GEORGIA INSTITUTE OF TECHNOLOGY
OFFICE OF CONTRACT ADMINISTRATION
SPONSORED PROJECT INITIATION

Date: 9/27/79

Project Title: Develop Software Package for Test Receiver Hardware

Project No: A-2448

Project Director: Ms. C. M. Agee

Sponsor: Scientific Atlanta; Atlanta, Georgia 30340

Agreement Period: From 8/27/79 Until 12/17/79

Type Agreement: Purchase Order No. 052588

Amount: Not to exceed $18,981

Reports Required: Monthly Status Reports; Final Project Report

Sponsor Contact Person (s):

Technical Matters

Contractual Matters
(thru OCA)

Mr. Sonny Morris
Purchasing Agent
Scientific Atlanta
Instrumentation Division
3845 Pleasantdale Road
Atlanta, Georgia 30340
(404) 449-2000

Defense Priority Rating: None

Assigned to: CSTL/SAD (School/Laboratory)

COPIES TO:

- Project Director
- Division Chief (EES)
- School/Laboratory Director
- Dean/Director–EES
- Accounting Office
- Procurement Office
- Security Coordinator (OCA)
- Reports Coordinator (OCA)
- Library, Technical Reports Section
- EES Information Office
- EES Reports & Procedures
- Project File (OCA)
- Project Code (GTRI)
- Other

GA-3 (3/79)

GEORGIA INSTITUTE OF TECHNOLOGY OFFICE OF CONTRACT ADMINISTRATION

SPONSORED PROJECT TERMINATION SHEET

Date 12/14/81

Project Title: Develop Software Package for Test Receiver Hardware

Project No: A-2448

Project Director: Ms. C. M. Agee

Sponsor: Scientific-Atlanta

Effective Termination Date: 11/2/81

Clearance of Accounting Charges: 11/2/81

Grant/Contract Closeout Actions Remaining:

- [x] Final Invoice ~~and Closing Documents~~
- [ ] Final Fiscal Report
- [ ] Final Report of Inventions
- [ ] Govt. Property Inventory & Related Certificate
- [ ] Classified Material Certificate
- [ ] Other _____

NOTE: This termination is effective for entire project including subbudgets.

Assigned to: ECSL/CTAD (~~School~~/Laboratory)

COPIES TO:

Administrative Coordinator	Research Security Services	EES Public Relations (2)
Research Property Management	Reports Coordinator (OCA)	Computer Input
Accounting	Legal Services (OCA)	Project File
Procurement/EES Supply Services	Library	Other _____

FORM OCA 10:781

TEST RECEIVER SOFTWARE DEVELOPMENT FOR SCIENTIFIC ATLANTA

Monthly Status Report
27 August through 30 September 1979

EES/GIT Project A-2448

Prepared by

Computer Science and Technology Laboratory
Engineering Experiment Station
Georgia Institute of Technology
Atlanta, Georgia 30332

C. M. Agee

for

Scientific Atlanta
3845 Pleasantdale Road
Atlanta, GA 30340

Following special accounting arrangements, EES was able to begin work on the Scientific Atlanta (SA)test receiver software on August 27, 1979. As of October 1, design, code, entry and compilation of the FORTRAN program has been accomplished. Portions of the I/O drivers, and the interface between the FORTRAN routines and the floating point package remain to be coded.

Assembly of I/O drivers will be accomplished beginning today. Some work was done on the clock drivers according to the plan to use the Fairchild chip, but now code of the clock routines is dependent on detailed information on the revised clock hardware.

Since there were aspects of the operator interface of the test receiver which were not defined at project initiation, the operator's interface has been specified as completely as possible during the past month during discussions between EES and SA personnel. A front panel description was completed September 10 and given to our technical contact, R. Larson. A revised version of the front panel description is attached. As currently configured, the software will be driven by the one interrupt available in the system. The interrupt (Restart 7) will trigger on the events (or some combination of them) listed below in order of priority.

(1) 1 million bits

(2) 1 second tick

(3) front panel input

We also understand that 2 extra E-prom sockets can be made available as insurance that the FORTRAN software can be accomodated after an IEEE488 (or RS-232) interface is added.

We request that the front panel test hardware be set up as soon as possible. We estimate at least one-two weeks of software tests w/front panel will be needed prior to system test.

The information that we need on the system clock has been delayed due to the new chip chosen, lack of information about it, and the inability to acheive operation of chip as manufacturer described. Work on the clock routines has to be delayed until more information is available.

Any feedback on the front panel operation as described in the attached description would be appreciated.

MAN MONTHS

August:	Carol Agee	.25
	Total expenditure: $785.00	
	Percent of budget remaining: 95	
September:	Carol Agee	.90
	John Scoville	.10
	Gary Peckham	.05
	Secretarial	.01
	Total expenditure: $3,589.00	
	Percent of budget remaining: 77	

TEST RECEIVER SOFTWARE DEVELOPMENT FOR SCIENTIFIC ATLANTA

Monthly Status Report
October 1979

EES/GIT Project A-2448

Prepared by

Computer Science and Technology Laboratory
Engineering Experiment Station
Georgia Institute of Technology
Atlanta, Georgia 30332

C. M. Agee

for

Scientific Atlanta
3845 Pleasantdale Road
Atlanta, GA 30340

During the month of October 1979, coding needed for the test receiver was completed after the final information on the time of day clock was made available. Test of the routines using breakpoints then proceeded. By the end of October, all assembly level code had been tested in this manner.

On October 16, an MDS with 32k of memory was moved temporily from Scientific Atlanta to EES to facilitate the software development. On October 25, 1979 test of the software driver for the front panel with corresponding front panel hardware began using the ICE80 emulator.

Tasks yet to be accomplished as of October 31, 1979:

1. Integration of I/O drivers and Fortran routines
2. Test of portions of Fortran code
3. Test of time of day clock driver with clock hardware
4. System test of software
5. Documentation

The delay in the availability of the test receiver hardware has slowed software development and test. We are aware that system test was to begin October 15 per our proposal for this project. We are anxious to test the software with additional parts of the hardware as soon as it is available.

MAN MONTHS

October:	Carol Agee	.80
	Jeff Hopper	.70
	John Scoville	.10
	Gary Peckham	.05

Total expenditure: $5,790.

Percent of budget remaining: 46

A-2448

Georgia Institute of Technology
ENGINEERING EXPERIMENT STATION
ATLANTA, GEORGIA 30332

28 November 1979

Mr. Sam Davis
Scientific Atlanta
3845 Pleasantdale Road
Atlanta, GA 30340

Dear Mr. Davis:

We are anxious to help you get the test receiver operable as soon as possible. All the software has been undergoing a close check using the ICE80 emulator and the front panel prototype hardware. Debug has gone well especially since the faulty disk controller boards in the development system were replaced at the first of this month. Software is ready for system test whenever the hardware can be set up. In the meantime, some documentation can be accomplished.

Of course the actual hardware-software interface for the clock, error counters, and one second and one million bits remains to be checked when the hardware is completed.

The use of Scientific Atlanta's MDS here at EES has certainly been a help during development. With the addition of the second 32k of memory, the Fortran programs can be compiled on it. SA's support and open communication on the project has been greatly appreciated.

Some of the software has taken longer to check out than was anticipated because the corresponding hardware was not available. Also the change in the choice of time of day clock slowed the coding somewhat.

System test, now planned to begin November 28, is occurring several weeks later than October 15 which we quoted in our proposal written in August. I have voiced my concern about this timing to Robin Larson during our communication concerning the technical aspects of the project.

Our original estimate for development of the test receiver software including an IEEE488 bus interface was $18,981. At the end of October $8,817 was as yet unused. With expenditures for November estimated to be $5,000, $3,800 would remain. Depending on how quickly system test proceeds and on how many modifications, if any, are needed, there is probably enough money to get

your system operable with the front panel and to complete documentation. As it looks now, there won't be adequate funding for EES to write an IEE488 bus interface.

I am aware that there has been a question of who would write the IEEE-488 (or RS-232) interface routines. We would be happy to assist you with that phase of the project. The additional interface can probably be written and tested quite easily with the bulk of the system already operable. If you do decide to have EES complete this second phase of the project, an estimate of cost for the work (over the original $19,000) would need your authorization. For your information, the paperwork for such as addendum usually takes about two weeks.

If you have any questions, I am, of course, available to talk with you further at any time.

Sincerely,

Carol Agee

Carol Agee
Project Director

CA/jg

Georgia Institute of Technology
ENGINEERING EXPERIMENT STATION
ATLANTA, GEORGIA 30332

14 December 1979

Mr. Sam Davis
Scientific Atlanta
3845 Pleasantdale Road
Atlanta, GA 30340

Dear Mr. Davis:

As you know, system test of the test receiver has been underway for approximately two weeks. Necessary modifications have been made to the software as hardware debug has progressed. All modules of the test receiver (with front panel) software are operable.

This week, the need to reduce processing during interrupt service was identified. A plan has been formulated to eliminate conversion of floating point results to BCD during any interrupt service. We have been working on a revision to accomplish this goal.

Documentation of the project remains to be completed. It has also been suggested that some self-check features be added to the front panel utilizing some of the undefined combinations of switches.

The project budget which you originally authorized for this project has been exhausted by our efforts to date. We anticipate being able to complete documentation and provide necessary support to you through December and on a half-time basis in January with an extension of $4,600.

Sincerely,

Carol Agee

Carol M. Agee
Project Director

CMA/jg

12-18-79

TEST RECEIVER SOFTWARE DEVELOPMENT

MANMONTHS

NOVEMBER 1979

CAROL AGEE	.92	
JEFF HOPPER	.50	
GARY PECKHAM	.05	
Total Expenditure:		$4781
Per cent of budget remaining:		21

DECEMBER 1979

CAROL AGEE	.96	
JEFF HOPPER	.50	
GARY PECKHAM	.15	
Total expenditure:		$5670
over run:		$1640

JANUARY 1980

CAROL AGEE/JEFF HOPPER	.50	
GARY PECKAM	.05	
Total expenditure:		$2150
Over run:		$2150

Total over-run: $3790

C. Agee

A-2448

Georgia Institute of Technology
ENGINEERING EXPERIMENT STATION
ATLANTA, GEORGIA 30332

April 30, 1980

Mr. Sam Davis
Scientific Atlanta
3845 Pleasantdale Road
Atlanta, Georgia 30340

RE: Project Status Feb-April, 1980 (GIT/EES A-2448)

Dear Mr. Davis:

During March, EES received authorization of $17,000 from Scientific Atlanta's Instrumentation Division for software development of peripheral interfaces for the BERTS Receiver (as described in letter of proposal 22 February 1980). Aside from minor modification to the BERTS firmware during March, the new phase of development actually began on March 31.

As you are aware, the BERTS receiver software (first phase) was essentially completed in February. At the request of Robin Larson of S.A., documentation completed at that time will be delivered in updated form once the peripheral interfaces are implemented.

During April, major definition of operator interfaces was achieved, the printer driver was written and tested, and design/code of new routines and modifications to BERTS main data acquisition and processing routines has commenced.

Project expenditures for Feb-March and man hours thru April are attached.

Sincerely,

Carol Agee

Carol Agee

CA/cp

Enclosure

Man Months

February:	Carol Agee	.06
	Total Expenditures: $249.19	
	Percent of Budget Remaining: 0	
March:	Carol Agee	.12
	Total Expenditures: $587.67	
	Percent of Budget Remaining: 96	
April:	Carol Agee	.68

A-2448

TEST RECEIVER SOFTWARE DEVELOPMENT FOR SCIENTIFIC ATLANTA

Monthly Status Report
May through July 1980

EES/GIT Project A-2448

Prepared by

Computer Science and Technology Laboratory
Engineering Experiment Station
Georgia Institute of Technology
Atlanta, Georgia 30332

C. M. Agee

for

Scientific Atlanta
3845 Pleasantdale Road
Atlanta, GA 30340

Georgia Institute of Technology
ENGINEERING EXPERIMENT STATION
ATLANTA, GEORGIA 30332

July 22, 1980

Robin Larson, Instrumentation Division
Scientific Atlanta
3845 Pleasantdale Road
Atlanta, Georgia 30340

Dear Mr. Larson:

As you are aware, the BERTS software including the local logger, an RS-232C interface, and an I-EEE488 Bus interface is approaching completion. The bulk of the coding was done during May. System debug began the first week of June. A semi-operable interface to the local logger was available for demonstration at a Seattle conference by June 5. Debug preceded through June and into July. All software functions are essentially complete and tested. Only minor adjustment to the operator interface and documentation remains. An E-PROM version was made on July 17.

Please refer to the attached man hours for a breakdown of expenditures. The amount of funding left available for July was $3,662. My calculation of July charges to date exceeds that figure by $970. Some additional effort is also indicated to provide documentation and final adjustment to the software.

Several factors have slowed the software effort including time spent getting SA's RS-232 and IEE488 cards to function with the program. The development prototype was unavailable to us the week of July 7 because it was at a customer site. Also the BERTS software and software functions had to be trimmed to fit in 12K of Eproms.

As we discussed one additional man month should be allowed to complete documentation and provide final software support. A total authorization of $4,800 is needed to complete the project.

Sincerely,

Carol M. Agee
Research Scientist

CMA:ae
cc: JFP, GLP

MAN MONTHS

April:	Carol Agee	.68
	Total Expenditure	$2,370
	Percent of Budget Remaining:	82
May:	Carol Agee	1.00
	Jeff Hopper	.25
	Gary Peckham	.10
	Total Expenditure:	4,932
	Percent of budget remaining:	53
June:	Carol Agee	.88
	Jeff Hopper	.50
	Gary Peckham	.10
	Total	5,431
	Percent remaining:	21
	Amount remaining:	3,662.97
July 1 - July 22:		
	Carol Agee	.50
	Jeff Hopper	.50
	Gary Peckham	.10
	Aproximate Total	4662
	Percent overrun: .04	
	Amount overrun:	$1,000

Estimated to complete project:

Carol Agee	.50
Jeff Hopper	.50
	$3800

Total authorization needed: $4,800

Georgia Institute of Technology
ENGINEERING EXPERIMENT STATION
ATLANTA, GEORGIA 30332

March 20, 1981

Mr. Charles Trawick
Scientific Atlanta
Instrumentation
3845 Pleasantdale Road
Atlanta, Georgia 30340

Dear Mr. Trawick:

Subject: Progress Report for February, 1981 on Project #A-2448-002

The following items have been accomplished in the last month:

1. New driver clock tasks have been completed and tested out.

2. I have become familiar with Carol Agee's program and with the scope of the changes required in the FORTRAN.

3. Completed work in the driver section for the new buffering scheme and have mapped out the changes required in the FORTRAN to produce the new formatting.

4. Carol has produced a revision for the original version of the software.

5. Considered options for the old software that would provide ETX terminations on both RS-232 and GPIB communication.

The following items will be undertaken in the next month:

1. Fixing the old software to incorporate the ETX add-on discussed above.

2. Complete the FORTRAN coding and driver coding to incorporate the new thresholds and the new formatting.

Below I have a few notes about some decisions we made in the meeting on March 9, 1981 and some subsequent phone calls concerning the formatting:

1. For the printer: There will be only one mode of formatting (human mode) with four types of measurement data format which are selected by threshold and print control.

2. For RS-232: There will be two modes of formatting, human mode and machine mode. Human mode is the same as the printer. Machine mode is of the same form as the GPIB format for the old BERTS with "threshold" and "total seconds in test" added.

   In machine mode, there is only one type of measurement data format. That type is shown in the examples appended to this letter. In the machine mode, there will be a "switch" to either enable or disable the date, time, elapsed time, "to go" time, seconds in test and threshold information. There will also be query commands to allow you to see any threshold data. Basically, in the machine mode, the print control is essentially set to "partial" at all times. There will still be a switch to allow measurement transmissions only when requested. This applies to both machine and human mode.

3. For GPIB: Same as RS-232.

Notes:

1. Query commands for each threshold while in human mode will produce formats of that threshold setting with the selected print control.

2. Flag data can be only requested while in human mode and using RS-232. Flag data is passed in serial poll register in the GPIB.

3. Time request is the same for both machine and human form and will not differ between RS-232 and GPIB interfaces.

4. Carriage returns and line feeds will be inserted in GPIB as well as RS-232 interfaces for both modes.

5. GPIB messages will end with ETX. RS-232 messages will end with ETX, CR, LF.

6. Messages such as "Power up," "signal loss," etc. will come out o RS-232 and GPIB only in human form and will never come out in the machine form.

Please let me know if you believe these notes are not right or if you have corrections or additions.

As you know, I will be out on vacation the last week in March. My present goal is to be testing the new threshold measurements and formatting by the first of May. So if possible, I would like to have a new clock card and new front panel card incorporated with the unit by that time. I may possibly do some driver checkout of the front panel before that time.

After the first of May, I plan to be making the changes needed to incorporate the ASYNC/SYNC mode of operation and the Framed/Unframed PRBS measurements. My target date at this time is still around the first of June, but the changes to the old BERTS and the fact that it will be harder for me to make changes in the original software, since I am just learning the system, could be time delaying factors. Carol will be somewhat available for consultation, which could help quite a bit.

Please let me know when the hardware becomes complete and when any changes, additions, or questions come up concerning the software.

Sincerely yours,

JCH for Jeff Hopper

Mr. Jefferey C. Hopper
Research Engineer I
Computer Technology and
Applications Division
Electronics and Computer
Systems Laboratory

JCH/jm

Enclosure

Approved by: JTScott

John T. Scoville
Division Chief

FORMATTING APPENDIX

R09/11 14:33:58, 43824 (total seconds elapsed in test),
-6 (threshold)

L002:02:17

G001:34:56

BZAA, 213, 901E+9, 1.17E-3

PAAA, 009, 876, 2.20E-8

VAAA, 1.06E+4, 1.60E+8, 1.00E+9

Machine Mode

Can Suppress Top Three Lines

Human Mode is the type already decided on by Scientific Atlanta.

A 2448

Georgia Institute of Technology
ENGINEERING EXPERIMENT STATION
ATLANTA, GEORGIA 30332

April 7, 1981

Mr. Charles Trawick
Scientific Atlanta
Instrumentation Division
4311 Communications Drive
Norcross, Georgia 30093

Dear Mr. Trawick:

Subject: Status Report for March, 1981 for Project A-2448-002

The following items were accomplished in March:

1. Initial coding of the new formatting procedure has been started.

2. Items were added and changed to existing BERTS software for the Collins version.

3. I went on vacation to California and had a good time.

Changes and corrections for Collins are continuing to delay work on the "A" version and consequently are budgetary threats.

At this time, the outlook for April involves finishing up the Collins version of the "old" BERTS and then beginning serious work towards having the new formatting changes tested around the first week in May.

If you have any questions concerning this report or the project in general, please feel free to call me at 894-3553.

Sincerely,

Mr. Jeffrey C. Hopper
Research Engineer I
Computer Technology and
Applications Division
Electronics and Computer
Systems Laboratory

JCH/jm

cc: Robin Larson
Sonny Morris

Approved by:
John T. Scoville, Division Chief

Georgia Institute of Technology
ENGINEERING EXPERIMENT STATION
ATLANTA, GEORGIA 30332

May 19, 1981

Mr. Charles Trawick
Instrumentation Division
Scientific Atlanta
3845 Pleasantdale Road
Atlanta, Georgia 30340

Reference: Purchase Order #053467
Update for BERTS Receiver Software
Georgia Tech Project A-2448-002

Dear Mr. Trawick:

Subject: Letter Progress Report for April, 1981 for Reporting Period of April 1, 1981 to April 30, 1981

The following items have been accomplished in April:

1. The Collins version for the original BERTS was finished and is operating satisfactorily in the field.

2. All "A" version driver software for the new queing routines is finished. This completes all the new driver revisions that were needed for the new BERTS.

3. FORTRAN for the new queing and formatting routines have essentially been finished. The BURST format and the ability to express the percent error efficiency in a three decimal place accuracy have not yet been fully coded.

4. FORTRAN for the parsing of the new commands that were listed on the memorandum from Scientific Atlanta on April 15th has been completed. The "Q" command for the BURST information (05) is the only new command that has not been coded.

5. I am presently working on the decision processing involved with calculating the threshold error counts.

The following items should be accomplished during the month of May:

1. Finish coding the above-mentioned items.

2. Incorporate the framed/unframed mode.

3. Start the testing of the code.

Unfortunately, the present funds for the project will be exhausted around the first of June. It is my estimate that it will take an additional month and a half from the time I begin checkout to complete the job. This time involves checking out the changes outlined earlier and incorporating and checking out the SYNC/ASYNC operation. This means that assuming that I begin checkout on May 25th, I should expect to be finished around July 10th. Funds to cover through that period would amount to $6,000.

The reasons for this overrun are due primarily to the following facts:

1. Time spent on the Collins version of the original BERTS.

   Virtually 100% of Carol's charges to the project were involved in making Collins' changes. I also used over a month of my time in making changes. I concede that some of the GPIB changes that were made were items that would have needed attention in either version. Still, at the time of the project estimate, we were not aware that this effort was necessary.

   Carol's time along amounted to over $5,000.00.

2. Carol's resignation forced some minimal amount of time for me to become acquainted with the system.

I hope that these circumstances will not pose a problem. I have kept you aware in previous progress letters of the fact that time was being used up on changes to the original BERTS.

If you have any problems or questions, please let me know.

Sincerely,

Mr. Jefferey C. Hopper
Research Engineer I
Computer Technology and
Applications Division
Electronics and Computer
Systems Laboratory

JCH/jm

cc: Mr. J. L. Morris
Instrumentation Division

Approved:

John T. Scoville, Chief
Computer Technology and
Applications Division

Georgia Institute of Technology
ENGINEERING EXPERIMENT STATION
ATLANTA, GEORGIA 30332

June 10, 1981

Mr. Charles Trawick
Instrumentation Division
Scientific Atlanta
3845 Pleasantdale Road
Atlanta, Georgia 30340

Reference: Purchase Order #053467
Update for BERTS Receiver Software
Georgia Tech Project A-2448-002

Dear Mr. Trawick:

Subject: Letter Progress Report for May, 1981 for Reporting Period of May 1, 1981 to May 31, 1981

The following items have been accomplished in the month of May:

1. The new threshold information and the new front panel configuration were checked out.

2. The original software with the new changes was made operational in the newer hardware environment.

This essentially leaves the following items that still need to be done (in the order that I plan to do them):

1. Checkout new formatting changes.

2. Add burst information.

3. Framed/unframed mode.

4. Sync/Async mode.

5. Saving of previous measurements.

I believe the first three items will come along rather quickly. The last two items willprobably be tougher, mainly due to the fact that RAM and possibly ROM size constraints will start to curve into play.

As of now, I am still planning to work hard so I can finish up by the second week in July.

If you have any questions or comments, please call me.

Sincerely,

Mr. Jeffrey C. Hopper
Research Engineer I
Computer Technology and
Applications Division
Electronics and Computer
Systems Laboratory

JCH/jm

cc: Mr. J. L. Morris
Instrumentation Division

Approved by:

John T. Scoville, Chief
Computer Technology and
Applications Division

Computer Technology and Applications Division
Electronics and Computer Systems Laboratory
ENGINEERING EXPERIMENT STATION
Georgia Institute of Technology
Atlanta, Georgia 30332

FINAL REPORT

FIRMWARE LISTINGS FOR BERTS

TO:

Instrumentation Division
Scientific Atlanta
3845 Pleasantdale Road
Atlanta, Georgia 30340

FOR:

GIT Project A-2448-002

Scientific Atlanta Purchase Order #053467

CONTRACTING THROUGH:

Georgia Tech Research Institute
Georgia Institute of Technology
Atlanta, Georgia 30332

OUTLINE OF SECTIONS

I. FORTRAN Listings

- A. Common and Constants
- B. Utility Routines
- C. Front Panel Input Routine
- D. Remote Input Routine
- E. Once A Second Routine
- F. Million-Bit Routine
- G. Formatting Routines
- H. Measurement and Status Lights
- I. Main Line

II. Drivers Listings

- A. Clock Drivers
- B. Panel Drivers
- C. Remote Communications Drivers
- D. Error Counter Drivers
- E. Jump Table and Interrupt Service
- F. Floating Point Utilities

