                                     .WORD  T$HILIM
9370 035606          GPRMD  ISRUN,24,0,7,0,7,YES
      (4) 035606      012032
      (4) 035610      035730
      (4) 035612      000007
      (4) 035614      000000
      (4) 035616      000007
                                     .WORD  T$CODE
                                     .WORD  ISRUN
                                     .WORD  7
                                     .WORD  T$LLOLIM
                                     .WORD  T$HILIM
9371
9372 035620          ENDHRD
      (2)
      (3) 035620
                                     .EVEN
                                     L10077:
9373
9374 035620 042504 044526 042503  ADDRES: .ASCIZ /DEVICE CSR ADDRESS : /
      035626 041440 051123 040440
      035634 042104 042522 051523
      035642 035040 000040
9375 035646 042504 044526 042503  VECTOR: .ASCIZ /DEVICE VECTOR ADDRESS : /
      035654 053040 041505 047524
      035662 020122 042101 051104
      035670 051505 020123 020072
      035676      000
9376 035677      104 053105 041511  PRIRTY: .ASCIZ /DEVICE PRIORITY LEVEL : /
      035704 020105 051120 047511
      035712 044522 054524 046040
```

CZDMRE M8203 STATIC DIAG #1  
CZDMRE.P11 08-JUN-81 13:27

MACY11 30A(1052) 15-JUN-81 15:34 PAGE 7-132  
HARDWARE PARAMETER CODING SECTION

SEQ 0172

	035720	053105	046105	035040	
	035726	000040			
9377	035730	034115	030062	020067	ISRUN: .ASCIZ /M8207 RUN SWITCH (E28 SW7) - TYPE 0 IF OFF, 1 IF ON : /
	035736	052522	020116	053523	
	035744	052111	044103	024040	
	035752	031105	020070	053523	
	035760	024467	026440	052040	
	035766	050131	020105	020060	
	035774	043111	047440	043106	
	036002	020054	020061	043111	
	036010	047440	020116	020072	
	036016	000			
9378		036020			.EVEN
9379					
9380					
9381					
9382					
9383					
9384					

.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.

9386  
9387  
9388  
9389  
9390  
9391  
9392  
9393  
9394  
9395  
9396  
9397  
9398  
9399  
9400  
9401  
9402  
9403  
9404  
9405  
9406  
9407  
9408  
9409  
9410  
9411  
9412  
9413  
9414  
9415  
9416  
9417  
9418  
9419  
9420  
9421  
9422  
9423  
9424  
9425  
9426

036020  
(3) 036020 000000  
(3) 036022  
036022  
(2)  
(3) 036022  
036022  
036222  
036222 000240  
036224 000240  
036226 000240  
036230  
036230  
(2)  
(4) 036230 000000  
(4) 036232 000000  
(3) 036234  
000001

BGNSFT  
ENDSFT  
.EVEN

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

ENDMOD  
LASTAD  
L\$LAST::  
.END

.WORD L10100-L\$SOFT/2  
L\$SOFT::  
.EVEN  
L10100:

.EVEN 0  
.WORD 0  
.WORD 0



CZDMRE M8203 STATIC DIAG #1  
CZDMRE.P11 08-JUN-81 13:27

MACY11 30A(1052) 15-JUN-81 15:34 PAGE 8-1  
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0175

BIT10 = 002000 G	4449#	4708	4720											
BIT11 = 004000 G	4449#	4707	4719											
BIT12 = 010000 G	4449#													
BIT13 = 020000 G	4449#													
BIT14 = 040000 G	4449#													
BIT15 = 100000 G	4449#	4751	4753	4754	5676	5830								
BIT2 = 000004 G	4449#	4469	4481	4490	4509	4519	4531	4543	4555	4567	4579	4591	4603	
	4615	4627	4639	4648	4660	4673	4685	4696						
BIT3 = 000010 G	4449#	4468	4480	4489	4508	4518	4530	4542	4554	4566	4578	4590	4602	
	4614	4626	4638	4647	4659	4672	4684							
BIT4 = 000020 G	4449#	4467	4479	4507	4517	4529	4541	4553	4565	4577	4589	4601	4613	
	4625	4637	4658	4671	4683									
BIT5 = 000040 G	4449#	4478	4499	4506	4516	4528	4540	4552	4564	4576	4588	4600	4612	
	4624	4636	4657	4670	4682	4695								
BIT6 = 000100 G	4449#	4466	4477	4498	4505	4515	4527	4539	4551	4563	4575	4587	4599	
	4611	4623	4635	4656	4669	4681	4694							
BIT7 = 000200 G	4449#	4465	4476	4488	4497	4504	4526	4538	4550	4562	4574	4586	4598	
	4610	4622	4634	4646	4655	4668	4680	4693	6159	6160				
BIT8 = 000400 G	4449#	4710	4722											
BIT9 = 001000 G	4449#	4709	4721											
BOE = 000400 G	4449#													
BPOLL = 000100	4498#													
BSEL1 002450	4831#	5324*	5326*	5327*	5341*	5342*	5347*	5439*	5441*	5542*	5680*	5681*	5683*	
	5773*	5774*	5779*	5792*	6775*	6776*	6777	7012*	7712*					
BSEL2 002452	4832#	6777*	6778*	7348	7368	7387								
BSEL4 002454	4833#	5377	5398*	7051*										
CARR = 000001	4581#													
CHPCHK= 100000	4751#	8037	8086	8137	8195	8236	8280	8333	8417	9047	9199	9313		
CHPTYP 002430	4821#	5677	5787*	5791*	5831	6729*	8821	8843						
CKDATA 007750	6150#	8963	8992	9158	9161	9202	9205	9208	9211	9214	9251	9256	9264	
	9320	9323	9326	9329										
CKTBIT 011176	6365#	8419	9200	9314										
CRCCHK= 100000	4754#	6420												
CRCTY0= 000001	4675#													
CRCTY1= 000002	4674#													
CRCTY2= 000004	4673#													
CRC1 = 000100	4527#	8131	8609	8672	8791	9035	9087	9152	9190	9241	9297			
CRC2 = 000200	4526#	8131	8609	8672	8791	9035	9087	9152	9190	9241	9297			
CS = 000004	4579#													
C\$AU = 000052	4306#	6871												
C\$AUTO= 000061	4306#	6816												
C\$BRK = 000022	4306#													
C\$BSEG= 000004	4306#	6950	7181	7215	7249	7283	7558	7609	7656	7716	7778			
C\$BSUB= 000002	4306#	7776	7802	7838										
C\$CEFG= 000045	4306#													
C\$CLK= 000062	4306#													
C\$CLEA= 000012	4306#	6833												
C\$CLOS= 000035	4306#													
C\$CLP1= 000006	4306#													
C\$CVEC= 000036	4306#													
C\$DCLN= 000044	4306#													
C\$DODU= 000051	4306#	6813												
C\$DRPT= 000024	4306#													
C\$DU = 000053	4306#	6852												
C\$EDIT= 000003	4306#	4346												
C\$ERDF= 000055	4306#	5585	5591	5601	5607	5714	5720	5878	5910	5916	5926	5932	5967	









CZDMRE M8203 STATIC DIAG #1  
CZDMRE.P11 08-JUN-81 13:27

MACY11 30A(1052) 15-JUN-81 15:34 PAGE 8-5  
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0179

FMT6	011613	6658	6672	6560	6564	6579	6583	6599	6603	6618	6622	6636	6640	6656
FMT7	011646	6445#	6545	6575	6632	6651	6670							
FMT8	011656	6660	6674	6594	6614									
FMT9	011712	6446#	6540	6559	6563	6578	6582	6598	6602	6617	6621	6635	6639	6655
FRSTIM	002412	6447#	6555											
F\$AU =	000015	6448#	6544											
F\$AUTO=	000020	6659	6673											
F\$BGN =	000040	4814#	6732	6739*										
		4306#	6870	5871										
		4306#	6804	6816										
		4306#	4312	6531	6537	6552	6571	6590	6610	6629	6647	6667	6690	6706
		6724	6804	6830	6847	6870	6892	6905	6917	6930	6945	6950	6968	6979
		6995	7006	7028	7044	7064	7081	7105	7110	7125	7157	7164	7176	7181
		7198	7210	7215	7232	7244	7249	7266	7278	7283	7300	7327	7401	7417
		7430	7444	7456	7467	7472	7488	7516	7526	7533	7554	7558	7573	7588
		7605	7609	7624	7638	7652	7656	7671	7685	7710	7716	7734	7750	7767
		7776	7778	7798	7802	7816	7817	7835	7838	7862	7872	7877	7878	7900