SECTION I. FORTRAN LISTINGS

A. COMMON AND CONSTANTS

```
UTI FORT//80 COMPILER  .3A1                    PAGE 0001

0000 C }}}}}}}}}}}}}}}}}}}}}}}}} FILE:  COMMON.FTN {{{{{{{{{{{{{{{{

0000 C* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
0000 C.....................................................................


0000 C              B I T  E R R O R  R A T E  T E S T  S E T

0000 C           R E   E I V E R   F O R T R A N  R O U T I N E S

0000 C                S C I E N T I F I C   A T L A N T A


0000 C* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
0000 C.....................................................................

0000 C               CAROL AGEE, CSTL, EES, GA TECH

0000 C                       OCTOBER  1979
0000 C               REVISED 10 DECEMBER 1979
0000 C               REVISED 13 DECEMBER 1979
0000 C               REVISED 14 JANUARY 1980
0000 C               REVISED 24 JANUARY 1980
0000 C               REVISED 10 MARCH 1980
0000 C               ADDED PERIPHERAL INTERFACES  MAY-JULY 1980
0000 C               CORRECTIONS DECEMBER 1980 - V1.1
0000 C               ADD ONE SEC RATE FEBRUARY 1981 - V1.2
0000 C               ADDED MORE COLLINS FEATURES
0000 C                  MARCH,APRIL,1981  JEFF HOPPER
0000 C               'A' VERSION OF BERTS -START MAY 1981
0000 C                                    JEFF HOPPER
0000 C

0000 C ***************************************************************
0000 C ...............................................................

0000 C                         E-PROM CONSTANTS !!!

0000 C CONSTANTS ARE AT TOP OF LAST PROM
0000         COMPILER(3)=7FFFH
0000
0000 C ***************************************************************
0000 C **********************    FLOATING POINT CONSTANTS

0000 C ******************  TABLE OF POWERS OF TEN FOR
0000 C                 FLOATING POINT  TO  BCD  CONVERSION
0000 C       ( 3 BYTES OF FLOATING POINT REPRESENTATION & 1 BYTE WITH
0000 C                 EXPONENT )

0000         INTEGER*1       FPTEN12 (4)     / 68H,80H,82H,0CH /
0000         INTEGER*1       FPTEN11 (4)     / 65H,43H,0BAH , 0BH/,
0000      &                  FPTEN10 (4)     / 62H,2,95H,0AH /
0000         INTEGER*1       FPTEN9  (4)     / 5EH,6AH,0EEH, 9 /,
0000      &                   FP@TEN8 (4)    /  5BH, 0BCH, 0BEH, 8/
0000         INTEGER*1       FP3TEN7 (4)     /  58H, 96H, 98H, 7/,
```

```
0000       &                    FPTEN6 (4)     /  54H, 24H, 0F4H, 6/,
0000       &                   FPTEN5  (4)     / 51H, 50H, 0C3H, 5/,
0000       &                   FPTEN4  (4)     / 4EH, 40H, 9CH, 4/,
0000       &                   FP1000 (4)      /  4AH, 00 , 0FAH, 3/,
0000       &                    FP100 (4)      /  47H, 00, 0C8H, 2/,
0000       &                    FP10 (4)       /  44H, 0, 0A0H, 1/,
0000       &                    FP1 (4)        /  41H, 0, 80H, 0/
0000        INTEGER*1     FPTENTH  (4)  / 3DH, 0CCH, 0CCH, -1H/,
0000       &                   FPTEN@NEG2      (4)     /3AH, 0D6H, 0A3H, -2H/,
0000       &                   FPTEN@NEG3      (4)     / 37H, 12H, 83H, -3H/,
0000       &                   FPTEN@NEG4      (4)     / 33H,0B6H,0D1H, -4H/,
0000       &                   FPTEN@NEG5      (4)     /30H, 0C4H, 0A7H, -5H/,
0000       &                   FPTEN@NEG6      (4)     /2DH, 36H, 86H, -6H/,
0000       &                   FPTEN@NEG7      (4)     / 29H, 0C0H, 0D6H, -7H/,
0000       &                   FPTEN@NEG8      (4)     / 26H, 0CCH, 0ABH, -8H/
0000       &                   FPTEN@NEG9      (4)     / 23H, 70H, 89H, -9H/,
0000       &                   FPTEN@NEG10     (4)     / 1FH,0E8H, 0DBH, -10/,
0000       &                   FPTEN@NEG11     (4)     / 1CH, 0ECH, 0AFH, -11/
0000

0000 C ************************* INTEGER*5  CONSTANTS

0000          INTEGER*1  DUMMY@BIT@ERRORS (5) /  20H,0A1H,7,0,0/,
0000       &                   DUMMY@PARITY@ERRORS (5)       /  69H,0,0,0,0/
0000 C        DUMMY@BIT@ERRORS=500,000   DUMMY@PARITY@ERRORS=105
0000 C        ( USED WHEN FRAME@LOSS AND/OR PRBS@LOSS OCCURS WITH SIGNAL)

0000          INTEGER*1       DUMMY@ALL@BIT@ERRORS (5)
0000       &                          / 60H, 11H, 0AAH, 2, 0 /,
0000       &                   DUMMY@ALL@PARITY@ERRORS (5)
0000       &                         /  0ACH, 24H,0,0,0 /
0000 C DUMMY ALL ERRORS WHEN LOSE SIGNAL DURING TEST, ALL ERRORS PER SECOND
0000 C DUMMY NUMBER OF MILLION BITS PER SECOND FOR SIGNAL LOSS DURING RUN

0000          INTEGER*1       I@TEN   (5)     /0AH,0,0,0,0/
0000          INTEGER*1       ONE     (5)     /1,0,0,0,0/
0000          INTEGER*1       I@HUNDRED (5)   /64H,0,0,0,0/
0000          INTEGER*1       ZEROS   (5)     /0,0,0,0,0/
0000 C  CONSTANTS THAT REPRESENT THE THRESHOLDS+1 THESE USED
0000 C   TO COMPARE SECOND TOTALS WITH.
0000          INTEGER*1       T7@CONS@1 (5)   /5,0,0,0,0/
0000          INTEGER*1       T6@CONS@1 (5)   /45,0,0,0,0/
0000          INTEGER*1       T3@CONS@1 (5)   /0E1H,0AEH,0,0,0/
0000          INTEGER*1       T7@CONS@2 (5)   /5,0,0,0,0/
0000          INTEGER*1       T6@CONS@2 (5)   /46,0,0,0,0/
0000          INTEGER*1       T3@CONS@2 (5)   /0C9H,0AFH,0,0,0/
0000

0000 C ***************************************************************

0000 C ERROR DISPLAY SPECIAL CONFIGURATIONS
0000          INTEGER*1       BLANKS          (4)     /0FH, 0FH, 0FH, 0FH /
0000          INTEGER*1 BLU@CONFIG            (4)     /0FH,0DH,0CH,8 /
0000
0000 C  TABLES TO OFFSET INTO ACCUMULATORS ACCORDING TO ERROR TYPE AND
0000 C  THRESHOLD. WAS DEVELOPED TO SAVE CODE AND TIME.
0000          INTEGER*1  ETABLE (4)   /80,40,0,0/
0000          INTEGER*1  TTABLE (8)   /15,0,0,5,0,0,0,10/
```

```
0000
0000 C ***************************************************************
0000 C MAXIMUM VALUE FOR SECS, MINS, HOURS
0000 C       ( USED BY BUMP@ELAPSED@TIME & COUNT@DOWN@TIMER ROUTINES )
0000         INTEGER*2 MAX@TIME@DISPLAY(4) /59H,59H,99H,99H/
0000
0000 C MAX VALUES FOR SECS, MINS, HRS FOR REAL TIME USED BY BUMP CLOCK
0000         INTEGER*2 MAX@CLOCK@DISPLAY(3) /59H,59H,23H/
0000


0000 C ***************************************************************
0000 C SPECIAL TIME@DISPLAY CONFIGURATIONS:  CLEAR AND ERROR

0000 C       INTEGER*1         CLEAR@TIME(4) /0,0,0,0/ - USE ZEROS(5)
0000         INTEGER*1        TIME@ERROR(4) /0ABH,0B5H,0FEH,0FFH/
0000

0000 C ***************************************************************
0000 C



0000 C  ***************************  FLOATING POINT ROUTINE OPERATORS
0000 C      INTEGER*1 GT  /'G'/,
0000 C     & GE  /'E'/,
0000 C     & LT  /'L'/,
0000         INTEGER*1  FIX  /'X'/,
0000       & FLOAT  /'F'/
0000

0000 C ***************************************************************
0000 C *************** TABLE TO CONVERT REMOTE COMMANDS TO HEX CODE
0000 C TABLE ENTRIES:  5 MSB'S ARE 5 LSB'S OF ALPHA,
0000 C                 3 LSB'S ARE 3 LSB'S OF NUMERIC-1 (0-7)
0000 C               ( WHEN 2 LETTERS OF ALPHA A UNIQUE ALPHA HAS
0000 C                       BEEN SUBSTITUTED:       TH - 5AH
0000 C                                               XT - 5BH
0000 C                                               XM - 5CH
0000 C                                               XL - 5DH
0000 C                                               XF - 5EH
0000 C                                               XH - 5FH
0000 C ENTRY POSITION IN TABLE GIVES HEX CODE FOR COMMAND,
0000 C       I.E. 'D1' IS COMBINED FOR TABLE ENTRY OF 20H, FIRST IN TABLE W/
0000 C               HEX CODE OF 1H

0000 C COMMAND THAT CORRESPONDS TO EACH TABLE ENTRY FOLLOWS
0000 C       1.D1, 2.D2, 3.D3,  10.C3
0000 C       11.H1, 12.H2, 14.T1, 15.T2, 16.T3, 17.T4, 18.T5, 19.C1, 20.C2
0000 C       21.K1, 22.K2, 23.K3, 24.K4, 25.K5, 26.K6, 27.K7, 28.K8, 29.XM1,
0000 C       31.XM2, 32.J1, 33.J2, 34.XT1, 35.XT2, 38.XL1,
0000 C        39.XL2,  0.Q2, 41.F1, 42.F2, 43.Q4
0000 C       44.Q1,  45.Q3, 46.Y1, 47.Y2, 48.S1, 49.S2
0000 C       50.M1, 51.M2, 52.M3, 53.M4, 54.M5, 55.M6, 56.XF1, 57.XF2
0000 C       58.XH1, 59.XH2, 60.TH1, 61.TH2, 62.TH3, 63.TH4, 64.05
0000 C               NOTE: 13,30 NOT USED
0000 C                     M1-M6 MOVED BECAUSE OF THE FACT THAT
```

```
0000 C                        STATE DEPENDS ON CLOSURE AND LAST STATE
0000 C

0000          INTEGER*1 COMMAND@TABLE (64)
0000       &  / 20H, 21H, 22H, 0, 0, 0, 0, 0, 1AH,
0000       &    40H, 41H, 00H, 0A0H,0A1H,0A2H,0A3H,0A4H,18H, 19H,
0000       &    58H, 59H, 5AH, 5BH, 5CH, 5DH, 5EH, 5FH, 0E0H, 00H,
0000       &    0E1H,50H, 51H,0D8H,0D9H, 0, 0,0E8H,0E9H, 89H,
0000       &    30H, 31H, 8BH, 88H, 8AH, 0C8H,0C9H, 98H,99H,
0000       &    68H, 69H, 6AH, 6BH, 6CH, 6DH, 0F0H, 0F1H,
0000       &    0F8H, 0F9H, 0D0H, 0D1H, 0D2H, 0D3H, 8CH /
0000

0000          INTEGER*1       J@HEX@SEQUENCE (8)
0000       &          / 8AH, 10H, 0AH, 2, 4, 1FH, 21H  /
0000 C    CLEAR,T3,RESET,PARITY,ERRSEC,XM2,J2

0000 C ***  MASKS TO BE USED BY BRANCH ON INPUT
0000          INTEGER*1       BIT@MASKS (8) / 1,2,4,8,10H,20H,40H,80H /
0000

0000 C                                        MESSAGE TEXT:
0000          INTEGER*1       MES@SL (17) /'SIGNAL LOSS',0DH,0AH,3,0DH,0AH,0/
0000          INTEGER*1       MES@FL (16) /'FRAME LOSS',0DH,0AH,03H,0DH,0AH,0/
0000          INTEGER*1       MES@PL (15) /'PRBS LOSS',0DH,0AH,3,0DH,0AH,0/
0000          INTEGER*1       MES@BS (17) /'BLUE SIGNAL',0DH,0AH,3,0DH,0AH,0/
0000          INTEGER*1       MES@GS (17) /'GOOD SIGNAL',0DH,0AH,3,0DH,0AH,0/
0000          INTEGER*1       MES@FK (16) /'FRAME LCKD',0DH,0AH,3,0DH,0AH,0/
0000          INTEGER*1       MES@PK (15) /'PRBS LCKD',0DH,0AH,3,0DH,0AH,0/
0000          INTEGER*1       MES@NL (13) /'NO LOSS',0DH,0AH,3,0DH,0AH,0/
0000          INTEGER*1       MES@NB (13) /'NO BLUE',0DH,0AH,3,0DH,0AH,0/
0000



0000          INTEGER*1       MES@L   (21) /'START ELAPSED TEST',0DH,0AH,0/
0000          INTEGER*1       MES@S   (20) /'START SINGLE TEST',0DH,0AH,0/
0000          INTEGER*1       MES@R   (20) /'START REPEAT TEST',0DH,0AH,0/
0000          INTEGER*1  MES@T (27) /'P=BIT P=PARITY V=BIPV',0DH,0AH,3,
0000       &                          0DH,0AH,0/
0000          INTEGER*1       MES@PW  (14)  /'POWER-UP',0DH,0AH,3,0DH,0AH,0/
0000          INTEGER*1       MES@CE  (12) /' ERROR',0DH,0AH,3,0DH,0AH,0/
0000          INTEGER*1       MES@OV  (6)  /' OVFL '/
0000          INTEGER*1       MES@UN  (6)  /' UNFL '/
0000 C      &                 MES@BR (9)  /0DH,0AH,' BER',0DH,0AH,0/,

0000 C    NEW HEADINGS FOR THE "A" VERSION
0000          INTEGER*1       MES@PD (16) /'POWER DOWN',0DH,0AH,3,0DH,0AH,0/,
0000       &                  ERMES(3)    /'ERR'/,
0000       &                  EFFMES(3)   /'%EF'/,
0000       &                  ESMES(3)    /'ES '/,
0000       &                  PESMES(3)   /'%ES'/,
0000       &                  BERMES(3)   /'BER'/
0000          INTEGER*1 T@@S@HEAD(20) /' ES',82H,'TOTERR AVGBER',0DH,0AH,0/
0000
0000          INTEGER*1       MES@PD@M (8) /'PWRD',0DH,0AH,3,0/
```

```
0000          INTEGER*1 T0@HEAD(17) /86H,'TOTAL',83H,'BER>E-3',0DH,0AH,0/
0000          INTEGER*1 T6@HEAD(17) /86H,'TOTAL',83H,'BER>E-6',0DH,0AH,0/
0000          INTEGER*1 T7@HEAD(17) /86H,'TOTAL',83H,'BER>E-7',0DH,0AH,0/
0000          INTEGER*1 T6@T7@HEAD(19) /86H,'BER>E-6 BER>E-7',0DH,0AH,0/
0000          INTEGER*1 BURST@HEAD(39) /'ERROR BURSTS, MSEC',0DH,0AH,
0000       &                            ' 1-10',83H,'11-99',82H,'100+',
0000       &                           0DH,0AH,0/
0000          INTEGER*1 T0@HEAD@MAC(4) /'T=0',0/
0000          INTEGER*1 T3@HEAD@MAC(4) /'T-3',0/
0000          INTEGER*1 T6@HEAD@MAC(4) /'T-6',0/
0000          INTEGER*1 T7@HEAD@MAC(4) /'T-7',0/
0000
0000 C        INTEGER*1        BLANKS@PER@MODE (4) / 0,7,0,14 /
0000          INTEGER*1        ERROR@DESIGNATOR (4) / 'B', 'P', 0, 'V' /
0000

0000          COMPILER         (3)=43FFH
0000

0000 C ..............................................................
0000 C    *************   SYSTEM  COMMON  VARIABLES   **************
0000 C ..............................................................

0000 C WARNING:  THESE COMMON VARIABLES ARE USE BY DRIVER ROUTINES
0000 C       DON'T CHANGE THEM HERE WITHOUT CHANGING EQUATES IN THE
0000 C       DRIVERS!!!!



0000 C    *************   VARIABLES FOR INTERFACE WITH FRONT PANEL,
0000 C                     CLOCK AND COUNTER ASSEMBLY I/O DRIVERS


0000 C       FRONT PANEL OUTPUT STATE:

0000 C       ( CONTENTS OF THESE VARIABLES SHOULD ONLY BE CHANGED W/
0000 C               INTERRUPTS DI--DURING INTERRUPT SERVICE THESE
0000 C               V RIABLES USED TO UPDATE FRONT PANEL)

0000          INTEGER*1        TIME@DISPLAY(4),
0000       &                   ERROR@DISPLAY(4),
0000       &                   ERROR@MODE,
0000       &                   TIME@MODE,
0000       &                   INPUT@TYPE,
0000       &                   ERROR@TYPE,
0000       &                   RUN@STOP@LIGHT,
0000       &                   TRAFFIC@LIGHTS,
0000       &                   THRESHOLD,
0000       &                   PRINT@CONTROL,
0000       &                   HORN@ENABLE,
0000       &                   HORN
0000
0000 C      FRONT PANEL INPUT CODE
0000        INTEGER 1 DECODED@INPUT
0000
0000 C       FLAGS FOR INTERPRETING COUNTER CONTENTS
0000          INTEGER*1         SIGNAL@LOSS,
```

```
0000      &                   FRAME@LOSS,
0000      &                   PRBS@LOSS
0000
0000 C       COUNTERS READ INTO HERE:
0000       INTEGER*2 BP@CTR, PAR@CTR, BER@CTR
0000
0000 C       FLAG FOR PRESERVING THE TIME@DISPLAY
0000         INTEGER*1        SAVE@TIME@DISPLAY
0000 C       FLAG FOR UPDATING THE FRONT PANEL DISPLAY
0000         INTEGER*1        UPDATE@FRONT@PANEL
0000 C       STATUS FLAG FOR BLUE SIGNAL
0000         INTEGER*1        BLU@CONDITION
0000
0000         INTEGER*1        RUN@STOP
0000 C       RUN@STOP INDICATES ONGOING MEASUREMENT OR NOT,
0000 C              SEE RUN@STOP@LIGHT IN COMMON FOR RUN/STOP BIT ASSIGNMENT
0000         INTEGER*1         RUN@ON@TICK
0000

0000 C ***********************************************************
0000 C ***   DEFINITIONS FOR COMMON FOR PERIPHERAL INTERFACES : ***

0000 C FLAG FOR REMOTE CONTROL:
0000         INTEGER*1        GPIB@THERE
0000

0000 C       NEW INPUT CHARACTER
0000         INTEGER*1        RXCHAR
0000
0000 C       SOFTWARE CLOCK REGISTERS
0000         INTEGER*2        CLOCK (4)
0000         INTEGER*2        DATE (4)
0000
0000 C       IEEE-488 BUS VARIABLES
0000         INTEGER*1        BUS@REMOTE@ENABLE
0000         INTEGER*1        PARALLEL@POLL@MASK
0000 C   RING BUFFER PARAMETER BLOCK FOR AUTO

0000         INTEGER*1        BUFFER@TOTAL@A
0000         INTEGER*2        RBUFFER@STRT@A
0000         INTEGER*1        NO@POSITIONS@A
0000         INTEGER 1        SPC@COUNTER@A
0000         INTEGER*2        RBUFFER@IN@A
0000         INTEGER*2        RBUFFER@OFF@A
0000         INTEGER*2        RBUFFER@OUT@A
0000         INTEGER*2        ARNG@BUFFER(10)
0000 C       RS-232 SPECIAL VARIABLES:
0000         INTEGER*1        RS@232@THERE
0000         INTEGER*1        RS@LOCAL@LOCKOUT
0000         INTEGER*1        ASYNC
0000         INTEGER*1        UNFRAMED
0000

0000 C       PRINTER SPECIAL VARIABLES:
0000         INTEGER*1        PRINTER@THERE
0000
0000 C       HOLD VALUE FROM CONVERSION(FP TO BCD) HERE:
0000         INTEGER*1        BCDNUM (4)
```

```
0000
0000 C RING BUFFER PARAMETER BLOCK FOR PRINTER

0000          INTEGER*1       BUFFER@TOTAL@P
0000          INTEGER*2       RBUFFER@STRT@P
0000          INTEGER*1       NO@POSITIONS@P
0000          INTEGER*1       SPC@COUNTER@P
0000          INTEGER*2       RBUFFER@IN@P
0000          INTEGER*2       RBUFFER@OFF@P
0000          INTEGER*2       RBUFFER@OUT@P
0000          INTEGER*2       PRNG@BUFFER(10)
0000 C        TEMPORARIES FOR ASSEMBLER ROUTINES
0000          INTEGER*1 DISPLA@ASM@TEMP(8)
0000          INTEGER*1 FPBCD@ASM@TEMP(7)
0000
0000          INTEGER*1       NV@RAM@TEST1,NV@RAM@TEST2
0000          INTEGER*1       FIRST@FLAG
0000
0000 C        FOR RS-232 DELAY ON CR
0000          INTEGER*1       RS@DELAY@FLAG
0000          INTEGER*1       RS@DELAY@CTR
0000          INTEGER*1       TEST@CTR
0000          INTEGER*1       TRANSMIT@ENABLE
0000          INTEGER*1       BURST@LENGTH
0000          INTEGER*1       BURST@COUNT
0000          INTEGER*1       LAST@BURST@COUNT
0000



0000 C ***********************************************************
0000 C        ***     DEFINITIONS  FOR  COMMON  VARIABLES :  ***


0000 C DISPLAY CODES :  0 - 9,  BCD DIGITS  0 - 9
0000 C                          0EH,    UPPERCASE E
0000 C                          0BH,    LOWERCASE r
0000 C                          0AH,    LOWERCASE o
0000 C                          0CH     LOWERCASE B
0000 C                          0DH     LOWERCASE U
0000 C                          0FH,    BLANK
0000 C        ( FOR ERROR AND TIME DISPLAYS )

0000 C

0000 C        *** ERROR@MODE
0000 C                BIT 0   (1H)    ERROR SECONDS
0000 C                BIT 1   (2H)    TOTAL ERRORS
0000 C                BIT 2   (4H)    TOTAL ERRORS/TOTAL BITS
0000 C                BIT 3   (8H)    ERRORS 10**7 BITS
0000 C                BIT 4   (10H)   ERRORS 10**8 BITS
0000 C                BIT 5   (20H)   % ERROR SECONDS


0000 C        *** ERROR TYPE
```

```
0000 C                BIT 0   (1H)    BIT ERRORS
0000 C                BIT 1   (2H)    PARITY ERRORS
0000 C                BIT 2   (4H)    BIPOLAR VIOLATIONS


0000 C        *** INPUT TYPE
0000 C                BIT 0   (1H)    LOW
0000 C                B T 1   (2H)    DSX-3
0000 C                B T 2   (4H)    HIGH


0000 C            TIME@MODE
0000 C                        (0H)    MONTH-DAY
0000 C                BIT 0   (1H)    TIME OF DAY
0000 C                BIT 1   (2H)    ELAPSED
0000 C                BIT 2   }4      SINGLE TIMED
0000 C                BIT 3   (8)     REPEAT TIMED
0000 C                BIT 4   (10H)   DESELECT TIME@MODE
0000 C ( BITS 0 THRU 3 ONLY USED FOR FRONT PANEL UPDATE!!)

0000 C        *** THRESHOLD
0000 C                BIT 0   (1H)    THRESHOLD 0
0000 C                BIT 1   (2H)    THRESHOLD E-7
0000 C                BIT 2   (4H)    THRESHOLD E-6
0000 C                BIT 3   (8H)    THRESHOLD E-3

0000 C        *** PRINT@CONTROL
0000 C                BIT 0   (1H)    PRINT CONTROL ALL
0000 C                B T 1   (2H)    PRINT CONTROL PARTIAL

0000 C        *** HORN@ENABLE
0000 C                BIT 0     1=HORN ENABLED
0000 C                          0=HORN DISABLED

0000 C        *** HORN
0000 C                BIT 0     1=HORN ON
0000 C                          0=HORN OFF
0000 C        *** TRAFFIC LIGHTS -- SEE SET@TRAFFIC@LIGHTS ROUTINE

0000 C        *** RUN@STOP@LIGHT :   BIT 1    (1H)    STOP
0000 C                               BIT 2    (2H)    RUN

0000 C        *** ERROR@DISPLAY :    4 BYTES, UNPACKED BCD
0000 C                ERROR@DISPLAY(1) - EXPONENT
0000 C                ERROR@DISPLAY(2) - HUNDREDTHS
0000 C                ERROR@DISPLAY(3) - TENTHS
0000 C                ERROR@DISPLAY(4) - UNITS
0000 C                ( DECIMAL POINT OR MINUS SIGN MASK IS BIT 7)


0000 C        *** TIME@DISPLAY :     4 BYTES, PACKED BCD
0000 C                TIME@DISPLAY(1) - SECONDS
0000 C                TIME@DISPLAY(2) - MINUTES
0000 C                T ME@DISPLAY(3) - HOURS
0000 C                TIME@DISPLAY(4) - HUNDRED HOURS
0000 C        ( 7 DIGITS IN DISPLAY -- TOP NIBBLE IS UNUSED )
```

```
0000          COMPILER(2)= 3H
0003
0003 C WARNING:  THE FLTPT@PACK PARAMETERS ARE USED BY CALLFP MODULE
0003 C        DON'T CHANGE LOCATION HERE WITHOUT CORRESPONDING CHANGE
0003 C        IN CALLFP

0003 C JUMP TO FP LOCATED AT 3H
0003 C ************************************************************

0003        SUBROUTINE FLTPT@PACK(VALUE1,OPERATOR,VALUE2,RESULT)
0003
0003 C ************************************************************
0003          INTEGER*1       VALUE1(3),VALUE2(3),RESULT(3)
0003          INTEGER*1       OPERATOR
0003
0003 C THIS ROUTINE CALLS FLOATING POINT  ROUTINE INDICATED BY THE
0003 C        OPERATOR:  I.E.  VALUE1,VALUE2,'*',RESULT
0003 C CODE FOR THIS R UTINE IS IN ASSEMBLER AND IS FOUND IN THE
0003 C        CALLFP.ASM FILE
0003 C THIS ROUTINE LOOKS AS IF ITS LOCATED AT 3H FOR FORT
0003 C        ACTUALLY A JUMP TO THE ROUTINE IS AT 3H --
0003 C        THE JUMP IS INSERTED BY AN ASEG IN THE ASSEMBLER ROUTINE

0003          END


PROGRAM  STORAGE: 0000

VARIABLE STORAGE: 0016

SYMBOL TABLE
435B RESULT
435F VALUE2
4363 OPERATOR
4367 VALUE1

0003
```

```
0003 C .....................................................


0003 C ********************  EQUIVALENCES *************************


0003          INTEGER*2       FLOAT@TEMP
0003
0003          INTEGER*1       BEFORE@FLOAT@TEMP       (1)
0003 C INTEGER*1 ARRAY NEEDED AT NEXT LOWER ADDRESS FOR CALL TO FLOAT
0003 C        NORMAL CALL TO FLT.PT.PACK. REQUIRES I*1(3), BUT WANT
0003 C        TO INSERT THE 16 BIT NUMBER TO FLOAT IN LAST TWO BYTES



0003 C ********************    INTEGER VARIABLES FOR ERROR COUNTS

0003        INTEGER*2 LST@BP@CTR, LST@PAR@CTR, LST@BER@CTR, CTR@TEMP
0003


0003 C ***************** **  INTEGER*5 INTERNAL ACCUMUALTORS
0003          INTEGER*1       BER@SEC@TEMP (5)
0003          INTEGER*1       BER@SYNC@CTR
0003          INTEGER*1       PAR@SEC@TEMP (5)
0003          INTEGER*1       PAR@SYNC@CTR
0003          INTEGER*1       BP@SEC@TEMP  (5)
0003          INTEGER*1       BP@SYNC@CTR
0003          INTEGER*1       BURST@1@10 (5)
0003          INTEGER*1       BURST@11@99  (5)
0003          INTEGER*1       BURST@GT@100 (5)
0003
0003          INTEGER*1       BER@SEC@FLAG,
0003       &                  BER@SEC@FLAG@T3,
0003       &                  BER@SEC@FLAG@T6,
0003       &                  BER@SEC@FLAG@T7,
0003       &                  BER@INT@A,
0003       &                  BER@INT@P,
0003       &                  BIT@ERRORS                          (5),
0003       &                  THIS@SEC@BIT@ERRORS                 (5),
0003       &                  THIS@INT@BIT@ERRORS@P               (5),
0003       &                  THIS@INT@BIT@ERRORS@A               (5),
0003       &                  PAR@SEC@FLAG,
0003       &                  PAR@SEC@FLAG@T3,
0003       &                  PAR@SEC@FLAG@T6,
0003       &                  PAR@SEC@FLAG@T7,
0003       &                  PAR@INT@A,
0003       &                  PAR@INT@P,
```

```
0003       &                    PARITY@ERRORS                    (5),
0003       &                    THIS@SEC@PARITY@ERRORS           (5),
0003       &                    THIS@INT@PARITY@ERRORS@P                 (5),
0003       &                    THIS@INT@PARITY@ERRORS@A                 (5),
0003       &                    BP@SEC@FLAG,
0003       &                    BP@SEC@FLAG@T3,
0003       &                    BP@SEC@FLAG@T6,
0003       &                    BP@SEC@FLAG@T7,
0003       &                    BP@INT@A,
0003       &                    BP@INT@P,
0003       &                    BIPOLAR@VIOLATIONS               (5),
0003       &                    THIS@SEC@BIPOLAR@VIOLATIONS      (5),
0003       &                    THIS@INT@BIPOLAR@VIOLATIONS@P    (5),
0003       &                    THIS@INT@BIPOLAR@VIOLATIONS@A    (5),
0003       &                    TOTAL@BIT@ERRORS                 (5),
0003       &                    TOTAL@BIT@ERRORS@T3              (5),
0003       &                    TOTAL@BIT@ERRORS@T6              (5),
0003       &                    TOTAL@BIT@ERRORS@T7              (5)
0003          INTEGER*1         BIT@ERROR@SECS                   (5),
0003       &                    BIT@ERROR@SECS@T3                (5),
0003       &                    BIT@ERROR@SECS@T6                (5),
0003       &                    BIT@ERROR@SECS@T7                (5),
0003       &                    TOTAL@PARITY@ERRORS              (5),
0003       &                    TOTAL@PARITY@ERRORS@T3           (5),
0003       &                    TOTAL@PARITY@ERRORS@T6           (5),
0003       &                    TOTAL@PARITY@ERRORS@T7           (5)
0003          INTEGER*1         PARITY@ERROR@SECS                (5),
0003       &                    PARITY@ERROR@SECS@T3             (5),
0003       &                    PARITY@ERROR@SECS@T6             (5),
0003       &                    PARITY@ERROR@SECS@T7             (5),
0003       &                    TOTAL@BIPOLAR@VIOLATIONS         (5),
0003       &                    TOTAL@BIPOLAR@VIOLATIONS@T3      (5),
0003       &                    TOTAL@BIPOLAR@VIOLATIONS@T6      (5),
0003       &                    TOTAL@BIPOLAR@VIOLATIONS@T7      (5)
0003          INTEGER 1         BIPOLAR@VIOLATION@SECS           )5~,
0003       &                    BIPOLAR@VIOLATION@SECS@T3        (5),
0003       &                    BIPOLAR@VIOLATION@SECS@T6        (5),
0003       &                    BIPOLAR@VIOLATION@SECS@T7        (5)
0003
0003 C ACCUMULATOR FOR TOTAL BITS
0003          INTEGER*1         TOTAL@BITS                       (5)
0003
0003 C ACCUMULATOR FOR TOTAL SECONDS IN TEST
0003          INTEGER*1         TOTAL@SECONDS                    (5)
0003 C ACCUMULATOR FOR TOTAL ERRORS IN SECOND
0003          INTEGER*1         TOTAL@THIS@SECOND
0003          INTEGER*1         TOTAL@THIS@SEC@ACC               (5)
0003 C ACCUMULATOR FOR 10**7 AND 10**8 MEASUREMENT
0003          INTEGER*1         ERRORS@TEN@7@8                   (5)
0003
0003 C
0003 C TEMPORARY FLOATING POINT ACCUMULATOR FOR COMPUTATION PURPOSES
0003          INTEGER*1         RESULT@DENOM                     (5),
0003       &                    RESULT                           (5)
0003          INTEGER*1         RDTMP                            (5),
0003       &                    RTMP                             (7)
0003
0003 C
```

```
0003          INTEGER*1       BURST@LETTER
0003
0003 C ***********************   FLAGS
0003          INTEGER*1        RESET@TEST@ON@TICK,
0003        &                  RESET@TIMER@NEXT@RUN,
0003        &                  SIGNAL@OBTAINED,
0003        &                  FRAME@OBTAINED,
0003        &                  BLINK@FRAME@LOSS,
0003        &                  BLINK@PRBS@LOSS,
0003        &                  BLINK@SIGNAL@LOSS,
0003        &                  RESULT@READY,
0003        &                  RESULT@TASK,
0003        &                  PRBS@OBTAINED
0003          INTEGER 1       CLOCK@TASK
0003
0003 C ***** DEFINITIONS OF THE FLAGS:
0003 C        AUTO@CONTROL IS SET TO FALSE ON POWER-UP.
0003 C          IF REMOTE CONTROL BECOMES POSSIBLE W/
0003 C          ADDITION OF SOME INTERFACE, THEN
0003 C          AUTO@CONTROL WOULD BE SET TO TRUE
0003 C          WHEN NOT IN MANUAL OPERATION.
0003 C          USED TO SET TRAFFIC@LIGHTS FOR FRONT
0003 C          PANEL
0003 C        CLOCK@TASK IS SET IN INTERRUPT ROUTINE
0003 C          TO REQUEST READ OR SET OF TIME OF DAY
0003 C          CLOCK TO BE EXECUTED BY MAINLINE.
0003 C          SEE CLOCK@TASKS LISTED IN E-PROM CONSTANTS
0003 C          SECTION BELOW.
0003 C        RESULT@RE DY IS SET IN INTERRUPT ROUTINE
0003 C          TO REQUEST CONVERSION OF AN ACCUMULATOR
0003 C          INTO THE ERROR@DISPLAY WHICH MUST TAKE
0003 C          PLACE IN THE MAINLINE
0003 C        RESULT@FL G IS USED BY THE MAINLINE
0003 C          TO SAVE THE VALUE OF REULT@READY
0003 C        RUN@ON@TICK WHEN SET TRUE, CAUSES
0003 C          A TEST TO BEGIN ON THE NEXT ONE SECOND
0003 C          TICK INTERRUPT.  THIS FLAG SET TRUE
0003 C          BY BUTTON PUSH OR TIME DOWN OF A REPEAT
0003 C          TEST.
0003 C        RESET@TEST@ON@TICK CAUSES ALL ACCUMS TO
0003 C          GET CLEARED WHEN RUN@ON@TICK AND A
0003 C          ONE SECOND INTERRUPT.
0003 C        CLEAR@ACCUMS@NEXT@RUN IS SET ON POWER-UP,
0003 C          ON BEGIN OR TIME DOWN OF A TIMED TEST.
0003 C          FLAG CA SES RESET@TEST@ON@TICK TO BE SET
0003 C          WHEN RU  BUTTON PUSHED.
0003 C        OBTAINED FLAGS:
0003 C        SIGNAL@OBTAINED, FRAME@OBTAINED, & PRBS@OBTAINED
0003 C          ARE SET TO FALSE ON POWER-UP,
0003 C          SET TO TRUE WHENEVER SIGNAL@LOSS, FRAME@LOSS
0003 C          & PRBS@LOSS BECOME FALSE.  PRBS@OBTAINED WILL
0003 C          REMAIN FALSE DURING LIVE TRAFFIC RECEPTION
0003 C        BLINK FLAGS:
0003 C        BLINK@SIGNAL@LOSS, BLINK@FRAME@LOSS, {
0003 C          BLINK@PRBS@LOSS ARE SET TO FALSE ON POWER-UP
0003 C          AND ON RESET.  WHENEVER THE CORRESPONDING
```

```
0003 C         OBTAINED FLAG IS SET TRUE AND THEN THE
0003 C         CORRESPONDING LOSS FLAG BECOMES FALSE
0003 C         WHILE A TEST IS IN PROGRESS,
0003 C         THE BLINK FLAG IS SET TRUE AND THE
0003 C       DISCRETE LED WILL BLINK.



0003 C ***********************      TIME REGISTERS
0003       INTEGER*2          TIMER                (4),
0003      &                   TIMER@START@TIME           (4),
0003      &                   ELAPSED@TIME     (4),
0003      &                   TIME@TEMP        (4)
0003
0003          INTEGER*1       ASCII@TIMER (7)
0003



0003 C ***************************  COUNTERS FOR TEN**7 OR TEN**8 SAMPLING
0003 C  10**7 OR 10**8 SAMPLE CTR
0003        INTEGER*1 TEN@7@8@CTR
0003


0003 C ****************  SECOND COUNTER FOR TEMPORARY MO-DAY DISPLAY
0003          INTEGER*1       MO@DISPLAY@CTR
0003          INTEGER*1       LAST@TIME@MODE
0003


0003 C ****************  COUNTER FOR BLINKING SIGNAL LOSS LED
0003          INTEGER*1       BLINK@CTR
0003          INTEGER*1       BLINKER
0003          INTEGER*1       AT@LEAST@ONE@TO@BLINK
0003

0003 C ****************  COUNTER TO BLINK GATING LIGHT WHEN
0003 C                        ERROR@DISPLAY UPDATED
0003          INTEGER 1       GATING@WINK@CTR
0003



0003 C                                MEASUREMENT TEST TYPE:
0003 C                              BIT 2  ELASPED          2
0003 C                              BIT 3  SINGLE-TIMED     4
0003 C                              BIT 4   REPEAT-TIMED    8
0003 C                         (SAME BIT ASSIGNMENT AS TIME@MODE)
0003        INTEGER*1 TEST@TYPE
0003 C                         (TEST@TYPE IS NEVER TIME OF DAY)


0003 C
```

```
0003 C ***************************************************************
0003 C ***   VARIABLES ADDED FOR PERIPHERAL INTERFACE FOLLOW:        ***

0003 C        FLAGS AND MASKS TO CONTROL TRANSMISSON:
0003          INTEGER*1       TX@MODE,
0003       &                  TX@TIME
0003          INTEGER*1       MACHINE@FORMAT
0003
0003 C        FLAGS AND TEMPORARIES FOR RECEIVING AND DECODING PERIPH INPUT:
0003          INTEGER*1       RX@NEED@ALPHA1,
0003       &                  ALPHA1
0003

0003 C        TEMPORARY ARRAYS FOR HOLDING ACCUMULATED COUNTS FOR CONVERSION
0003 C                AND FOR HOLDING THE RESULTING ASCII STRING FOR FORMATTING
0003          INTEGER*1       BLOCK@TEMPS@FULL@A,
0003       &                  BLOCK@TEMPS@FULL@P,
0003       &                  BLOCK@MASK@A,
0003       &                  BLOCK@MASK@P
0003          INTEGER*1       B@MEAS@1A  (7),
0003       &                  B@MEAS@1A@T3 (7),
0003       &                  B@MEAS@1A@T6 (7),
0003       &                  B@MEAS@1A@T7 (7),
0003       &                  B@MEAS@2A (7),
0003       &                  B@MEAS@2A@T3 (7),
0003       {                  B@MEAS@2A@T6 }7~,
0003       &                  B@MEAS@2A@T7 (7),
0003       {                  P@MEAS@1A }7~,
0003       &                  P@MEAS@1A@T3 (7),
0003       {                  P@MEAS@1A@T6 }7~,
0003       &                  P@MEAS@1A@T7 (7),
0003       {                  P@MEAS@2A }7~,
0003       &                  P@MEAS@2A@T3 (7),
0003       {                  P@MEAS@2A@T6 }7~,
0003       &                  P@MEAS@2A@T7 (7),
0003       &                  V@MEAS@1A (7),
0003       &                  V@MEAS@1A@T3 (7),
0003       &                  V@MEAS@1A@T6 (7),
0003       &                  V@MEAS@1A@T7 (7),
0003       &                  V@MEAS@2A (7),
0003       &                  V@MEAS@2A@T3 (7),
0003       &                  V@MEAS@2A@T6 (7),
0003       &                  V@MEAS@2A@T7 (7),
0003       &                  B@MEAS@3A (7),
0003       &                  B@MEAS@3A@T3 (7),
0003       &                  B@MEAS@3A@T6 (7),
0003       &                  B@MEAS@3A@T7 (7),
0003       &                  B@MEAS@4A (7),
0003       &                  B@MEAS@4A@T3 (7),
0003       &                  B@MEAS@4A@T6 (7),
0003       &                  B@MEAS@4A@T7 (7),
0003       &                  P@MEAS@3A (7),
0003       &                  P@MEAS@3A@T3 (7),
0003       &                  P@MEAS@3A@T6 (7),
0003       &                  P@MEAS@3A@T7 (7),
```

```
0003      &                   P@%EAS@4A (7),
0003      &                   P@%EAS@4A@T3 (7),
0003      &                   P@%EAS@4A@T6 (7),
0003      &                   P@%EAS@4A@T7 (7),
0003      &                   V@%EAS@3A (7),
0003      &                   V@%EAS@3A@T3 (7),
0003      &                   V@%EAS@3A@T6 (7),
0003      &                   V@%EAS@3A@T7 (7),
0003      {                   V@%EAS@4A }7~,
0003      &                   V@%EAS@4A@T3 (7),
0003      {                   V@%EAS@4A@T6 }7~,
0003      &                   V@%EAS@4A@T7 (7),
0003      &                   B@%EAS@1P (7),
0003      &                   B@%EAS@1P@T3 (7),
0003      {                   B@%EAS@1P@T6 }7~,
0003      &                   B@%EAS@1P@T7 (7),
0003      &                   B@%EAS@2P (7),
0003      &                   B@%EAS@2P@T3 (7),
0003      &                   B@%EAS@2P@T6 (7),
0003      &                   B@%EAS@2P@T7 (7),
0003      &                   P@%EAS@1P (7),
0003      &                   P@%EAS@1P@T3 (7),
0003      &                   P@%EAS@1P@T6 (7),
0003      &                   P@%EAS@1P@T7 (7),
0003      &                   P@%EAS@2P (7),
0003      &                   P@%EAS@2P@T3 (7),
0003      &                   P@%EAS@2P@T6 (7),
0003      &                   P@%EAS@2P@T7 (7),
0003      &                   V@%EAS@1P (7),
0003      &                   V@%EAS@1P@T3 (7),
0003      &                   V@%EAS@1P@T6 (7),
0003      &                   V@%EAS@1P@T7 (7),
0003      &                   V@%EAS@2P (7),
0003      &                   V@%EAS@2P@T3 (7),
0003      &                   V@%EAS@2P@T6 (7),
0003      &                   V@%EAS@2P@T7 (7),
0003      &                   B@%EAS@3P (7),
0003      &                   B@%EAS@3P@T3 (7),
0003      &                   B@%EAS@3P@T6 (7),
0003      &                   B@%EAS@3P@T7 (7),
0003      &                   B@%EAS@4P (7),
0003      &                   B@%EAS@4P@T3 (7),
0003      &                   B@%EAS@4P@T6 (7),
0003      &                   B@%EAS@4P@T7 (7),
0003      {                   P@%EAS@3P }7~,
0003      &                   P@%EAS@3P@T3 (7),
0003      {                   P@%EAS@3P@T6 }7~,
0003      &                   P@%EAS@3P@T7 (7),
0003      {                   P@%EAS@4P }7~,
0003      &                   P@%EAS@4P@T3 (7),
0003      {                   P@%EAS@4P@T6 }7~,
0003      &                   P@%EAS@4P@T7 (7),
0003      {                   V@%EAS@3P }7~,
0003      &                   V@%EAS@3P@T3 (7),
0003      &                   V@%EAS@3P@T6 (7),
0003      &                   V@%EAS@3P@T7 (7),
0003      &                   V@%EAS@4P (7),
0003      &                   V@%EAS@4P@T3 (7),
```

```
0003      &                    V@MEAS@4P@T6 (7),
0003      &                    V@MEAS@4P@T7 (7),
0003      &                    DENOM@MEAS@A (5),
0003      &                    DENOM@MEAS@SEC@A (5)
0003         INTEGER*1         DENOM@MEAS@P (5),
0003      &                    DENOM@MEAS@SEC@P (5)
0003         INTEGER*1         BST1@TMP@A (7)
0003         INTEGER*1         BST2@TMP@A (7)
0003         INTEGER*1         BST3@TMP@A (7)
0003         INTEGER*1         BST1@TMP@P (7)
0003         INTEGER*1         BST2@TMP@P (7)
0003         INTEGER*1         BST3@TMP@P (7)
0003
0003         INTEGER*1         P@COUNTER,A@COUNTER
0003         INTEGER*1         T@TEMP@A,T@TEMP@P
0003 C NOTE: NO DIFFERENTIAL RATE FOR BPV !!  ALWAYS SAME !!

0003 C ** PASS MESSAGE NUMBER FROM INTERRUPT ROUTINES TO MAIN FOR PROCESSING
0003         INTEGER*1         MESSAGE@NUMBER
0003
0003         INTEGER*1         FPERROR@TO@PERIPH,
0003      &                    NEW@ERROR@MASK
0003


0003 C       FLAGS TO COMPARE PREVIOUS STATUS TO PRESENT
0003         INTEGER*1         LAST@SIGNAL@LOSS,
0003      &                    LAST@FRAME@LOSS,
0003      &                    LAST@PRBS@LOSS,
0003      &                    LAST@BLU@COND
0003
0003         INTEGER*1         BURST@FLAG
0003         INTEGER*1         LETTER
0003

0003 C       LOGGING TO PRINTER FLAGS
0003         INTEGER*1         LOG@MADE@IN@LAST@HOUR
0003         INTEGER*1         NUMBER@SEQUENTIAL@LOGS
0003         INTEGER*2         COUNTER@15@MIN
0003
0003 C       FORMATTING FLAGS
0003         INTEGER*1         FORMAT@INDEX
0003         INTEGER*1         FORMAT@SEPARATOR
0003         INTEGER*1         SEPARATOR1,
0003      &                    SEPARATOR2
0003         INTEGER*1         SCALE
0003         INTEGER*1         SELECT,MASK
0003         INTEGER*1         BUS@FORMAT
0003         INTEGER*1         FAULT@FLAG
0003         INTEGER*1         TEMP1
0003         INTEGER*1         LOG@COUNTER
0003         INTEGER*2         BUFFER@INSERT@PTR
0003
0003 C       PRINTER BUFFER
0003         INTEGER*1         PBUF1(80)
0003         INTEGER*1         REAL@TIME@P(28)
0003
0003 C       AUTO BUFFER
```

```
0003          INTEGER*1       ABUF1(100)
0003          INTEGER*1       REAL@TIME@A(100)
0003
0003 C        REQUEST MEAS DATA FLAG:  0 - OFF, 1 - FPERROR, 2 - BLOCK TOTALS
0003          INTEGER*1       Q@COMMAND@A
0003          INTEGER*1       Q@COMMAND@P
0003          INTEGER*1       INT@CTR
0003

0003          INTEGER*1       LOSS@REPORT@CTR
0003          INTEGER 1       SFLAG
0003
0003          INTEGER*1       UNLOGGED@ERRORS@TO@PRINT
0003          INTEGER*1       UNLOGGED@ERRORS@TO@AUTO
0003
```