		7977	8000	8010	8030	8059	8079	8105	8127	8165	8185	8254	8269	8310
		8326	8390	8405	8425	8441	8505	8521	8588	8604	8651	8667	8734	8747
		8766	8783	8912	8936	9013	9031	9065	9083	9133	9148	9165	9186	9218
		9237	9268	9291	9338	9365	9398	9422						
F\$CLEA=	000007	4306#	6830	6833										
F\$DU =	000016	4306#	6847	6852										
F\$END =	000041	4306#	4312	6533	6546	6565	6584	6604	6623	6641	6661	6675	6692	6790
		6816	6833	6852	6871	6892	6905	6917	6930	6945	6964	6968	6979	6995
		7006	7028	7044	7064	7081	7105	7110	7125	7157	7164	7176	7194	7198
		7210	7228	7232	7244	7262	7266	7278	7296	7300	7327	7401	7417	7430
		7444	7456	7467	7472	7488	7516	7526	7533	7554	7573	7584	7588	7605
		7624	7634	7638	7652	7671	7681	7685	7710	7734	7745	7750	7767	7776
		7793	7798	7802	7816	7817	7835	7838	7862	7872	7877	7878	7900	7977
		8000	8010	8030	8059	8079	8105	8127	8165	8185	8254	8269	8310	8326
		8390	8405	8425	8441	8505	8521	8588	8604	8651	8667	8734	8747	8766
		8783	8912	8936	9013	9031	9065	9083	9133	9148	9165	9186	9218	9237
		9268	9291	9338	9372	9401	9422							
F\$HARD=	000004	4306#	9365	9372										
F\$HW =	000013	4306#	4391	4405										
F\$INIT=	000006	4306#	6724	6790										
F\$JMP =	000050	4306#												
F\$MOD =	000000	4306#	4312	9422										
F\$MSG =	000011	4306#	6531	6533	6537	6546	6552	6565	6571	6584	6590	6604	6610	6623
		6629	6641	6647	6661	6667	6675							
F\$PROT=	000021	4306#	6706	6710										
F\$PWR =	000017	4306#												
F\$RPT =	000012	4306#	6690	6692										
F\$SEG =	000003	4306#	6950	6964	7181	7194	7215	7228	7249	7262	7283	7296	7558	7584
		7609	7634	7656	7681	7716	7745	7778	7793					
F\$SOFT=	000005	4306#	9398	9401										
F\$SRV =	000010	4306#												
F\$SUB =	000002	4306#	7776	7798	7802	7816	7838	7877						
F\$SW =	000014	4306#	4419	4422										
F\$TEST=	000001	4306#	6892	6905	6917	6930	6945	6968	6979	6995	7006	7028	7044	7064
		7081	7110	7125	7164	7176	7198	7210	7232	7244	7266	7278	7300	7327
		7401	7417	7472	7488	7533	7554	7588	7605	7638	7652	7685	7710	7750
		7767	7817	7835	7878	7900	8010	8030	8059	8079	8105	8127	8165	8185
		8254	8269	8310	8326	8390	8405	8425	8441	8505	8521	8588	8604	8651
		8667	8734	8747	8766	8783	8912	8936	9013	9031	9065	9083	9133	9148

CZDMRE M8203 STATIC DIAG #1  
CZDMRE.P11 08-JUN-81 13:27

MACY11 30A(1052) 15-JUN-81 15:34 PAGE 8-6  
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0180

GETALL 004526	9165	9186	9218	9237	9268	9291	9338							
	5535#	5583	5589	5599	5605	5712	5718	5876	5908	5914	5924	5930	5965	
	5971	6005	6011	6019	6025	6171	6176	6195	6201	6210	6216	6225	6231	
	6240	6246	6378	6384	7453	7464	7513	7523	7570	7580	7621	7630	7668	
	7677	7731	7741	7789	7812	7859	7869	7974	7997	8159	8202	8211	8223	
	8242	8250	8473	8488	8571	8636	8717	8761	8865					
GETPRM 021306	6752	6763#	6770											
GETREG 004016	5412#	5541	6926	6960	6991	7024	7060	7102	7154	7190	7224	7258	7292	
	7356	7375	7394	7427	7441									
GOAH = 000010	4489#													
GOODAT 002404	4811#	6163*	6164*	6185*	6186*	6541	6556	6595	6652	6923*	6924*	6958*	6988*	
	6989*	7021*	7022*	7057*	7058*	7099*	7100*	7151*	7152*	7188*	7222*	7256*	7290*	
	7354*	7373*	7392*	7450*	7451*	7461*	7462*	7510*	7511*	7520*	7521*	7568*	7577*	
	7578*	7619*	7628*	7666*	7675*	7728*	7729*	7738*	7739*	7786*	7787*	7809*	7810*	
	7856*	7857*	7866*	7867*	7972*	7995*	8471*	8486*	8496*	8569*	8634*	8715*	8759*	
	8863*	8883*	8899*											
G\$CNT0= 000200	4306#													
G\$DELM= 000372	4306#													
G\$DISP= 000003	4306#													
G\$EXCP= 000400	4306#													
G\$HILI= 000002	4306#													
G\$LOLI= 000001	4306#													
G\$NO = 000000	4306#													
G\$OFFS= 000400	4306#	9367	9368	9369	9370									
G\$OF SI= 000376	4306#	9367	9368	9369	9370									
G\$PRMA= 000001	4306#	9367	9368											
G\$PRMD= 000002	4306#	9369	9370											
G\$PRML= 000000	4306#													
G\$RADA= 000140	4306#													
G\$RADB= 000000	4306#													
G\$RADD= 000040	4306#													
G\$RADL= 000120	4306#													
G\$RADO= 000020	4306#	9367	9368	9369	9370									
G\$XFER= 000004	4306#													
G\$YES = 000010	4306#	9367	9368	9369	9370									
HDX = 000020	4507#	4577#												
HELP = 000001	4292#	4338	4348	4371	5297									
HOE = 100000 G	4449#													
IACT = 000100	4563#	5963	5969											
IACTIV 006714	5953#	6080	6101	8461	8466	8481	8494	8502	8541	8550	8557	8562	8579	
	8585	8624	8629	8644	8648	8687	8696	8703	8708	8725	8731	8829	8854	
	8875	8891												
IBE = 010C00 G	4449#													
IC = 000200	4497#	4562#	8583	8646	8729									
ICIR = 000010	4602#	5922	5928											
IDL = 000010	4672#													
IDLE = 000040	4528#	8233	8273	9035	9152	9241								
IDU = 000040 G	4449#													
IER = 020000 G	4449#													
IERR = 000002	4532#	6125	6131	8446	8526	8609	8672	8791						
INITRN 005540	5744#	8032	8081	8129	8190	8231	8271	8328	8406	8785	8948	9033	9085	
	9150	9188	9239	9295										
INTFLG 002354	4798#													
INTGRL= 000010	4684#	4688												
IRDY = 000020	4565#	5906	5912	6091										
ISIRDY 006430	5896#	6082	6095	6099	6103	6341	6347	8841	8850	8858	8871	8879	8887	

CZDMRE M8203 STATIC DIAG #1  
CZDMRE.P11 08-JUN-81 13:27

MACY11 30A(1052) 15-JUN-81 15:34 PAGE 8-7  
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0181

	8895	8942	8967	8977	8986	8998								
ISR = 000100 G	4449#													
ISRUN 035730	9370	9377#												
IXE = 004000 G	4449#													
I\$AU = 000041	4306#	6870#	6871#											
I\$AUTO= 000041	4306#	6804#	6816#											
I\$CLN = 000041	4306#	6830#	6833#											
I\$DU = 000041	4306#	6847#	6852#											
I\$HRD = 000041	9365#	9372#												
I\$INIT= 000041	4306#	6724#	6790#											
I\$MOD = 000041	4306#	4312#	9422#											
I\$MSG = 000041	4306#	6531#	6533#	6537#	6546#	6552#	6565#	6571#	6584#	6590#	6604#	6610#	6623#	
	6629#	6641#	6647#	6661#	6667#	6675#								
	4306#	6706#												
I\$PROT= 000040	4306#													
I\$PTAB= 000041	4306#													
I\$PWR = 000041	4306#													
I\$RPT = 000041	4306#	6690#	6692#											
I\$SEG = 000041	4306#	6892	6917	6945	6950#	6964#	6979	7006	7044	7081	7125	7176	7181#	
	7194#	7210	7215#	7228#	7244	7249#	7262#	7278	7283#	7296#	7327	7417	7488	
	7554	7558#	7573	7584#	7605	7609#	7624	7634#	7652	7656#	7671	7681#	7710	
	7716#	7734	7745#	7767	7776	7778#	7793#	7802	7835	7838	7900	8030	8079	
	8127	8185	8269	8326	8405	8441	8521	8604	8667	8747	8783	8936	9031	
	9083	9148	9186	9237	9291									
I\$SETU= 000041	4306#													
I\$SFT = 000041	9398#	9401#												
I\$SRV = 000041	4306#													