B. UTILITY ROUTINES

```
0003 C }}}}}}}}}}}}}}}}}}}}}}}}}}}} UTIL.FTN {{{{{{{{{{{{{{{{{{{{{{{{
0003 C ** JUMP TABLE OCCUPIES 0 - 3AH
0003         COMPILER(2) = 003BH
003B
003B C ************************************************************

003B C                U T I L I T Y     R O U T I N E S

003B C ************************************************************


003B C *************** ************************************************

003B        SUBROUTINE         MOVE@MEMORY
003B
003B C ************************************************************
003B C CALL THIS CODE AFTER REGISTERS SET UP: CTR IN C REG
003B C DES ARRAY POINTER IN D
003B C SRC ARRAY POINTER IN H
003B C  LOOP:  INX H  INX D --  COMPENSATE FOR FORT PTRS OFF BY 1
003B C MOV A,M  STAX D  DCR C  JNZ LOOP

003B 131      INLINE /  3H, 23H /
003D          INLINE / 7EH, 12H, 0DH, 0C2H, ADDRESS(131) /
0043          RETURN
0044          END


PROGRAM  STORAGE: 0009

VARIABLE STORAGE: 0000

SYMBOL TABLE
003B 131

0044 C ************************************************************

0044        SUBROUTINE         MOVE@ACCUM(MSRC,MDES)
0044
0044 C ************************************************************

0044 C MOVE CONTENTS OF 5 BYTE ARRAY(MSRC) TO ANOTHER 5 BYTE ARRAY (MDES)

0044          INTEGER*1 MSRC(5),MDES(5)
0044
0044 C MOVE CONTENTS OF MSRC ARRAY TO MDES
0044 C LHLD MSRC  XCHG  LHLD MDES  MVI C,5
0044          INLINE  / 2AH, ADDRESS(MDES), 0EBH, 2AH, ADDRESS(MSRC) /
004B          INLINE / 0EH, 5 /
004D          CALL MOVE@MEMORY
0050          RETURN
0051          END


PROGRAM  STORAGE: 0013

VARIABLE STORAGE: 0004
```

```
SYMBOL TABLE
3DBF MDES
3DC1 MSRC

0051 C ************************************************************

0051       SUBROUTINE        ADD@ACCUM (INR,M@ACCUM)
0051
0051 C ************************************************************
0051 C ADD I*5 TO I*5 ,  RESULT IN SECOND PARAMETER

0051         INTEGER*1       INR(5), M@ACCUM(5)
0051
0051 C LHLD INR  XCHG  LHLD M@ACCUM
0051 C MVI C,5  STC  CMC
0051 C LOOP:  INX H  INX D  LDAX D  ADC M  MOV M,A  DCR C  JNZ LOOP

0051         INLINE /  AH, ADDRESS(INR), 0EBH, 2AH, ADDRESS(M@ACCUM) /
0058         INLINE / 0EH, 5, 37H, 3FH/
005C 99      INLINE / 13H, 23H, 1AH, 8EH, 77H, 0DH, 0C2H, ADDRESS(99)/
0065         RETURN
0066         END


PROGRAM  STORAGE: 0021

VARIABLE STORAGE: 0004

SYMBOL TABLE
005C 99
3DBB M@ACCUM
3DBD INR

0066 C ************************************************************

0066       SUBROUTINE        ZERO@ACCUM(M@ACCUM)
0066
0066 C ************************************************************
0066       INTEGER*1 M@ACCUM(5)
0066 C FILL AN ACCUMULATOR WITH ZEROS
0066         CALL MOVE@ACCUM(ZEROS,M@ACCUM)
0075         RETURN
0076       END


PROGRAM  STORAGE: 0016

VARIABLE STORAGE: 0002

SYMBOL TABLE
3DB9 M@ACCUM

0076 C ************************************************************

0076       SUBROUTINE ACCUM@GE@LIM?  (M@ACCUM,LIM,I@FLAG)
0076
0076 C ************************************************************
```

```
0076 C COMPARE ACCUM TO LIM-- LIM IS ONLY 1 BYTE !!
0076         INTEGER*1       LIM, M@ACCUM(5),I@FLAG
0076
0276         I@FLAG = .TRUE.
027B
027B C LHLD M@ACCUM
027B C INX H  LDA LIM  CPR M  RZ  RC  ;COMPARE LIM TO LSB
027B C INX H  MOV A,M
027B C INX H  ORA M
027B C INX H  ORA M
027B C INX H  ORA M      ; SEE IF TOP 4 BYTES .NE. 0
027B C RNZ

027B         INLINE /  AH, ADDRESS(M@ACCUM)/
027E         INLINE / 23H, 3AH, ADDRESS(LIM), 0BEH, 0C8H, 0D8H /
0285         INLINE / 23H,7EH,  23H, 0B6H, 23H, 0B6H, 23H, 0B6H/
028D         INLINE / 0C0H/
028E
028E         I@FLAG = .FALSE.
0293         RETURN
0294         END


PROGRAM  STORAGE: 0030

VARIABLE STORAGE: 0006

SYMBOL TABLE
3DB3 I@FLAG
3DB5 LIM
3DB7 M@ACCUM

0294
0294 C ***************************************************************

0294 C ***************************************************************

0294         SUBROUTINE       PUT@TIME(MSRC,MDES)
0294
0294 C ***************************************************************

0294 C PUT THE CONTENTS OF 8 CONSECUTIVE BYTES INTO ANOTHER 8 CONSECUTIVE BYTES

0294         INTEGER*2 MSRC(4),MDES(4)
0294
0294 C  LHLD MDES  XCHG
0294 C  LHLD MSRC
0294 C  MVI C, 8
0294 C  INX H  INX D          I*2 PTRS OFF BY TWO
0294 C  LOOP:  INX H  INX D --  COMPENSATE FOR FORT PTRS OFF BY 1
0294 C MOV A,M  STAX D  DCR C  JNZ LOOP

0294         INLINE /  2AH, ADDRESS(MDES), 0EBH/
0298         INLINE /  AH, ADDRESS(MSRC) /
029B         INLINE / 0EH, 8 /
029D         INLINE /  3H,23H /
029F 131     INLINE / 13H, 23H /
02A1         INLINE / 7EH, 12H, 0DH, 0C2H, ADDRESS(131) /
```

```
00A7
00A7          RETURN
00A8          END


PROGRAM  STORAGE: 0020

VARIABLE STORAGE: 0004

SYMBOL TABLE
009F 131
3DAF MDES
3DB1 MSRC

00A8 C ************************************************************

00A8          SUBROUTINE        MOVE@TIME@TO@DIS(IME@REG)
00A8
00A8 C ************************************************************

00A8 C PUT THE CONTENTS OF 4 LOW BYTES OF I*2 ARRAY
00A8 C        INTO TIME@DISPLAY ( I*1 (4) )

00A8          INTEGER*2 IME@REG(4)
00A8
00A8 C  LXI D, TIME@DISPLAY
00A8 C  LHLD IME@REG
00A8 C  INX H                          I*2 PTRS OFF BY TWO
00A8 C  MVI C, 4
00A8 C  LOOP:  INX H  INX D --  COMPENSATE FOR FORT PTRS OFF BY 1
00A8 C MOV A,M  STAX D  INX H DCR C  JNZ LOOP

00A8           INLINE /  11H, ADDRESS(TIME@DISPLAY)/
00AB           INLINE / 2AH, ADDRESS(IME@REG) /
00AE           INLINE / 0EH, 4 /
00B0           INLINE / 23E/
00B1 131       INLINE /  3H, 23H /
00B3           INLINE / 7EH, 12H, 0DH, 23H, 0C2H, ADDRESS(131) /
00BA
00BA          RETURN
00BB          END


PROGRAM  STORAGE: 0019

VARIABLE STORAGE: 0002

SYMBOL TABLE
00B1 131
3DAD IME@REG

00BB
00BB C ************************************************************

00BB          SUBROUTINE         ZERO@TIME@REG(MDES)
00BB
00BB C ********** ***************************************************
```

```
UTI FORT//80 COMPILER 3.3A1                    PAGE 0022

00BB C PUT 0 IN 8 BYTES ( I*2 ARRAY OF 4)

00BB          INTEGER*2 MDES(4)
00BB
00BB C LHLD MDES
00BB C INX H  INX H  --  COMPENSATE FOR FORT PTRS OFF BY 2

00BB C XRA A  MOV M,A  INX H  MOV M,A  INX H  MOV M,A
00BB C    INX H  MOV M,A  INX H  MOV M,A  INX H  MOV M,A  INX H  MOV M,A

00BB          INLINE / 0AFH, 2AH, ADDRESS(MDES), 23H, 23H /
00C1
00C1          INLINE /  77H,23H, 77H, 23H,  77H,  23H, 77H, 23H  /
00C9          INLINE /  77H, 23H, 77H, 23H, 77H,  23H, 77H  /
00D0          RETURN
00D1          END


PROGRAM  STORAGE: 0022

VARIABLE STORAGE: 0002

SYMBOL TABLE
3DAB MDES

00D1
00D1 C ************************************************************
00D1 C ************************************************************

00D1        SUBROUTINE        MOVE4(I@CONFIG, I@WHERE)
00D1
00D1 C ************************************************************

00D1 C MOVE I*1 (4) ARRAY  I.E. MOVE A SPECIAL I@CONFIGURATION INTO
00D1 C         TIME@DISPLAY  OR  ERROR@DISPLAY

00D1          INTEGER*1 I@CONFIG (4), I@WHERE (4)
00D1
00D1 C  LHLD I@WHERE
00D1 C  XCHG
00D1 C  LHLD I@CONFIG
00D1 C  MVI C, 4
00D1          INLINE / 2AH, ADDRESS(I@WHERE), 0EBH /
00D5          INLINE / 2AH, ADDRESS(I@CONFIG) /
00D8          INLINE /  EB, 4 /
00DA          CALL MOVE@MEMORY
00DD          RETURN
00DE          END


PROGRAM  STORAGE: 0013

VARIABLE STORAGE: 0004

SYMBOL TABLE
3DA7 I@WHERE
3DA9 I@CONFIG
```

```
00DE C ****************************************
00DE C ************************************************************

00DE       SUBROUTINE  TIME@ZERO?(IME@REG,I@FLAG)
00DE
00DE C                                                        ****

00DE C CHECK IF A TIME REGISTER IS EQUAL TO 0
00DE C ONLY THE LSB  ARE USED (MSB IS ALWAYS = 0, SO IGNORE THEM)

00DE         INTEGER*2  IME@REG(4)
00DE         INTEGER*1  I@FLAG
00DE
00DE C /* INITIALIZE THE FUNCTION TO  FALSE
00DE         I@FLAG = .FALSE.
00E3
00E3 C LHLD IME@REG  XRA A   MVI C,4
00E3 C  INX H  INX H  --  COMPENSATE FOR FORT PTRS OFF BY 2

00E3         INLINE  / 2AH,  ADDRESS(IME@REG),  0AFH,  0EH, 4 /
00E9
00E9 C /* CHECK EACH LOW BYTE, RETURN IF FIND A NON-ZERO BYTE
00E9 C LOOP:  INX H  I X H  CMP M  RNZ    DCR C    JNZ  LOOP
00E9 114     INLINE  / 23H,23H, 0BEH, 0C0H, 0DH,  0C2H, ADDRESS(114)  /
00F1
00F1 C /*  IF ALL ZERO SET FUNCTION = .TRUE.  AND RETURN
00F1         I@FLAG = .TRUE.
00F5         RETURN
00F7         END


PROGRAM  STORAGE: 0025

VARIAPLE STORAGE: 0004

SYMBOL TABLE
00E9 114
3DA3 I@FLAG
3DA5 IME@REG

00F7 C ************************************************************

00F7         SUBROUTINE      STRING@TO@BUFFER(MSRC,M@CT)
00F7
00F7 C ************************************************************
00F7 C GLOBAL FORMAT@INDEX IS USED AS START INDEX FOR MDESTINATION
00F7 C PTR OF MDESTINATION IS IN BUFFER@INSERT@PTR

00F7         INTEGER*1       MSRC(50),  M@CT
00F7
00F7 C       DO 5 I = 1,M@CT
00F7 C       MDES(FORMAT@INDEX) = MSRC(I)
00F7 C5      FORMAT@INDEX = FORMAT@INDEX + 1

00F7 C LXI H,FORMAT@INDEX  MOV E,M MVI D,0  PUSH H
00F7 C LHLD BUFFER@INSERT@PTR  DAD D  XCHG
00F7 C LHLD MSRC  INX H  MOV B,H  MOV C,L
00F7 C  POP H  LDA M@CT
```

```
00F7 C LOOP:          PUSH PSW
00F7 C                LDAX B  STAX D  INR M  POP PSW
00F7 C                DCR A  RZ
00F7 C                INX B  INX D  JMP  LOOP

00F7          INLINE /  1H, ADDRESS(FORMAT@INDEX), 5EH, 16H, 0,  0E5H/
00FE          INLINE /  AH, ADDRESS(BUFFER@INSERT@PTR), 19H, 0EBH /
0103          INLINE / 2AH, ADDRESS(MSRC), 23H, 44H, 4DH/
0109          INLINE / 0E1H,  3AH, ADDRESS(M@CT)/
010D 100      INLINE / 0F5H, 0AH, 12H, 34H, 0F1H, 3DH, 0C8H/
0114          INLINE / 13H, 03H, 0C3H, ADDRESS(100)/
0119
0119          RETURN
011A          END


PROGRAM  STORAGE: 0035

VARIABLE STORAGE: 0004

SYMBOL TABLE
010D 100
3D9F M@CT
3DA1 MSRC

011A
011A C ************************************************************

011A C       I N I T I A L I Z A T I O N   R O U T I N E S

011A C ************************************************************


011A C ************************************************************

011A        SUBROUTINE         RESET@7@8
011A
011A C ************************************************************

011A C RESET CTR AND ZERO ACCUMULATOR FOR ERROR RATE 10**7(8)

011A          CALL ZERO ACCUM(ERRORS@TEN@7@8)
0123          TEN@7@8@C R = 10
0128          IF (ERROR MODE .EQ. 10H) TEN@7@8@CTR = 100
0132          RETURN
0133          END


PROGRAM  STORAGE: 0025

VARIABLE STORAGE: 0000

SYMBOL TABLE

0133 C ************************************************************

0133        SUBROUTINE         CLEAR@THIS@INT@ACCUMS@AUTO
0133
```

```
0133 C ************************************************************
0133           CALL ZERO@ACCUM(THIS@INT@BIT@ERRORS@A)
013C           CALL ZERO@ACCUM(THIS@INT@PARITY@ERRORS@A)
0145           CALL ZERO@ACCUM(THIS@INT@BIPOLAR@VIOLATIONS@A)
014E           BER@INT@A=.FALSE.
0153           PAR@INT@A=.FALSE.
0157           BP@INT@A=.FALSE.
015C           RETURN
015D           END


PROGRAM  STORAGE: 0042

VARIABLE STORAGE: 0000

SYMBOL TABLE

015D C ************************************************************

015D        SUBROUTINE         CLEAR@THIS@INT@ACCUMS@PRT
015D
015D C ************************************************************
015D           CALL ZERO@ACCUM(THIS@INT@BIT@ERRORS@P)
0166           CALL ZERO@ACCUM(THIS@INT@PARITY@ERRORS@P)
016F           CALL ZERO@ACCUM(THIS@INT@BIPOLAR@VIOLATIONS@P)
0178           BER@INT@P=.FALSE.
017D           PAR@INT@P=.FALSE.
0181           BP@INT@P=.FALSE.
0186           RETURN
0187           END


PROGRAM  STORAGE: 0042

VAPIABLE STORAGE: 0000

SYMBOL TABLE

0187 C

0187           SUBROUTINE ZERO@ASCII@TIMER
0187
0187 C************************************************************

0187 C         INTEGER*1 I

0187 C         ASCII@TIMER(1)=30H
0187 C         DO 20 I=2,7
0187 C20       ASCII@TIMER(I)=20H
0187 C         RETURN
0187 C    ASSEMBLY TO SAVE SPACE
0187 C     LXI H,ADDRESS(ASCII@TIMER)  INX H   MVI M,30  MVI A,6
0187 C5    INX H  MVI M,20H  DCR A  JNZ 5  RET
0187           INLINE /21H,ADDRESS(ASCII@TIMER),23H,36H,30H,3EH,6/
018F 5         INLINE /23H,36H,20H,3DH,0C2H,ADDRESS(5)/
0195           RETURN
0197           END
```

```
PROGRAM  STORAGE: 0016

VARIABLE STORAGE: 0000

SYMBOL TABLE
018F 5

0197
0197 C ************************************************************

0197       SUBROUTINE         ZERO@INTERNAL@ACCUMS
0197
0197 C ************************************************************

0197 C  4/1/81 CHANGE TO GET MORE ROOM
0197 C  ZERO ALL ACCUMULATORS BY SINGLE LOOP ROUTINE
0197 C
0197 C  MVI B,247  LXI H,TOTAL@THIS@SEC@ACC  XRA A
0197 C  LOOP:   INX H,  MOV M,A ,  DCR B    JNZ LOOP
0197 C          RET
0197 C
0197          INLINE / 06H,247,  21H, ADDRESS(TOTAL@THIS@SEC@ACC), 0AFH /
019D 157      INLINE / 23H, 77H, 05H ,0C2H, ADDRESS(157) /
01A3          CALL ZERO@ASCII@TIMER
01A6          RETURN
01A7
01A7          END


PROGRAM  STORAGE: 0016

VARIABLE STORAGE: 0000

SYMBOL TABLE
019D 157

01A7 C************************************************

01A7       SUBROUTINE         RESET@SYNTAX@FLAGS
01A7
01A7 C************************************************
01A7          RX@NEED@ALPHA1 = 1
21AC          RETURN
01AD          END


PROGRAM  STORAGE: 0006

VAPIABLE STORAGE: 0000

SYMBOL TABLE

01AD C ***************************************************************************

01AD          SUBROUTINE       SELECT@PRT@BUF
01AD
01AD C ***************************************************************************
```

```
01AD C  TRIES TO FIND A BUFFER THAT HAS A 1 IN STARTING LOCATION.THIS
01AD C  INDICATES THAT THE BUFFER IS READY TO USE.
01AD C  FIRST MAKE SURE THERE IS A RING BUFFER POSITION
01AD C  IN ANY CASE IF UNSUCCESSFUL- SET THE RETURN FLAG
01AD          FORMAT@INDEX=1
01B2 C        LXI H,PBUF1
01B2 C        SHLD BUFFER@INSERT@POINTER
01B2 C        RET
01B2          INLINE /21H,ADDRESS(PBUF1)/
01B5          INLINE /22H,ADDRESS(BUFFER@INSERT@PTR)/
01B8          RETURN
01B9          END


PROGRAM  STORAGE: 0012

VARIABLE STORAGE: 0000

SYMBOL TABLE

01B9 C ***************************************************************************

01B9          SUBROUTINE      SELECT@AUTO@BUF
01B9
01B9 C ***************************************************************************
01B9 C SAME TYPE CODE  S SELECT@PRT@BUF
01B9          FORMAT@INDEX=1
01BE          INLINE /21H,ADDRESS(ABUF1)/
01C1          INLINE /22H,ADDRESS(BUFFER@INSERT@PTR)/
01C4          RETURN
01C5          END


PROGRAM  STORAGE: 0012

VARIABLE STORAGE: 0000

SYMBOL TABLE

01C5 C*******************************************************

01C5        SUBROUTINE          INITIAL@SYSTEM@STATE
01C5
01C5 C*******************************************************

01C5 C INITIALIZE THE START- UP STATE OF THE SYSTEM
01C5          IF(.NOT.FIRST@FLAG) GO TO 5
01CD
01CD          TEST@TYPE =
01CD      $           ERROR@TYPE = 2
01D5          ERROR@MODE = 8
01DA
01DA          THRESHOLD=1
01DE          HORN@ENABLE=0
01E2          HORN=0
01E5          PRINT@CONTROL=1
01E9          PARALLEL@POLL@MASK=10H
01ED          TX@MODE=2
```

```
01F2          TX@TIME=.TRUE.
01F5          RS@LOCAL@LOCKOUT=
01F5      &       RS@DELAY@FLAG=
01F5      &       RS@DELAY@CTR=.FALSE.
0200          BUS@REMOTE@ENABLE=.FALSE.
0205
0205          CALL ZERO@INTERNAL@ACCUMS
0208          CALL RESET@7@8
020B

020B          CALL ZERO@TIME@REG(TIMER@START@TIME )
0214          CALL ZERO@TIME@REG(TIMER)
021D          CALL ZERO@TIME@REG(ELAPSED@TIME)
0226
0226          INPUT@TYPE =
0226      &           TIME@MODE =
0226      &           LAST@TIME@MODE =
0226      &           CLOCK@TASK = 1
0234
0234          RUN@ON@TICK=
0234      &           SAVE@TIME@DISPLAY =
0234      &           BLU@CONDITION =
0234      &           BLINKER =
0234      &           UPDATE@FRONT@PANEL  = .FALSE.
0245
0245 C INIT. VARIABLES SOME 'OBTAINED','BLINKING','RESET' AND
0245 C  'RESULT' VARIABLES TO ZERO
0245 C     LXI H, ADD(PRBS@OBTAINED)  MVI A,10
0245 C1    MVI M,0  INX H  DCR A  JNZ 1
0245          INLINE /21H,ADDRESS(PRBS@OBTAINED),3EH,10/
024A 1        INLINE /36H,0,23H,3DH,0C2H,ADDRESS(1)/
0251
0251          RUN@STOP = RUN@STOP@LIGHT = 2
0259
0259 C    SIGNAL LOSS, FRAME LOSS, PRBS LOSS ON
0259          SIGNAL@LOSS =
0259      &           FRAME@LOSS =
0259      &           PRBS@LOSS = .TRUE.
0264
0264          GATING@WI K@CTR =
0264      &           BLINK@CTR = 0
026C
026C          CALL MOVE4(BLANKS, ERROR@DISPLAY)
027B          CALL MOVE4(ZEROS, TIME@DISPLAY)
028A

028A          BURST@LETTER='P'
028F
028F C INITIALIZATION FOR PERIPHERAL INTERFACES


028F 5        MESSAGE@NUMBER =
028F      &           Q COMMAND@A = Q@COMMAND@P =0
029A
029A          BURST@FLAG=.FALSE.
029F          A@COUNTER=P@COUNTER=0
02A6
02A6          INT@CTR=0
```

```
02AB
02AB          LOG@MADE@IN@LAST@HOUR=
02AB       &          BLOCK@TEMPS@FULL@A =BLOCK@TEMPS@FULL@P = .FALSE.
02B6
02B6 C        FPERROR@TO@PERIPH = .FALSE.


02B6          IF(RUN@STOP.EQ.2)MACHINE@FORMAT=.FALSE.
02C3          CALL RESET@SYNTAX@FLAGS
02C6          ABUF1(1)=PBUF1(1)=1
02D1          REAL@TIME@A(1)=REAL@TIME@P(1)=1
02DD

02DD          RETURN
02DE          END


PROGRAM  STORAGE: 0281

VARIABLE STORAGE: 0000

SYMBOL TABLE
024A 1
028F 5

02DE
02DE C*****************************************************************

02DE          SUBROUTINE      CHECK@MIL@INT
02DE
02DE C*****************************************************************

02DE C  IN 21H   ANI 10H  RNZ  IN 20H  LXI H,INT@CTR  INR M   RET

02DE          INLINE /0DBH,21H,0E6H,10H,0C0H,0DBH,20H/
02E5          INLINE /21H,ADDRESS(INT@CTR),34H/
02E9          RETURN
02EA          END


PROGRAM  STORAGE: 0012

VARIABLE STORAGE: 0000

SYMBOL TABLE

02EA C ***************************************************************************

02EA          SUBROUTINE      SAVE@TOTAL@BLOCK(SRC,DES1,DES2)
02EA
02EA C ***************************************************************************
02EA          INTEGER*1 SRC(5),DES1(5),DES2(5)
02EA
02EA C  THIS ROUTINE IS USED TO PUT THE ACCUMULATORS IN TEMPORARY BLOCKS
02EA C  IN ORDER FOR THE INFORMATION TO BE PRINTED OUT. IT IS IN THIS FORM
02EA C  FOR SPATIAL AS WELL AS TIME PROBLEMS. SRC STARTS AT THE LOWER PART
02EA C  OF THE BIPOLAR VIOLATIONS ERROR SEC ACCUMULATORS AND THE
02EA C  DESTINATIONS ARE THE BPV MEAS ACCUMULATORS FOR ERROR SECONDS
```

```
02EA C  AND PER CENT E ROR SECONDS. WITH THE APPROPRIATE ENTRYS INTO
02EA C  THE PARAMETERS, THE TOTAL ERRORS, BER , ERROR SEC, AND PER
02EA C  CENT ERROR SECONDS WILL ALL BE UPDATED. THIS ROUTINE SHOULD
02EA C  BE CALLED TO UPDATE THE AUTO AND THE PRINTER TEMPORARIES.
02EA C    NOTE: THE VALUES IN THE BER TEMPORARY WILL ONLY BE VALID
02EA C          FOR CUMULATIVE RATE. ANOTHER ROUTINE BELOW UPDATES
02EA C          FOR INSTANTANEOUS.

02EA C     MVI A,24  LHLD ADD(SRC) INX H  PUSH H  LHLD ADD(DES1)
02EA C     INX H  XCHG  LHLD ADD(DES2)  MOV B,H  MOV C,L  INX B
02EA C     POP H
02EA C 10  PUSH PSW  MVI A,5
02EA C 20  PUSH PSW  MOV A,M  STAX D  STAX B INX H  INX D  INX B
02EA C      POP A  DCR A
02EA C     JNZ 20  INX D  INX D  INX B  INX B  POP PSW  DCR A
02EA C     JNZ 10  RET
02EA         INLINE /3EH,24,2AH,ADDRESS(SRC),23H,0E5H,2AH/
02F2         INLINE /ADDRESS(DES1),23H,0EBH,2AH,ADDRESS(DES2)/
02F9         INLINE /44H,4DH,03H,0E1H/
02FD 10      INLINE /0F5H,3EH,5/
0300 20      INLINE /0F5H,7EH,12H,02H,23H,13H,03H,0F1H,3DH/
0309         INLINE /0C2H,ADDRESS(20),13H,13H,03H,03H,0F1H/
0311         INLINE /3DH,0C2H,ADDRESS}10~/
0315         RETURN
0316         END


PROGRAM  STORAGE: 0044

VARIABLE STORAGE: 0012

SYMBOL TABLE
0300 20
02FD 10
3D93 DES2
3D97 DES1
3D9B SRC

0316 C ***************************************************************************

0316         SUBROUTINE      SAVE@TOTAL@BLOCK@W@CUMULATIVE@RATE@A
0316
0316 C ***************************************************************************
0316 C ** BE SURE THE TEMPORARIES ARE EMPTY
0316         IF(BLOCK@TEMPS@FULL@A) RETURN
031B         BLOCK@MASK@A=NEW@ERROR@MASK
0321         BLOCK@TEMPS@FULL@A=.TRUE.
0326         CALL SAVE@TOTAL@BLOCK(BIPOLAR@VIOLATION@SECS@T7,
032C      &   V@MEAS@2A@T7,V@MEAS@4A@T7)
033B         CALL MOVE@ACCUM(TOTAL@BITS,DENOM@MEAS@A)
034A         CALL MOVE@ACCUM(TOTAL@SECONDS,DENOM@MEAS@SEC@A)
0359 C       CALL MOVE@ACCUM(BURST@1@10,BST1@TMP@A)
0359 C       CALL MOVE@ACCUM(BURST@11@99,BST2@TMP@A)
0359 C       CALL MOVE@ACCUM(BURST@GT@100,BST3@TMP@A)
0359 C  ASSEMBLY TO SAVE SPACE
0359 C       LXI H,ADD(BURST@GT@100)  LXI D,ADD(BST3@TMP@A)
0359 C       MVI C,3
0359 C1      MVI B,5
```

```
0359 C2       INX H   INX D  MOV A,M  STAX D  DCR B  JNZ 2
0359 C        INX D  INX D  DCR C  JNZ 1
0359          INLINE /21H,ADDRESS(BURST@GT@100),11H,ADDRESS(BST3@TMP@A)/
035F          INLINE /0EH,3/
0361 1        INLINE /6,5/
0363 2        INLINE /23H,13H,7EH,12H,5,0C2H,ADDRESS(2)/
036B          INLINE /13H,13H,0DH,0C2H,ADDRESS}1 /
0371          RETURN
0372          END


PROGRAM  STORAGE: 0092

VARIABLE STORAGE: 0000

SYMBOL TABLE
0363 2
0361 1

0372 C*****************************************************************************

0372          SUBROUTINE    SAVE@TOTAL@BLOCK@W@CUMULATIVE@RATE@P
0372
0372 C*****************************************************************************
0372          IF(BLOCK@TEMPS@FULL@P) RETURN
0377          BLOCK@MASK@P=NEW@ERROR@MASK
037D          BLOCK@TEMPS@FULL@P=.TRUE.
0382          CALL SAVE@TOTAL@BLOCK(BIPOLAR@VIOLATION@SECS@T7,
0388       $   V@MEAS@2P@T7,V@MEAS@4P@T7)
0397          CALL MOVE@ACCUM(TOTAL@BITS,DENOM@MEAS@P)
03A6          CALL MOVE@ACCUM(TOTAL@SECONDS,DENOM@MEAS@SEC@P)
03B5 C        CALL MOVE@ACCUM(BURST@1@10,BST1@TMP@P)
03B5 C        CALL MOVE@ACCUM(BURST@11@99,BST2@TMP@P)
03B5 C        CALL MOVE@ACCUM(BURST@GT@100,BST3@TMP@P)
03B5 C  SAME AS ABOVE
03B5          INLINE /21H,ADDRESS(BURST@GT@100),11H,ADDRESS(BST3@TMP@P)/
03BF          INLINE /0EH,3/
03BD 1        INLINE /6,5/
03BF 2        INLINE /23H,13H,7EH,12H,5,0C2H,ADDRESS(2)/
03C7          INLINE /13H,13H,0DH,0C2H,ADDRESS(1)/
03CD          RETURN
03CE          END


PROGRAM  STORAGE: 0092

VARIABLE STORAGE: 0000

SYMBOL TABLE
03BF 2
03BD 1

03CE C*****************************************************************

03CE          SUBROUTINE  SAVE@INTERVAL@BLOCK(FLAGS,TEMPS,SRC)
03CE
03CE C*****************************************************************
```

```
03CE          INTEGER*1 FLAGS,TEMPS(5),SRC(5)
03CE C THIS ROUTINE IS ALSO ADDED TO SAVE SPACE AND TIME. IT LOOKS
03CE C AT A BLOCK OF FLAGS FOR A CHOSEN ERROR TYPE AND THEN BASED
03CE C ON THAT FLAG IT LOADS THE THRESHOLD "0" INTERVAL TOTAL OR
03CE C ZERO INTO THE OTHER THRESHOLD TEMPORARIES.
03CE C ROUTINE MUST BE CALLED FOR EACH ERROR TYPE FOR BOTH AUTO
03CE C AND PRINTER.

03CE C      LDA  ADD(FLAGS) RLC RLC RLC RLC
03CE C      LHLD ADD(TEMPS)  INX H  XCHG MOV L,A
03CE C      MVI A,4
03CE C 10   PUSH PSW   LXI B,ADD(ZEROS) INX B   MOV A,L RLC MOV L,A JNC 20
03CE C      PUSH H  LHLD ADD(SRC) INX H MOV B,H  MOV C,L  POP H
03CE C 20   MVI A,5
03CE C 30   PUSH PSW  LDAX B  STAX D  INX B  INX D  POP PSW  DCR A
03CE C      JNZ 30  INX D  INX D   POP PSW  DCR A  JNZ 10
03CE C      RET

03CE          INLINE /3AH,ADDRESS(FLAGS),7,7,7,7,2AH,ADDRESS(TEMPS)/
03D8          INLINE /23H,0EBH,6FH,3EH,4/
03DD 10       INLINE /0F5H,01H,ADDRESS(ZEROS),3,07DH,7,6FH,0D2H,ADDRESS(20)/
03E8          INLINE /0E5H,2AH,ADDRESS(SRC),23H,44H,4DH,0E1H/
03F0 20       INLINE /3EH,5/
03F2 30       INLINE /0F5H,0AH,12H,3,13H,0F1H,3DH,0C2H,ADDRESS(30)/
03FC          INLINE /13H,13H,0F1H,3DH,0C2H,ADDRESS(10)/
0403          RETURN
0404          END


PROGRAM  STORAGE: 0054

VARIABLE STORAGE: 0012

SYMBOL TABLE
03F2 30
03F0 20
03DD 10
3D87 SRC
3D8B TEMPS
3D8F FLAGS

0404
0404 C ***************************************************************************

0404          SUBROUTINE      SAVE@TOTAL@BLOCK@W@INTERVAL@RATE
0404
0404 C ********************************************
0404          IF(BLOCK@TEMPS@FULL@A) GO TO 10
040B          CALL SAVE@TOTAL@BLOCK@W@CUMULATIVE@RATE@A
040E C IF RUN@STOP .EQ. @STOP THEN NEED THE BLOCK W/ CUMLATIVE RATE
040E C LDA RUN@STOP  CPI 1  JNZ 10
040E          INLINE / 3AH,ADDRESS}RUN@STOP~, 0FEH , 01H /
0413          INLINE /0C2H,ADDRESS(10)/
0416
0416          CALL SAVE@INTERVAL@BLOCK(BER@INT@A,
0416       &    B@MEAS@3A@T7,THIS@INT@BIT@ERRORS@A)
0431          CALL SAVE@INTERVAL@BLOCK(PAR@INT@A,
0431       &    P@MEAS@3A@T7,THIS@INT@PARITY@ERRORS@A)
```

```
044C          CALL SAVE@INTERVAL@BLOCK(BP@INT@A,V@MEAS@3A@T7,
0458      &     THIS@INT@BIPOLAR@VIOLATIONS@A)
0467          CALL CHECK@MIL@INT
046A
046A          CALL MOVE@ACCUM(T6@CONS@1,DENOM@MEAS@A)
0479 10       IF(BLOCK@TEMPS@FULL@P) RETURN
047E          CALL SAVE@TOTAL@BLOCK@W@CUMULATIVE@RATE@P
0481          INLINE /3AH,ADDRESS(RUN@STOP),0FEH,01H/
0486          INLINE /0C0H/
0487          CALL CHECK@MIL@INT
048A
048A          CALL SAVE@INTERVAL@BLOCK(BER@INT@P,
048A      &     B@MEAS@3P@T7,THIS@INT@BIT@ERRORS@P)
04A5          CALL SAVE@INTERVAL@BLOCK(PAR@INT@P,
04A5      &     P@MEAS@3P@T7,THIS@INT@PARITY@ERRORS@P)
04C0          CALL SAVE@INTERVAL@BLOCK(BP@INT@P,
04C0      &     V@MEAS@3P@T7,THIS@INT@BIPOLAR@VIOLATIONS@P)
04DB          CALL MOVE@ACCUM(T6@CONS@1,DENOM@MEAS@P)
04EA          RETURN
04EB
04EB          END


PROGRAM  STORAGE: 0231

VARIABLE STORAGE: 0000

SYMBOL TABLE
0479 10

04EB
```

C. FRONT PANEL INPUT ROUTINE

```
04EB C }}}}}}}}}}}}}}}}}}}}}}}}}}}} FILE: INPUT.FTN {{{{{{{{{{{{{{{{{{{{{{{{{{{{

04EB C ************************************************************
04EB         SUBROUTINE ADJUST@NEW@ERROR@MASK
04EB C ************************************************************
04EB C IF BLUE SIGNAL NEVER PRINT BIT ERRORS
04EB         IF (BLU@CONDITION) GO TO 3
04F2         IF (PRBS@OBTAINED) RETURN
04F7 C IF LIVE TRAFFIC, IGNORE BIT ERRORS
04F7 C LDA NEW@ERROR@MASK   ANI 6   STA NEW@ERROR@MASK
04F7 3       INLINE / 3AH, ADDRESS(NEW@ERROR@MASK), 0E6H, 6 /
04FC         INLINE / 32H, ADDRESS(NEW@ERROR@MASK) /
04FF         RETURN
0500         END


PROGRAM  STORAGE: 0021

VARIABLE STORAGE: 0000

SYMBOL TABLE
04F7 3

0500
0500 C ************************************************************

0500 C       F R O N T   P A N E L   I N P U T   R O U T I N E S

0500 C ************************************************************



0500 C ************************************************************

0500       SUBROUTINE        ROTATE@DIGIT@INTO@TIME
0500
0500 C ************************************************************

0500 C ROTATE THE BCD DIGITS IN TIME@DISPLAY TO THE LEFT
0500 C       -- TO LOOK LIKE A CALCULATOR
0500 C       -- INSERT THE DIGIT IN DECODE@INPUT IN THE
0500 C       -- LEAST SIGNIFICANT NIBBLE
0500 C TIME ON FRONT PANEL WILL BE CHANGED ON NEXT UPDATE OF FRONT PANEL


0500 C MVI C,3   LXI H, TIME@DISPLAY(4), INX H
0502 C  INX H  --  COMPENSATE FOR FORT PTRS OFF BY 1
0503         INLINE / 0EH, 3,  21H, ADDRESS(TIME@DISPLAY), 23H /
0506         INLINE /  23H, 23H, 23H /
0509
0509 C LOOP:   MOV D,M  DCX H  MOV E,M  XCHG   DAD H  DAD H  DAD H  DAD H
0509 113     INLINE / 56H, 2BH, 5EH, 0EBH, 29H, 29H, 29H, 29H /
0511
0511 C XCHG  INX H  MOV M,D  DCX H   DCR C  JNZ LOOP
0511         INLINE /  EBH, 23H, 72H, 2BH, 0DH, 0C2H, ADDRESS(113) /
0519
0519 C LDA DECODE@INPUT  ADD E  MOV M,A
0519         INLINE / 3AH, ADDRESS(DECODED@INPUT), 83H, 77H /
```

```
051E
051E          RETURN
051F          END


PROGRAM  STORAGE: 0031

VARIABLE STORAGE: 0000

SYMBOL TABLE
0509 113

051F C ************************************************************

051F        SUBROUTINE         COMMAND@ERROR
051F
051F C ************************************************************
051F C ** COMMAND ERROR CAN BE GENERATED BY SEVERAL CONDITIONS:
051F C        ERRORS SETTING THE CLOCK
051F C        ON CODE INPUT, INVALID FIRST LETTER, INVALID 2ND LETTER,
051F C                NUMERIC TOO LARGE OR OUT OF PLACE
051F          MESSAGE@NUMBER = 1
0524          RETURN
0525          END


PROGRAM  STORAGE: 0006

VARIABLE STORAGE: 0000

SYMBOL TABLE

0525
0525 C ************************************************************

0525        SUBROUTINE         TIME@DISPLAY@ERROR
0525
0525 C ************************************************

0525 C PUT 'Error' INTO THE TIME@DISPLAY
0525          CALL MOVE4(TIME@ERROR,TIME@DISPLAY)
0534          CALL COMMAND@ERROR
0537          RETURN
0538          END


PROGRAM  STORAGE: 0019

VARIABLE STORAGE: 0000

SYMBOL TABLE

0538
0538 C ************************************************************

0538        SUBROUTINE    CATCH@JUNK@IN@TIME(JUNK@FLAG)
0538
0538 C ************************************************************
```

```
0538 C THIS ROUTINE CALLED WHEN A SET REQUESTED
0538 C ERROR CHECK THE ENTRY IN THE TIME@DISPLAY
0538 C  ( THE ENTRY MUST HAVE BEEN MOVED TO TIME@TEMP ALREADY!)
0538 C IF FIND AN ERRORNEOUS CHARACTER IN THE DISPLAY OR TOO
0538 C   BIG FOR THE UNIT  .I.E. 66 SEC, THEN PUT 'Error' IN THE
0538 C   DISPLAY
0538 C MON/DAY RELATIONSHIP IS CHECKED IN THE ASSEMBLER ROUTINE
0538 C       TO SET THE MO/DAY
0538 C BLANKS(F'S) ARE CHANGED TO ZEROS

0538         INTEGER*2 K, TEMP
0538         INTEGER*1 JUNK@FLAG
0538
0538 C INTIALIZE RETURN VALUE TO FALSE
0538         JUNK@FLAG=.TRUE.
053D
053D         DO 32 K=1,4
0543
0543 C CHANGE BLANK IN UPPER NIBBLE TO 0
0543         IF ( TIME@TEMP(K) .GE. 0F0H)
054F      {           TIME@TEMP}K  = TIME@TEMP}K) .AND. 0FH
0577
0577 C ANYTHING BIGGER THAN 99 IS INVALID
0577         IF ( TIME@TEMP(K) .GT. 99H )  RETURN
0587
0587 C LOOK AT LOWER NIBBLE
0587         TEMP = TIME@TEMP(K) .AND. 0FH
059E
059E C CHANGE A BLANK  N LOWER NIBBLE TO 0
059E         IF ( TEMP .EQ. 0FH)
059E      &          TIME@TEMP(K) = TIME@TEMP(K) .AND. 0F0H
05CC
05CC C LOWER NIBBLE > 9 IS INVALID
05CC 32      IF ( TEMP .GT. 9 ) RETURN
05E6

05E6 C BE SURE MIN AND SEC ARE <= 59
05E6         IF ( TIME@TEMP(1) .GT. 59H)   RETURN
05F1         IF ( TIME@TEMP(2) .GT. 59H) RETURN
05FA
05FA         JUNK@FLAG = .FALSE.
05FF         RETURN
0600         END


PROGRAM  STORAGE: 0200

VARIABLE STORAGE: 0006

SYMBOL TABLE
05CC 32
3D81 TEMP
3D83 K
3D85 JUNK@FLAG

0600 C ************************************************************
```

```
0600       SUBROUTINE        GET@SET@TO@SET
0600
0600 C ***********************************************
0600 C CALLED BY BRANCH@ON@INPUT ROUTINE
0600 C ROUTINE CALLED WHEN COMBO OF SET BUTTON AND TIME@MODE SELECTION
0600 C       PUSHED.  CALL ERROR CHECK ROUTINE AND THEN BRANCH ON THE
0600 C       TIME@MODE SELECTED.  TIMER OR ELASPED CAN BE SET IN THE
0600 C       ROUTINE.  CLOCK SETS RESULT IN CLOCK@TASK INDICATOR FOR
0600 C       SETTING THE CLOCK( TIME OF DAY  OR  MO/DAY) IN THE MAIN
0600 C       LINE.

0600         INTEGER*2 G@TEMP
0600         INTEGER*1 T@FLAG
0600

0600         CALL    CATCH@JUNK@IN@TIME(T@FLAG)
060F         IF      (T@FLAG)        GO TO 21
0616
0616         GO TO (20,22,24,28,28), DECODED@INPUT - 0DH

0624         RETURN
0625
0625 C .....................................................................
0625 C  SET THE MONTH AND DAY OF REAL TIME CLOCK
0625 20      CLOCK@TASK=3
062A         RETURN
062B

062B C .....................................................................
062B C SET TIME OF DAY
062B 22      IF ((TIME@TEMP(3) .GE. 24H) .OR. (TIME@TEMP(4) .NE. 0)) GO TO 21
064D C CAN'T SET THE SECONDS !!!!  THEREFORE ZERO THEM
064D         CLOCK@TASK = 2
0652         RETURN
0653

0653 C .....................................................................
0653 C PRESET ELAPSED TIME
0653 24      CALL PUT@TIME(TIME@TEMP,ELAPSED@TIME)
0661         GO TO 30
0664
0664 C .....................................................................
0664 C SET TIMER
0664 C   SETTING TIMER=0 IS AN ERROR !!!!!
0664 28      CALL TIME@ZERO?(TIME@TEMP,T@FLAG)
0679         IF (T@FLAG) GO TO 21
0680
0680         CALL PUT@TIME(TIME@TEMP,TIMER@START@TIME)
068E         CALL PUT@TIME(TIME@TEMP,TIMER)
069C C   TEST@TYPE IS EITHER SINGLE (4) OR REPEAT (8) TIMED
069C         TEST@TYPE = (DECODED@INPUT - 10H) * 4
06A9         RESET@TIMER@NEXT@RUN = .TRUE.
06AE
06AE C IF NOT SETTING HARDWARE CLOCK THEN CLEAR FLAG TO UPDATE TIME@DISPLAY
06AE 30      SAVE@TIME@DISPLAY = .FALSE.
06B3
06B3         RETURN
```

```
06B4
06B4 C *** ERROR RETURN -- INVALID ENTRY DETECTED IN THIS ROUTINE
06B4 21       CALL TIME@DISPLAY@ERROR
06B7          RETURN
06B8
06B8          END


PROGRAM  STORAGE: 0194

VARIABLE STORAGE: 0003

SYMBOL TABLE
06AE 30
0664 28
0653 24
062B 22
0625 20
06B4 21
3D7E T@FLAG
3D7F G@TEMP

06B8
06B8 C ***************************************************

06B8        SUBROUTINE        BRANCH@ON@INPUT
06B8
06B8 C ***************************************************

06B8 C THIS ROUTINE  BRANCHES ON THE DECODED@INPUT, SETTING FLAGS AND
06B8 C        FRONT PANEL OUTPUT INFO
06B8 C  *             FRONT PANEL INTERRUPT ROUTINE
06B8 C THIS ROUTINE  CALLED BY ASSEMBLY INTERRUPT ROUTINE
06B8 C        WHEN AN INPUT IS DETECTED
06B8 C THIS IS THE LOWEST PRIORITY INTERRUPT EVENT
06B8 C THE INPUT MUST BE IN DECODED@INPUT



06B8 C FLAG USED TO UNFREEZE TIME@DISPLAY WHEN THERE IS A TIME@MODE
06B8 C        SELECTION ( UNLESS THERE IS A SET )
06B8          INTEGER*1 THIS@IS@SET
06B8          INTEGER*1 B@TEMP, ZERO@FLAG
06B8
06B8 C GUARD AGAINST ILLEGAL CODES
06B8          IF ((DECODED@INPUT .AND. 7FH) .GT. 40H) RETURN
06C2
06C2          UPDATE@FRONT@PANEL = .TRUE.
06C7          THIS@IS@SET = .FALSE.
06CC
06CC C FIRST DETERMINE IF SET BUTTON MASHED !!!
06CC C                ( SET BUTTON MASK IS 80H !! )
06CC C ALLOW SET OF TIMER OR ELAPSED W/ OUT SET BUTTON
06CC          IF ((TIME@MODE .EQ. 10H) .AND. (DECODED@INPUT .GE. 10H)
06D4       &          .AND. (DECODED@INPUT .LE. 12H ) ) GO TO 37
06EF
06EF          IF(DECODED@INPUT.GE.0H) GO TO 45
06F5
```

```
06F5 C ************************************************************

06F5 C                S E T T I N G

06F5 C ************************************************************

06F5          DECODED@INPUT = DECODED@INPUT - 80H
06F9
06F9 C /*  N O !! SETTING TIME IF RUNNING  !!!!
06F9 37       IF ( RUN@STOP .EQ. 1 ) GO TO 38
0701
0701          IF (DECODED@INPUT .GE. 0EH ) GO TO 42
0709
0709 38       IF (DECOD D@INPUT - 0AH )  40,  39,  44
070F


070F C .........................*** CLEAR ***...................
070F 39       IF (RUN@STOP.EQ.1) RETURN
071B          CALL MOVE4(ZEROS,TIME@DISPLAY)
072A C CLEAR ALL THE TIME REGISTERS ON A CLEAR
072A          CALL ZERO@TIME@REG(TIMER)
0733          CALL ZERO@TIME@REG(TIMER@START@TIME)
073C          CALL ZERO@TIME@REG(ELAPSED@TIME)
0745 C RESET TEST@TYPE TO ELAPSED
0745          TEST@TYPE = 2
074A          GO TO 41
074D

074D C .....................*** DIGITS 0-9 ***...................
074D 40       IF (RUN@S OP.EQ.1) RETURN
0753          CALL ROTATE@DIGIT@INTO@TIME
0756 41       SAVE@TIME DISPLAY = .TRUE.
075B C DESELECT THE TIME@MODE
075B          TIME@MODE = 10H
075F          RETURN
0760

0760 C .....................*** SPECIAL FUNCTIONS ***.................
0760 C (SOME CODES MAY BE NO SPECIAL FUNCTION WITH SET BUTTON DOWN )
0760 44       GO TO ( 445, 265 ),     DECODED@INPUT - 0AH

076E C        BH BECOMES 1 AND CH BECOMES 2

076E          RETURN
076F C  LIGHT TEST - CALL THROUGH JUMP TABLE
076F 445      INLINE /0CDH,1BH,0/
0772
0772          RETURN
0773
0773 C ..................*** READY TO SET TIME ***.................

0773 C MOVE TIME@DISPLAY DIGITS INTO A I*2 ARRAY BECAUSE OF LIMITATIONS
0773 C                OF RELATIONAL OPERATORS

0773 42       THIS@IS@SET = .TRUE.
```

```
0778 C ONLY 7 DIGITS APPEAR IN DISPLAY, CLEAR TOP NIBBLE
0778         TIME@DISPLAY(4) = TIME@DISPLAY(4) .AND. 0FH
0781
0781 C PUT DIGITS IN TIME@DISPLAY INTO 16 BIT TIME@TEMP ARRAY
0781 C       MVI C,4          LXI D, TIME@TEMP  INX D  INX D
0781 C       LXI H, TIME@DISPLAY  INX H
0781 C LOOP: MOV A,M          STAX D
0781 C       INX H            INX D
0781 C       XRA A            STAX D  INX D
0781 C       DCR C            JNZ LOOP

0781         INLINE / 0EH, 4,  11H, ADDRESS(TIME@TEMP), 13H, 13H /
0788         INLINE / 21H, ADDRESS(TIME@DISPLAY), 23H /
078C 43      INLINE / 7EH, 12H, 13H, 23H, 0AFH, 12H, 13H /
0793         INLINE / 0DH, 0C2H, ADDRESS(43) /
0797
0797         SAVE@TIME@DISPLAY = .TRUE.
079C         CALL GET@SET@TO@SET
079F
```

```
079F C ************************************************************

079F C                S W I T C H E S

079F C ************************************************************

079F C SCALE DECODE@INPUT (0 - 12H) BY ONE TO  1 - 13H
079F 45      B@TEMP = DECODED@INPUT +1
07A7

07A7 C ***************        SWITCHES        *********
07A7         IF ( B@TEMP .GT. 13H) GOTO 145
07B0       GO TO (50,55,60,65,70,75,80,85,90,95,100,105,105,115,120,125,
07B0      & 130,135,140) , B@TEMP

07BA         RETURN
07BB
07BB C.....***  RUN/STOP ( 0 ) ***.....
07BB C STOP IMMEDIATELY,  RUN ON THE NEXT TICK
07BB 50      IF ( RUN@ON@TICK) RETURN
07C0         IF ( RUN@STOP .EQ. 2 ) GO TO 52
07C8 C RUNNING > STOP
07C8 51      RUN@STOP = 2
07CD         RUN@ON@TICK = RESET@TEST@ON@TICK =  .FALSE.
07D4         GO TO 265
07D7
07D7 C STOP >>  RUN
07D7 C SCREEN FOR TIMER=0 AND STARTING A TIMED TEST:
07D7 C               IF SO, RETURN AND DON'T START THE TEST
07D7 52      CALL TIME@ZERO?(TIMER@START@TIME,ZERO@FLAG)
07EC         IF }}TEST@TYPE .GT. 2~ .AND. }ZERO@FLAG~~ RETURN
07FA C ** IF TIME@MODE DESELECTED IT WILL GO TO ELAPSED ON TEST START
```

```
07FA          RUN@ON@TICK = .TRUE.
07FF          RESET@TEST@ON@TICK =  RESET@TIMER@NEXT@RUN
0805          RESET@TIMER@NEXT@RUN = .FALSE.
080A C ** BE SURE TIME DISPLAY IS NOT HUNG UP WITH SETTING PROTECT
080A          GO TO 143
080D
080D C.....*** BIT ERROR (1) ***.....
080D 55       ERROR@TYPE = 1
0812          GO TO 67
0815

0815 C.....*** PARITY ERROR (2) ***.....
0815 60       ERROR@TYPE = 2
081A          GO TO 67
081D
081D C.....*** BIPOLAR VIOLATION (3) ***.....
081D 65       ERROR@TYPE = 4
0822 C * IF SELECTING  RROR@TYPE RESTART 10**7(8) ACCUMULATION!!!
0822 67       CALL RESET@7@8
0825 C BLANK IF DISPLAY IF ERROR@DISPLAY IS NOT READY IMMEDIATELY IN THIS MODE
0825          IF ( ERROR@MODE .GE. 8 )  RESULT@READY = 0F0H
0832          RETURN
0833

0833 C.....*** 0 OR 10-7 THRESH. (4) ***.....
0833 70       IF(THRESHOLD.GT.1) THRESHOLD=0
083E          THRESHOLD=THRESHOLD+1
0846          RETURN
0847
0847 C.....*** 10-6 OR 10-3 THRESH. (5) ***.....
0847 75       IF (THRESHOLD.NE.4) THRESHOLD=0
0854          THRESHOLD=THRESHOLD+4
085C          RETURN
085D
085D C.....*** TOT. OR % ERR. SEC(6) ***.....
085D 80       IF(ERROR@MODE.NE.1) ERROR@MODE=-30
086A          ERROR@MODE=ERROR@MODE+31
0872          RETURN
0873
0873 C.....*** TOT ERR OR BER (7) ***.....
0873 85       IF(ERROR@MODE.NE.2) ERROR@MODE=0
0880          ERROR@MODE=ERROR@MODE+2
0888          RETURN
0889
0889 C.....*** 10-7 OR 10-8 RATE (8) ***.....
0889 90       IF (ERROR@MODE.NE.8) ERROR@MODE=0
0896          ERROR@MODE=ERROR@MODE+8
089E C GO RESET 708 ACCUM AND BLANK DISPLAY
089E          GO TO 101
08A1
08A1 C.....*** PRINT CONTROL (9) ***.....
08A1 95       IF (PRINT@CONTROL.NE.1) PRINT@CONTROL=0
08AE          PRINT@CONTROL=PRINT@CONTROL+1
08B6          RETURN
08B7
08B7 C.....*** RESET ***
08B7 100      IF(RUN@STOP.EQ. 1) RETURN
08BD
```

```
08BD          BLINK@SIGNAL@LOSS =
08BD       &          BLINK@FRAME@LOSS = BLINK@PRBS@LOSS = .FALSE.
08C8
08C8 102      CALL ZERO@INTERNAL@ACCUMS
08CB
08CB C SET FLAG TO BLANK ERROR DISPLAY IN MAINLINE
08CB 101      RESULT@RE DY = 0F0H
08D0
08D0          CALL RESET@7@8
08D3          RETURN
08D4
08D4 C.....*** LOW (0BH) ***.....
08D4 C                *** DSX3 (0CH)***
08D4 105      IF (RUN@STOP.EQ.1) RETURN
08DA C     LXI H,ADD(DECODED@INPUT)  MVI A,0AH  SUB M  CMA  INR A
08DA C     STA ADD(INPUT@TYPE)
08DA          INLINE /21H,ADDRESS(DECODED@INPUT),3EH,0AH,96H,2FH,3CH/
08E2          INLINE /32H,ADDRESS(INPUT@TYPE)/
08E5          RETURN
08E6
08E6 C.....*** HORN ENABLE (0DH) ***.....
08E6 C       LXI H,ADD(HORN@ENABLE)  MOV A,M  CMA  ANI 1  MOV M,A
08E6 115      INLINE /21H,ADDRESS(HORN@ENABLE),7EH,2FH,0E6H,1,77H/
08EE          RETURN
08EF
08EF C.....*** MO/DA (0EH) ***.....
08EF C GOT SAVE THE TIME@MODE SELECTION TO RETURN TO WHEN DONE W/ MO-DA
08FF 120      IF ((TIME@MODE .AND. 0FH) .NE. 0) LAST@TIME@MODE = TIME@MODE
08FF          MO@DISPLAY@CTR = 3
0904          TIME@MODE = 0
0909          IF ((CLOCK@TASK.GT.1).OR.THIS@IS@SET) RETURN
0917          CLOCK@TASK = 5
091C          GO TO 143
091F
091F C.....***  TIME/DA (0FH) ***.....
091F 125      TIME@MODE = 1
0924          IF ((CLOCK@TASK.GT.1).OR.THIS@IS@SET) RETURN
0932          CLOCK@TASK = 4
0937          GO TO 143
093A

093A C.....*** ELAPSED (10H) ***.....
093A 130      TIME@MODE = 2
093F          GO TO 142
0942
0942 C.....*** TIMER SINGLE(11H) ***.....
0942 135      B@TEMP = 4
0947          GO TO 141
094A
094A C.....***   REPEAT TIMED (12H) ***.....
094A 140      B@TEMP = 8
094F
094F C CAN'T ACCESS TIMER IF RUNNING AND ELAPSED TEST
094F 141      IF ( RUN@STOP .EQ. 2) GO TO 144
0957 C ** RUNNING
0957          IF ( TEST@TYPE .LE. 2) RETURN
095E          TEST@TYPE = B@TEMP
0962 C OK TO SWITCH TEST@TYPE :
```

```
0962 C                REPEATED TO SINGLE AND VICE VERSA WHILE RUNNING
0962 144      TIME@MODE = B@TEMP
0968 C IF NO CLOCK TASK IN PROGRESS & NOT A SET -- BE SURE TIME@DISPLAY
0968 C                IF FREE TO BE UPDATED AFTER TIME@MODE SELECTION
0968 142      IF (( CLOCK@TASK.GT.1) .OR.THIS@IS@SET) RETURN
0976 143      SAVE@TIME@DISPLAY = .FALSE.
097B          RETURN
097C


097C C ***************************************************************************

097C C                M O R E    S W I T C H E S

097C C ***************************************************************************

097C 145      B@TEMP = DECODED@INPUT - 12H
0984          GO TO (146,150,186,186,186,186,186,186,186,186,190,147,190,205,
0984       &          210,215,215,147,147,241,241,245,250,255,267,265,270,
0984       &          275,280,147,147,285,285,285,285,285,285,305,305,310,
0984       &          310,320,320,
0984       &          320,320,330), B@TEMP

0990 147      RETURN
0991
0991 C ..................... ***  CLEAR FLAGS (STOP BLINKING LIGHTS) - C1 (13H)
0991 146      IF ( RUN@STOP.EQ.1) RETURN
0997          BLINK@SIGNAL@LOSS = BLINK@FRAME@LOSS = BLINK@PRBS@LOSS = .FALSE.
09A2          RETURN
09A3

09A3 C ..................... *** ZERO ACCUMULATORS - C2 (14H)
09A3 C NOTE: PROGRAM JUMPS TO 102 TO PROCESS A C2 WHICH IS A SUBSET OF F.P.RESET
09A3 150      IF ( RUN@STOP .EQ. 2) GO TO 102
09AB          RETURN
09AC

09AC C .........................*** PARALLEL POLL MASK ***.................
09AC C *** K1 (15H)  PARALLEL@POLL@MASK = 1
09AC C *** K2 (16H)  PARALLEL@POLL@MASK = 2
09AC C *** K3 (17H)  P RALLEL@POLL@MASK = 4
09AC C *** K4 (18H)  P RALLEL@POLL@MASK = 8
09AC C *** K5 (19H)  PARALLEL@POLL@MASK = 10H
09AC C *** K6 (1AH)  PARALLEL@POLL@MASK = 20H
09AC C *** K7 (1BH)  PARALLEL@POLL@MASK = 40H
09AC C *** K8 (1CH)  PARALLEL@POLL@MASK = 80H
09AC 186      IF ( .NOT. BUS@REMOTE@ENABLE  ) GO TO 300
09B4          PARALLEL@POLL@MASK = BIT@MASKS(DECODED@INPUT-14H)
09C4          RETURN
09C5
09C5 C ..................... *** CONTROL MEASUREMENT TRANSMISSION ***.........
09C5 C *** TRANSMIT NOTHING - XM1 (1DH)       TX@MODE = 0
09C5 C *** TRANSMIT ALL(TIMED TEST)/ANY(ELAPSED TEST) TOTALS
09C5 C                                  - XM2 (1FH)     TX@MODE = 2
09C5 190      TX@MODE =  DECODED@INPUT - 1DH
09CD          RETURN
09CE
```

```
09CE C ..................  *** REMOTE CONTROL TIMED TEST *** ......
09CE C *** STOP TEST REMOTE -- J1 (20H)
09CE 205      GO TO 51
09D1
09D1 C *** RUN TEST REMOTE(UNLESS ALREADY RUNNING) -- J2 (21H)
09D1 210      IF ((RUN@STOP.EQ.2).AND.(.NOT.RUN@ON@TICK)) GO TO 52
09E4          RETURN
09E5
09E5 C ..............  *** CONTROL TIME SENT WITH MEASUREMENT TRANSMISSION ***....

09E5 C *** SEND NO TIME - XT1 (22H)  TX@TIME = 0
09E5 C *** SEND REAL TIME - XT2 (23H)        TX@TIME = 1
09E5 215      IF (.NOT.MACHINE@FORMAT) GO TO 300
09ED          TX@TIME = DECODED@INPUT - 22H
09F5          RETURN
09F6

09F6 C.......... PRINT CONTROL..........
09F6 C *** PRINT PARTIAL --XL1  (26H)
09F6 C *** PRINT ALL     --XL2  (27H)
09F6 241      PRINT@CONTROL=28H-DECODED@INPUT
09FF          RETURN
0A00
0A00 C ..................  *** REQUEST FOR SINGLE TRANSMISSION
0A00 C *** REQUEST TIMES (28H) - Q3:
0A00 245      MESSAGE@NUMBER=5
0A05          RETURN
0A06
0A06 C *** DELAY ON CARRIAGE RETURN -- 29H
0A06 250      IF (.NOT.RS@232@THERE) GO TO 300
0A0E          RS@DELAY@FLAG = .TRUE.
0A13          RETURN
0A14 C     NO DELAY ON CARRIAGE RETURN -- 2AH
0A14 255      IF (.NOT.RS@232@THERE) GO TO 300
0A1C          RS@DELAY@FLAG = .FALSE.
0A21          RETURN
0A22
0A22 C    *** REQUEST ALL TOTAL MEASUREMENTS - Q2 (2CH)
0A22 265      CALL SAVE@TOTAL@BLOCK@W@INTERVAL@RATE
0A25          IF(Q@COMMAND@P.NE.0) GO TO 2651
0A2D          Q@COMMAND@P=2
0A32 2651     IF(Q@COMMAND@A.NE.0) RETURN
0A38          Q@COMMAND@A=2
0A3D          RETURN
0A3E 266      CALL SAVE@TOTAL@BLOCK@W@INTERVAL@RATE
0A41          IF(Q@COMMAND@A.NE.0) RETURN
0A47          Q@COMMAND@A=4
0A4C          RETURN
0A4D
0A4D C*** NEW Q COMMAND -Q4 REQEST SELECTED ERROR DISPLAY DATA
0A4D 267      CALL SAVE@TOTAL@BLOCK@W@INTERVAL@RATE
0A50          IF(Q@COMMAND@A.NE.0) RETURN
0A56          Q@COMMAND@A=8
0A5B          RETURN
0A5C
0A5C C *** REQUEST FLAGS - Q3 (2DH)
0A5C 270      IF ( .NOT. RS@232@THERE ) GO TO 300
0A64          MESSAGE@NUMBER = 6
0A69          RETURN
```

```
0A6A
0A6A C ..................  *** CONTROL LOCAL LOCKOUT FROM RS-232
0A6A C *** ALLOW LOCAL CONTROL - Y1 (2EH)
0A6A 275      IF (.NOT. RS@232@THERE ) GO TO 300
0A72          RS@LOCAL@LOCKOUT = .FALSE.
0A77          RETURN
0A78 C *** LOCKOUT LOCAL CONTROL - Y2 (2FH)
0A78 280      IF (.NOT. RS@232@THERE ) GO TO 300
0A82          RS@LOCAL@LOCKOUT = .TRUE.
0A85          RETURN
0A86
0A86 C    ERROR MODES FROM REMOTE  32H- 37H
0A86 285      ERROR@MODE=BIT@MASKS(DECODED@INPUT-31H)
0A96          RETURN
0A97 C** MACHINE FORMAT
0A97 C    MACHINE@FORMAT=TRUE --XF1  38H
0A97 C    MACHINE@FORMAT=FALSE  XF2  39H
0A97 305      MACHINE@FORMAT=.NOT.(38H-DECODED@INPUT)
0AA1          RETURN
0AA2 C** HORN@ENABLE
0AA2 C** HORN@ENABLE=1 XH1 3AH
0AA2 C** HORN@ENABLE=0 XH2 3BH
0AA2 310      HORN@ENABLE=3BH-DECODED@INPUT
0AAE          RETURN
0AAC C** THRESHOLD SETTING
0AAC C    THRESHOLD=1   0 THRESH  TH1 3CH
0AAC C    THRESHOLD=2   -7 THRESH TH2 3DH
0AAC C    THRESHOLD=4   -6 THRESH TH3 3EH
0AAC C    THRESHOLD=8   -3 THRESH TH4 3FH
0AAC 320      THRESHOLD=BIT@MASKS(DECODED@INPUT-3BH)
0ABC          RETURN
0ABD
0ABD C  BURST Q@COMMAND Q5  -- 40H
0ABD 330      CALL SAVE@TOTAL@BLOCK@W@INTERVAL@RATE
0AC0          IF(Q@COMMAND@A.EQ.0) Q@COMMAND@A=1
0ACD          RETURN
0ACE 300      CALL COMMAND@ERROR
0AD1          RETURN
0AD2
0AD2          END


PROGRAM  STORAGE: 1184

VARIABLE STORAGE: 0003

SYMBOL TABLE
0A32 2651
0ACE 300
0ABD 330
0AAC 320
0AA2 310
0A97 305
0A86 285
0A78 280
0A6A 275
0A5C 270
0A3E 266
```

```
0A4D 267
0A14 255
0A06 250
0A00 245
09F6 241
09E5 215
09D1 210
09CE 205
0990 147
09C5 190
09AC 186
09A3 150
0991 146
0962 144
094F 141
0968 142
08C8 102
08CB 101
0822 67
0976 143
07C8 51
07D7 52
094A 140
0942 135
093A 130
091F 125
08EF 120
08E6 115
08D4 105
08B7 100
08A1 95
0889 90
0873 85
085D 80
0847 75
0833 70
081D 65
0815 60
080D 55
07BB 50
097C 145
078C 43
0A22 265
076F 445
0756 41
0760 44
0715 39
074D 40
0773 42
0709 38
079F 45
06F9 37
3D7B ZERO@FLAG
3D7C B@TEMP
3D7D THIS@IS@SET

0AD2
0AD2 C ************************************************************
```

```
0AD2        SUBROUTINE        SYNTAX@ERROR
0AD2
0AD2 C ************************************************************
0AD2           CALL RESET@SYNTAX@FLAGS
0AD5           CALL COMMAND@ERROR
0AD8           RETURN
0AD9           END
```