I\$SUB = 000041	4306#	6892	6917	6945	6979	7006	7044	7081	7125	7176	7210	7244	7278	
	7327	7417	7488	7554	7605	7652	7710	7767	7776#	7798#	7802#	7816#	7835	
	7838#	7862	7872	7877#	7900	8030	8079	8127	8185	8269	8326	8405	8441	
	8521	8604	8667	8747	8783	8936	9031	9083	9148	9186	9237	9291		
I\$TST = 000041	4306#	6892#	6905#	6917#	6930#	6945#	6968#	6979#	6995#	7006#	7028#	7044#	7064#	
	7081#	7105	7110#	7125#	7157	7164#	7176#	7198#	7210#	7232#	7244#	7266#	7278#	
	7300#	7327#	7401#	7417#	7430	7444	7456	7467	7472#	7488#	7516	7526	7533#	
	7554#	7588#	7605#	7638#	7652#	7685#	7710#	7750#	7767#	7776	7802	7817#	7835#	
	7838	7878#	7900#	7977	8000	8010#	8030#	8059#	8079#	8105#	8127#	8165#	8185#	
	8254#	8269#	8310#	8326#	8390#	8405#	8425#	8441#	8505#	8521#	8588#	8604#	8651#	
	8667#	8734#	8747#	8766#	8783#	8912#	8936#	9013#	9031#	9065#	9083#	9133#	9148#	
	9165#	9186#	9218#	9237#	9268#	9291#	9338#							
	4680#	4688												
I422 = 000200	4306#													
J\$JMP = 000167	6410#													
LDMSG1 011374	5651#	5764	5765	5823	6314	7912	7913	7937	7935	8216	8412	8455	8535	
LDTXSI 005174	8618	8681	8754	8953	8975	9005	9009	9038	9090	9154	9195	9196	9244	
	9300	9307	9308											
LOADAT 002410	4813#	6592	7784*	7785*										
LODMSG 011016	6307#	6412	9191	9303										
LOE = 040000 G	4449#													
LOGDEV 002344	4794#	6758*	6764*	6765	6767	6813	6851							
LOOPIN 004074	5438#	7341												
LOT = 000010 G	4449#													
LULOOP= 000010	4468#	5347	5542	5680	5773	7712								
LULP = 000040	4499#	4564#	8451	8531	8614	8677	8799							
LUREG 002302	4730	4731	4732	4733	4734	4735	4736	4737	4738	4739	4740	4741	4742	
	4743	4744	4745	4788#										
LUR10 = 002302	4730#	5414	6543	6558	6577	6597	6616	6634	6654	6672				
LUR11 = 002304	4731#	6543	6558	6577	6597	6616	6634	6654	6672					

LUR12 = 002306	4732#	6543	6558	6577	6597	6616	6634	6654	6672
LUR13 = 002310	4733#	6543	6558	6577	6597	6616	6634	6654	6672
LUR14 = 002312	4734#	6545	6560	6579	6599	6618	6636	6656	6674
LUR15 = 002314	4735#	6545	6560	6579	6599	6618	6636	6656	6674
LUR16 = 002316	4736#	6545	6560	6579	6599	6618	6636	6656	6674
LUR17 = 002320	4737#	6545	6560	6579	6599	6618	6636	6656	6674
LUSWI1 002466	4839#								
LUSWI2 002470	4840#								
LUSWI3 002472	4841#								
LU1MOD 002000 G	4312#								
L\$ACP 002110 G	4346#								
L\$APT 002036 G	4346#								
L\$AU 021632 G	4346	6870#							
L\$AUT 002070 G	4346#								
L\$AUTO 021466 G	4346	6804#							
L\$CCP 002106 G	4346#								
L\$CLEA 021546 G	4346	6830#							
L\$CO 002032 G	4346#								
L\$DEPO 002011 G	4346#								
L\$DESC 003470 G	4346	5290#							
L\$DESP 002076 G	4346#								
L\$DEVP 002060 G	4346#								
L\$DISP 002124 G	4346	4369#							
L\$DLY 002116 G	4346#								
L\$DTP 002040 G	4346#								
L\$DTYP 002034 G	4346#								
L\$DU 021550 G	4346	6847#							
L\$DUT 002072 G	4346#								
L\$DVTY 003462 G	4346	5285#							
L\$EF 002052 G	4346#								
L\$ENVI 002044 G	4346#								
L\$ETP 002102 G	4346#								
L\$EXP1 002046 G	4346#								
L\$EXP4 002064 G	4346#								
L\$EXP5 002066 G	4346#								
L\$HARD 035554 G	4346	9365#							
L\$HIME 002120 G	4346#								
L\$HPCP 002016 G	4346#								
L\$HPTP 002022 G	4346#								
L\$HW 002252 G	4346	4391#							
L\$ICP 002104 G	4346#								
L\$INIT 021116 G	4346	6724#							
L\$LADP 002026 G	4346#								
L\$LAST 036234 G	4346	9424#							
L\$LOAD 002100 G	4346#								
L\$LUN 002074 G	4346#								
L\$MREV 002050 G	4346#								
L\$NAME 002000 G	4346#								
L\$PRIO 002042 G	4346#								
L\$PROT 021110 G	4346	6706#							
L\$PRT 002112 G	4346#								
L\$REPP 002062 G	4346#								
L\$REV 002010 G	4346#								
L\$RPT 021106 G	6690#								
L\$SOFT 036022 G	9398#								
L\$SPC 002056 G	4346#								

L\$SPCP	002020	G	4346#				
L\$SPTP	002024	G	4346#				
L\$STA	002030	G	4346#				
L\$SW	002302	G	4419#				
L\$TEST	002114	G	4346#				
L\$TIML	002014	G	4346#				
L\$UNIT	002012	G	4346#	6765			
L10000	002300		4391	4405#			
L10001	002302		4419	4422#			
L10002	014630		6533#				
L10003	015136		6546#				
L10004	015620		6565#				
L10005	016276		6584#				
L10006	017010		6604#				
L10007	017466		6623#				
L10010	020120		6641#				
L10011	020626		6661#				
L10012	021104		6675#				
L10013	021106		6692#				
L10015	021464		6790#				
L10016	021544		6816#				
L10017	021546		6833#				
L10020	021576		6852#				
L10021	021632		6871#				
L10022	021714		6905#				
L10023	022000		6930#				
L10024	022124		6968#				
L10025	022226		6995#				
L10026	022354		7028#				
L10027	022474		7064#				
L10030	022650		7105	7110#			
L10031	023104		7157	7164#			
L10032	023222		7198#				
L10033	023340		7232#				
L10034	023456		7266#				
L10035	023574		7300#				
L10036	024224		7401#				
L10037	024556		7430	7444	7456	7467	7472#
L10040	025046		7516	7526	7533#		
L10041	025256		7588#				
L10042	025462		7638#				
L10043	025666		7685#				
L10044	026124		7750#				
L10045	026356		7817#				
L10046	026262		7798#				
L10047	026354		7816#				
L10050	026630		7878#				
L10051	026626		7862	7872	7877#		
L10052	027306		7977	8000	8010#		
L10053	027426		8059#				
L10054	027536		8105#				
L10055	027732		8165#				
L10056	030276		8254#				
L10057	030506		8310#				
L10060	031036		8390#				
L10061	031136		8425#				









RX7 = 000200	4538#	4610#											
R14NRW 002560	4886#	6952	6955										
SAVE4 002414	4815#	6734*	6737	6814	6903								
SAVE6 002416	4816#	6735*	6738	6815	6904								
SAVLEN 002432	4822#	5353*	6157	6290*	6731*								
SCRACH 002342	4793#												
SEC = 000020	4671#												
SECA = 000020	4529#												
SEFR = 000040	4506#												
SELSBY= 000002	4510#												
SEL4 002454	4834#	6779*	6780*										
SEL6 002456	4835#	5325*	5440*	6781*	6782*								
SETUP 010640	6275#	8444	8524	8607	8670	8789							
SFPTBL 002302 G	4419#												
SIGQ = 000100	4599#												
SIGR = 000200	4598#												
SOM = 000001	4492#												
STALL 005164	5638#	5682	5684	5775	5780	5793							
STARES 002444	4827#	6754*	6759*	7332									
STARST 021256	6742	6745	6753#										
STBY = 000002	4580#												
STEPLU= 000020	4467#	5681	5683	5774	5779	5792							
STEPMP= 000001	4471#	5326	5327										
STPCLK 003540	5323#	5375	5399	7052									
STPERR 007646	6122#												
STPLU 005254	5672#	5840	6086	6129	6253	6343	6371	7922	7954	7990	8153	8196	8206
	8218	8237	8416	8459	8463	8477	8492	8500	8539	8545	8575	8622	8626
	8640	8685	8691	8721	8812	8833	8848	8869	8885	8972	8984	9000	9046
	9155	9198	9262	9312									
STR = 000040	4670#												
STRIP = 000010	4530#	8034	8131	8192	8233	8408	8446	8609	8950	9190	9297		
SUBRPC 002352	4797#	5572	5573	5575*	5576*	5613*	5701	5702	5704*	5705*	5726*	5747*	5748*
	5800*	5820*	5821*	5847*	5862*	5863*	5879*	5897	5898	5900*	5901*	5938*	5954
	5955	5957*	5958*	5979*	5994	5995	5997*	5998*	6033*	6072*	6073*	6109*	6152*
	6153*	6263*	6295*	6334*	6335*	6350*	6367*	6368*	6398*	6572	6611	6648	6727*
SVCGBL= 000000	4306#	4312	4320#	4346	4369	4391	4419	5285	5290	6531	6537	6552	6571
	6590	6610	6629	6647	6667	6690	6706	6724	6804	6830	6847	6870	9365
	9398	9424#											
SVCINS= 000001	4306#	4317#	4346	4369	4391	4419	5285	5290	5585	5591	5601	5607	5714
	5720	5878	5910	5916	5926	5932	5967	5973	6007	6013	6021	6027	6173
	6178	6197	6203	6212	6218	6227	6233	6242	6248	6380	6386	6532	6533
	6538	6539	6540	6541	6542	6543	6544	6545	6546	6553	6554	6555	6556
	6557	6558	6559	6560	6561	6562	6563	6564	6565	6572	6573	6574	6575
	6576	6577	6578	6579	6580	6581	6582	6583	6584	6591	6592	6593	6594
	6595	6596	6597	6598	6599	6600	6601	6602	6603	6604	6611	6612	6613
	6614	6615	6616	6617	6618	6619	6620	6621	6622	6623	6630	6631	6632
	6633	6634	6635	6636	6637	6638	6639	6640	6641	6648	6649	6650	6651
	6652	6653	6654	6655	6656	6657	6658	6659	6660	6661	6668	6669	6670
	6671	6672	6673	6674	6675	6692	6741	6742	6744	6745	6747	6748	6750
	6751	6767	6768	6790	6806	6813	6816	6833	6849	6851	6852	6871	6894
	6902	6905	6928	6930	6950	6962	6964	6968	6993	6995	7011	7026	7028
	7062	7064	7104	7105	7110	7156	7157	7164	7181	7192	7194	7198	7215
	7226	7228	7232	7249	7260	7262	7266	7283	7294	7296	7300	7334	7358
	7360	7377	7379	7396	7398	7401	7429	7430	7443	7444	7455	7456	7466
	7467	7472	7515	7516	7525	7526	7533	7558	7572	7573	7582	7584	7588
	7609	7623	7624	7632	7634	7638	7656	7670	7671	7679	7681	7685	7716



TXCHAR 006122	5818#	8035	8038	8041	8044	8047	8050	8053	8084	8087	8090	8093	8096
	8099	8135	8138	8141	8144	8147	8150	8193	8234	8278	8281	8286	8291
	8296	8299	8302	8305	8331	8334	8337	8344	8349	8354	8359	8364	8369
	8374	8379	8382	8385									
TXDATA= 000040	4600#	6376	6382										
TXEN = 000100	4515#	4587#	7018	8133	8795	8797							
TXEOM = 001000	4709#	5245	5246	5247	5248	5254	5255	8042	8045	8048	8051	8054	8091
	8094	8097	8100	8142	8145	8148	8151	8297	8300	8303	8306	8380	8383
	8386	9008	9194	9306									
TXGA = 000010	4647#												
TXGOA = 004000	4707#	9194											
TXLENO= 000040	4695#	6157	8276	8289	8342	8352	8362	8372					
TXLEN1= 000100	4694#	6157	8284	8289	8347	8352	8367	8372					
TXLEN2= 000200	4693#	6157	8276	8284	8289	8357	8362	8367	8372				
TXSOM = 000400	4710#	5238	5239	5756	7911	8215	9298						
TXWORD 002422	4818#	5652*	5654	5657	5756*	5757*	5822*	6313*	7911*	7936*	7984*	8215*	8411*
	8454*	8534*	8617*	8680*	8753*	8952*	8974*	9004*	9008*	9037*	9089*	9153*	9194*
	9243*	9298*	9306*										
TXC = 000001	4483#	4641#											
TX1 = 000002	4482#	4640#											
TX2 = 000004	4481#	4639#											
TX3 = 000010	4480#	4638#											
TX4 = 000020	4479#	4637#											
TX5 = 000040	4478#	4636#											
TX6 = 000100	4477#	4635#											
TX7 = 000200	4476#	4634#											
T\$ARGC= 000001	4346#	6532#	6538#	6539#	6540#	6541#	6542#	6543#	6544#	6545#	6553#	6554#	6555#
	6556#	6557#	6558#	6559#	6560#	6561#	6562#	6563#	6564#	6572#	6573#	6574#	6575#
	6576#	6577#	6578#	6579#	6580#	6581#	6582#	6583#	6591#	6592#	6593#	6594#	6595#
	6596#	6597#	6598#	6599#	6600#	6601#	6602#	6603#	6611#	6612#	6613#	6614#	6615#
	6616#	6617#	6618#	6619#	6620#	6621#	6622#	6630#	6631#	6632#	6633#	6634#	6635#
	6636#	6637#	6638#	6639#	6640#	6648#	6649#	6650#	6651#	6652#	6653#	6654#	6655#
	6656#	6657#	6658#	6659#	6660#	6668#	6669#	6670#	6671#	6672#	6673#	6674#	6851#
	7334#	7360#	7379#	7398#									
T\$CODE= 012032	9367#	9368#	9369#	9370#									
T\$ERRN= 000032	4306#	5585#	5591#	5601#	5607#	5714#	5720#	5878#	5910#	5916#	5926#	5932#	5967#
	5973#	6007#	6013#	6021#	6027#	6173#	6178#	6197#	6203#	6212#	6218#	6227#	6233#
	6242#	6248#	6380#	6386#	6902#	6928#	6962#	6993#	7026#	7062#	7104#	7156#	7192#
	7226#	7260#	7294#	7358#	7377#	7396#	7429#	7443#	7455#	7466#	7515#	7525#	7572#
	7582#	7623#	7632#	7670#	7679#	7733#	7743#	7791#	7814#	7861#	7871#	7976#	7999#
	8161#	8204#	8213#	8225#	8244#	8252#	8475#	8490#	8573#	8638#	8719#	8763#	8867#
T\$EXCP= 000000	9367#	9368#	9369#	9370#									
T\$FLAG= 000C40	7105#	7157#	7430#	7444#	7456#	7467#	7516#	7526#	7573#	7624#	7671#	7734#	7862#
	7872#	7977#	8000#										
T\$GMAN= 000000	4306#												
T\$HILI= 000007	9367#	9368#	9369#	9370#									
T\$LAST= 000001	4306#	9424#											
T\$LOLI= 000000	9367#	9368#	9369#	9370#									
T\$LSYM= 010000	4306#	4405	4422	6533	6546	6565	6584	6604	6623	6641	6661	6675	6692
	6790	6816	6833	6852	6871	6905	6930	6968	6995	7028	7064	7110	7164
	7198	7232	7266	7300	7401	7472	7533	7588	7638	7685	7750	7798	7816
	7817	7877	7878	8010	8059	8105	8165	8254	8310	8390	8425	8505	8588
	8651	8734	8766	8912	9013	9065	9133	9165	9218	9268	9338	9372	9401
T\$LTNO= 000052	9424#												
T\$NEST= 177777	4306#	4312#	4391#	4405#	4419#	4422#	6531#	6533#	6537#	6546#	6552#	6565#	6571#
	6584#	6590#	6604#	6610#	6623#	6629#	6641#	6647#	6661#	6667#	6675#	6690#	6692#

T\$NSO = 000000  
 T\$NS1 = 000005

T\$NS2 = 000002

T\$NS3 = 000003  
 T\$PTNU= 000000  
 T\$SAVL= 177777  
 T\$SEGL= 177777

T\$SEKO= 010000

T\$SUBN= 000000

T\$TAGL= 177777  
 T\$TAGN= 010101

T\$TEMP= 000000

T\$TEST= 000052

T\$TSTM= 177777

6706#	6710#	6724#	6790#	6804#	6816#	6830#	6833#	6847#	6852#	6870#	6871#	6892#
6905#	6917#	6930#	6945#	6950#	6964#	6968#	6979#	6995#	7006#	7028#	7044#	7064#
7081#	7110#	7125#	7164#	7176#	7181#	7194#	7198#	7210#	7215#	7228#	7232#	7244#
7249#	7262#	7266#	7278#	7283#	7296#	7300#	7327#	7401#	7417#	7472#	7488#	7533#
7554#	7558#	7584#	7588#	7605#	7609#	7634#	7638#	7652#	7656#	7681#	7685#	7710#
7716#	7745#	7750#	7767#	7776#	7778#	7793#	7798#	7802#	7816#	7817#	7835#	7838#
7877#	7878#	7900#	8010#	8030#	8059#	8079#	8105#	8127#	8165#	8185#	8254#	8269#
8310#	8326#	8390#	8405#	8425#	8441#	8505#	8521#	8588#	8604#	8651#	8667#	8734#
8747#	8766#	8783#	8912#	8936#	9013#	9031#	9065#	9083#	9133#	9148#	9165#	9186#
9218#	9237#	9268#	9291#	9338#	9365#	9372#	9398#	9401#	9422#			
4312#	9422											
4391#	4405	4419#	4422	6531#	6533	6537#	6546	6552#	6565	6571#	6584	6590#
6604	6610#	6623	6629#	6641	6647#	6661	6667#	6675	6690#	6692	6706#	6710
6724#	6790	6804#	6816	6830#	6833	6847#	6852	6870#	6871	6892#	6905	6917#
6930	6945#	6968	6979#	6995	7006#	7028	7044#	7064	7081#	7110	7125#	7164
7176#	7198	7210#	7232	7244#	7266	7278#	7300	7327#	7401	7417#	7472	7488#
7533	7554#	7588	7605#	7638	7652#	7685	7710#	7750	7767#	7817	7835#	7878
7900#	8010	8030#	8059	8079#	8105	8127#	8165	8185#	8254	8269#	8310	8326#
8390	8405#	8425	8441#	8505	8521#	8588	8604#	8651	8667#	8734	8747#	8766
8783#	8912	8936#	9013	9031#	9065	9083#	9133	9148#	9165	9186#	9218	9237#