PROGRAM STORAGE: 0007

VARIABLE STORAGE: 0000

SYMBOL TABLE

0AD9

D. REMOTE INPUT ROUTINE

```
ØAD9 C }}}}}}}}}}}}}}}}}}}}}}}}}}}} RX.FTN {{{{{{{{{{{{{{{{{{{{{{{{{{
ØAD9 C ************************************************************
ØAD9       SUBROUTINE        RX@PARSE
ØAD9
ØAD9 C ************************************************************
ØAD9         INTEGER*1         I,COMBO
ØAD9
ØAD9 C ** IF NEW CHARACTER IS NUMERIC BRANCH TO 6
ØAD9         INLINE / 3AH, ADDRESS(RXCHAR) /
ØADC C IF RXCHAR >= 3AH, NOT NUMERIC
ØADC         INLINE / ØFEH, 3AH, ØD2H, ADDRESS(1)/
ØAE1 C BUT IF >= 3AH & IF RXCHAR >= 30H, IT IS NUMERIC, GO TO 6
ØAE1         INLINE / ØFEH, 30H, ØD2H, ADDRESS(6)/
ØAE6
ØAE6 C ** COMMAS CAUSE A RESET OF ALPHA FLAGS
ØAE6 1       INLINE / ØFEH, ',', ØC2H, ADDRESS(2)/
ØAEB         CALL RESET@SYNTAX@FLAGS
ØAEE         RETURN
ØAEF
ØAEF C ** ALPHA CHARACTERS
ØAEF 2       IF(RX@NEED@ALPHA1.NE.Ø) GO TO 22
ØAF7         IF(ALPHA1.NE.'T') GO TO 22
ØAFF         IF(RXCHAR.NE.'H') GO TO 22
ØB07         ALPHA1=ØDAH
ØBØC         RETURN
ØBØD 22      IF ( RX@NEED@ALPHA1.NE.2) GOTO 5
ØB15
ØB15 C ** LOOKING FOR SECOND ALPHA CHARACTER OF DOUBLE LETTER SEQUENCE, IE XM1
ØB15 C       IF VALID 2ND CHAR PUT CODE FOR SEQENCE IN ALPHA1
ØB15 C       IF NOT VALID GO TO 18
ØB15 C       LDA RXCHAR
ØB15 C       MVI C,DBH        ; 5BH + 80H
ØB15 C       CPI 'T'   JZ  32
ØB15 C       INR C
ØB15 C       CPI 'M'   JZ  32
ØB15 C       INR C
ØB15 C       CPI 'L'   JZ 32
ØB15 C       INR C
ØB15 C       CPI 'F' J  32
ØB15 C       INR C
ØB15 C       CPI 'H' JNZ 18
ØB15 C 32    MOV A,C   STA ALPHA1
ØB15 C *** SET RX@NEED FLAG TO LOOK FOR NUMERIC CODE NEXT
ØB15 C       XRA A     STA RXNEED

ØB15         INLINE / 3AH, ADDRESS(RXCHAR)/
ØB18         INLINE / ØEH, ØDBH, ØFEH, 'T', ØCAH, ADDRESS(32) /
ØB1F         INLINE / ØCH, ØFEH, 'M', ØCAH, ADDRESS(32) /
ØB25         INLINE / ØCH, ØFEH, 'L', ØCAH, ADDRESS(32) /
ØB2B         INLINE / ØCH, ØFEH, 'F', ØCAH, ADDRESS(32) /
ØB31         INLINE / ØCH, ØFEH, 'H', ØC2H, ADDRESS(18) /
ØB37 32      INLINE / 79H, 32H, ADDRESS(ALPHA1)/
ØB3B         INLINE / ØAFH, 32H, ADDRESS(RX@NEED@ALPHA1)/
ØB3F         RETURN
ØB40
ØB40 C ************************************************************
```

```
0B47 C ** START OF NEW COMMAND, REGARDLESS IF ONE ALREADY STARTED
0B4Ø 5       ALPHA1 = RXCHAR
0B46         RX@NEED@ALPHA1 = Ø
0B4B C ** CHECK IF FIRST LETTER OF TWO LETTER SEQUENCE:
0B4B         IF (RXCHAR.EQ.'X') RX@NEED@ALPHA1 = 2
0B55         IF ( RXCHAR .NE. 'N') RETURN
0B5B
0B5B C ** CLEAR THE DISPLAY IF BEGINNING AN ENTER('N') COMMAND
0B5B         DECODED@INPUT = 8AH
0B6Ø         CALL BRANCH@ON@INPUT
0B63         RETURN
0P64
0B64 C ************************************************
0B64 C **DIGITS -- READY TO BRANCH ON LETTER/DIGIT SEQUENCE
0B64 C ** GOT AN ERROR IF EXPECTING AN ALPHA CHARACTER
0B64 6       IF ( RX@NEED@ALPHA1 .NE. Ø) GO TO 18
0B6C 9       IF (ALPHA1 .NE. 'N') GO TO 11
0B74
0B74 C ************************************************************************
0B74 C .......................   N  COMMAND   ..........................
0B74 C ADD SET BIT TO DIGIT
0B74         DECODED@INPUT = (RXCHAR-3ØH) .OR. 8ØH
0B7E         CALL BRANCH@ON@INPUT
0B81         RETURN
0B82
0B82 11      CALL RESET@SYNTAX@FLAGS
0B85         IF (ALPHA1 .NE. 'J') GO TO 12
0B8D
0B8D C ************************************************************
0B8D C ....................   J3 & J4  COMMANDS    ......................
0B8D         IF (RXCHAR.NE.'3') GO TO 103
0B95 C ** RUN STANDARD ELAPSED TEST
0B95         I=1
0B9A         GO TO 105
0B9D 103     IF (RXCHAR.NE.'4') GO TO 12
0BA5 C ** RUN STANDARD TIMED TEST
0BA5         I=3
0BAA 105     DO 106  I=I, 7
0BB2         DECODED@INPUT = J@HEX@SEQUENCE(I)
0BBD 106     CALL BRANCH@ON@INPUT
0BCA         RETURN
0BCB
0BCB C ************************************************************************
0BCB C .................  MAIN LOOK-UP TABLE  ..........................

0BCB 12      IF (( RXCHAR .GT. '8').OR.(RXCHAR.EQ.Ø))  GO TO 18
0BE2 C ** BE SURE OPERATOR DIDN'T ENTER THE 5BH-5DH USED FOR 2 LETTER SEQ
0BE2         IF ( ALPHA1 .LT. Ø ) GO TO 13
0BE8         IF } ALPHA1 .GT. |Y|~ GO TO 18
0BF1 C **  LDA ALPHA1  ORA A  JM 13  CPI 'Z'  JNC 18
0BF1 13      COMBO = ((ALPHA1.AND.7FH) * 8) + ( RXCHAR-31H)
0C06
0C06 C ** SEARCH THE COMMAND TABLE TO FIND THE ALPHA/NUMERIC COMBINATION
0C06 C       LXI H,COMMAND@TABLE  INX H  MVI B,64  LDA COMBO
0C06 C 14    CMP M   JZ 20
0C06 C       INX H  DCR B  JNZ  14

0C06         INLINE / 21H, ADDRESS(COMMAND@TABLE),23H/
```

```
0C0A            INLINE / 6, 64, 3AH, ADDRESS(COMBO) /
0C0F 14         INLINE / 0BEH, 0CAH, ADDRESS(20) /
0C13            INLINE / 23H, 5, 0C2H, ADDRESS(14) /
0C18
0C18 18         CALL SYNTAX@ERROR
0C1B            RETURN
0C1C
0C1C C ** USE COUNTER NUMBER TO FIGURE THE HEX CODE NOW THAT FOUND THE ASCII
0C1C C          IN TABLE
0C1C C          MVI A,65  SUB B
0C1C C          CPI 48  JZ  200
0C1C C          CPI 49  JNZ 21
0C1C C ** THE LAST TWO COMMANDS HAVE HEX CODES <> TO THEIR INDEX (SET CLOCKS)
0C1C C200        SUI 48  ADI 8EH

0C1C 20         INLINE / 3EH, 65, 90H/
0C1F            INLINE /0FEH, 48, 0CAH, ADDRESS(200)/
0C24            INLINE /0FEH,49, 0C2H,ADDRESS(21)/
0C29 200        INLINE / 0D6H,48,0C6H,8EH/
0C2D
0C2D 21         INLINE / 32H, ADDRESS(DECODED@INPUT)/
0C30            CALL BRANCH@ON@INPUT
0C33            RETURN
0C34            END


PROGRAM  STORAGE: 0347

VARIABLE STORAGE: 0002

SYMBOL TABLE
0C2D 21
0C29 200
0C1C 20
0C0F 14
0BF1 13
0BBD 106
0BAA 105
0B9D 103
0BCB 12
0B82 11
0B6C 9
0C18 18
0B37 32
0B40 5
0B0D 22
0AEF 2
0B64 6
0AE6 1
3D79 COMBO
3D7A I

0C34
```

E. ONCE A SECOND ROUTINE

```
0C34 C }}}}}}}}}}}}}}}}}}}}}}}} FILE:  COUNT.FTN {{{{{{{{{{{{{{{{{{{

0C34 C ************************************************************

0C34 C       O N E   S E C O N D   T I C K   R O U T I N E S  :

0C34 C ************************************************************


0C34 C ************************************************************

0C34       SUBROUTINE        BUMP@CLOCK
0C34
0C34 C ************************************************************

0C34 C INCREMEMT BCD NUMBER IN CLOCK REG BY 1 SEC, MAINTAINING
0C34 C        VALID TIME ENTRY USING MAX@CLOCK@DISPLAY
0C34 C NOTE:  ONLY LSB OF I*2 (4) CLOCK  ARE USED, MSB ARE ALWAYS 0.

0C34          INTEGER*1 HOUR
0C34
0C34 C THIS IS THE SAME ALGORITHM AS IN BUMP ELAPSED TIME
0C34 C        MVI C,3
0C34 C        LXI H, CLOCK
0C34 C        LXI D, MAX@CLOCK@DISPLAY
0C34 C  LOOP:INX H    INX H
0C34 C        INX D     INX D
0C34 C        LDAX D
0C34 C        CMP M
0C34 C        JNZ N2
0C34 C        XRA A             ; IF MAX VALUE CHANGE TO 0
0C34 C        MOV M,A
0C34 C        MVI A,1    CMP C  JNZ N0          ;HOURS=0??
0C34 C        MVI A,5  STA CLOCK@TASK ;NEW DATE
0C34 C    N0: MVI A,2    CMP C  JNZ N1          ;MINUTES=0??
0C34 C        MVI A,FFH  STA HOUR               ;HOUR ROLL-OVER
0C34 C    N1: DCR C      JNZ LOOP    JMP 275
0C34 C    N2: MVI A, 0FH
0C34 C        ANA M
0C34 C        CPI 9
0C34 C        MOV A,M
0C34 C        JNZ  N4
0C34 C        CPI      0F0H
0C34 C        JC       N3
0C34 C        ANI      0FH
0C34 C   N3:  ADI 6
0C34 C     N4: ADI 1
0C34 C        MOV M,A

0C34          HOUR = .FALSE.
0C39          INLINE / 0EH,3, 21H, ADDRESS(CLOCK)/
0C3E          INLINE /  11H, ADDRESS(MAX@CLOCK@DISPLAY) /
0C41
0C41 271      INLINE / 23H, 23H, 13H, 13H/
0C45          INLINE /1AH, 0BEH, 0C2H, ADDRESS(272)/
0C4A          INLINE / 0AFH, 77H/
```

```
0C4C
0C4C          INLINE / 3EH, 1, 0B9H, 0C2H, ADDRESS(222)/
0C52          INLINE / 3EH, 5, 32H, ADDRESS(CLOCK@TASK)/
0C57
0C57 222      INLINE /3EH, 2, 0B9H, 0C2H, ADDRESS(223)/
0C5D          INLINE / 3EH, 0FFH, 32H, ADDRESS(HOUR)/
0C62
0C62 223      INLINE / 0DH, 0C2H, ADDRESS(271)/
0C66          GO TO 275
0C69
0C69 272      INLINE / 3EH, 0FH, 0A6H, 0FEH, 9, 7EH, 0C2H, ADDRESS(274) /
0C72          INLINE / 0FEH, 0F3H, 0DAH, ADDRESS(273), 0E6H, 0FH /
0C79 273      INLINE / 0C6H, 6/
0C7B 274      INLINE / 0C6H, 1, 77H/
0C7E
0C7E C ** SEE IF HOUR ROLLOVER?? MAY NEED TO LOG TIME
0C7E 275      IF (RUN@STOP.EQ.2) RETURN
0C84          IF (.NOT. HOUR) RETURN
0C8A          IF (.NOT.LOG@MADE@IN@LAST@HOUR) MESSAGE@NUMBER=4
0C97          LOG@MADE@IN@LAST@HOUR=.FALSE.
0C9C          RETURN
0C9D          END


PROGRAM  STORAGE: 0105

VARIABLE STORAGE: 0001

SYMBOL TABLE
0C79 273
0C7B 274
0C7E 275
0C62 223
0C57 222
0C69 272
0C41 271
3D78 HOUR

0C9D
0C9D C ************************************************************

0C9D        SUBROUTINE        BUMP@ELAPSED@TIME
0C9D
0C9D C ************************************************************

0C9D C INCREMEMT BCD NUMBER IN ELAPSED TIME BY 1 SEC, MAINTAINING
0C9D C        VALID TIME ENTRY USING MAX@TIME@DISPLAY
0C9D C NOTE:  ONLY LSB OF I 2 }4  ELAPSED@TIME  ARE USED, MSB ARE ALWAYS 0.


0C9D C        INTEGER 1 I
0C9D C        DO 260 I=1,4
0C9D C        IF (ELAPSED@TIME(I).EQ.MAX@TIME@DISPLAY(I)) GO TO 260
0C9D C        IF ((ELAPSED@TIME(I).AND. 0FH) .EQ. 9)
0C9D C     &          ELAPSED@TIME(I) = ELAPSED@TIME(I) + 6
0C9D C        ELAPSED@TIME(I) = ELAPSED@TIME(I) + 1
0C9D C        RETURN
0C9D C ADJUST FOR TIME ARITHMETIC-- I.E. 59 SEC + 1 = 00 SE
```

```
0C9D C260     ELAPSED@TIME(I) = 0
0C9D C270      RETURN

0C9D C ASSEMBLY VERSION FOR SPEED, SAME LOGIC AS FORTRAN ABOVE
0C9D C        MVI C,4
0C9D C        LXI H, ELAPSED@TIME
0C9D C        LXI D, MAX@TIME
0C9D C  LOOP:INX H    INX H
0C9D C        INX D     INX D
0C9D C        LDAX D
0C9D C        CMP M
0C9D C        JNZ N2
0C9D C        XRA A             ; IF MAX VALUE CHANGE TO 0
0C9D C        MOV M,A
0C9D C        DCR C      JNZ LOOP     RET
0C9D C    N2: MVI A, 0FH
0C9D C        ANA M
0C9D C        CPI 9
0C9D C        MOV A,M
0C9D C        JNZ  N4
0C9D C        ADI 6
0C9D C     N4: ADI 1
0C9D C        MOV M,A
0C9D C        RET

0C9D          INLINE / 0EH,4, 21H, ADDRESS(ELAPSED@TIME)/
0CA2          INLINE /  11H, ADDRESS(MAX@TIME@DISPLAY) /
0CA5
0CA5 271      INLINE / 23H, 23H, 13H, 13H/
0CA9          INLINE /1AH, 0BEH, 0C2H, ADDRESS(272)/
0CAE          INLINE /  AFH, 77H, 0DH, 0C2H, ADDRESS(271)/
0CB4          RETURN
0CB5
0CB5 272      INLINE / 3EH, 0FH, 0A6H, 0FEH, 9, 7EH, 0C2H, ADDRESS(274) /
0CBE          INLINE / 0C6H, 6/
0CC0 274      INLINE / 0C6H, 1, 77H/
0CC3
0CC3          RETURN
0CC4
0CC4          END


PROGRAM  STORAGE: 0039

VARIABLE STORAGE: 0000

SYMBOL TABLE
0CC0 274
0CB5 272
0CA5 271

0CC4



0CC4 C ************************************************
```

```
0CC4          SUBROUTINE COUNT@DOWN@TIMER
0CC4
0CC4 C ************************************************

0CC4 C DECREMENT THE TIMER BY ONE SECOND, MAINTAINING VALID
0CC4 C        TIME VALUE
0CC4 C        ( I.E. 2 MINUTES, 0 SECONDS  MINUS 1
0CC4 C        EQUALS  1 MINUTE, 59 SECONDS)
0CC4 C RETURN FALSE UNLESS TIMER DECREMENTS TO ZERO


0CC4 C FORTRAN VERSION:
0CC4 C        INTEGER*2 I

0CC4 C        DO 220 I = 1,4
0CC4 C         IF(TIMER(I).NE.0) GO TO 200
0CC4 C220     TIMER(I)=MAX@TIME@DISPLAY(I)

0CC4 C200      IF((TIMER(I).AND.0FH).EQ.0)  TIMER(I) = TIMER(I) - 6
0CC4 C        TIMER(I)=TIMER(I)-1

0CC4 C
0CC4 C ASSEMBLY VERSION TO SAVE SPACE
0CC4 C        MVI C,4
0CC4 C        LXI H, MAX@TIME
0CC4 C        LXI D, TIMER
0CC4 C LOOP:  INX H   INX H
0CC4 C        INX D   INX D
0CC4 C        LDAX D
0CC4 C        ORA A
0CC4 C        JNZ  L2
0CC4 C        MOV A,M          ;IF BYTE=0, SET IT = MAX & GO DCR THE NEXT BYTE
0CC4 C        STAX D
0CC4 C        DCR C       JNZ LOOP     RETURN

0CC4 C  L2:   ANI 0FH          ;ELSE, DCR W/ A DEC. ADJUST & ITS DONE
0CC4 C        LDAX D
0CC4 C        JNZ  L3
0CC4 C        SUI 6
0CC4 C  L3:   SUI 1
0CC4 C        STAX D


0CC4          INLINE / 0EH,4,  21H, ADDRESS(MAX@TIME@DISPLAY) /
0CC9          INLINE / 11H,  ADDRESS(TIMER) /
0CCC
0CCC 221      INLINE / 13H, 13H, 23H, 23H /
0CD0          INLINE / 1AH, 0B7H, 0C2H, ADDRESS(223)/
0CD5          INLINE / 7EH, 12H, 0DH, 0C2H, ADDRESS(221)/
0CDB
0CDB          RETURN
0CDC
0CDC 223      INLINE / 0E6H, 0FH, 1AH, 0C2H, ADDRESS(224) /
0CE2          INLINE / 0D6H, 6/
0CE4 224      INLINE / 0D6H, 1, 12H /
0CE7
0CE7          RETURN
0CE8          END
```

```
PROGRAM  STORAGE: 0036

VARIABLE STORAGE: 0000

SYMBOL TABLE
0CE4 224
0CDC 223
0CCC 221

0CE8 C*****************************************************************

0CE8          SUBROUTIN  BUMP@ASCII@TIMER
0CE8
0CE8 C*****************************************************************
0CE8 C     FORTRAN ROUTINE REPLACED BY ASSEMBLY
0CE8 C        INTEGER*1 I
0CE8 C        I=1
0CE8 C10      ASCII@TIMER(I)=ASCII@TIMER(I).OR.30H
0CE8 C        ASCII@TIMER(I)=ASCII@TIMER(I)+1
0CE8 C        IF (ASCII@TIMER(I).NE.3AH) RETURN
0CE8 C        ASCII@TIMER(I)=30H
0CE8 C        I=I+1
0CE8 C        GO TO 10
0CE8 C        RETURN
0CE8 C     ASSEMBLY ROUTINE TO GO FASTER AND SAVE SPACE
0CE8 C     LXI H,ADDRESS(ASCII@TIMER) INX H  MVI A,7
0CE8 C10   PUSH PSW  MOV A,M  ORI 30H  INR A  MOV M,A  CPI 3AH
0CE8 C     JNZ 20  MVI M,30H  INX H  POP PSW  DCR A  JNZ 10
0CE8 C     RET
0CE8 C20   POP PSW   RET
0CE8          INLINE /21H,ADDRESS(ASCII@TIMER), 23H,3EH,7/
0CEE 10       INLINE /0F5H,7EH,0F6H,30H,3CH,77H,0FEH,3AH/
0CF6          INLINE /0C2H,ADDRESS(20),36H,30H,23H,0F1H,3DH/
0CFF          INLINE /0C2H,ADDRESS(10),0C9H/
0D02 20       INLINE /0F1H/
0D03          RETURN
0D04          END


PROGRAM  STORAGE: 0028

VARIABLE STORAGE: 0000

SYMBOL TABLE
0D02 20
0CEE 10

0D04 C *****************************************************************

0D04        SUBROUTINE        UPDATE@TIME
0D04
0D04 C *****************************************************************

0D04 C THIS ROUTINE USES THE TIME@MODE SELECTION TO PUT THE TIME@DISPLAY
0D04 C        INTO THE COMMON FOR THE FRONT PANEL UPDATE DRIVER
0D04 C        FIRST IF RUN=TRUE, THE ELASPED TIME IS INCREMENTED AND
```

```
0D04 C        IF THIS IS A TIMED TEST THE TIMER IS DECREMENTED
0D04 C        IF TIMER=0 OF A SINGLE TIMED TEST RUN@STOP IS SET FALSE
0D04 C        WHILE IF TIMER=0 OF REPEATED SET A FLAG TO RESTART IN THE MAIN ONE
0D04 C               S COND ROUTINE
0D04          INTEGER*1 ZERO@FLAG
0D04
0D04          CALL BUMP@CLOCK
0D07          IF (SAVE@TIME@DISPLAY) RETURN
0D0C          IF(RUN@STOP.EQ.2) GOTO 330
0D14 C IF RUNNING INCREMENT ELAPSED TIME AND DECREMENT TIMER IF TIMED TEST
0D14          CALL BUMP@ELAPSED@TIME
0D17          CALL BUMP@ASCII@TIMER
0D1A          CALL ADD@ACCUM(ONE,TOTAL@SECONDS)
0D29 C .....................................................
0D29 C  BRANCH ON TIME@MODE SELECTED
0D29 330      IF (TIME@MODE .NE. 0) GO TO 334
0D31 C /* MONTH-DAY
0D31 C                TEMPORARY DISPLAY TIMES OUT W/ CTR
0D31 C                W EN CTR=0 THEN RESTORE PREVIOUS MODE
0D31          IF (MO@DISPLAY@CTR .NE. 0) GO TO 331
0D39          TIME@MODE = LAST@TIME@MODE
0D3F          LAST@TIME@MODE = 1
0D44          GO TO 330
0D47 331      MO@DISPLAY@CTR = MO@DISPLAY@CTR -1
0D4F          RETURN
0D50 334      IF ( TIME@MODE .NE. 1 ) GO TO 336
0D58 C /* TIME OF DAY
0D58          CALL MOVE@TIME@TO@DIS(CLOCK)
0D61          RETURN
0D62 336      IF ( TIME@MODE .NE. 2 ) GO TO 338
0D6A C /* ELAPSED
0D6A          CALL MOVE@TIME@TO@DIS}ELAPSED@TIME~
0D73          RETURN
0D74 338      IF (TIME@MODE .GT. 8) RETURN
0D7B C /* TIMER
0D7B          CALL MOVE@TIME@TO@DIS(TIMER)
0D84          RETURN
0D85          END


PROGRAM  STORAGE: 0129

VARIABLE STORAGE: 0001

SYMBOL TABLE
0D74 338
0D62 336
0D47 331
0D50 334
0D29 330
3D77 ZERO@FLAG

0D85
0D85 C*************************************************************************

0D85          SUBROUTINE      UPDATE@TEST
0D85
0D85 C*************************************************************************
```

```
0D85          INTEGER*1  ZERO@FLAG
0D85
0D85          IF(TEST@TYPE .LE. 2)  RETURN
0D8C C IF A TIMED MEASUREMENT
0D8C          CALL COUNT@DOWN@TIMER
0D8F          CALL TIME@ZERO?(TIMER,ZERO@FLAG)
0DA4          IF (.NOT. ZERO@FLAG) RETURN
0DAA C ..........................................................
0DAA C IF TIMER IS NOW = 0 THEN END OF THIS MEASUREMENT
0DAA          RUN@STOP=2
0DAF          CALL SAVE@TOTAL@BLOCK@W@INTERVAL@RATE
0DB2          IF(Q@COMMAND@A.EQ.0) Q@COMMAND@A=2
0DBF          IF(Q@COMMAND@P.EQ.0) Q@COMMAND@P=2
0DCC          IF( TEST@TYPE .EQ. 8) GO TO 320
0DD4 C     SINGLE TIMER MODE - TERMINATES
0DD4 C  CLEAR TIME MODE FLAG FOR MEASUREMENT
0DD4          TEST@TYPE =  2
0DD9          RETURN
0DDA C  REPEAT TIME MODE - SET FLAG TO RESET
0DDA 320      RESET@TEST@ON@TICK = RUN@ON@TICK = .TRUE.
0DE2
0DE2          RETURN
0DE3          END


PROGRAM  STORAGE: 0094

VARIABLE STORAGE: 0001

SYMBOL TABLE
0DDA 320
3D76 ZERO@FLAG

0DE3 C***********************************************************

0DE3          SUBROUTINE      SET@RESULT (RES)
0DE3
0DE3 C***********************************************************

0DE3          INTEGER*1  RES(5)
0DE3
0DE3 C  ASSEMBLY USED TO SPEED UP
0DE3 C        LXI H,ADD(ETABLE) LDA ADD(ERROR@TYPE)  MOV E,A MVI D,0
0DE3 C        DAD D    MOV C,M  LXI H,ADD(TTABLE)  LDA ADD(THRESHOLD)
0DE3 C        MOV E,A  DAD D   MOV A,M  ADD C  MOV E,A
0DE3 C        LHLD ADD(RES)  DAD D  SHLD ADD(RES)
0DE3          INLINE /21H,ADDRESS(ETABLE),3AH,ADDRESS(ERROR@TYPE),5FH,16H,0/
0DFC          INLINE /19H,4EH,21H,ADDRESS(TTABLE),3AH,ADDRESS(THRESHOLD)/
0DF4          INLINE /5FH,19H,7EH,81H,5FH/
0DF9          INLINE /2AH,ADDRESS(RES),19H,22H,ADDRESS(RES)/
0E00
0E00          CALL MOVE@ACCUM(RES,RESULT)
0E0F          RETURN
0E10          END


PROGRAM  STORAGE: 0045
```

```
VARIABLE STORAGE: 0004

SYMBOL TABLE
3D72 RES

0E10 C ************************************************************

0E10        SUBROUTINE         TOTAL@ERRORS
0E10
0E10 C ************************************************************

0E10           RESULT@READY=10H
0E15           CALL SET@RESULT(TOTAL@BIPOLAR@VIOLATIONS@T7)
0E1D           RETURN
0E1E           END


PROGRAM  STORAGE: 0014

VARIABLE STORAGE: 0000

SYMBOL TABLE

0E1E


0E1E C ************************************************************

0E1E C ************************************************************

0E1F        SUBROUTINE         CUMULATIVE@RATE
0E1E
0E1E C ************************************************************

0E1E           IF(.NOT.((RESET@TEST@ON@TICK).OR.(RUN@STOP.EQ.2)))
0E2A       $           GO TO 729
0E2F           CALL MOVE@ACCUM(TOTAL@BITS,RESULT@DENOM)
0E3C           RESULT@READY=6
0E41           CALL SET@RESULT(TOTAL@BIPOLAR@VIOLATIONS@T7)
0E49           RETURN
0E4A 729       CALL MOVE@ACCUM(TOTAL@THIS@SEC@ACC,RESULT@DENOM)
0E58           RESULT@READY=6
0E5D C  ASSEMBLY VERSION TO SPEED UP AND SAVE SPACE
0E5D C
0E5D C     LXI H,ADD(THIS@SEC@BP@VIOL)  LXI D,ADD(BP@SEC@FLAG@T7)
0E5D C     LXI B,26  INX H  LDA ADD(ERROR@TYPE) RLC  RLC  RLC  RLC  RLC
0E5D C10   RLC  JC 20  DAD B  XCHG   DAD B  XCHG  JMP 10
0E5D C20   LDA ADD(THRESHOLD)
0E5D C     RRC  JC 50
0E5D C30   RRC  JC 40  INX D  JMP 30
0E5D C40   LDAX D  ANA A  JNZ 50  LXI H,ADD(ZEROS)  INX H
0E5D C50   LXI D,ADD(RESULT) INX D  MVI C,5
0E5D C60   MOV A,M  STAX D  INX D  INX H  DCR C JNZ 60
0E5D C     RET

0E5D           INLINE /21H,ADDRESS(THIS@SEC@BIPOLAR@VIOLATIONS)/
0E60           INLINE /11H,ADDRESS(BP@SEC@FLAG@T7),1,26,0,23H/
```

```
0E67          INLINE /3AH,ADDRESS(ERROR@TYPE),7,7,7,7,7/
0E6F 10       INLINE /7,0DAH,ADDRESS(20),9,0EBH,9,0EBH/
0E77          INLINE /0C3H,ADDRESS(10)/
0E7A 20       INLINE /3AH,ADDRESS(THRESHOLD),0FH,0DAH,ADDRESS(50)/
0E81 30       INLINE /0FH,0DAH,ADDRESS(40),13H,0C3H,ADDRESS(30)/
0E89 40       INLINE /1AH,0A7H,0C2H,ADDRESS(50),21H,ADDRESS(ZEROS),23H/
0E92 50       INLINE /11H,ADDRESS(RESULT),13H,0EH,5/
0E98 60       INLINE /7EH,12H,23H,13H,0DH,0C2H,ADDRESS(60)/
0EA0          RETURN
0EA1
0EA1          END


PROGRAM  STORAGE: 0131

VARIABLE STORAGE: 0000

SYMBOL TABLE
0E98 60
0E89 40
0E81 30
0E92 50
0E7A 20
0E6F 10
0E4A 729

0EA1
0EA1 C ***************************************************************

0EA1        SUBROUTINE         ERROR@SECONDS
0EA1
0EA1 C ***************************************************************

0EA1          RESULT@READY=10H
0EA6          CALL SET@RESULT(BIPOLAR@VIOLATION@SECS@T7)
0EAE          RETURN
0EAF          END


PROGRAM  STORAGE: 0014

VARIABLE STORAGE: 0000

SYMBOL TABLE

0EAF
0EAF C***************************************************************

0EAF          SUBROUTINE       PERCENT@ERROR@SECONDS
0EAF
0EAF C***************************************************************
0EAF          CALL MOVE@ACCUM(TOTAL@SECONDS,RESULT@DENOM)
0EBD          RESULT@READY=-2
0EC2          CALL SET@RESULT(BIPOLAR@VIOLATION@SECS@T7)
0ECA          RETURN
0ECB          END
```

```
UTI FORT//80 COMPILER  .3A1                    PAGE 0060

PROGRAM  STORAGE: 0028

VARIABLE STORAGE: 0000

SYMBOL TABLE

0ECB C ************************************************************

0ECB           SUBROUTINE     DUMMY@ALL@ERRORS@DURING@SIGNAL@LOSS
0ECB
0ECB C        DUMMY BIT ERRORS WITH 4.7 MILLION AND PARITY ERRORS WITH 9398
0ECB C        DURING SIGNAL LOSS  AND RUNNING
0ECB C        ALSO, SET ERROR SECONDS FLAGS

0ECB          CALL MOVE@ACCUM(DUMMY@ALL@BIT@ERRORS,THIS@SEC@BIT@ERRORS)
0EDA          CALL MOVE@ACCUM(DUMMY@ALL@PARITY@ERRORS,THIS@SEC@PARITY@ERRORS)
0EE9
0EE9          CALL ADD@ACCUM(DUMMY@ALL@BIT@ERRORS,TOTAL@BIT@ERRORS)
0EF8          CALL ADD@ACCUM(DUMMY@ALL@PARITY@ERRORS,TOTAL@PARITY@ERRORS)
0F07          BER@SEC@FLAG =
0F07       &           PAR@SEC@FLAG = .TRUE.
0F0F          CALL ADD@ACCUM(T6@CONS@1,TOTAL@BITS)
0F1E          CALL MOVE@ACCUM(T6@CONS@1,TOTAL@THIS@SEC@ACC)
0F2D          RETURN
0F2E          END


PROGRAM  STORAGE: 0099

VARIABLE STORAGE: 0000

SYMBOL TABLE

0F2E
0F2E C ************************************************************

0F2E           SUBROUTINE     UPDATE@THIS@INT@ACCUMS(OLD,NEW)
0F2E
0F2E C ************************************************************
0F2E          INTEGER*1 OLD(5), NEW(5)
0F2E C IF NEW IS BIGGE  THAN OLD THEN OLD = NEW
0F2E C ( ACCUMS ARE STORED W LSB FIRST, WANT TO COMPARE MSB FIRST )

0F2E C LXI B,6    LHLD NEW   DAD B  XCHG   LHLD OLD   DAD B
0F2E C MAKE POINTERS TO MSB'S,  REG C IS CTR
0F2E C 100:    DCR C  RZ  DCX H  DCX D  LDAX D  CMP M  JZ 100  RC

0F2E          INLINE /01H, 6,0, 2AH,ADDRESS(NEW), 09,0EBH, 2AH,ADDRESS(OLD)/
0F39          INLINE / 09 /
0F3A 100      INLINE / 0DH, 0C8H, 1BH,2BH,1AH, 0BEH, 0CAH, ADDRESS(100)/
0F43          INLINE / 0D8H /
0F44 110      CALL MOVE@ACCUM(NEW,OLD)
0F53          RETURN
0F54          END


PROGRAM  STORAGE: 0038
```

```
VARIABLE STORAGE: 0006

SYMBOL TABLE
0F44 110
0F3A 100
3D6C NEW
3D6E OLD

0F54 C*****************************************************************

0F54          SUBPOUTINE ACCUM@GE@THR?
0F54
0F54 C*****************************************************************

0F54 C  THIS ROUTINE WRITTEN TO FIT THE CALL IN "UPDATE@ERROR@SECONDS".
0F54 C  IT WAS DONE THIS WAY TO SAVE SPACE AND TIME. THE ROUTINE MUST
0F54 C  BE ENTERED WITH THE POINER TO THE MOST SIG. BYTE OF THE THRESHOLD
0F54 C  IN B&C AND THE POINTER TO THE MOST SIG. BYTE OF THE ACCUMULATOR IN
0F54 C  H&L. THE FLAG TO INDICATE THE DECISION IS RETURNED IN THE  A REG.
0F54 C   FFH-->= THAN THRESHOLD   00--< THAN THRESHOLD
0F54 C
0F54 C    MVI E,5  MVI D,0FFH
0F54 C10  LDAX B  CMP M  JC 13  JNZ 12  DCX H  DCX B  DCR E
0F54 C .  JNZ 10
0F54 C    RET
0F54 C12  MVI D,0
0F54 C13  DCX B
0F54 C    DCR E  JNZ 13  RET

0F54          INLINE /1EH,5,16H,0FFH/
0F58 10       INLINE /0AH,0BEH,0DAH,ADDRESS(13),0C2H,ADDRESS(12),2BH,0BH,1DH/
0F63          INLINE /0C2H,ADDRESS(10)/
0F66          RETURN
0F67 12       INLINE /16H,0/
0F69 13       INLINE /0BH,1DH,0C2H,ADDRESS(13)/
0F6E          RETURN
0F6F          END


PROGRAM  STORAGE: 0027

VARIABLE STORAGE: 0000

SYMBOL TABLE
0F67 12
0F69 13
0F58 10

0F6F
0F6F C*****************************************************************

0F6F          SUBROUTINE      UPDATE@ERROR@SECONDS
0F6F
0F6F C*****************************************************************

0F6F C  THIS ROUTINE WAS REWRITTEN TO SAVE SPACE. IT UPDATES ALL THE ERROR
0F6F C  SECONDS ACCUMULATORS BY FIRST CHECKING THE THRESHOLDS AND THEN BASED
0F6F C  ON THE "SECOND FLAGS" IT INCREMENTS THE ERROR SECONDS ACCUMS.
```

```
ØF6F C     LXI H,ADD(BIPOLAR@VIOLATIONS)  LXI D,ADD(BP@SEC@FLAG@T7)
ØF6F C     MVI A,3
ØF6F C10   PUSH PSW  PUSH D   LXI B,ADD(ZEROS)  LDA ADD(TOTAL@THIS@SECOND)
ØF6F C     CPI 45  JNZ 15  LXI B,ADD(T3@CONS@1)
ØF6F C15   MVI A,3
ØF6F C16   PUSH PSW  PUSH H  PUSH D

ØF6F         INLINE /21H,ADDRESS(BIPOLAR@VIOLATIONS)/
ØF72         INLINE /11H,ADDRESS(BP@SEC@FLAG@T7),3EH,3/
ØF77 10      INLINE /ØF5H,ØD5H,1,ADDRESS(ZEROS)/
ØF7C         INLINE /3AH,ADDRESS(TOTAL@THIS@SECOND)/
ØF7F         INLINE /ØFEH,45,ØC2H,ADDRESS(15),1,ADDRESS(T3@CONS@1)/
ØF87 15      INLINE /3EH,3/
ØF89 16      INLINE /ØF5H,ØE5H,ØD5H/
ØF8C
ØF8C         CALL ACCUM@GE@THR?
ØF8F
ØF8F C    MOV A,D  POP D  POP H  STAX D  INX D  POP PSW  DCR A
ØF8F C    JNZ 16  POP D  LXI B,26  DAD B  XCHG  DAD B  XCHG
ØF8F C    POP PSW  DCR A  JNZ 10

ØF8F         INLINE /7AH,ØD1H,ØE1H,12H,13H,ØF1H,3DH/
ØF96         INLINE /ØC2H,ADDRESS(16),ØD1H,1,26,Ø,9,ØEBH,9,ØEBH/
ØFA1         INLINE /ØF1H,3DH,ØC2H,ADDRESS(10)/
ØFA6
ØFA6 C  NOW FINISHED WITH SETTING THRESHOLD FLAGS - MUST GO ON TO UPDATE
ØFA6 C  ERROR SECONDS REGISTER INCLUDING THRESHOLD "Ø".

ØFA6 C    LXI H,ADD(BIPOLAR@VIOLATION@SECS@T7)
ØFA6 C    LXI D,ADD(BP SEC@FLAG@7)  MVI A,3
ØFA6 C17  PUSH PSW  MVI A,4
ØFA6 C18  PUSH PSW  LDAX D  ANA A  CNZ 30  INX D
ØFA6 C    LXI B,5  DAD B  POP PSW  DCR A
ØFA6 C    JNZ 18  LXI B,20  DAD B  XCHG  LXI B,22  DAD B  XCHG
ØFA6 C    POP PSW  DCR A  JNZ 17  RET
ØFA6 C
ØFA6 C30  PUSH H  STC  MVI C,5
ØFA6 C31  INX H  MVI A,Ø  ADC M  MOV M,A
ØFA6 C    DCR C  JNZ 31  POP H RET

ØFA6         INLINE /21H,ADDRESS(BIPOLAR@VIOLATION@SECS@T7)/
ØFA9         INLINE /11H,ADDRESS(BP@SEC@FLAG@T7),3EH,3/
ØFAE 17      INLINE /ØF5H,3EH,4/
ØFB1 18      INLINE /ØF5H,1AH,ØA7H,ØC4H,ADDRESS(30),13H/
ØFB8         INLINE /1,5,Ø,9,ØF1H,3DH/
ØFBE         INLINE /ØC2H,ADDRESS(18),1,20,Ø,9,ØEBH,1,22,Ø,9,ØEBH/
ØFCB         INLINE /ØF1H,3DH,ØC2H,ADDRESS(17)/
ØFD0         RETURN
ØFD1 30      INLINE /ØE5H,37H,ØEH,5/
ØFD5 31      INLINE /2 H,3EH,2,8EH,77H,ØDH,ØC2H,ADDRESS(31)/
ØFDE         INLINE /ØE1H/
ØFDF         RETURN
ØFE0
ØFE0         END
```