9268	9291#	9338	9365#	9372	9398#	9401						
6950#	6964	7181#	7194	7215#	7228	7249#	7262	7283#	7296	7558#	7584	7609#
7634	7656#	7681	7716#	7745	7776#	7798	7802#	7816	7838#	7877		
7778#	7793											
4306#												
4306#												
4306#	6950#	6964#	7181#	7194#	7215#	7228#	7249#	7262#	7283#	7296#	7558#	7573
7584#	7609#	7624	7634#	7656#	7671	7681#	7716#	7734	7745#	7778#	7793#	
6950#	6964	7181#	7194	7215#	7228	7249#	7262	7283#	7296	7558#	7573	7584
7609#	7624	7634	7656#	7671	7681	7716#	7734	7745	7778#	7793		
4306#	6892#	6917#	6945#	6979#	7006#	7044#	7081#	7125#	7176#	7210#	7244#	7278#
7327#	7417#	7488#	7554#	7605#	7652#	7710#	7767#	7776#	7802#	7835#	7838#	7900#
8030#	8079#	8127#	8185#	8269#	8326#	8405#	8441#	8521#	8604#	8667#	8747#	8783#
8936#	9031#	9083#	9148#	9186#	9237#	9291#						
4306#												
4306#	4391#	4419#	6531#	6537#	6552#	6571#	6590#	6610#	6629#	6647#	6667#	6690#
6706#	6724#	6804#	6830#	6847#	6870#	6892#	6917#	6945#	6979#	7006#	7044#	7081#
7125#	7176#	7210#	7244#	7278#	7327#	7417#	7488#	7554#	7605#	7652#	7710#	7767#
7776#	7802#	7835#	7838#	7900#	8030#	8079#	8127#	8185#	8269#	8326#	8405#	8441#
8521#	8604#	8667#	8747#	8783#	8936#	9031#	9083#	9148#	9186#	9237#	9291#	9365#
9398#												
4369#	4405#	4422#	6533#	6546#	6565#	6584#	6604#	6623#	6641#	6661#	6675#	6692#
6710#	6790#	6816#	6833#	6852#	6871#	6905#	6930#	6964#	6968#	6995#	7028#	7064#
7105#	7110#	7157#	7164#	7194#	7198#	7228#	7232#	7262#	7266#	7296#	7300#	7401#
7477#	7444#	7456#	7467#	7472#	7516#	7526#	7533#	7573#	7584#	7588#	7624#	7634#
7638#	7671#	7681#	7685#	7734#	7745#	7750#	7793#	7798#	7816#	7817#	7862#	7872#
7877#	7878#	7977#	8000#	8010#	8059#	8105#	8165#	8254#	8310#	8390#	8425#	8505#
8588#	8651#	8734#	8766#	8912#	9013#	9065#	9133#	9165#	9218#	9268#	9338#	9367#
9368#	9369#	9370#	9372#	9401#	9422#							
4306#	6892#	6917#	6945#	6979#	7006#	7044#	7081#	7125#	7176#	7210#	7244#	7278#
7327#	7417#	7488#	7554#	7605#	7652#	7710#	7767#	7776	7802	7835#	7838	7900#
8030#	8079#	8127#	8185#	8269#	8326#	8405#	8441#	8521#	8604#	8667#	8747#	8783#
8936#	9031#	9083#	9148#	9186#	9237#	9291#	9424					
4306#	5585	5591	5601	5607	5714	5720	5878	5910	5916	5926	5932	5967
5973	6007	6013	6021	6027	6173	6178	6197	6203	6212	6218	6227	6233
6242	6248	6380	6386	6532	6533	6538	6539	6540	6541	6542	6543	6544

6545	6546	6553	6554	6555	6556	6557	6558	6559	6560	6561	6562	6563
6564	6565	6572	6573	6574	6575	6576	6577	6578	6579	6580	6581	6582
6583	6584	6591	6592	6593	6594	6595	6596	6597	6598	6599	6600	6601
6602	6603	6604	6611	6612	6613	6614	6615	6616	6617	6618	6619	6620
6621	6622	6623	6630	6631	6632	6633	6634	6635	6636	6637	6638	6639
6640	6641	6648	6649	6650	6651	6652	6653	6654	6655	6656	6657	6658
6659	6660	6661	6668	6669	6670	6671	6672	6673	6674	6675	6692	6741
6744	6747	6750	6767	6790	6806	6813	6816	6833	6849	6851	6852	6871
6894	6902	6905	6928	6930	6950	6962	6964	6968	6993	6995	7011	7026
7028	7062	7064	7104	7105	7110	7156	7157	7164	7181	7192	7194	7198
7215	7226	7228	7232	7249	7260	7262	7266	7283	7294	7296	7300	7334
7358	7360	7377	7379	7396	7398	7401	7429	7430	7443	7444	7455	7456
7466	7467	7472	7515	7516	7525	7526	7533	7558	7572	7573	7582	7584
7588	7609	7623	7624	7632	7634	7638	7656	7670	7671	7679	7681	7685
7716	7733	7734	7743	7745	7750	7776	7778	7791	7793	7798	7802	7814
7816	7817	7838	7861	7862	7871	7872	7877	7878	7976	7977	7999	8000
8010	8059	8105	8161	8165	8204	8213	8225	8244	8252	8254	8310	8390
8425	8475	8490	8505	8573	8588	8638	8651	8719	8734	8763	8766	8867
8912	9013	9065	9133	9165	9218	9268	9338					
4306#	6892#	6917#	6945#	6979#	7006#	7044#	7081#	7125#	7176#	7210#	7244#	7278#
7327#	7417#	7488#	7554#	7605#	7652#	7710#	7767#	7835#	7900#	8030#	8079#	8127#
8185#	8269#	8326#	8405#	8441#	8521#	8604#	8667#	8747#	8783#	8936#	9031#	9083#
9148#	9186#	9237#	9291#									
6870#	6871											
6804#	6816											
6830#	6833											
6847#	6852											
9365#	9372											
4391#	4405											
6724#	6790											
6531#	6533	6537#	6546	6552#	6565	6571#	6584	6590#	6604	6610#	6623	6629#
6641	6647#	6661	6667#	6675								
6706#												
6690#	6692											
6950#	6964#	7181#	7194#	7215#	7228#	7249#	7262#	7283#	7296#	7558#	7573	7584#
7609#	7624	7634#	7656#	7671	7681#	7716#	7734	7745#	7778#	7793#		
9398#	9401											
7776#	7798	7802#	7816	7838#	7862	7872	7877					
4419#	4422											
6892#	6905	6917#	6930	6945#	6968	6979#	6995	7006#	7028	7044#	7064	7081#
7105	7110	7125#	7157	7164	7176#	7198	7210#	7232	7244#	7266	7278#	7300
7327#	7401	7417#	7430	7444	7456	7467	7472	7488#	7516	7526	7533	7554#
7588	7605#	7638	7652#	7685	7710#	7750	7767#	7817	7835#	7878	7900#	7977
8000	8010	8030#	8059	8079#	8105	8127#	8165	8185#	8254	8269#	8310	8326#
8390	8405#	8425	8441#	8505	8521#	8588	8604#	8651	8667#	8734	8747#	8766
8783#	8912	8936#	9013	9031#	9065	9083#	9133	9148#	9165	9186#	9218	9237#
9268	9291#	9338										
4369	6892#											
4369	7210#											
4369	7244#											
4369	7278#											
4369	7327#											
4369	7417#											
4369	7488#											
4369	7554#											
4369	7605#											

T\$TSTS= 000001

T\$\$AU = 010021  
T\$\$AUT= 010016  
T\$\$CLE= 010017  
T\$\$DU = 010020  
T\$\$HAR= 010077  
T\$\$HW = 010000  
T\$\$INI= 010015  
T\$\$MSG= 010012

T\$\$PRO= 010014  
T\$\$RPT= 010013  
T\$\$SEG= 010000

T\$\$SOF= 010100  
T\$\$SUB= 010051  
T\$\$SW = 010001  
T\$\$TES= 010076

T1 021634 G  
T10 023224 G  
T11 023342 G  
T12 023460 G  
T13 023576 G  
T14 024226 G  
T15 024560 G  
T16 025050 G  
T17 025260 G



T18	025464	G	4369	7652#															
T19	025670	G	4369	7710#															
T2	021716	G	4369	6917#															
T20	026126	G	4369	7767#															
T20.1	026156		7776#																
T20.2	026264		7802#																
T21	026360	G	4369	7835#															
T21.1	026372		7838#																
T22	026632	G	4369	7900#															
T23	027310	G	4369	8030#															
T24	027430	G	4369	8079#															
T25	027540	G	4369	8127#															
T26	027734	G	4369	8185#															
T27	030300	G	4369	8269#															
T28	030510	G	4369	8326#															
T29	031040	G	4369	8405#															
T3	022002	G	4369	6945#															
T30	031140	G	4369	8441#															
T31	031510	G	4369	8521#															
T32	032052	G	4369	8604#															
T33	032322	G	4369	8667#															
T34	032664	G	4369	8747#															
T35	033014	G	4369	8783#															
T36	033726	G	4369	8936#															
T37	034210	G	4369	9031#															
T38	034410	G	4369	9083#															
T39	034710	G	4369	9148#															
T4	022126	G	4369	6979#															
T40	035000	G	4369	9186#															
T41	035142	G	4369	9237#															
T42	035310	G	4369	9291#															
T5	022230	G	4369	7006#															
T6	022356	G	4369	7044#															
T7	022476	G	4369	7081#															
T8	022652	G	4369	7125#															
T9	023106	G	4369	7176#															
UAM	= 000200	G	4449#																
UNIT	002440		4825#																
UNRR	= 000001		4557#	6169	8200	8209	8221	8240	8248										
UPBITS	002550		4876#	7093	7138	7144	7185	7219	7253	7287									
VECTOR	035646		9368	9375#															
V35	= 000020		4683#	4688															
WAIT50	005146		5624#	5766	5824	5867	6088	6317	6337	7849	7914	7938	8217	8415	8458				
			8538	8621	8684	8907	9041	9093	9197	9247	9309								
WAX	= 000010		4518#	4590#	5513														
WAX15	002374		4807#	5503*	5504	5755*	5757	6278*	6287*	7421*	7497*	7499	7560*	7562*	7566				
			7568	7611*	7613*	7617	7619	7658*	7660*	7664	7666	7718*	7722	8275*	8341*				
			8805*	8817*	8845*	9043*	9095*												
WAX16	002376		4808#	5507*	5508	5758*	6279*	6289*	6290	7422*	7498*	7500	7561*	7563*	7612*				
			7614*	7626	7628	7659*	7661*	7673	7675	7719*	8276*	8284*	8289*	8294*	8342*				
			8347*	8352*	8357*	8362*	8367*	8372*	8377*	8806*	8818*	8823*	8846*	9044*	9048*				
			9052*	9056*	9098*	9102*	9106*	9110*	9114*	9118*	9122*	9126*							
WRDYTO=	000002		4771#	5496	5522	7424													
WRIBYT	002366		4804#	5349*	5397*	5398	5460*	5461*	5462*	5463*	5498*	5499*	5500*	5504*	5508*				
			5511*	5512*	5513*	5514*	5654*	5657*	5762*	5865*	6124*	6125*	6131*	6283*	6951*				
			6952*	6956	6958	6982*	7009*	7085*	7136*	7137	7182*	7186	7188	7216*	7220				

CZDMRE M8203 STATIC DIAG #1  
CZDMRE.P11 08-JUN-81 13:27

L 15  
MACY11 30A(1052) 15-JUN-81 15:34 PAGE 8-19  
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0193

	7222	7250*	7254	7256	7284*	7288	7290	7779*	7803*	7843*	7846*	8133*	8451*
	8531*	8583*	8614*	8646*	8677*	8729*	8750*	8795*	8799*	8902*	8905*	9260*	
WRITAX 004312	5494#	5759	6280	6291	7423	7501	7564	7615	7662	7720	8277	8285	8290
	8295	8343	8348	8353	8358	8363	8368	8373	8378	8807	8826	8847	9045
	9049	9053	9057	9099	9103	9107	9111	9115	9119	9123	9127		
WRITLU 003750	5350	5391#	5464	5501	5505	5509	5515	5655	5658	5763	5866	6126	6132
	6284	6953	6983	7010	7086	7139	7183	7217	7251	7285	7780	7804	7844
	7847	7848	8134	8452	8532	8584	8615	8647	8678	8730	8751	8796	8800
	8903	8906	9261										
XYZ = 000100	4681#	4688											
X\$ALWA= 000000	4306#												
X\$FALS= 000040	4306#												
X\$OFFS= 000400	4306#												
X\$TRUE= 000020	4306#												
\$LSTIN= 000001	4315#												
\$LSTTA= 000001	4316#												
. = 036234	4298#	4788#	5262#	5290#	6525#	6855#	7105	7157	7430	7444	7456	7467	7516
	7526	7573	7624	7671	7734	7862	7872	7977	8000	9378#	9414#		



ENDSRV	580#	4306#													
ENDSUB	596#	4306#	7798	7816	7877										
ENDSW	614#	4306#	4422												
ENDTST	624#	4306#	6905	6930	6968	6995	7028	7064	7110	7164	7198	7232	7266	7300	7401
	7472	7533	7588	7638	7685	7750	7817	7878	8010	8059	8105	8165	8254	8310	8390
	8425	8505	8588	8651	8734	8766	8912	9013	9065	9133	9165	9218	9268	9338	
EQUALS	642#	4306#	4449												
ERRDF	714#	4306#	5585	5591	5601	5607	5714	5720	5878	5910	5916	5926	5932	5967	5973
	6007	6013	6021	6027	6173	6178	6197	6203	6212	6218	6227	6233	6242	6248	6380
	6386	6902	6928	6962	6993	7026	7062	7104	7156	7192	7226	7260	7294	7358	7377
	7396	7429	7443	7455	7466	7515	7525	7572	7582	7623	7632	7670	7679	7733	7743
	7791	7814	7861	7871	7976	7999	8161	8204	8213	8225	8244	8252	8475	8490	8573
	8638	8719	8763	8867											
ERRHRD	718#	4306#													
ERROR	722#	4306#													
ERRSF	726#	4306#													
ERRSOF	730#	4306#													
ERRTBL	734#	4306#													
ESCAPE	744#	4306#	7105	7157	7430	7444	7456	7467	7516	7526	7573	7624	7671	7734	7862
	7872	7977	8000												
EXIT	771#	4306#													
FEQUAL	810#	4306#													
GETBYT	824#	4306#													
GETPRI	834#	4306#													
GETWOR	829#	4306#													
GMANIA	839#	4306#													
GMANID	848#	4306#													
GMANIL	859#	4306#													
GPHARD	868#	4306#	6767												
GPRMA	874#	4306#	9367	9368											
GPRMD	903#	4306#	9369	9370											
GPRML	934#	4306#													
HEADER	954#	4306#	4346												
INLOOP	962#	4306#													
IOSETU	966#	4306#													
IOSTAR	974#	4306#													
KT11	982#	4306#													
LASTAD	1147#	4306#	9424												
MANUAL	1162#	4306#													
MEMORY	1166#	4306#													
M\$BYTE	2000#	4306#	4346#												
M\$CHEC	2118#	4306#													
M\$CNT0	2182#	4306#	9367#	9368#	9369#	9370#									
M\$COUN	2066#	4306#	6532#	6538#	6539#	6540#	6541#	6542#	6543#	6544#	6545#	6553#	6554#	6555#	6556#
	6557#	6558#	6559#	6560#	6561#	6562#	6563#	6564#	6572#	6573#	6574#	6575#	6576#	6577#	6578#
	6579#	6580#	6581#	6582#	6583#	6591#	6592#	6593#	6594#	6595#	6596#	6597#	6598#	6599#	6600#
	6601#	6602#	6603#	6611#	6612#	6613#	6614#	6615#	6616#	6617#	6618#	6619#	6620#	6621#	6622#
	6630#	6631#	6632#	6633#	6634#	6635#	6636#	6637#	6638#	6639#	6640#	6648#	6649#	6650#	6651#
	6652#	6653#	6654#	6655#	6656#	6657#	6658#	6659#	6660#	6668#	6669#	6670#	6671#	6672#	6673#
	6674#	6851#	7334#	7360#	7379#	7398#									
M\$DATA	1867#	4306#	4346#	5285#	5290#										
M\$DECR	2029#	4306#	4405#	4422#	6533#	6546#	6565#	6584#	6604#	6623#	6641#	6661#	6675#	6692#	6710#
	6790#	6816#	6833#	6852#	6871#	6905#	6930#	6964#	6968#	6995#	7028#	7064#	7110#	7164#	7194#
	7198#	7228#	7232#	7262#	7266#	7296#	7300#	7401#	7472#	7533#	7584#	7588#	7634#	7638#	7681#
	7685#	7745#	7750#	7793#	7798#	7816#	7817#	7877#	7878#	8010#	8059#	8105#	8165#	8254#	8310#
	8390#	8425#	8505#	8588#	8651#	8734#	8766#	8912#	9013#	9065#	9133#	9165#	9218#	9268#	9338#

M\$DEFA	9372#	9401#	9422#												
M\$ENDE	2170#	4306#	9367#	9368#	9369#	9370#									
	2074#	4306#	4405#	4422#	6533#	6546#	6565#	6584#	6604#	6623#	6641#	6661#	6675#	6692#	6790#
	6816#	6833#	6852#	6871#	6905#	6930#	6964#	6968#	6995#	7028#	7064#	7110#	7164#	7194#	7198#
	7228#	7232#	7262#	7266#	7296#	7300#	7401#	7472#	7533#	7584#	7588#	7634#	7638#	7681#	7685#
	7745#	7750#	7793#	7798#	7816#	7817#	7877#	7878#	8010#	8059#	8105#	8165#	8254#	8310#	8390#
	8425#	8505#	8588#	8651#	8734#	8766#	8912#	9013#	9065#	9133#	9165#	9218#	9268#	9338#	9372#
	9401#	9422#													
M\$ERRI	1649#	4306#	5585#	5591#	5601#	5607#	5714#	5720#	5878#	5910#	5916#	5926#	5932#	5967#	5973#
	6007#	6013#	6021#	6027#	6173#	6178#	6197#	6203#	6212#	6218#	6227#	6233#	6242#	6248#	6380#
	6386#	6902#	6928#	6962#	6993#	7026#	7062#	7104#	7156#	7192#	7226#	7260#	7294#	7358#	7377#
	7396#	7429#	7443#	7455#	7466#	7515#	7525#	7572#	7582#	7623#	7632#	7670#	7679#	7733#	7743#