PROGRAM STORAGE: 0113

VARIABLE STORAGE: 0000

SYMBOL TABLE
0FD5 31
0FD1 30
0FB1 18
0FAE 17
0F89 16
0F87 15
0F77 10

```
0FE0
0FE0 C*****************************************************************

0FE0          SUBROUTINE UPDATE@TOTAL@ERRORS(DUM1,DUM2)
0FE0
0FE0 C*****************************************************************

0FE0          INTEGER*1  DUM1(5),DUM2(5)
0FE0
0FE0 C    ROUTINE TO SPEED UP LOADING THE THRESHOLD TOTAL ERROR ACCUMS.

0FE0 C    LXI H,ADD(TOTAL@BIPOLAR@VIOLATIONS@T7)
0FE0 C    LXI D,ADD(THIS@SEC@BIPOLAR@VIOLATIONS)
0FE0 C    LXI B,ADD(BP@SEC@FLAG@T7)  MVI A,3
0FE0 C3   PUSH PSW   PUSH H  PUSH B  PUSH D  XCHG  SHLD ADD(DUM1)
0FE0 C    XCHG  MVI A,3
0FE0 C5   PUSH PSW  SHLD ADD(DUM2)  PUSH D  PUSH H  PUSH B
0FE0 C    LDAX B  ANA A  JZ 10

0FE0          INLINE /21H,ADDRESS(TOTAL@BIPOLAR@VIOLATIONS@T7)/
0FE3          INLINE /11H,ADDRESS(THIS@SEC@BIPOLAR@VIOLATIONS)/
0FE6          INLINE /1,ADDRESS(BP@SEC@FLAG@T7),3EH,3/
0FEB 3        INLINE /0F5H,0E5H,0C5H,0D5H,0EBH,22H,ADDRESS(DUM1)/
0FF3          INLINE /0EBH,3EH,3/
0FF6 5        INLINE /0F5H,22H,ADDRESS(DUM2),0D5H,0E5H,0C5H/
0FFD          INLINE /0AH,0A7H,0CAH,ADDRESS(10)/
1002
1002          CALL ADD@ACCUM(DUM1,DUM2)
1011
1011 C10  POP B  INX B  POP H  LXI D,5  DAD D  POP D
1011 C    POP PSW  DCR A  JNZ 5  POP D  LXI H,26  DAD D
1011 C    XCHG  POP B  LXI H,26  DAD B  MOV B,H  MOV C,L  POP H
1011 C    PUSH B  LXI B,40   DAD B  POP B  POP PSW  DCR A  JNZ 3  RET

1011 10       INLINE /0C1H,3,0E1H,11H,5,0,19H,0D1H/
1019          INLINE /0F1H,3DH,0C2H,ADDRESS(5),0D1H,21H,26,0,19H/
1023          INLINE /0EBH,0C1H,21H,26,0,9,44H,4DH,0E1H/
102C          INLINE /0C5H,1,40,0,9,0C1H,0F1H,3DH,0C2H,ADDRESS(3)/
1037          RETURN
1038

1038          END
```

PROGRAM STORAGE: 0088

```
VARIABLE STORAGE: 0008

SYMBOL TABLE
1011 10
0FF6 5
0FEB 3
3D64 DUM2
3D68 DUM1

1038 C ************************************************************

1038          SUBROUTINE     ELAPSED@TEST@DATA@TO@PERIPH
1038
1038 C ************************************************************
1038 C /* DETERMINE IF ERRORS SHOULD BE LOGGED, 1 IF NEW ERROR(S) OCCURED
1038 C                2 IF OLD ERROR(S) NOT YET LOGGED
1038 C                3 IF 15 MINUTE INTERVAL ON LOGGING IS IN EFFECT
1038 C                4 ( NO LOGGING DURING SIGNAL OR FRAME LOSS )
1038 C                5 COUNT A CONSECUTIVE ERROR IF A NEW ERROR
1038 C /* UPDATE THE MAXIMUM ERRORS IN A SECOND DURING LOG INTERVAL




1038 C /* BLOCK THE LOG DURING LOSSES, ALSO DON'T COUNT AS LOG
1038 737      IF (PRBS@LOSS.AND.FRAME@LOSS) GO TO 999
1043
1043 C MASK OUT BIT ERRORS IF LIVE TRAFFIC:
1043          CALL ADJUST@NEW@ERROR@MASK
1046
1046 C BUMP NUMBER SEQUENTIAL LOGS IF NEW ERRORS
1046 C ERRORS TO PRINT = NEW ERRORS .OR. ERRORS TO PRINT
1046 C ERRORS TO AUTO = NEW ERRORS .OR. ERRORS TO AUTO
1046 C NEW ERRORS = ERRORS TO AUTO .OR. ERRORS TO PRINT

1046 C        LXI     D, NEW@ERROR@MASK
1046 C        LDAX    D
1046 C        ORA     A       JZ      101
1046 C        LXI     H,NUMBER@SEQUENTIAL@LOGS
1046 C        INR     M
1046 C  101   LXI     H, UNLOGGED@ERRORS@TO@PRINT
1046 C        ORA     M
1046 C        MOV     M,A     MOV     B,A
1046 C        LXI     H, UNLOGGED@ERRORS@TO@AUTO
1046 C        LDAX    D
1046 C        ORA     M
1046 C        MOV     M,A
1046 C        ORA     B
1046 C        STAX    D
1046 C        JZ      999

1046          INLINE / 11H, ADDRESS(NEW@ERROR@MASK)/
1049          INLINE /  1AH, 0B7H, 0CAH, ADDRESS(101) /
104E
104E          INLINE / 21H, ADDRESS(NUMBER@SEQUENTIAL@LOGS), 34H/
1052
1052 101      INLINE / 21H, ADDRESS(UNLOGGED@ERRORS@TO@PRINT)/
```

```
1055          INLINE / 0B6H, 77H, 47H /
1058
1058          INLINE / 21H, ADDRESS(UNLOGGED@ERRORS@TO@AUTO)/
105B          INLINE / 1AH, 0B6H, 77H /
105E          INLINE / 0B0H, 12H, 0CAH, ADDRESS(999) /
1063
1063          IF(COUNTE @15@MIN.EQ.0) GO TO 850
106F          COUNTER@15@MIN=COUNTER@15@MIN-1
1079          IF(COUNTE @15@MIN.NE.0) RETURN
1083 900      IF(Q@COMMAND@A.EQ.0) Q@COMMAND@A=2
1090          IF(Q@COMMAND@P.EQ.0) Q@COMMAND@P=2
109D          CALL SAVE@TOTAL@BLOCK@W@CUMULATIVE@RATE@A
10A0          CALL SAVE@TOTAL@BLOCK@W@CUMULATIVE@RATE@P
10A3          GO TO 998
10A6


10A6 850      IF((HORN.EQ.0).AND.(PRINT@CONTROL.EQ.2)) RETURN
10BB C /* TRY TO LOG
10BB 799      CALL SAVE@TOTAL@BLOCK@W@INTERVAL@RATE
10BE 801      IF (NUMBER@SEQUENTIAL@LOGS.LT.40) RETURN
10C4 C /* HAVE HAD 40 CONSECUTIVE ERROR SECONDS, THEREFORE SET 15 MIN CTR
10C4 998      COUNTER@15@MIN=900
10CA 999      NUMBER@SEQUENTIAL@LOGS = 0
10CF          RETURN
10D0          END


PROGRAM  STORAGE: 0152

VARIABLE STORAGE: 0000

SYMBOL TABLE
10BE 801
10BB 799
10C4 998
1083 900
10A6 850
1052 101
10CA 999
1038 737

10D0
10D0 C*************************************************************

10D0          SUBROUTINE  SET@HORN
10D0
10D0 C*************************************************************

10D0 C    THIS ASSEMBLY ROUTINE SETS THE HORN. THIS ROUTINE DOES NOT
10D0 C   TAKE INTO ACCOUNT THE HORN ENABLE .

10D0 C        XRA A  STA ADDRESS(HORN)  LXI H,ADD(BP@SEC@FLAG@T7)
10D0 C        LXI D,26  LDA ADD(THRESHOLD)  RRC  JNC 10  ORI 08H
10D0 C10      RRC   JC 12   INX H  JMP 10
10D0 C12      MOV C,M  DAD D   MOV A,M  ORA C  MOV C,A  DAD D
10D0 C        MOV A,M  ORA C  RZ  MVI A,1  STA ADD(HORN)  RET
10D0          INLINE  /0AFH,32H,ADDRESS(HORN),21H,ADDRESS(BP@SEC@FLAG@T7)/
```

```
10D7          INLINE  /11H,26,0,3AH,ADDRESS(THRESHOLD),0FH,0F6H,08H/
10E0 10       INLINE  /0FH,0DAH,ADDRESS(12),23H,0C3H,ADDRESS(10)/
10E8 12       INLINE  /4EH,19H,7EH,0B1H,4FH,19H/
10EE          INLINE  /7EH,0B1H,0C8H,3EH,1,32H,ADDRESS(HORN)/
10F6          RETURN
10F7          END


PROGRAM  STORAGE: 0039

VARIABLE STORAGE: 0000

SYMBOL TABLE
10E8 12
10E0 10

10F7 C ************************************************************

10F7           SUBROUTINE     ONCE@PER@SEC
10F7
10F7 C ************************************************************

10F7 C THINGS THAT GET DONE ON A ONE SECOND TICK:
10F7 C       1. KEEP TIME IF RUNNING &
10F7 C              EITHER SET A CLOCK TASK OR UPDATE THE TIME@DISPLAY
10F7 C              DIRECTLY WITH TIMER OR ELAPSED TIME
10F7 C       2. IF RUNNING UPDATE ERROR SECOND ACCUMS
10F7 C       3. PULL ERROR SECONDS, TOTAL ERRORS OR CUMULATIVE RATE ACCUMS
10F7 C              FOR CONVERSION IN THE MAIN LINE OF THE PROGRAM
10F7 C       4. START A NEW TEST IF RUN@ON@TICK IS SET



10F7 C ..............................................................
10F7 C  KEEP THE TIME
10F7          CALL UPDATE@TIME
10FA C  TRANSFER THE SECOND TOTAL BIT COUNT
10FA C  LXI H,ADD(TOTAL@THIS@SEC@ACC), INX H, LDA ADD(TOTAL@THIS@SECOND)
10FA C  MOV M,A
10FA          INLINE /21H,ADDRESS(TOTAL@THIS@SEC@ACC),23H/
10FE          INLINE /3AH,ADDRESS(TOTAL@THIS@SECOND),77H/
1102 7402     IF ( RUN@STOP .EQ. 2) GO TO 741
110A
110A          IF ( SIGNAL@LOSS)  CALL DUMMY@ALL@ERRORS@DURING@SIGNAL@LOSS
1111

1111 C /* UPDATE THE ERROR SECOND COUNTERS
1111 7401     CALL CHECK@MIL@INT
1114
1114          CALL UPDATE@ERROR@SECONDS
1117          CALL UPDATE@TOTAL@ERRORS(ZEROS,TOTAL@BIPOLAR@VIOLATIONS)
1126          CALL SET@HORN
1129          BER@SEC@FLAG=0
112E          PAR@SEC@FLAG=0
1132          BP@SEC@FLAG=0
1137
1137          CALL CHECK@MIL@INT
113A          CALL UPDATE@TEST
```

```
113D          IF(RUN@STOP.EQ.2) GO TO 7409
1145
1145          IF (UNLOGGED@ERRORS@TO@PRINT.EQ.0)
1145       &         CALL CLEAR@THIS@INT@ACCUMS@PRT
114D          IF (UNLOGGED@ERRORS@TO@AUTO.EQ.0)
114D       &         CALL CLEAR@THIS@INT@ACCUMS@AUTO
1155
1155          CALL UPDATE@THIS@INT@ACCUMS
1155       &   (THIS@INT@PARITY@ERRORS@A,THIS@SEC@PARITY@ERRORS)
1164          CALL UPDATE@THIS@INT@ACCUMS
1164       &   (THIS@INT@PARITY@ERRORS@P,THIS@SEC@PARITY@ERRORS)
1173          CALL UPDATE@THIS@INT@ACCUMS
1173       &   (THIS@INT@BIT@ERRORS@A,THIS@SEC@BIT@ERRORS)
1181          CALL UPDATE@THIS@INT@ACCUMS
1181       &   (THIS@INT@BIT@ERRORS@P,THIS@SEC@BIT@ERRORS)
118F          CALL UPDATE@THIS@INT@ACCUMS
118F       &  (THIS@INT@BIPOLAR@VIOLATIONS@A,
1195       &   THIS@SEC@BIPOLAR@VIOLATIONS)
119D          CALL UPDATE@THIS@INT@ACCUMS
119D       &  (THIS@INT@BIPOLAR@VIOLATIONS@P,
11A3       &   THIS@SEC@BIPOLAR@VIOLATIONS)
11AB          CALL CHECK@MIL@INT
11AE
11AE C     ASSEMBLY VERSION TO SET THE INTERVAL FLAGS FOR THE
11AE C     THRESHOLDS. EACH ERROR TYPE HAS ONE BYTE WITH BITS
11AE C     SET TO INDICATE WHICH THRESHOLD HAS BEEN BROKEN IN THE
11AE C     LAST PRINT INTERVAL.
11AE C       LXI D,ADD(BP@SEC@FLAG@T7) LXI H,ADD(BP@INT@P)
11AE C       MVI A,3
11AE C5      PUSH PSW   PUSH D
11AE C       PUSH H  MOV B,M  INX H  MOV C,M  MVI L,8  MVI A,3
11AE C10     PUSH PSW  LDAX D  ANA L  ORA B   MOV B,A  LDAX D  ANA L
11AE C       ORA C  MOV C,A  MOV A,L  RRC  MOV L,A  INX D  POP PSW
11AE C       DCR A  JNZ 10  POP H  MVI A,1  ORA B  MOV M,A MVI A,1
11AE C       ORA C INX H  MOV M,A  DCX H  POP D  LXI B,26  DAD B   XCHG
11AE C       DAD B  XCHG  POP PSW  DCR A  JNZ 5   RET
11AE          INLINE /1 H,ADDRESS(BP@SEC@FLAG@T7),21H,ADDRESS(BP@INT@P)/
11B4          INLINE /3EH,3/
11B6 5        INLINE /0F5H,0D5H,0E5H,46H,23H,4EH,2EH,8,3EH,3/
11C0 10       INLINE /0F5H,1AH,0A5H,0B0H,47H,1AH,0A5H,0E1H,4FH,7DH,0FE/
11CB          INLINE /6FH,13H,0F1H,3DH,0C2H,ADDRESS(10),0E1H,3EH,1/
11D5          INLINE /0B0H,77H,3EH,1,0E1H,23H,77H,2BH,0D1H,1,26,0/
11E1          INLINE /9,0EBH,9,0EBH,0F1H,3DH,0C2H,ADDRESS(5)/
11EA


11FA          IF ( TEST@TYPE.EQ.2) GO TO 7405
11F2 C ON TIMED TEST AT LEAST 1 HOUR LONG, OUTPUT INTERMEDIATE LOGS
11F2 C        IF ((TIMER@START@TIME(4).OR.TIMER@START@TIME(3)).EQ.0) GO TO 741
11F2 C        LXI H,TIMER@START@TIME  LXI B,6  DAD B
11F2 C        MOV A,M  INX H  INX H  ORA M -- ONLY LSB OF TIME REG ARE SIGNIFICAN

11F2 C        JZ 741
11F2          INLINE/ 21H,ADDRESS(TIMER@START@TIME), 1,6,0,9 /
11F9          INLINE/ 7EH, 23H,23H,0B6H, 0CAH, ADDRESS(741) /
1200 7405     CALL CHECK@MIL@INT
1203          IF(SFLAG.EQ.0) GO TO 7406
120B          SFLAG=SFLAG-1
1213          IF(SFLAG.EQ.0) GO TO 7410
```

```
121B 7406     CALL ELAPSED@TEST@DATA@TO@PERIPH
121E
121E C ..........................................................
121E C /* SELECT ON THE TIME DEPENDENT ERROR MODE OPERATOR HAS SELECTED, IF ANY
121E C        PUT THE ACCUMULATOR CONTENTS IN RESULT TO BE CONVERTED


121E C ..........................................................
121E 741      IF ( RUN@ON@TICK ) MESSAGE@NUMBER = 3
122A C /* AVOID OUTPUTTING MESSAGE IF REPEATING A TIMED TEST
122A 7409     CALL CHECK@MIL@INT
122D          IF (ERROR@MODE.EQ.32) CALL PERCENT@ERROR@SECONDS
1235          IF (ERROR@MODE.EQ.1) CALL ERROR@SECONDS
123D          IF (ERROR@MODE.EQ.2) CALL TOTAL@ERRORS
1245          IF (ERROR@MODE.EQ.4) CALL CUMULATIVE@RATE
124D 7410     CALL ZERO@ACCUM(THIS@SEC@PARITY@ERRORS)
1256          CALL ZERO@ACCUM(THIS@SEC@BIPOLAR@VIOLATIONS)
125F          CALL ZERO@ACCUM(THIS@SEC@BIT@ERRORS)
1268          NEW@ERROR@MASK=0
126D          TOTAL@THIS@SECOND=0
1272          HORN=HORN.AND.HORN@ENABLE
127A          UPDATE@FRONT@PANEL = .TRUE.
127E          IF (.NOT. RUN@ON@TICK) RETURN
1284
1284 C ..........................................................
1284 C /* START A NEW TEST
1284          RUN@ON@TICK = .FALSE.
1286          RUN@STOP = 1
1289          FURST@LETTER=ERROR@DESIGNATOR(ERROR@TYPE)
1296 C INITIALIZE FLAGS TO TELL IF LOSS OF FRAME DURING RUNNING SHOULD>BLINK
1296          PRBS@OBTAINED = .NOT. PRBS@LOSS
129D          FRAME@OBTAINED = .NOT. FRAME@LOSS
12A4          COUNTER@15@MIN =
12A4       &           NUMBER@SEQUENTIAL@LOGS =
12A4       &           UNLOGGED@ERRORS@TO@AUTO =
12A4       &           UNLOGGED@ERRORS@TO@PRINT =
12A4       &                   LOSS@REPORT@CTR =
12A4       &           BER@SYNC@CTR=
12A4       &           PAR@SYNC@CTR=
12A4       &           BP@SYNC@CTR=
12A4       &           SFLAG=0
12C5
12C5 C START A TEST WITH TIME@MODE REFLECTING THE TEST@TYPE
12C5          TIME@MODE = TEST@TYPE
12CB
12CB          IF (.NOT.RESET@TEST@ON@TICK) RETURN
12D1
12D1 C /*  SET TIMER FOR A NEW TEST
12D1          CALL PUT@TIME(TIMER@START@TIME,TIMER)
12DF          CALL ZERO@TIME@REG(ELAPSED@TIME)
12E8          CALL ZERO@INTERNAL@ACCUMS
12EF          RESET@TEST@ON@TICK = .FALSE.
12F2
12F7          RETURN
12F1          END
```

PROGRAM STORAGE: 0506

F. MILLION-BIT ROUTINE

VARIABLE STORAGE: 0000

SYMBOL TABLE
124D 7410
121B 7406
1200 7405
11C0 10
11B6 5
122A 7409
1111 7401
121E 741
1102 740

```
12F1
12F1 C*****************************************************

12F1 C       O N E   M I L L I O N   B I T S   R O U T I N E S   :

12F1 C*****************************************************

12F1 C*****************************************************

12F1       SUBROUTINE        UPDATE@BLOCK(NEW,I@TYPE,N@TOT,STMP,
12F1      &                                          I@SECFLG,N@SECTOT)
12F1
12F1 C*****************************************************

12F1 C       (POINTERS TO ACCUMS FOR ONE OF ERROR@TYPES ARE PASSED)
12F1 C       (TAKES THREE CALLS TO UPDATE ACCUMS FOR ALL THREE)
12F1 C RETURN IF NEW COUNTS = ZERO !!
12F1 C ELSE ALWAYS
12F1 C IF RUNNING
12F1 C SERVICE TOTAL COUNTS ACCUMULATOR AND SET SECFLAG = TRUE

12F1         INTEGER*1 NEW(5),  I@TYPE, N@SECTOT(5)
12F1         INTEGER*1 N@TOT(5), I@SECFLG ,STMP(5)
12F1         INTEGER*1 LIM@FLAG
12F1
12F1 C       IF(SCTR.NE.0) GO TO 330
12F1         INLINE /2AH,ADDRESS(STMP),7EH,0A7H,0C2H,ADDRESS(330)/
12F9
12F9 C /* COMPARE NEW COUNTS TO ONE
12F9         CALL ACCUM@GE@LIM?(NEW,1,LIM@FLAG)
1313         IF (.NOT.LIM@FLAG)  RETURN
1319 C /* RETURN IF NEW = 0


1319 330     IF (RUN@STOP .EQ. 2~ RETURN
131F         CALL ADD@ACCUM(NEW,N@TOT)
132F         IF(.NOT.ASYNC) GO TO 335
1336         CALL ADD@ACCUM(NEW,N@SECTOT)
1345
1345 C /* GOT AT LEAST ONE ERROR THIS SEC
1345 331     I@SECFLG = .TRUE.
134A         NEW@ERROR@MASK=NEW@ERROR@MASK.OR.I@TYPE
1352         RETURN
1353
```

```
1353                RETURN
1354 335       CALL ADD@ CCUM(NEW,STMP)
1363 C         SCTR=SCTR+1+INT@CTR
1363 C         IF(SCTR.LT.45) RETURN
1363 C         SCTR=0
1363           INLINE /2AH,ADDRESS(STMP),3AH,ADDRESS(INT@CTR)/
1369           INLINE /B6H,3CH,77H/
136C           INLINE /0FEH,45,0D3H,36H,0/
1371           IF(SFLAG.EQ.0) SFLAG=2
137E           CALL MOVE@ACCUM(STMP,N@SECTOT)
138D           CALL ZERO@ACCUM(STMP)
1396           GO TO 331
1399           END


PROGRAM  STORAGE: 0168

VARIABLE STORAGE: 0015

SYMBOL TABLE
1345 331
1354 335
1319 330
3D55 LIM@FLAG
3D56 N@SECTOT
3D58 I@SECFLG
3D5A STMP
3D5E N@TOT
3D60 I@TYPE
3D62 NEW

1399
1399 C***********************************************

1399        SUBROUTINE          SERVICE@BLOCK@ACCUMS
1399
1399 C***********************************************************

1399 C UPDATE THE TOTAL ERRORS AND ERROR SEC FLAG IF RUNNING &
1399 C UPDATE THE ERROR-RATE-100-ERRORS ACCUMS FOR EACH ERROR@TYPE
1399 C IF ERROR-RATE-100-ERRORS IS SELECTED AS MEASUREMENT MODE,
1399 C        PUT THE RATE OF ERROR@TYPE SELECTED IN THE ERROR@DISPLAY

1399           CALL UPDATE@BLOCK (BIT@ERRORS,
139F       &            1, TOTAL@BIT@ERRORS,
13AA       &            BER@SEC@TEMP,BER@SEC@FLAG,
13B0       &           THIS@SEC@BIT@ERRORS )
13C4
13C4           CALL UPDATE@BLOCK (PARITY@ERRORS,
13CA       &            2,
13CA       &           TOTAL@PARITY@ERRORS,
13D5       &           PAR@SEC@TEMP,PAR@SEC@FLAG,
13DB       &           THIS@SEC@PARITY@ERRORS)
13EF
13EF           CALL UPDATE@BLOCK (BIPOLAR@VIOLATIONS,4,
13F5       &           TOTAL@BIPOLAR@VIOLATIONS,
13FF       &           BP@SEC@TEMP,BP@SEC@FLAG,
```

```
1405      &          THIS@SEC@BIPOLAR@VIOLATIONS )
141A
141A          IF (RUN@STOP .EQ. 2) GO TO 6
1422 2        CALL ADD@ACCUM(ONE,TOTAL@BITS)
1431          TOTAL@THIS@SECOND=TOTAL@THIS@SECOND+1
1439          IF(INT@CTR.EQ.0) GO TO 5
1441          INT@CTR=INT@CTR-1
1449          GO TO 2
144C 5        IF(.NOT.ASYNC) TOTAL@THIS@SECOND=45
1459 6        INT@CTR=0
145E               RETURN
145F          END


PROGRAM  STORAGE: 0198

VARIABLE STORAGE: 0000

SYMBOL TABLE
144C 5
1422 2
1459 6

145F C*******************************************************

145F        SUBROUTINE         SERVICE@TEN@7@8@ACCUM
145F
145F C*******************************************************

145F C ERROR@MODE SELECTED IS 10*7 OR 10*8
145F C ADD TO THE 10**7 (**8) ACCUMULATOR OF THE ERROR@TYPE SELECTED

145F          GO TO (360,361,370,362), ERROR@TYPE

146B 370      RETURN
146C 360      CALL ADD@ACCUM (BIT@ERRORS,ERRORS@TEN@7@8)
147B          GO TO 363
147E
147E 361      CALL ADD@ACCUM (PARITY@ERRORS,ERRORS@TEN@7@8)
148D          GO TO 363
1492
1490 362      CALL ADD@ACCUM (BIPOLAR@VIOLATIONS,ERRORS@TEN@7@8)
149E
149E 363      TEN@7@8@CTR=TEN@7@8@CTR-1-INT@CTR
14AA          IF(TEN@7@8@CTR.GT.0) RETURN
14B1
14B1 C ENOUGH BITS TO COMPUTE A RATE
14B1          CALL MOVE@ACCUM (ERRORS@TEN@7@8,RESULT)
14BE
14BF          IF (ERROR@MODE .EQ. 10H ) GO TO 366
14C6 C /* ERROR RATE / 10**7
14C6          CALL MOVE@ACCUM (I@TEN,RESULT@DENOM)
14D5          GO TO 368
14D8 C /* ERROR RATE / 10**8
14D8 366      CALL MOVE@ACCUM (I@HUNDRED,RESULT@DENOM)
14E7
14E7 368      RESULT@READY = 6
14EC          CALL RESET@7@8
```

```
14EF C        CALL INSTANTANEOUS@TO@PERIPH?

14EF          RETURN
14F0          END


PROGRAM  STORAGE: 0153

VARIABLE STORAGE: 0000

SYMBOL TABLE
14E7 368
14D8 366
149E 363
1490 362
146B 370
147E 361
146C 360

14F0
14F0 C*************************************************************

14F0        SUBROUTINE          MOVE@CTR@TEMP@TO@ACCUM  (FP@DES)
14F0
14F0 C*************************************************************
14F0 C MOVE CTR@TEMP CONTENTS TO I*5 ACCUM

14F0          INTEGER*1         FP@DES(5)
14F0
14F0 C /* FIRST ZERO THE ENTIRE ARRAY
14F0          CALL ZERO@ACCUM (FP@DES)
14F9
14F9 C /* THEN PUT CTR@TEMP IN LOWEST 2 BYTES

14F9 C LHLD CTR@TEMP
14F9 C XCHG  LHLD FPDES
14F9 C INX H  MOV M,L
14F9 C INX H  MOV M,H

14F9          INLINE / 2AH, ADDRESS(CTR@TEMP)/
14FC          INLINE / 0EBH, 2AH, ADDRESS(FP@DES) /
1500          INLINE / 23H, 73H, 23H, 72H /
1504
1504          RETURN
1505          END


PROGRAM  STORAGE: 0021

VARIABLE STORAGE: 0004

SYMBOL TABLE
3D51 FP@DES

1505 C*************************************************************

1505        SUPROUTINE          GET@ERROR@COUNTS
1505
```

```
1505 C*****************************************************

1505 C PUT NEW COUNTS OF EACH TYPE IN 5 BYTE ARRAY

1505 C THE NEW COUNTS ARE THE DIFFERENCE BETWEEN THIS CTR READING AND THE LAST
1505 C BER@CTR HAS 12 BITS, PAR@CTR HAS 8 BITS, BP@CTR HAS 12 BITS

1505 C AS NEEDED,  DUMMY THE COUNTS INSTEAD (BIT: 500,000  PARITY: 105 )


1505 C.....................................................
1505 C BIT ERRORS
1505 383      IF(PRBS@LOSS) GO TO 384
150C          PRBS@OBTAINED = .TRUE.
1511          IF ((CTR@TEMP= (BER@CTR - LST@BER@CTR)) .LT.0)
1521       &           C R@TEMP = CTR@TEMP + 1000H
1534          CALL MOVE@CTR@TEMP@TO@ACCUM(BIT@ERRORS)
153D          GO TO 385
1540
1540 C DUMMY BIT ERRORS
1540 384      CALL MOVE@ACCUM(DUMMY@BIT@ERRORS,BIT@ERRORS)
154F
154F C.....................................................
154F C PARITY ERRORS
154F 385      IF(UNFRAMED) GO TO 387
1556          IF(FRAME@LOSS) GO TO 390
155D          FRAME@OBTAINED = .TRUE.
1562 387      IF ( (CTR@TEMP =(PAR@CTR - LST@PAR@CTR)) .LT. 0)
1572       &           CTR@TEMP = CTR@TEMP + 100H
1585          CALL MOVE@CTR@TEMP@TO@ACCUM(PARITY@ERRORS)
158F          GO TO 386
1591
1591 C DUMMY PARITY ERRORS
1591 390      CALL MOVE@ACCUM}DUMMY@PARITY@ERRORS,PARITY@ERRORS)
15A0
15A0 C.....................................................
15A0 C BIPOLAR VIOLATIONS
15A0 386      IF ( (CTR@TEMP =(BP@CTR - LST@BP@CTR) ).LT. 0 )
15B0       &            CTR@TEMP = CTR@TEMP + 1000H
15C3          CALL MOVE@CTR@TEMP@TO@ACCUM(BIPOLAR@VIOLATIONS)
15CC
15CC          RETURN
15CD          END


PROGRAM  STORAGE: 0200

VARIABLE STORAGE: 0000

SYMBOL TABLE
15A0 386
1591 390
1562 387
154F 385
1540 384
1505 383

15CD
```

```
15CD C*******************************************************

15CD       SUBROUTINE        TEN@6@BITS
15CD
15CD C****************** ************************************


15CD C               ONE MILLION BITS INTERRUPT ROUTINE

15CD C CALLED BY ASSEMBLER INTERRUPT HANDLER AFTER ONE MILLION BITS ARE
15CD C       RECEIVED
15CD C COUNTER VALUES MUST HAVE BEEN INSERTED INTO COMMON VARIABLES:
15CD C       BP@CTR, BER@CTR, PAR@CTR



15CD C IF TESTING FRONT PANEL DON'T CHANGE IT

15CD C IF SIGNAL LOSS, THIS INTERRUPT ONLY OCCURRS BECAUSE OF NOISE, IGNORE IT
15CD         IF ( SIGNAL@LOSS) GOTO 389
15D4


15D4 C COUNTING

15D4 C IF FIRST COUNTER READING -- NO COUNTS -- SAVE READING
15D4 381        IF( .NOT.SIGNAL@OBTAINED) GO TO 387
15DC
15DC C SEE HOW MANY COUNTS FOR EACH ERROR TYPE:
15DC         CALL GET@ERROR@COUNTS
15DF

15DF C NOW UPDATE INTERNAL ACCUMULATORS WITH NEW ERRORS:

15DF C UPDATE BLOCKS OF ACCUMS FOR EACH ERROR TYPE:

15DF C UPDATE THE 10**7 OR 10**8 ACCUM IF THAT MODE IS SELECTED
15DF C       IF((ERROR@MODE .EQ. 8) .OR. (ERROR@MODE .EQ. 10H))
15DF C     & CALL SERVICE@TEN@7@8@ACCUM
15DF C LDA ERROR@MODE,  ANI  18H, CNZ SERVICE
15DF         INLINE / 3AH, ADDRESS(ERROR@MODE),0E6H, 18H/
15E4         INLINE / 0C4H, ADDRESS(SERVICE@TEN@7@8@ACCUM) /
15E7         CALL SERVICE@BLOCK@ACCUMS
15EA
15EA 387     SIGNAL@OBTAINED = .TRUE.
15EF
15EF C ALWAYS  SAVE THIS READING FOR NEXT TIME
15EF 389     LST@BER@CTR     =       BER@CTR
15F5         LST@PAR@CTR     =       PAR@CTR
15FB         LST@BP@CTR      =       BP@CTR
1601
1601         RETURN
1602         END
```

```
PROGRAM  STORAGE: 0053

VARIABLE STORAGE: 0000

SYMBOL TABLE
15EA 387
15D4 381
15EF 389

1602 C*****************************************************************

1602          SUBROUTINE      BURST@DATA
1602
1602 C*****************************************************************
1602          INTEGER*1 I
1602
1602          IF(RUN@STOP.EQ.2) RETURN
1608
1608          I=BURST@COUNT-1
1610          IF(BURST@LENGTH.EQ.0) GO TO 20
1618
1618          IF(BURST@LENGTH.GT.10) GO TO 10
1621 5        CALL ADD@ACCUM(ONE,BURST@1@10)
1630          GO TO 40
1633 10       IF(BURST@LENGTH.GT.99) GO TO 20
163C          CALL ADD@ACCUM(ONE,BURST@11@99)
164B          GO TO 40
164E 20       CALL ADD@ACCUM(ONE,BURST@GT@100)
165D 40       IF(I.EQ.0) RETURN
1663          I=I-1
166F          GO TO 5
166E          END


PROGRAM  STORAGE: 0108

VARIABLE STORAGE: 0001

SYMBOL TABLE
165D 40
1621 5
1633 10
164F 20
3D50 I

166E
```