	7791#	7814#	7861#	7871#	7976#	7999#	8161#	8204#	8213#	8225#	8244#	8252#	8475#	8490#	8573#
	8638#	8719#	8763#	8867#											
M\$ESCA	2006#	4306#	7105#	7157#	7430#	7444#	7456#	7467#	7516#	7526#	7573#	7624#	7671#	7734#	7862#
	7872#	7977#	8000#												
M\$ESCS	2010#	4306#	7105#	7157#	7430#	7444#	7456#	7467#	7516#	7526#	7573#	7624#	7671#	7734#	7862#
	7872#	7977#	8000#												
M\$EXCP	2101#	4306#	9367#	9368#	9369#	9370#									
M\$EXIT	2014#	4306#													
M\$EXSE	2022#	4306#													
M\$EXTJ	2018#	4306#													
M\$GEN	2038#	4306#	4312#	4346#	4369#	4391#	4405#	4419#	4422#	5285#	5290#	6531#	6533#	6537#	6546#
	6552#	6565#	6571#	6584#	6590#	6604#	6610#	6623#	6629#	6641#	6647#	6661#	6667#	6675#	6690#
	6692#	6706#	6724#	6790#	6804#	6816#	6830#	6833#	6847#	6852#	6870#	6871#	6892#	6905#	6917#
	6930#	6945#	6964#	6968#	6979#	6995#	7006#	7028#	7044#	7064#	7081#	7110#	7125#	7164#	7176#
	7194#	7198#	7210#	7228#	7232#	7244#	7262#	7266#	7278#	7296#	7300#	7327#	7401#	7417#	7472#
	7488#	7533#	7554#	7584#	7588#	7605#	7634#	7638#	7652#	7681#	7685#	7710#	7745#	7750#	7767#
	7776#	7793#	7798#	7802#	7816#	7817#	7835#	7838#	7877#	7878#	7900#	8010#	8030#	8059#	8079#
	8105#	8127#	8165#	8185#	8254#	8269#	8310#	8326#	8390#	8405#	8425#	8441#	8505#	8521#	8588#
	8604#	8651#	8667#	8734#	8747#	8766#	8783#	8912#	8936#	9013#	9031#	9065#	9083#	9133#	9148#
	9165#	9186#	9218#	9237#	9268#	9291#	9338#	9365#	9372#	9398#	9401#	9424#			
M\$GENB	1938#	4306#													
M\$GETS	2035#	4306#	4405#	4422#	6533#	6546#	6565#	6584#	6604#	6623#	6641#	6661#	6675#	6692#	6710#
	6790#	6816#	6833#	6852#	6871#	6905#	6930#	6964#	6968#	6995#	7028#	7064#	7110#	7164#	7194#
	7198#	7228#	7232#	7262#	7266#	7296#	7300#	7401#	7472#	7533#	7573#	7584#	7588#	7624#	7634#
	7638#	7671#	7681#	7685#	7734#	7745#	7750#	7793#	7798#	7816#	7817#	7877#	7878#	8010#	8059#
	8105#	8165#	8254#	8310#	8390#	8425#	8505#	8588#	8651#	8734#	8766#	8912#	9013#	9065#	9133#
	9165#	9218#	9268#	9338#	9372#	9401#	9422#								
M\$GETT	1877#	4306#	7105#	7157#	7430#	7444#	7456#	7467#	7516#	7526#	7573#	7624#	7671#	7734#	7862#
	7872#	7977#	8000#												
M\$GNGB	1902#	4306#	4312#	4346#	4369#	4391#	4419#	5285#	5290#	6531#	6537#	6552#	6571#	6590#	6610#
	6629#	6647#	6667#	6690#	6706#	6724#	6804#	6830#	6847#	6870#	9365#	9398#	9424#		
	2049#	4306#	4346#	4369#	4391#	4419#	5285#	5290#	5585#	5591#	5601#	5607#	5714#	5720#	5878#
	5910#	5916#	5926#	5932#	5967#	5973#	6007#	6013#	6021#	6027#	6173#	6178#	6197#	6203#	6212#
	6218#	6227#	6233#	6242#	6248#	6380#	6386#	6532#	6533#	6538#	6539#	6540#	6541#	6542#	6543#
	6544#	6545#	6546#	6553#	6554#	6555#	6556#	6557#	6558#	6559#	6560#	6561#	6562#	6563#	6564#
	6565#	6572#	6573#	6574#	6575#	6576#	6577#	6578#	6579#	6580#	6581#	6582#	6583#	6584#	6591#
	6592#	6593#	6594#	6595#	6596#	6597#	6598#	6599#	6600#	6601#	6602#	6603#	6604#	6611#	6612#
	6613#	6614#	6615#	6616#	6617#	6618#	6619#	6620#	6621#	6622#	6623#	6630#	6631#	6632#	6633#
	6634#	6635#	6636#	6637#	6638#	6639#	6640#	6641#	6648#	6649#	6650#	6651#	6652#	6653#	6654#
	6655#	6656#	6657#	6658#	6659#	6660#	6661#	6668#	6669#	6670#	6671#	6672#	6673#	6674#	6675#
	6692#	6741#	6742#	6744#	6745#	6747#	6748#	6750#	6751#	6767#	6768#	6790#	6806#	6813#	6816#
	6833#	6849#	6851#	6852#	6871#	6894#	6902#	6905#	6928#	6930#	6950#	6962#	6964#	6968#	6993#
	6995#	7011#	7026#	7028#	7062#	7064#	7104#	7105#	7110#	7156#	7157#	7164#	7181#	7192#	7194#
	7198#	7215#	7226#	7228#	7232#	7249#	7260#	7262#	7266#	7283#	7294#	7296#	7300#	7334#	7358#

	7360#	7377#	7379#	7396#	7398#	7401#	7429#	7430#	7443#	7444#	7455#	7456#	7466#	7467#	7472#
	7515#	7516#	7525#	7526#	7533#	7558#	7572#	7573#	7582#	7584#	7588#	7609#	7623#	7624#	7632#
	7634#	7638#	7656#	7670#	7671#	7679#	7681#	7685#	7716#	7733#	7734#	7743#	7745#	7750#	7776#
	7778#	7791#	7793#	7798#	7802#	7814#	7816#	7817#	7838#	7861#	7862#	7871#	7872#	7877#	7878#
	7976#	7977#	7999#	8000#	8010#	8059#	8105#	8161#	8165#	8204#	8213#	8225#	8244#	8252#	8254#
	8310#	8390#	8425#	8475#	8490#	8505#	8573#	8588#	8638#	8651#	8719#	8734#	8763#	8766#	8867#
	8912#	9013#	9065#	9133#	9165#	9218#	9268#	9338#	9365#	9367#	9368#	9369#	9370#	9372#	9398#
	9401#	9424#													
M\$GNLS	1913#	4306#	6964#	7194#	7228#	7262#	7296#	7584#	7634#	7681#	7745#	7793#			
M\$GNSU	1898#	4306#	7776#	7802#	7838#										
M\$GNTA	1890#	4306#	4405#	4422#	6533#	6546#	6565#	6584#	6604#	6623#	6641#	6661#	6675#	6692#	6790#
	6816#	6833#	6852#	6871#	6905#	6930#	6968#	6995#	7028#	7064#	7110#	7164#	7198#	7232#	7266#
	7300#	7401#	7472#	7533#	7588#	7638#	7685#	7750#	7798#	7816#	7817#	7877#	7878#	8010#	8059#
	8105#	8165#	8254#	8310#	8390#	8425#	8505#	8588#	8651#	8734#	8766#	8912#	9013#	9065#	9133#
	9165#	9218#	9268#	9338#	9372#	9401#									
M\$GNTE	1894#	4306#	6892#	6917#	6945#	6979#	7006#	7044#	7081#	7125#	7176#	7210#	7244#	7278#	7327#
	7417#	7488#	7554#	7605#	7652#	7710#	7767#	7835#	7900#	8030#	8079#	8127#	8185#	8269#	8326#
	8405#	8441#	8521#	8604#	8667#	8747#	8783#	8936#	9031#	9083#	9148#	9186#	9237#	9291#	
M\$HAPT	1739#	4306#	4346#												
M\$HNAP	1824#	4306#	4346#												
M\$INCR	2026#	4306#	4312#	4391#	4419#	5585#	5591#	5601#	5607#	5714#	5720#	5878#	5910#	5916#	5926#
	5932#	5967#	5973#	6007#	6013#	6021#	6027#	6173#	6178#	6197#	6203#	6212#	6218#	6227#	6233#
	6242#	6248#	6380#	6386#	6531#	6532#	6533#	6537#	6538#	6539#	6540#	6541#	6542#	6543#	6544#
	6545#	6546#	6552#	6553#	6554#	6555#	6556#	6557#	6558#	6559#	6560#	6561#	6562#	6563#	6564#
	6565#	6571#	6572#	6573#	6574#	6575#	6576#	6577#	6578#	6579#	6580#	6581#	6582#	6583#	6584#
	6590#	6591#	6592#	6593#	6594#	6595#	6596#	6597#	6598#	6599#	6600#	6601#	6602#	6603#	6604#
	6610#	6611#	6612#	6613#	6614#	6615#	6616#	6617#	6618#	6619#	6620#	6621#	6622#	6623#	6629#
	6630#	6631#	6632#	6633#	6634#	6635#	6636#	6637#	6638#	6639#	6640#	6641#	6647#	6648#	6649#
	6650#	6651#	6652#	6653#	6654#	6655#	6656#	6657#	6658#	6659#	6660#	6661#	6667#	6668#	6669#
	6670#	6671#	6672#	6673#	6674#	6675#	6690#	6692#	6706#	6724#	6741#	6744#	6747#	6750#	6767#
	6790#	6804#	6806#	6813#	6816#	6830#	6833#	6847#	6849#	6851#	6852#	6870#	6871#	6892#	6894#
	6902#	6905#	6917#	6928#	6930#	6945#	6950#	6962#	6964#	6968#	6979#	6993#	6995#	7006#	7011#
	7026#	7028#	7044#	7062#	7064#	7081#	7104#	7105#	7110#	7125#	7156#	7157#	7164#	7176#	7181#
	7192#	7194#	7198#	7210#	7215#	7226#	7228#	7232#	7244#	7249#	7260#	7262#	7266#	7278#	7283#
	7294#	7296#	7300#	7327#	7334#	7358#	7360#	7377#	7379#	7396#	7398#	7401#	7417#	7429#	7430#
	7443#	7444#	7455#	7456#	7466#	7467#	7472#	7488#	7515#	7516#	7525#	7526#	7533#	7554#	7558#
	7572#	7573#	7582#	7584#	7588#	7605#	7609#	7623#	7624#	7632#	7634#	7638#	7652#	7656#	7670#
	7671#	7679#	7681#	7685#	7710#	7716#	7733#	7734#	7743#	7745#	7750#	7767#	7776#	7778#	7791#
	7793#	7798#	7802#	7814#	7816#	7817#	7835#	7838#	7861#	7862#	7871#	7872#	7877#	7878#	7900#
	7976#	7977#	7999#	8000#	8010#	8030#	8059#	8079#	8105#	8127#	8161#	8165#	8185#	8204#	8213#
	8225#	8244#	8252#	8254#	8269#	8310#	8326#	8390#	8405#	8425#	8441#	8475#	8490#	8505#	8521#
	8573#	8588#	8604#	8638#	8651#	8667#	8719#	8734#	8747#	8763#	8766#	8783#	8867#	8912#	8936#
	9013#	9031#	9065#	9083#	9133#	9148#	9165#	9186#	9218#	9237#	9268#	9291#	9338#	9365#	9398#
M\$IOSE	1700#	4306#													
M\$LDRO	1942#	4306#	6741#	6744#	6747#	6750#	6767#	6806#	6813#	6894#					
M\$MASK	1671#	4306#													
M\$MCHI	4#	4306#													
M\$MCLO	1624#	4306#													
M\$MSK1	1677#	4306#													
M\$POP	1881#	4306#	4405#	4422#	6533#	6546#	6565#	6584#	6604#	6623#	6641#	6661#	6675#	6692#	6710#
	6790#	6816#	6833#	6852#	6871#	6905#	6930#	6964#	6968#	6995#	7028#	7064#	7110#	7164#	7194#
	7198#	7228#	7232#	7262#	7266#	7296#	7300#	7401#	7472#	7533#	7584#	7588#	7634#	7638#	7681#
	7685#	7745#	7750#	7793#	7798#	7816#	7817#	7877#	7878#	8010#	8059#	8105#	8165#	8254#	8310#
	8390#	8425#	8505#	8588#	8651#	8734#	8766#	8912#	9013#	9065#	9133#	9165#	9218#	9268#	9338#
	9372#	9401#	9422#												
M\$PRIN	1636#	4306#	6532#	6538#	6539#	6540#	6541#	6542#	6543#	6544#	6545#	6553#	6554#	6555#	6556#

	6557#	6558#	6559#	6560#	6561#	6562#	6563#	6564#	6572#	6573#	6574#	6575#	6576#	6577#	6578#
	6579#	6580#	6581#	6582#	6583#	6591#	6592#	6593#	6594#	6595#	6596#	6597#	6598#	6599#	6600#
	6601#	6602#	6603#	6611#	6612#	6613#	6614#	6615#	6616#	6617#	6618#	6619#	6620#	6621#	6622#
	6630#	6631#	6632#	6633#	6634#	6635#	6636#	6637#	6638#	6639#	6640#	6648#	6649#	6650#	6651#
	6652#	6653#	6654#	6655#	6656#	6657#	6658#	6659#	6660#	6668#	6669#	6670#	6671#	6672#	6673#
M\$PUSH	6674#	6851#	7334#	7360#	7379#	7398#									
	1631#	4306#	4312#	4391#	4419#	6531#	6537#	6552#	6571#	6590#	6610#	6629#	6647#	6667#	6690#
	6706#	6724#	6804#	6830#	6847#	6870#	6892#	6917#	6945#	6950#	6979#	7006#	7044#	7081#	7125#
	7176#	7181#	7210#	7215#	7244#	7249#	7278#	7283#	7327#	7417#	7488#	7554#	7558#	7605#	7609#
	7652#	7656#	7710#	7716#	7767#	7776#	7778#	7802#	7835#	7838#	7900#	8030#	8079#	8127#	8185#
	8269#	8326#	8405#	8441#	8521#	8604#	8667#	8747#	8783#	8936#	9031#	9083#	9148#	9186#	9237#
	9291#	9365#	9398#												
M\$PUT	1972#	4306#	6532#	6538#	6539#	6540#	6541#	6542#	6543#	6544#	6545#	6553#	6554#	6555#	6556#
	6557#	6558#	6559#	6560#	6561#	6562#	6563#	6564#	6572#	6573#	6574#	6575#	6576#	6577#	6578#
	6579#	6580#	6581#	6582#	6583#	6591#	6592#	6593#	6594#	6595#	6596#	6597#	6598#	6599#	6600#
	6601#	6602#	6603#	6611#	6612#	6613#	6614#	6615#	6616#	6617#	6618#	6619#	6620#	6621#	6622#
	6630#	6631#	6632#	6633#	6634#	6635#	6636#	6637#	6638#	6639#	6640#	6648#	6649#	6650#	6651#
	6652#	6653#	6654#	6655#	6656#	6657#	6658#	6659#	6660#	6668#	6669#	6670#	6671#	6672#	6673#
	6674#	6851#	7334#	7360#	7379#	7398#									
M\$PUT1	1981#	4306#	6532#	6538#	6539#	6540#	6541#	6542#	6543#	6544#	6545#	6553#	6554#	6555#	6556#
	6557#	6558#	6559#	6560#	6561#	6562#	6563#	6564#	6572#	6573#	6574#	6575#	6576#	6577#	6578#
	6579#	6580#	6581#	6582#	6583#	6591#	6592#	6593#	6594#	6595#	6596#	6597#	6598#	6599#	6600#
	6601#	6602#	6603#	6611#	6612#	6613#	6614#	6615#	6616#	6617#	6618#	6619#	6620#	6621#	6622#
	6630#	6631#	6632#	6633#	6634#	6635#	6636#	6637#	6638#	6639#	6640#	6648#	6649#	6650#	6651#
	6652#	6653#	6654#	6655#	6656#	6657#	6658#	6659#	6660#	6668#	6669#	6670#	6671#	6672#	6673#
	6674#	6851#	7334#	7360#	7379#	7398#									
M\$RADI	2077#	4306#	9367#	9368#	9369#	9370#									
M\$RBRO	1952#	4306#													
M\$RNRO	1962#	4306#	6767#												
M\$SETS	2032#	4306#	4312#	4391#	4419#	6531#	6537#	6552#	6571#	6590#	6610#	6629#	6647#	6667#	6690#
	6706#	6724#	6804#	6830#	6847#	6870#	6892#	6917#	6945#	6950#	6979#	7006#	7044#	7081#	7125#
	7176#	7181#	7210#	7215#	7244#	7249#	7278#	7283#	7327#	7417#	7488#	7554#	7558#	7605#	7609#
	7652#	7656#	7710#	7716#	7767#	7776#	7778#	7802#	7835#	7838#	7900#	8030#	8079#	8127#	8185#
	8269#	8326#	8405#	8441#	8521#	8604#	8667#	8747#	8783#	8936#	9031#	9083#	9148#	9186#	9237#
	9291#	9365#	9398#												
M\$STAR	1733#	4306#													
M\$SVC	1933#	4306#	5585	5591	5601	5607	5714	5720	5878	5910	5916	5926	5932	5967	5973
	6007	6013	6021	6027	6173	6178	6197	6203	6212	6218	6227	6233	6242	6248	6380
	6386	6532#	6533#	6538#	6539#	6540#	6541#	6542#	6543#	6544#	6545#	6546#	6553#	6554#	6555#
	6556#	6557#	6558#	6559#	6560#	6561#	6562#	6563#	6564#	6565#	6572#	6573#	6574#	6575#	6576#
	6577#	6578#	6579#	6580#	6581#	6582#	6583#	6584#	6591#	6592#	6593#	6594#	6595#	6596#	6597#
	6598#	6599#	6600#	6601#	6602#	6603#	6604#	6611#	6612#	6613#	6614#	6615#	6616#	6617#	6618#
	6619#	6620#	6621#	6622#	6623#	6630#	6631#	6632#	6633#	6634#	6635#	6636#	6637#	6638#	6639#
	6640#	6641#	6648#	6649#	6650#	6651#	6652#	6653#	6654#	6655#	6656#	6657#	6658#	6659#	6660#
	6661#	6668#	6669#	6670#	6671#	6672#	6673#	6674#	6675#	6692#	6741#	6744#	6747#	6750#	6767#
	6790#	6806#	6813#	6816#	6833#	6849#	6851#	6852#	6871#	6894#	6902	6905#	6928	6930#	6950#
	6962	6964#	6968#	6993	6995#	7011#	7026	7028#	7062	7064#	7104	7105#	7110#	7156	7157#
	7164#	7181#	7192	7194#	7198#	7215#	7226	7228#	7232#	7249#	7260	7262#	7266#	7283#	7294
	7296#	7300#	7334#	7358	7360#	7377	7379#	7396	7398#	7401#	7429	7430#	7443	7444#	7455
	7456#	7466	7467#	7472#	7515	7516#	7525	7526#	7533#	7558#	7572	7573#	7582	7584#	7588#
	7609#	7623	7624#	7632	7634#	7638#	7656#	7670	7671#	7679	7681#	7685#	7716#	7733	7734#
	7743	7745#	7750#	7776#	7778#	7791	7793#	7798#	7802#	7814	7816#	7817#	7838#	7861	7862#
	7871	7872#	7877#	7878#	7976	7977#	7999	8000#	8010#	8059#	8105#	8161	8165#	8204	8213
	8225	8244	8252	8254#	8310#	8390#	8425#	8475	8490	8505#	8573	8588#	8638	8651#	8719
	8734#	8763	8766#	8867	8912#	9013#	9065#	9133#	9165#	9218#	9258#	9338#			
M\$TLAB	1929#	4306#	5585#	5591#	5601#	5607#	5714#	5720#	5878#	5910#	5916#	5926#	5932#	5967#	5973#





CZDMRE M8203 STATIC DIAG #1  
CZDMRE.P11 08-JUN-81 13:27

MACY11 30A(1052) 15-JUN-81 15:34 PAGE 9-6  
CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0200

REDEF	1403#	4306#	6741	6744	6747	6750
RFLAGS	1408#	4306#				
SETPRI	1413#	4306#	6806	6894		
SETVEC	1418#	4306#				
SLASH	1424#	4306#				
STARS	1438#	4306#				
SVC	1452#	4305#	4306			
XFER	1612#	4306#				
XFERF	1616#	4306#				
XFERT	1620#	4306#				

. ABS. 036234 000

ERRORS DETECTED: 0

CZDMRE.BIC,CZDMRE.SEG/CRF/DOC/NL:TOC=SVC34R.MLB,CZDMRE.P11  
RUN-TIME: 35 45 4 SECONDS  
RUN-TIME RATIO: 106/85=1.2  
CORE USED: 17K (33 PAGES)

DOCUMENT PAGES: 200