G. FORMATTING ROUTINES

```
166E C }}}}}}}}}}}}}}}}}}}}}}}}}}} FORMAT.FTN {{{{{{{{{{{{{{{{{{{{

166E C ***************************************************************************

166E          SUBROUTINE      PSERV
166E
166E C ***************************************************************************
166E C        IF NOT PRINTER@THERE SEE IF PRINTER'S BEEN TURNED ON,
166E C                ELSE SEE IF A CHAR TO SEND
166E          INLINE / 0C3H, 33H, 0/
1671          END


PROGRAM  STORAGE: 0003

VARIABLE STORAGE: 0000

SYMBOL TABLE

1671 C ***************************************************************************

1671          SUBROUTINE      PRINT
1671
1671 C ***************************************************************************
1671 C  ROUTINE PLACES BUFFER@INSERT@PTR IN RING BUFFER SLOT AND THEN
1671 C  ADJUSTS THE BUFFER PARAMETER BLOCK.

1671 C    LHLD ADD(BUFFER@INSERT@PTR)  INX H  XCHG  LHLD ADD(RBUFFER@IN@P)
1671 C    MOV M,E  INX H  MOV M,D  INX H  SHLD ADD(RBUFFER@IN@P) PUSH H
1671 C    LXI H,ADD(NO@POSITIONS@P) DCR M  LHLD ADD(BUFFER@STRT@P)
1671 C    MOV B,H  MOV C,L  LDA ADD(BUFFER@TOTAL@P)  RLC  MOV E,A
1671 C    MVI D,0  DAD D  MOV A,L  POP H  CMP L RNZ  MOV H,B
1671 C    MOV L,C  SHLD ADD(RBUFFER@IN@P)  RET

1671          INLINE /2AH,ADDRESS(BUFFER@INSERT@PTR),23H,0EBH/
1676          INLINE /2AH,ADDRESS(RBUFFER@IN@P),73H,23H,72H,23H/
167D          INLINE /22H,ADDRESS(RBUFFER@IN@P),0E5H/
1681          INLINE /21H,ADDRESS(NO@POSITIONS@P),35H/
1685          INLINE /2AH,ADDRESS(RBUFFER@STRT@P),44H,4DH/
168A          INLINE /3AH,ADDRESS(BUFFER@TOTAL@P),87H,5FH,16H,0/
1691          INLINE /19E,7DH,0E1H,0BDH,0C2H,ADDRESS(4),60H,69H/
169A          INLINE /22H,ADDRESS(RBUFFER@IN@P)/
169D 4        CALL PSERV
16A0          LOG@MADE@IN@LAST@HOUR = .TRUE.
16A5          RETURN
16A6          END


PROGRAM  STORAGE: 0053

VARIABLE STORAGE: 0000

SYMBOL TABLE
169D 4

16A6 C ***************************************************************************

16A6          SUBROUTINE      G@SERVICE(MASK)
```

```
16A6
16A6 C ***************************************************************************
16A6 C CONTROL IS TRANSFERED TO THE ASSEMBY LEVEL DRIVER FOR THE GPIB
16A6 C       MASK DEFINITIONS:
16A6 C               STATUS CHANGE:  40H
16A6 C               COMMAND ERROR:  60H
16A6 C               MEASURE COMPLETE:  8
16A6 C               TX REQUESTED INFO:  0

16A6         INTEGER*1 MASK
16A6
16A6 C JUMP THROUGH JUMP TABLE ENTRY
16A6 C       LDA MASK,  JMP 0027H
16A6         INLINE / 3AH, ADDRESS(MASK), 0C3H,27H,0 /
16AC         RETURN
16AD         END


PROGRAM  STORAGE: 0007

VARIABLE STORAGE: 0002

SYMBOL TABLE
3D4E MASK

16AD C ***************************************************************************

16AD         SUBROUTINE      QUE@AUTO
16AD

16AD C*****************************************************


16AD C  ROUTINE PLACES BUFFER@INSERT@PTR IN RING BUFFER SLOT AND THEN
16AD C  ADJUSTS THE BUFFER PARAMETER BLOCK.

16AD C    LHLD ADD(BUFFER@INSERT@PTR)  INX H  XCHG  LHLD ADD(RBUFFER@IN@A)
16AD C    MOV M,E  INX H  MOV M,D  INX H  SHLD ADD(RBUFFER@IN@A) PUSH H
16AD C    LXI H,ADD(NO@POSITIONS@A) DCR M  LHLD ADD(BUFFER@STRT@A)
16AD C    MOV B,F  MOV C,L  LDA ADD(BUFFER@TOTAL@A)  RLC  MOV E,A
16AD C    MVI D,0  DAD D  MOV A,L  POP H  CMP L RNZ  MOV H,B
16AD C    MOV L,C  SHLD ADD(RBUFFER@IN@A)  RET

16AD         INLINE /2AH,ADDRESS(BUFFER@INSERT@PTR),23H,0EBH/
16B2         INLINE /2AH,ADDRESS(RBUFFER@IN@A),73H,23H,72H,23H/
16B9         INLINE /22H,ADDRESS(RBUFFER@IN@A),0E5H/
16BD         INLINE /21H,ADDRESS(NO@POSITIONS@A),35H/
16C1         INLINE /2AH,ADDRESS(RBUFFER@STRT@A),44H,4DH/
16C6         INLINE /3AH,ADDRESS(BUFFER@TOTAL@A),07H,5FH,16H,0/
16CD         INLINE /19H,7DH,0E1H,0BDH,0C2H,ADDRESS(4),62H,69H/
16D6         INLINE /22H,ADDRESS(RBUFFER@IN@A)/
16D9

16D9 4       CALL G@SERVICE(8)
16E1         RETURN
16E2         END
```

```
UTI FORT//80 COMPILER  .3A1                    PAGE 0078

PROGRAM  STORAGE: 0053

VARIABLE STORAGE: 0000

SYMBOL TABLE
16D9 4

16E2 C ********************************************************************************

16E2          SUBROUTINE      ADD@TO@BUFFER(CHAR)
16E2
16E2 C ********************************************************************************
16E2          INTEGER*1 CHAR
16E2
16E2 C        ARRAY(FOR AT@INDEX) = CHAR
16E2 C        FORMAT@INDEX = FORMAT@INDEX + 1

16E2 C        LXI H,FORMAT@INDEX  MOV E,M INR M MVI D,0
16E2 C        LHLD BUFFER@INSERT@PTR  DAD D
16E2 C        LDA CHAR  MOV M,A

16E2          INLINE / 21H, ADDRESS(FORMAT@INDEX), 5EH, 34H, 16H, 0/
16F9          INLINE / 2AH, ADDRESS(BUFFER@INSERT@PTR), 19H/
16FD          INLINE / 3AH, ADDRESS(CHAR), 77H/
16F1          RETURN
16F2          END


PROGRAM  STORAGE: 0016

VARIABLE STORAGE: 0004

SYMBOL TABLE
3D4A CHAR

16F2 C ********************************************************************************

16F2          SUBROUTINE      FORMAT@TIME(REG,NUM,SEPARATOR,PREFIX)
16F2
16F2 C ********************************************************************************
16F2          INTEGER*2 REG (4)
16F2          INTEGER*1 TEMP,NUM,SEPARATOR,NIB,PREFIX
16F2 C ** FORMAT@INDEX IS USED AS INDEX FOR THIS ROUTINE
16F2          IF (PREFIX .NE. 0) CALL ADD@TO@BUFFER(PREFIX)
1709 C ** INITIALIZE NIBBLE SWITCH SINCE PACKED BCD
1709          NIB = 0
170E          IF (NUM.EQ.4) NIB =1
171A
171A C ** GET BCD
171A 1        TEMP = REG(NUM)
1729
1729 C        LDA NIB, CPI 1,
1729          INLINE / 3AH, ADDRESS(NIB), 0FEH, 1 /
172E
172E C        LDA TEMP, JZ 2, RAR, RAR, RAR, RAR
172E          INLINE / 3AH, ADDRESS(TEMP), 0CAH, ADDRESS(2), 1FH,1FH,1FH,1FH/
1738
1738 C2       ANI 0FH, CPI 0FH, JNZ 7, XPA A
```

```
1738 2        INLINE / 0E6H, 0FH, 0FEH, 0FH, 0C2H, ADDRESS(7), 0AFH /
1740
1740 C7       ADI 30H, STA TEMP
1742 7        INLINE/ 0C6H, 30H, 32H, ADDRESS(TEMP)/
1745
1745          CALL ADD@TO@BUFFER(TEMP)
1754          NIB = ( NIB + 1) .AND. 1
175E          IF (NIB.EQ.1) GO TO 1
1766 C ** DETERMINE WHETHER TO INSERT SEPARATOR, GO ON OR DONE
1766          GO TO (5,3,3,4), NUM

1772 3        IF(SEPARATOR.NE.0) CALL ADD@TO@BUFFER(SEPARATOR)
1789 4        NUM = NUM - 1
1791          GO TO 1
1794 5        RETURN
1795          END


PROGRAM  STORAGE: 0171

VARIABLE STORAGE: 0016

SYMBOL TABLE
1789 4
1772 3
1794 5
1742 7
1738 2
171A 1
3D3A NIB
3D3B TEMP
3D3C PREFIX
3D40 SEPARATOR
3D44 NUM
3D46 REG

1795 C ***************************************************************************

1795          SUBROUTINE      CLOSURE
1795
1795 C ***************************************************************************
1795          CALL ADD@TO@BUFFER(0DH)
179D          CALL ADD@TO@BUFFER(0AH)
17A5          RETURN
17A6          END


PROGRAM  STORAGE: 0017

VARIABLE STORAGE: 0000

SYMBOL TABLE

17A6 C ***************************************************************************

17A6          SUBROUTINE      END@MESSAGE(MSK)
17A6
17A6 C ***************************************************************************
```

```
17A6          INTEGER*1 MSK
17A6
17A6          IF ( MSK .GE. 4) CALL ADD@TO@BUFFER(03H)
17B6          IF (( MSK .AND. 2) .NE. 0) CALL CLOSURE
17C0          IF (( MSK .AND. 1) .NE. 0) CALL ADD@TO@BUFFER(0)
17D2          RETURN
17D3          END
```

PROGRAM STORAGE: 0045

VARIABLE STORAGE: 0002

SYMBOL TABLE
3D38 MSK

```
17D3 C*************************************************************

17D3          SUBROUTINE MOVE@ASCII@TIME
17D3
17D3 C*************************************************************

17D3          INTEGER*1 I
17D3          I=7
17D8 10       IF((BUS@F RMAT.NE.1).OR.(ASCII@TIMER(I).NE.20H))
17TE      &       CALL ADD@TO@BUFFER(ASCII@TIMER(I))
1808          I=I-1
1810          IF(I.NE.0) GO TO 10
1818          RETURN
1819          END
```

PROGRAM STORAGE: 0070

VARIABLE STORAGE: 0001

SYMBOL TABLE
17D8 10
3D37 I

```
1819 C ************************************************************************

1819          SUBROUTINE      BUFFER@REAL@TIME(PREFIX,END@MASK,SEP1)
1819
1819 C ************************************************************************
1819          INTEGER*1  PREFIX, END@MASK,SEP1
1819          CALL FORMAT@TIME(DATE,2,0,PREFIX)
1838          CALL FORMAT@TIME(CLOCK,3,':',' ')
1850          CALL ADD@TO@BUFFER(SEP1)
185F          CALL MOVE@ASCII@TIME
1862          CALL END@MESSAGE(END@MASK)
1871          RETURN
1872          END
```

PROGRAM STORAGE: 0089

VARIABLE STORAGE: 0012

```
SYMBOL TABLE
3D2B SEP1
3D2F END@MASK
3D33 PREFIX

1872 C ***************************************************************************

1872          SUBROUTINE      BUFFER@ELAPSED@TIME( END@MASK)
1872
1872 C ***************************************************************************
1872          INTEGER*1 END@MASK
1872          CALL FORMAT@TIME}ELAPSED@TIME,4,|:| ,|L|~
188A          CALL END@MESSAGE(END@MASK)
1899          RETURN
189A          END


PROGRAM  STORAGE: 0040

VARIABLE STORAGE: 0004

SYMBOL TABLE
3D27 END@MASK

189A C ***************************************************************************

189A          INTEGER*1 FUNCTION      PRINT@OK
189A
189A C ***************************************************************************
189A          PRINT@OK=.FALSE.
189F          IF(.NOT.PRINTER@THERE) RETURN
18A6          IF(PBUF1(1).NE.1) RETURN
18B0          IF(REAL@TIME@P(1).NE.1) RETURN
18B9          IF(NO@POSITIONS@P.LT.3) RETURN
18C2          PRINT@OK=.TRUE.
18C7          RETURN
18C9          END


PROGRAM  STORAGE: 0047

VARIABLE STORAGE: 0001

SYMBOL TABLE

18C9 C ***************************************************************************

18C9          INTEGER*1 FUNCTION      TO@AUTO@OK
18C9
18C9 C ***************************************************************************
18C9          TO@AUTO@OK=.FALSE.
18CE          IF(.NOT.(RS@232@THERE.OR.BUS@REMOTE@ENABLE)) RETURN
18DB          IF(ABUF1(1).NE.1) RETURN
18E5          IF(REAL@TIME@A(1).NE.1) RETURN
18EF          IF(NO@POSITIONS@A.LT.3) RETURN
18F8          TO@AUTO@OK=.TRUE.
18F8          RETURN
```

```
18FA          END


PROGRAM  STORAGE: 0049

VARIABLE STORAGE: 0001

SYMBOL TABLE

18FA C

18FA          SUBROUTINE      DATE@TO@RS@232
18FA
18FA C ********************************************************************************
18FA          FORMAT@INDEX = 1
18FF          INLINE /21H,ADDRESS(REAL@TIME@A)/
1902          INLINE /23H,7EH,0FEH,1,0C0H,2BH/
1908          INLINE /22H,ADDRESS(BUFFER@INSERT@PTR)/
190B          CALL CLOSURE
190E          CALL BUFFER@REAL@TIME(0,3,' ')
1920          CALL QUE@AUTO
1923          RETURN
1924          END


PROGRAM  STORAGE: 0042

VARIABLE STORAGE: 0000

SYMBOL TABLE

1924 C ********************************************************************************

1924          SUBROUTINE      PRINT@DATE
1924
1924 C ********************************************************************************
1924          FORMAT@INDEX = 1
1929          INLINE /21H,ADDRESS(REAL@TIME@P)/
192C          INLINE /23H,7EH,0FEH,1,0C0H,2BH/
1932          INLINE /22H,ADDRESS(BUFFER@INSERT@PTR)/
1935          CALL CLOSURE
1938          CALL BUFFER@REAL@TIME(0,3,' ')
194A          CALL PRINT
194D          RETURN
194E          END


PROGRAM  STORAGE: 0042

VARIABLE STORAGE: 0000

SYMBOL TABLE

194E C

194E          SUBROUTINE SEND@MESSAGE@A(MES)
194F
194E C********************************************************************
```

```
194E          INTEGER 1 MES}115
194E
194E          IF(NO@POSITIONS@A.LT.2) RETURN
1954          INLINE /2AH,ADDRESS(MES),22H,ADDRESS(BUFFER@INSERT@PTR)/
195A          CALL QUE@AUTO
195D          RETURN
195E          END


PROGRAM  STORAGE: 0016

VARIABLE STORAGE: 0002

SYMBOL TABLE
3D23 MES

195E C*************************************************************

195E          SUBROUTINE SEND@MESSAGE@P(MES)
195E
195E C*************************************************************

195E          INTEGER*1 MES(115)
195E
195E          IF(NO@POSITIONS@P.LT.2) RETURN
1964          INLINE /2AH,ADDRESS(MES),22H,ADDRESS(BUFFER@INSERT@PTR)/
196A          CALL PRINT
196D          RETURN
196E          END


PROGRAM  STORAGE: 0016

VARIABLE STORAGE: 0002

SYMBOL TABLE
3D21 MES

196E C ************************************************************************

196E          SUBROUTINE      LOG@MES(MES)
196E
196E C ************************************************************************
196E          INTEGER*1 MES(115)
196E          BUS@FORMAT=0
1973
1973          IF (MESSAGE@NUMBER.EQ.6) GO TO 2
197F          IF (.NOT. PRINTER@THERE ) GO TO 2
1983          IF(P@COUNTER.NE.0) RETURN
1989          CALL PRINT@DATE
198C          CALL SEND@MESSAGE@P(MES)
1995
1995 2        IF (MACHINE@FORMAT  ) RETURN
199A          IF(RS@232@THERE) GO TO 4
19A1          IF(.NOT.BUS@REMOTE@ENABLE) RETURN
19A7 4        IF(A@COUNTER.NE.0) RETURN
19AD          CALL DATE@TO@RS@232
```

```
19B0          CALL SEND@MESSAGE@A(MES)
19B9          RETURN
19BA          END


PROGRAM  STORAGE: 0076

VARIABLE STORAGE: 0002

SYMBOL TABLE
19A7 4
1995 2
3D1F MES

19BA C ********************************************************************

19BA          SUBROUTINE      OUTPUT@TRAFFIC@LIGHT@STATE
19BA
19BA C ********************************************************************
19BA          IF ( .NOT. SIGNAL@LOSS ) GO TO 2
19C2          CALL LOG@MES(MES@SL)
19CB          RETURN
19CC 2        IF ( .NOT. FRAME@LOSS ) GO TO 3
19D4          CALL LOG@MES(MES@FL)
19DD 3        IF ( .NOT. PRPS@LOSS ) GO TO 4
19E5          IF ( BLU@CONDITION ) GO TO 5
19EC          CALL LOG@MES(MES@PL)
19F5          RETURN
19F6 5        CALL LOG@MES(MES@BS)
19FF          RETURN
1A00 4        IF(FRAME@LOSS) RETURN
1A05          CALL LOG@MES(MES@NL)
1A0F          RETURN
1A0F          END


PROGRAM  STORAGE: 0085

VARIABLE STORAGE: 0000

SYMBOL TABLE
19F6 5
1A00 4
19DD 3
19CC 2

1A0F C ********************************************************************

1A0F          SUBROUTINE      FORMAT@1@LOG(ASCII)
1A0F
1A0F C ********************************************************************
1A0F C ROUTINE USE GLOBAL FORMAT@INDEX FOR INDEX OF ARRAY
1A0F          INTEGER*1       ASCII(7)
1A0F
1A0F          IF (FORMAT@SEPARATOR.NE.0)
1A0F      &           CALL ADD@TO@BUFFER(FORMAT@SEPARATOR)
1A26
1A26          IF ( (BUS@FORMAT.NE.1)) GO TO 0
```

```
1A2E          IF ( ASCII(1) .NE. ' ') GO TO 4
1A3B C /* MEASURE IN INTEGER FORMAT(' XXX  ') FOR GPIB MUST NOT HAVE BLANKS
1A3B          CALL ADD@TO@BUFFER(ASCII(2))
1A49          CALL ADD@TO@BUFFER(ASCII(3))
1A57          CALL ADD@TO@BUFFER(ASCII(4))
1A65          RETURN
1A66
1A66 C /* PRINTER/RS-232C CHECK FOR OVERFLOW, UNDERFLOW
1A66 0        IF (ASCII(7).NE.'W') GO TO 3
1A73          CALL STRING@TO@BUFFER(MES@OV, 6)
1A81          RETURN
1A82
1A82 3        IF (ASCII(7).NE.'U') GO TO 4
1A8F          CALL STRING@TO@BUFFER(MES@UN, 6)
1A9D          RETURN
1A9E
1A9E C /* GOOD PRINTER/RS-232C AND GPIB E-FORMAT:
1A9E 4        CALL STRING@TO@BUFFER(ASCII, 4)
1AAC          IF ((BUS@FORMAT.EQ.1)) CALL ADD@TO@BUFFER('E')
1ABC          CALL ADD@TO@BUFFER(ASCII(5))
1ACA          CALL ADD@TO@BUFFER(ASCII(6))
1AD8          RETURN
1AD9          END


PROGRAM  STORAGE: 0202

VARIABLE STORAGE: 0004

SYMBOL TABLE
1A82 3
1A9E 4
1A66 0
3D1B ASCII

1AD9 C ************************************************************

1AD9          SUBROUTINE      TIME@BEGINS@OUTPUT
1AD9
1AD9 C ************************************************************

1AD9          IF(MACHINE@FORMAT) GO TO 30
1AE0
1AE0          CALL DATE@TO@RS@232
1AE3          RETURN
1AE4
1AE4 30       INLINE /21H,ADDRESS(REAL@TIME@A)/
1AE7          INLINE /22H,ADDRESS(BUFFER@INSERT@PTR)/
1AEA          FORMAT@INDEX=1
1AEF          IF(TX@TIME.EQ.0) GO TO 31
1AF7          CALL CLOSURE
1AFA          CALL BUFFER@REAL@TIME('F',2,',')
1B0C          CALL BUFFER@ELAPSED@TIME(2)
1B14          CALL FORMAT@TIME(TIMER,4,':','G')
1B2C 31       CALL END@MESSAGE(3)
1B34          CALL QUE@AUTO
1B37          RETURN
1B38
```

```
1B38          END


PROGRAM  STORAGE: 0095

VARIABLE STORAGE: 0000

SYMBOL TABLE
1B2C 31
1AE4 30

1B38 C ************************************************************

1B38          SUBROUTINE FORMAT@3@OF@TYPE(TYPE,SEC,TOT,CUM )
1B38
1B38 C ************************************************************
1B38          INTEGER*1 TYPE,SEC(7),TOT(7),CUM(7)
1B38          INTEGER*1 BLUE, NXT@LETTER
1B39
1B38 C FIRST IS |B| OR |P| OR |V| FOR BIT, PARITY OR BIPOLAR VIOLATION MEASUREME
TS
1B38          NXT@LETTER = TYPE
1B3E C ** INSERT LETTER INTO BUFFER
1B3E          INLINE /  CDH, ADDRESS(999) /
1B41
1B41          BLUE = (TYPE.EQ.'B').AND.BLU@CONDITION
1B50
1B50          IF ((BUS@FORMAT.NE.1)) GO TO 20
1B58 C /* GPIB MEASURES PRECEEDED BY STRING OF 3 INDICATORS
1B58 C /* FAULT INDICATOR --
1B58 C         INDICTES STATUS EFFECTS THIS MEASURE I.E FRAMELOSS FOR PARITY
1B58          NXT@LETTER = 'A'
1B5D          IF (FAULT@FLAG) NXT@LETTER = 'F'
1B66          IF (BLUE) NXT@LETTER = 'Z'
1B72          IF (SIGNAL@LOSS) NXT@LETTER = 'F'
1B7E          IF(BURST@FLAG) NXT@LETTER='B'
1B8A C ** INSERT LETTE  INTO BUFFER
1B8A          INLINE / 0CDH, ADDRESS(999) /
1B8D
1B8D          NXT@LETTER = 'A'
1B92          IF (TOT(7).EQ.'W') NXT@LETTER = 'W'
1BA4          IF(BURST@FLAG) NXT@LETTER='U'
1BB0 C ** INSERT LETTER INTO BUFFER
1BB0          INLINE / 0CDH, ADDRESS(999) /
1BB3
1BB3          NXT@LETTER ='A'
1BB8          IF (CUM(7).EQ.'U') NXT@LETTER='U'
1BCA          IF(BURST@FLAG) NXT@LETTER='R'
1BD6 C **  INSERT LETTER INTO BUFFER
1BD6          INLINE / 0CDH, ADDRESS(999) /
1BD9
1BD9 C  /* PRINTER OR RS-232C FORMAT
1BD9 20       FORMAT@SEPARATOR = SEPARATOR1
1BDF          CALL FORMAT@1@LOG}SEC
1BE8          FORMAT@SEPARATOR = SEPARATOR2
1BEE          CALL FORMAT@1@LOG(TOT )
1BF7          CALL FORMAT@1@LOG(CUM )
1C00          CALL CLOSURE
1C03          RETURN
```

```
1C04
1C04 C /* PROCEDURE TO CALL FOR INSERT OF CHARACTER
1C04 999      CALL ADD@TO@BUFFER(NXT@LETTER)
1C13          RETURN
1C14          END
```

PROGRAM STORAGE: 0220

VARIABLE STORAGE: 0018

SYMBOL TABLE

```
1BD9 20
1C04 999
3D09 NXT@LETTER
3D0A BLUE
3D0B CUM
3D0F TOT
3D13 SEC
3D17 TYPE
```

1C14

H. MEASUREMENT AND STATUS LIGHTS

```
1C14 C }}}}}}}}}}}}}}}}}}}}} FILE:  TRAF.FTN {{{{{{{{{{{{{{{{{{{{

1C14 C ***************************************************

1C14       SUBROUTINE        NOTE@STAT@CHANGES
1C14
1C14 C ***************************************************

1C14 C **   PREVENT A FLICKERING STATUS CONDITION FROM GENERATING
1C14 C                SO MANY LOGS
1C14 C **  LDA LOSS@REPORT@CTR   CPI 150  RC
1C14         INLINE / 3AH, ADDRESS}LOSS@REPORT@CTR~ /
1C17         INLINE  / 0FEH,  150,  0D0H /
1C1A
1C1A         IF ((.NOT.SIGNAL@LOSS) .OR. LAST@SIGNAL@LOSS) GO TO 2
1C26         CALL LOG@MES(MES@SL)
1C2F         GO TO 45
1C32
1C32 2       IF (SIGNAL@LOSS) RETURN
1C37         IF ((.NOT.FRAME@LOSS) .OR. LAST@FRAME@LOSS)  GO TO 3
1C43         CALL LOG@MES(MES@FL)
1C4C         GO TO 45
1C4F
1C4F 3       IF ((.NOT.BLU@CONDITION) .OR. LAST@BLU@COND) GO TO 4
1C5B         CALL LOG@MES(MES@BS)
1C64         GO TO 45
1C67
1C67 4       IF ((.NOT.PRBS@LOSS) .OR. LAST@PRBS@LOSS)  GO TO 5
1C73         CALL LOG@MES(MES@PL)
1C7C
1C7C C **  BUMP THE LOSS@REPORT@CTR  LXI H, CTR  INR M
1C7C 45      INLINE /21H, ADDRESS(LOSS@REPORT@CTR), 34H /
1C80         GO TO 100
1C83
1C83 5       IF (PRBS@LOSS .OR. (.NOT.LAST@PRBS@LOSS))  GO TO 6
1C8F         CALL LOG@MES(MES@PK)
1C98         GO TO 100
1C9B
1C9B 6       IF (BLU@CONDITION .OR. (.NOT. LAST@BLU@COND)) GO TO 7
1CA7         CALL LOG@MES(MES@NB)
1CB0         GO TO 100
1CB3
1CB3 7       IF (FRAME@LOSS .OR. (.NOT. LAST@FRAME@LOSS)) GO TO 8
1CBF          CALL LOG@MES(MES@FK)
1CC8         GO TO 100
1CCB
1CCB 8       IF (SIGNA @LOSS .OR. (.NOT.LAST@SIGNAL@LOSS) ) RETURN
1CD5         CALL LOG@MES(MES@GS)
1CDE
1CDE 100     CALL G@SERVICE(42H)
1CE6         RETURN
1CE7         END

PROGRAM  STORAGE: 0211

VARIABLE STORAGE: 0000
```

```
SYMBOL TABLE
1CCB 8
1CB3 7
1C9B 6
1CDE 100
1C83 5
1C67 4
1C4F 3
1C7C 45
1C32 2

1CE7 C ******************************************************

1CE7       SUBROUTINE        SET@TRAFFIC@LIGHTS
1CE7
1CE7 C ******************************************************
1CE7 C THIS ROUTINES USES FLAGS AND MASKS FOR DISCRETE LED'S TO
1CE7 C       SET THE STATE OF THE FRONT PANEL DISCRETE LED'S
1CE7 C       TRAFFIC LIGHTS USED BY JEFF'S FRONT PANEL DISPLAY ROUTINE

1CE7 C       *** TRAFFIC LIGHTS
1CE7 C               BIT 0           AUTOCONTROL
1CE7 C               BIT 1           SIGNAL LOSS
1CE7 C               BIT 2           FRAME LOSS
1CE7 C               BIT 3           PRBS LOSS
1CE7 C               BIT 4           GATING


1CE7         INTEGER*1 TEMP,RNTEMP,BL@FLAG
1CE7

1CE7 C */ COMBINE AUTO LIGHTS, PRBS@LOSS, FRAMELOSS, & SIGNAL@LOSS

1CE7 C */ FIRST DECREMENT THE BLINK@CTR AND CHANGE STATE OF BLINK IF
1CE7 C                  CTR = 0
1CE7         IF ( BLINK@CTR .NE. 0 ) GO TO 300
1CEF         BLINKER = .NOT. BLINKER
1CF6         BLINK@CTR = 88
1CFB 300     BLINK@CTR = BLINK@CTR - 1
1D03
1D03 C USE FLAGS TO SET THE TRAFFIC@LIGHTS CONFIGURATION IN TEMP
1D03         TEMP = 0
1D08 C ** AUTO CONTROL LIGHT
1D08         IF ((RS@232@THERE.AND.RS@LOCAL@LOCKOUT) .OR.
1D0F      &                              BUS@REMOTE@ENABLE ) TEMP = TEMP + 1
1D1F C ** OTHERS:
1D1F         BL@FLAG = BLINKER .AND. (RUN@STOP.EQ.2)
1D2D         IF ( SIGNAL@LOSS .OR. ( BLINK@SIGNAL@LOSS .AND. BL@FLAG))
1D32      &          TEMP = TEMP + 2
1D42         IF ( FRAME@LOSS .OR. ( BLINK@FRAME@LOSS .AND. BL@FLAG))
1D49      &          TEMP = TEMP + 4
1D59         IF ( PRBS@LOSS .OR. (BLINK@PRBS@LOSS .AND. BL@FLAG))
1D60      &          TEMP = TEMP + 8
1D70
1D70 C */ DETERMINE IF GATING LIGHT SHOULD BE ON:
1D70 305     IF (( SIGNAL@LOSS) .OR. (.NOT. SIGNAL@OBTAINED))
1D74      &                  GO TO 303
```

```
1D7C          IF (GATING@WINK@CTR .EQ. 0) GO TO 302
1D84 C /* COUNT DOWN WINK LIGHT--INDICATES A NEW NUMBER PUT IN ERROR@DISPLAY
1D84          GATING@WINK@CTR = GATING@WINK@CTR - 1
1D8C          GO TO 303
1D8F C /*  ELSE TURN ON GATING LIGHT
1D8F 302      TEMP = TEMP + 10H
1D97
1D97 C */  FIGURE OUT WHAT THE RUN/STOP LIGHT CONFIGURATION SHOULD BE
1D97 303      RNTEMP = RUN@STOP
1D9D C /*  RUN OR STOP WILL BLINK IF ANY BLINKING FLAGS SET
1D9D          IF )} BLINK@SIGNAL@LOSS .OR. BLINK@FRAME@LOSS .OR.
1D9D       &          BLINK@PRBS@LOSS) .AND. BLINKER ) RNTEMP = 0
1DB2
1DB2 304      IF ((TEMP.EQ.TRAFFIC@LIGHTS) .AND.
1DB9       &                 (RNTEMP.EQ.RUN@STOP@LIGHT)) RETURN
1DC9          RUN@STOP@LIGHT = RNTEMP
1DCD          TRAFFIC@LIGHTS = TEMP
1DD2          UPDATE@FRONT@PANEL = .TRUE.
1DD6          RETURN
1DD7          END


PROGRAM  STORAGE: 0240

VARIABLE STORAGE: 0003

SYMBOL TABLE
1DB2 304
1D8F 302
1D97 303
1D70 305
1CFB 300
3D06 BL@FLAG
3D07 RNTEMP
3D08 TEMP

1DD7
1DD7 C ******************************************************

1DD7       SUBROUTINE        SELECT@MESSAGE@TEXT
1DD7
1DD7 C ******************************************************
1DD7 C ** MESSAGE@NUMBERS:   1 - COMMAND ERROR
1DD7 C                       2 - POWER-UP W/ REALTIME
1DD7 C                       3 - START TEST W/ REALTIME
1DD7 C                       4 - LOG REAL TIME ONLY
1DD7 C                       5 - SEND TIMES REQUEST
1DD7 C                       6 - SEND FLAG REQUEST
1DD7 C                       7 - BER HEADER

1DD7          INTEGER*1 MASK1,MASK2
1DD7
1DD7          MASK1 = MESSAGE@NUMBER
1DDD          MESSAGE@NUMBER=0
1DE2          GO TO ( 1, 11, 21, 31, 41, 51, 52), MASK1

1DEE          RETURN
1DEF C ......................................................
```

```
1DEF C     COMMAND ERRO
1DEF 1        IF (MACHINE@FORMAT ) GO TO 2
1DF6          CALL SEND@MESSAGE@A}MES@CE~
1DFF          RETURN
1E00 C ** SET COMMAND ERROR BIT IN SERIAL POLL REGISTER
1E00 C **     AND REQUEST SERVICE OVER GPIB IF THERE
1E00 2        IF(.NOT.BUS@REMOTE@ENABLE) RETURN
1E06          CALL G@SERVICE(60H)
1E0E          RETURN
1E0F C .....................................................
1E0F C ** POWER-UP
1E0F 11       CALL LOG@MES(MES@PW)
1E18          RETURN
1E19 C .....................................................
1E19 C ** START TEST
1E19 C ** IF INSTANTANEOUS, SET FOR BER HEADER
1E19 C21      IF ( ERROR@MODE.GT.4) MESSAGE@NUMBER = 7
1E19 21       IF (TEST@TYPE.NE.2) GO TO 23
1E21 C *** ELAPSED TEST
1E21          CALL LOG@MES(MES@L)
1E2A          GO TO 25
1E2D 23       IF (TEST@TYPE.NE.4) GO TO 24
1E35 C *** SINGLE TIMER TEST
1E35          CALL LOG@MES(MES@S)
1E3E          GO TO 25
1E41 C *** REPEAT TIMER TEST
1E41 24       CALL LOG@MES(MES@R)
1E4A 25       CALL LOG@MES(MES@T)
1E53          RETURN
1E54 C .....................................................
1E54 C ** REAL TIME LOG ON HOUR WHEN NO LOGS THIS HOUR
1E54 31       IF (PRINTER@THERE) CALL PRINT@DATE
1E5B          RETURN
1E5C C .....................................................
1E5C C ** REQUEST TIMES
1E5C 41       IF (TO@AUTO@OK) GO TO 42
1E63          MESSAGE@NUMBER = 5
1E68 C ** IF BUFFER IN USE ALREADY TRY AGAIN NEXT TIME
1E68          RETURN
1E69 42       CALL SELECT@AUTO@BUF
1E6C          MASK1=' '
1E71          MASK2=7
1E74          BUS@FORMAT=0
1E79          IF (.NOT.MACHINE@FORMAT) GO TO 43
1E81 C ** BUS OUTPUT:
1E81          MASK1=',
1E86          MASK2=5
1E89          BUS@FORMAT=1
1E8E
1E8E C **RS-232 OUTPUT:
1E8E 43       CALL CLOSURE
1E91
1E91 44       CALL BUFFER@REAL@TIME('R',2,MASK1)
1EAA          CALL BUFFER@ELAPSED@TIME(2)
1EB2          CALL FORMAT@TIME(TIMER,4,':','S')
1ECA          CALL CLOSURE
1ECD          CALL END@MESSAGE(MASK2)
1EDC          CALL QUE@AUTO
```

```
1EDF          RETURN
1EE0 C ..........................................................
1EE0 C ** FLAG REQUEST -- RS-232 ONLY
1EE0 51       MESSAGE@NUMBER=6
1EE5 52       CALL OUTPUT@TRAFFIC@LIGHT@STATE
1EE8          MESSAGE@NUMBER=0
1EED          RETURN
1EEE C..........................................................
1EEE C
1EEE C ** BER HEADER
1EEE C61      CALL LOG@MES(MES@BR)
1EEE C        RETURN
1EEE          END


PROGRAM  STORAGE: 0293

VARIABLE STORAGE: 0002

SYMBOL TABLE
1E91 44
1E8E 43
1E69 42
1E41 24
1E4A 25
1E2D 23
1E00 2
1EE5 52
1EE0 51
1E5C 41
1E54 31
1E19 21
1E0F 11
1DEF 1
3D04 MASK2
3D05 MASK1

1EEE
```

```
1EEE C }}}}}}}}}}}}}}}}}}}}}}}}}} FILE:  MEAS.FTN {{{{{{{{{{{{{{{{{{{{{{{{{{
1EEE C ************************************************************

1EEE          SUBROUTINE      BCDNUM@TO@ASCII(ASCII)
1EEE
1EEE C ************************************************************
1EEE C CONVERT BCD NUMBER TO ASCII(6 CHARACTERS), 7TH BYTE OF ASCII
1EEE C        IS USED FOR AN INDICATOR OF GOOD, OVERFLOW OR UNDERFLOW
1EEE C BCDNUM:         ONES(4),TENTHS(3),HUNDREDTHS(2),EXPONENT(1)

1EEE          INTEGER*1 ASCII(7),I,X,CT
1EEE C BCDNUM IS A GLOBAL

1EEE C ** CLEAR FLAG
1EEE          ASCII(7) = 'G'
1EF7          ASCII(1)=ASCII(5)=ASCII(6)=' '
1F15          IF ( BCDNUM(1) .NE. 0FH) GO TO 3
1F1E C ** NUMBER IN I FORMAT:
1F1E          X = 4
1F23          DO 2 I=2,4
1F26          ASCII(I) = BCDNUM(X) + 30H
1F3F 2        X = X - 1
1F51          RETURN
1F52
1F52 C ** E-FORMAT
1F52 C NOW THREE DIGITS, AT LEAST 1 HAS A D.P.(MSB=1)
1F52 C MORE THAN 1 D.P. INDICATES OVERFLOW OR UNDERFLOW
1F52 3        X=4
1F57          I=1
1F5A          CT=0
1F5E 5        ASCII(I) = (BCDNUM(X) .AND. 0FH) + 30H
1F79          I=I+1
1F81          IF ( BCDNUM(X) .GE. 0) GO TO 6
1F90          CT = CT + 1
1F98          IF (CT .GT. 1) GO TO 6
1FA1 C ** ONLY PUT ONE '.' IN !
1FA1          ASCII(I) = '.'
1FAD          I=I+1
1FB5 6        X=X-1
1FBD          IF ( X .GT. 1) GO TO 5
1FC6
1FC6 C  CHECK EXPONENT(BCDNUM(1)) + OR - BY MSB
1FC6          IF(BCDNUM(1).EQ.1FH) RETURN
1FCD          ASCII(6) = (BCDNUM(1) .AND. 0FH) + 30H
1FDF          ASCII(5) = '+'
1FE8          IF (CT .GT. 1) ASCII(7)='W'
1FFA C FLAG OVERFLOW='W', UNDERFLOW='U'
1FFA          IF ( BCDNUM(1).GE.0) RETURN
2001          ASCII(5) = '-'
200A          IF ( CT .GT. 1) ASCII(7)='U'
201C          RETURN
201D          END


PROGRAM  STORAGE: 0303

VARIABLE STORAGE: 0007
```

```
SYMBOL TABLE
1FB5 6
1F5E 5
1F3F 2
1F52 3
3CFD CT
3CFE X
3CFF I
3D00 ASCII

201D C*****************************************************

201D       SUBROUTINE        GO@FLOAT@TEMP   (FP@DES)
201D
201D C*****************************************************

201D C FLOAT FLOAT@TEMP INTO FPDES

201D         INTEGER*1 FP@DES        (5)
201D
201D         CALL FLTPT@PACK
201D      &          (BEFORE@FLOAT@TEMP,FLOAT,BEFORE@FLOAT@TEMP,FP@DES)
203D         RETURN
203E         END


PROGRAM  STORAGE: 0033

VARIABLE STORAGE: 0004

SYMBOL TABLE
3CF9 FP@DES

203E C ***********************************************************

203E       SUBROUTINE        FLOAT@ACCUM (ACCUM)
203E
203E C ***********************************************************

203E C FLOAT I*5 ACCUM   FLOATING POINT NUM RETURNS IN FIRST 3
203E C        BYTES OF ACCUM

203E         INTEGER*1 ACCUM(5), TMP1(5), TMP2(5)
203E
203E C LSB=(1), MSB=(5)

203E C CAN ONLY FLOAT 15 BITS AT A TIME THEREFORE SOME MANIPULATION
203E C       IS REQUIRED TO BREAK THE ACCUM INTO 15 BIT CHUNKS

203E C INITIALIZE TEMPORARY TO ZERO
203E         TMP1(1) =40H
2043         TMP1(2) = TMP1(3) = 0
204B
204B C BITS 0 TO 14
204B C       ***  BYTES 1 & 2 ***
204B C LFLD ACCUM  INX H  MOV E,M  INX H  MOV A,M  PUSH H ANI 7FH
204B C MOV D,A  ORA E  JZ 900
```

```
204B C       XCHG  SHLD FLOAT@TEMP

204B         INLINE / 2AH, ADDRESS(ACCUM), 23H,5EH, 23H, 7EH/
2052         INLINE /   0E5H, 0E6H,7FH/
2055         INLINE / 57H, 0B3H, 0CAH, ADDRESS(900) /
205A         INLINE / 0EBH,22H, ADDRESS(FLOAT@TEMP) /
205E
205E         CALL GO@FLOAT@TEMP(TMP1)
2067
2067 C BITS 15 TO 29
2067 C       *** BYTES 2, 3 & 4
2067 C POP H
2067 C MOV A,M  INX H                ;BYTE 2 IN A
2067 C  MOV E,M  INX H  MOV D,M   PUSH H     ;BYTE 3 IN L, BYTE 4 IN H
2067 C       XCHG  DAD H  ORA A  JP GO  INX H        ;SHIFT H PR, ADD MSB OF A
2067 C GO:  MOV A,H  ANI 7FH   MOV H,A  ORA L  JZ 905
2067 C   SHLD  FLOAT@TEMP

2067 900     INLINE /0E1H, 7EH,  23H /
206A         INLINE / 5EH, 23H, 56H,  0E5H, 0EBH /
206F         INLINE /  9H, 0B7H, 0F2H, ADDRESS( 901), 23H /
2075 901     INLINE / 7CH, 0E6H, 7FH, 67H, 0B5H /
207A         INLINE /  0CAH, ADDRESS(905) /
207D         INLINE / 22H, ADDRESS(FLOAT@TEMP) /
2080
2080         CALL GO@FLOAT@TEMP(TMP2)
2089
2089 C TMP2 *  2**16  --  DIRECTLY ADJUST THE FLT.PT. EXPONENT

2089         TMP2(1) = TMP2(1) + 15
2092
2092         CALL FLTPT@PACK(TMP1,'+',TMP2,TMP1)
20AA
20AA C BITS 30 -39
20AA C       *** BYTES 4 & 5
20AA C POP H   MOV A,M  INX H                ;BYTE 4 IN A
20AA C   MOV L,M  MVI H,0  DAD H  DAD H      ;BYTE 5 IN L
20AA C  ORA A  JP  906  INX H  INX H         ;SHIFT H PR 2, ADD MSB OF A
20AA C 906:   RAL  ORA A  JP 907  INX H
20AA C 907:   MOV A,H  ORA L  JZ 910
20AA C SHLD FLOAT@TEMP

20AA 905     INLINE / 0E1H, 7EH,  23H /
20AD         INLINE / 6EH, 26H,0, 29H,29H/
20B2         INLINE / 0B7H,0F2H, ADDRESS(906), 23H, 23H /
20B8 906     INLINE / 17H, 0B7H,  0F2H, ADDRESS(907), 23H/
20BF 907     INLINE / 7CH,  0B5H /
20C0         INLINE /  0CAH, ADDRESS(910)/
20C3         INLINE /  22H, ADDRESS(FLOAT@TEMP) /
20C6
20C6         CALL GO@FLOAT@TEMP(TMP2)
20CF
20CF C ADJUST EXPONENT:  TMP2 *  2**30
20CF         TMP2(1) = TMP2(1) + 30
20D8
20D8         CALL FLTPT@PACK(TMP1,'+',TMP2,TMP1)
20F0
20F0 910     ACCUM(1) = TMP1(1)
```

```
20FE          ACCUM (2) = TMP1(2)
210C          ACCUM(3) = TMP1 (3)
211A          RETURN
211B          END


PROGRAM  STORAGE: 0221

VARIABLE STORAGE: 0014

SYMBOL TABLE
20F0 910
20BE 907
20B8 906
20AA 905
2075 901
2067 900
3CEB TMP2
3CF0 TMP1
3CF5 ACCUM

211B C ************************************************************

211B        SUBROUTINE        INTEGER@BCD (RTMP,RDTMP,RESULT@TASK)
211B
211B C ************************************************************
211B          INTEGER*1 RTMP(5),RDTMP(5),RESULT@TASK
211B

211B C ***  CONVERT 5 BYTE FIXED ACCUM TO FLOATING POINT SO IT CAN BE SCALED

211B 1025     CALL FLOAT@ACCUM (RTMP)
2124          IF ((RESULT@TASK .NE. 6).AND.(RESULT@TASK.NE.-2)) GO TO 1030
213D C IF RESULT@TASK IS 6 THEN,
213D C   RESULT NEEDS TO BE DIVIDED -- IT'S A RATE ( RESULT WILL
213D C        STILL NEED TO BE DIVIDED BY 1 MILLION --
213D C        BUT THAT'S DONE BY DIRECTLY ADJUSTING THE EXPONENT
213D C        ON THE DISPLAY)
213D          CALL FLTPT@PACK( RTMP, '/', RDTMP, RTMP)
2157
2157 C ******  CALL CONVERSION ROUTINE THRU JUMP TABLE
2157 C        CHANGE FLOATING POINT TO BCD W/ EXPONENT

2157 C        PUT RTMP INTO  B & D PAIR OF REGISTERS
2157 C LXI H, RTMP  INX H  MOV B,M  INX H  MOV E,M  INX H   MOV D,M
2157 1030     INLINE / 2AH, ADDRESS(RTMP), 23H, 46H, 23H, 5EH, 23H, 56H /
2160
2160 C PUT BASE ZERO OF POWR10 TABLE IN H & L REG
2160          INLINE / 21H, ADDRESS(FPTEN@NEG10), 23H /
2164
2164          IF(RESULT@TASK.EQ.-2) GO TO 1031
216C C PUT LOWER NIBBLE OF RESULT TASK IN A FOR CALL--HIDDEN DIVIDE BY
216C C                10**6 IF 6,  ELSE 0
216C          INLINE / 3AH, ADDRESS(RESULT@TASK), 0E6H, 0FH /
2171
2171 C RESTART 7 THRU LOCATION 38H IN THE JUMP TABLE
2171          INLINE /  F7H/
2172          RETURN
```

```
2173
2173 1031     INLINE /3AH,ADDRESS(RESULT@TASK),0F7H/
2177
2177          RETURN
2178          END


PROGRAM  STORAGE: 0093

VARIABLE STORAGE: 0012

SYMBOL TABLE
2173 1031
2157 1030
211B 1025
3CDF RESULT@TASK
3CE3 RDTMP
3CE7 RTMP

2178 C ************************************************************

2178          SUBROUTINE      INTEGER@TO@ASCII(ACCUM7)
2178
2178 C ************************************************************
2178          INTEGER*1 ACCUM7(7)
2178          CALL INTEGER@BCD(ACCUM7,ONE,0)
218C          CALL BCDNUM@TO@ASCII(ACCUM7)
2195          RETURN
2196          END


PROGRAM  STORAGE: 0030

VARIABLE STORAGE: 0004

SYMBOL TABLE
3CDB ACCUM7

2196 C ************************************************************

2196          SUBROUTINE      SCALE@INTEGER@ASCII(SRC7,DENOM,SCALE)
2196
2196 C ************************************************************
2196          INTEGER*1 SRC7(7),DENOM(5),SCALE
2196
2196          CALL INTEGER@BCD(SRC7,DENOM,SCALE)
21B1          CALL BCDNUM@TO@ASCII(SRC7)
21BA          RETURN
21BB          END


PROGRAM  STORAGE: 0037

VARIABLE STORAGE: 0012

SYMBOL TABLE
3CCF SCALE
3CD3 DENOM
```

```
3CD7 SRC7

21BB C ************************************************************

21BB          SUBROUTINE      FORMAT@TOTAL@LOG
21BB       &     (BMES1,BMES2,BMES3,DENOM)
21BB
21FB C ************************************************************

21BB          INTEGER*1 BMES1(7),BMES2(7),BMES3(7)
21BB          INTEGER*1 DENOM(7)
21BB
21BB C  ADJUST POINTERS FOR PRINTER
21BB          IF(BUS@FO MAT.NE.80H) GO TO 1
21C3
21C3 C        LHLD BMES1  LXI D,-336  DAD D  SHLD BMES1
21C3 C        LHLD BMES2  DAD D  SHLD BMES2
21C3 C        LHLD BMES3  DAD D  SHLD BMES3
21C3 C        LHLD DENOM  LXI D,-10  DAD D  SHLD DENOM
21C3          INLINE /2AH,ADDRESS(BMES1),11H,0B0H,0FEH,19H/
21CA          INLINE /22H,ADDRESS(BMES1),2AH,ADDRESS(BMES2)/
21D0          INLINE /19H,22H,ADDRESS(BMES2),2AH,ADDRESS(BMES3)/
21D7          INLINE /19H,22H,ADDRESS(BMES3),2AH,ADDRESS(DENOM)/
21DE          INLINE /11H,0F6H,0FFH,19H,22H,ADDRESS(DENOM)/
21E5


21E5 C  INCLUDE BIT MEASUREMENTS
21E5 1        IF ((1.AND.MASK).EQ.0) GO TO 2
21EF          LETTER='B'
21F4          FAULT@FLAG=PRPS@LOSS
21FA          INLINE /0CDH,ADDRESS(40)/
21FD 2        INLINE /0CDH,ADDRESS(30)/
2200          IF ((2.AND.MASK).EQ.0) GO TO 4
220A          LETTER='P'
220F          FAULT@FLAG=FRAME@LOSS
2215          INLINE /0CDH,ADDRESS(40)/
2218
2218 C    INCLUDE BIPOLAR VIOLATION MEASUREMENTS
2218 4        INLINE /0CDH,ADDRESS(30)/
221F          IF ((4.AND.MASK).EQ.0) RETURN
2223          LETTER='V'
2228          FAULT@FLAG=.FALSE.
222C          INLINE /0CDH,ADDRESS(40)/
222F
222F C ** END THE BLOCK
222F          RETURN
2230
2230 C   THIS ADJUSTS THE ADDRESS DEPENDING ON ERROR TYPE
2230 C30      LHLD BMES1  LXI D,-56   DAD D   SHLD  BMES1
2230 C        LHLD BMES2  DAD D  SHLD BMES1
2230 C        LHLD BMES3  DAD D  SHLD BMES3
2230 30       INLINE /2AH,ADDRESS(BMES1),11H,0C8H,0FFH,19H,22H/
2238          INLINE /ADDRESS(BMES1)/
223A          INLINE /2AH,ADDRESS(BMES2),19H,22H,ADDRESS(BMES2)/
2241          INLINE /2AH,ADDRESS(BMES3),19H,22H,ADDRESS(BMES3)/
2248          RETURN
2249 40       CALL INTEGER@TO@ASCII(BMES1)
```

```
2252          CALL INTEGER@TO@ASCII(BMES2)
225B          CALL SCALE@INTEGER@ASCII(BMES3,DENOM,6)
226F          CALL FORMAT@3@OF@TYPE(LETTER,BMES1,BMES2,BMES3)
229@          RETURN
2291
2291          END
```

PROGRAM STORAGE: 0214

VARIABLE STORAGE: 0016

SYMBOL TABLE

```
2218 4
2230 30
2249 40
21FD 2
21E5 1
3CBF DENOM
3CC3 BMES3
3CC7 BMES2
3CCB BMES1
```

```
2291
2291 C*************************************************************
2291          SUBROUTINE CONVERT@ER@SEC@EFF(ASCII)
2291
2291 C*************************************************************
2291          INTEGER*1 TEMP(6),NUM,ASCII(7),I,J
2291
2291          TEMP(4)=TEMP(5)=TEMP(6)=20H
229F
229F          IF (ASCII(7).EQ.'U') GO TO 20
22AC          IF ((ASCII(1).EQ.' ').AND.(ASCII(2).EQ.'0')) GO TO 20
22CD          IF ((ASCII(1).EQ.'1').AND.(ASCII(4).EQ.'.')) GO TO 40
22EE          IF ((ASCII(5).EQ.'-').AND.(ASCII(6).GT.'4')) GO TO 20
230E          IF (ASCII(5).NE.' ') GO TO 3
231P          IF (ASCII(2).EQ.'.') GO TO 2
2328          ASCII(3)=ASCII(2)
233A          ASCII(2)='.'
2343          ASCII(5)='+'
234C          ASCII(6)='1'
2355          GO TO 3
2358 2        ASCII(5)='+'
2361          ASCII(6)='0'
236A
236A 3        IF(ASCII(4).EQ.'0') GO TO 10
2377          TEMP(3)=3AH-ASCII(4)+'0'
2389          TEMP(2)=39H-ASCII(3)+'0'
239B 9        TEMP(1)=39H-ASCII(1)+'0'
23AD          GO TO 12
23B@ 10       TEMP(3)='0'
23B5          IF(ASCII(3).EQ.'0') GO TO 11
23C2          TEMP(2)=3AH-ASCII(3)+'0'
23D4          GO TO 9
```

```
23D7 11       TEMP}2~=¦0¦
23DC          TEMP(1)=3AH-ASCII(1)+'0'
23EC
23EC 12       NUM=0
23F1          IF((ASCII(5).EQ.'+').AND.(ASCII(6).EQ.'1')) GO TO 15
2412          NUM=ASCII(6)+1-30H
2421
2421 15       J=I=1
2429          ASCII(3)='.'
2432 16       IF(NUM.EQ.0) GO TO 17
243A          IF(I.EQ.3) I=4
2447          ASCII(I)='9'
2453          I=I+1
245B          NUM=NUM-1
2463          GO TO 16
2466
2466 17       IF(I.EQ.3) I=4
2473          ASCII(I)=TEMP(J)
248A          J=J+1
2492          IF(I.EQ.7) GO TO 18
249A          I=I+1
24A2          GO TO 17
24A5
24A5 18       IF(TEMP(J).LT.5) RETURN
24B2 19       ASCII(I)=ASCII(I)+1
24CC          IF(ASCII(I).NE.3AH) RETURN
24DA          ASCII(I)='0'
24F6          I=I-1
24EE          GO TO 19
24F1
24F1 20       ASCII(7)=0
24FA          ASCII(1)='1'
2503          ASCII(2)=ASCII(3)='0'
2517          ASCII(4)='.'
2520          ASCII(5)=ASCII(6)=' '
2534          RETURN
2535  40      ASCII(1)=ASCII(5)=ASCII(6)=' '
2553          ASCII(2)=ASCII(3)=ASCII(4)='0'
2571          RETURN
2572          END
```

PROGRAM STORAGE: 0737

VARIABLE STORAGE: 0013

SYMBOL TABLE

```
24B2 19
24A5 18
2466 17
2432 16
2421 15
23D7 11
23EC 12
239B 9
23B0 10
2359 2
236A 3
```

```
2535 40
24F1 20
3CB2 J
3CB3 I
3CB4 NUM
3CB5 TEMP
3CBB ASCII

2572 C*************************************************************

2572          SUBROUTINE FORMAT@THRESH@PAIR
2572       &        (MES,PMES1,PMES2,DENOM)
2572
2572 C*************************************************************

2572          INTEGER*1 MES(3),PMES1(7),PMES2(7)
2572          INTEGER*1  DENOM(5)
2572 C  FIRST MUST ADJUST PMES1 AND PMES2 TO APPROP. THRESHOLDS
2572 C    THRESHOLD ZERO IS PASSED IN PMES1 AND THRESHOLD 10E-7
2572 C    IS PASSED IN PMES2

2572          IF(LOG@COUNTER.GT.5) GO TO 400
257B          GO TO (100,1,150,200,150,150,150,100),TEMP1

2586 150      RETURN
2587
2587 C     LHLD PMES2  LXI D,14  DAD D  SHLD PMES2
2587 100      INLINE /2AH,ADDRESS(PMES2),11H,14,0,19H/
258E          INLINE /22H,ADDRESS(PMES2)/
2591          GO TO 1
2594
2594 C     LHLD PMES2  LXI  D,7  DAD D   SHLD PMES2
2594 200      INLINE /2AH,ADDRESS(PMES2),11H,7,0,19H/
259B          INLINE /22H,ADDRESS(PMES2)/
259E          GO TO 1
25A1 C    LHLD PMES1  LXI  D,-14  DAD D  SHLD PMES1
25A1 400      INLINE /2AH,ADDRESS(PMES1),11H,0F2H,0FFH,19H/
25A8          INLINE /22H,ADDRESS(PMES1)/
25AB

25AB C    MUST ADJUST PMES1,PMES2,DENOM IF SERVICING PRINTER
25AB C
25AB 1        IF(BUS@FORMAT.NE.80H) GO TO 2
25B3 C
25B3 C       LHLD PMES1  LXI D,-336  DAD D  SHLD PMES1
25B3 C       LHLD PMES2  DAD D  SHLD PMES2  LHLD DENOM
25B3 C       LXI D,-10  DAD D  SHLD DENOM
25B3          INLINE /2AH,ADDRESS(PMES1),11H,0B0H,0FEH,19H/
25BA          INLINE /22H,ADDRESS(PMES1),2AH,ADDRESS(PMES2)/
25C0          INLINE /19H,22H,ADDRESS(PMES2),2AH,ADDRESS(DENOM)/
25C7          INLINE /11H,0F6H,0FFH,19H,22H,ADDRESS(DENOM)/
25CE
25CE 2        CALL STRING@TO@BUFFER(MES,3)
25DC          SEPARATOR1=' '
25E1
25E1          GO TO (5,25,45),SELECT

25ED
```

```
25ED 5        IF ((MASK.AND.2).EQ.0) GO TO 10
25F7          LETTER='P'
25FC          INLINE /0CDH,ADDRESS(70)/
25FF          SEPARATOR1=84H
2604 10       INLINE /0CDH,ADDRESS(60)/
2607          IF((MASK.AND.1).EQ.0) GO TO 20
2611          LETTER='B'
2616          INLINE /0CDH,ADDRESS(70)/
2619          SEPARATOR1=84H
261E 20       INLINE /0CDH,ADDRESS(65)/
2621          IF((MASK.AND.4).EQ.0) GO TO 22
262B          LETTER='V'
2630          INLINE /0CDH,ADDRESS(70)/
2633

2633 22       CALL ADD@TO@BUFFER(0)
263B
263B          RETURN
263C
263C 25       IF((MASK.AND.2).EQ.0) GO TO 28
2646          LETTER='P'
264B          INLINE /0CDH,ADDRESS(80)/
264E          SEPARATOR1=84H
2653
2653 28       INLINE /0CDH,ADDRESS(60)/
2656          IF((MASK.AND.1).EQ.0) GO TO 33
2660          LETTER='B'
2665          INLINE /0CDH,ADDRESS(80)/
2668          SEPARATOR1=84H
266D
266D 33       INLINE /0CDH,ADDRESS(65)/
2670          IF((MASK.AND.4).EQ.0) GO TO 35
267A          LETTER='V'
267F          INLINE /0CDH,ADDRESS(80)/
2682 35       CALL ADD@TO@BUFFER(0)
268A
268A          RETURN
268B 45       IF((MASK.AND.2).EQ.0) GO TO 48
2695          LETTER='P'
269A          INLINE /0CDH,ADDRESS(90)/
269D          SEPARATOR1=84H
26A2
26A2 48       INLINE /0CDH,ADDRESS(60)/
26A5          IF((MASK.AND.1).EQ.0) GO TO 53
26AF          LETTER='B'
26B4          INLINE /0CDH,ADDRESS(90)/
26B7          SEPARATOR1=84H
26BC
26BC 53       INLINE /0CDH,ADDRESS(65)/
26BF          IF((MASK.AND.4).EQ.0) GO TO 55
26C9          LETTER='V'
26CF          INLINE /0CDH,ADDRESS(90)/
26D1 55       CALL ADD@TO@BUFFER(0)
26D9
26D9 C    FIRST TO ADJUST FROM PARITY TO BIT ACCUMULATORS
26D9 C60   LHLD PMES1    LXI D,56  DAD D  SHLD PMES1
26D9 C     LHLD PMES2    DAD D  SHLD PMES2
26D9 60       INLINE /2AH,ADDRESS(PMES1),11H,56,0,19H,22H/
```

```
26E1          INLINE /ADDRESS(PMES1)/
26E3          INLINE /2AH,ADDRESS(PMES2),19H,22H,ADDRESS(PMES2)/
26EA          RETURN
26EB
26EB C    ADJUST FROM BIT TO BPV
26EB C65   LHLD PMES1   LXI D,-112   DAD D   SHLD PMES1
26EB C     LHLD PMES2    DAD D   SHLD PMES2
26EB 65       INLINE /2AH,ADDRESS(PMES1),11H,00H,0FFH,19H,22H/
26F3          INLINE /ADDRESS(PMES1)/
26F5          INLINE /2AH,ADDRESS(PMES2),19H,22H,ADDRESS(PMES2)/
26FC          RETURN
26FD
26FD C    INTEGER@TO@ASCII
26FD 70       CALL INTEGER@TO@ASCII(PMES1)
2706          CALL INTEGER@TO@ASCII(PMES2)
270F          INLINE /0CDH,ADDRESS(95)/
2712          RETURN
2713 80       CALL SCALE@INTEGER@ASCII(PMES1,DENOM,SCALE)
272F          CALL SCALE@INTEGER@ASCII(PMES2,DENOM,SCALE)
2749          INLINE /0CDH,ADDRESS(95)/
274C          RETURN
274D 90       CALL CONVERT@ER@SEC@EFF(PMES1)
2756          CALL CONVERT@ER@SEC@EFF(PMES2)
275F          INLINE /0CDH,ADDRESS(95)/
2762          RETURN
2763
2763 95       CALL ADD@TO@BUFFER(SEPARATOR1)
2772          CALL ADD@TO@BUFFER(LETTER)
2781          FORMAT@SEPARATOR=20H
2786          CALL FORMAT@1@LOG(PMES1)
278F          FORMAT@SEPARATOR=82H
2794          CALL FORMAT@1@LOG(PMES2)
279D          CALL CLOSURE
27A0          RETURN
27A1
27A1          END


PROGRAM  STORAGE: 0581

VARIABLE STORAGE: 0014

SYMPOL TABLE
2763 95
26D1 55
26BC 53
274D 90
26A2 48
2682 35
266D 33
2713 80
2653 28
2633 22
26EB 65
261E 20
26D9 60
26FD 70
2604 10
```

```
268B 45
263C 25
25ED 5
25CE 2
2594 200
2586 150
25AB 1
2587 100
25A1 400
3CA4 DENOM
3CA8 PMES2
3CAC PMES1
3CB0 MES

27A1 C*************************************************************

27A1          SUBROUTINE FORMAT@ERR@SEC
27A1
27A1 C*************************************************************

27A1          SELECT=1
27A6          SCALE=0
27A9
27A9          IF (BUS@FORMAT.EQ.80H) GO TO 60
27B1          MASK=BLOCK@MASK@A
27B6          CALL SELECT@AUTO@BUF
27B9          INLINE /0CDH,ADDRESS(10)/
27BC          CALL QUE@AUTO
27BF          RETURN
27C0

27C0 10       CALL FORMAT@THRESH@PAIR (ESMES,
27C6       &   P@MEAS@2A,P@MEAS@2A@T7,
27D1       &   DENOM@MEAS@A)
27DA
27DA          RETURN
27DB

27DB C    FOR PRINTER
27DB 60       MASK=BLOCK@MASK@P
27E1          CALL SELECT@PRT@BUF
27E4          INLINE /0CDH,ADDRESS(10)/
27E7          CALL PRINT
27EA          RETURN
27EB

27EB          END


PROGRAM  STORAGE: 0074

VARIABLE STORAGE: 0000

SYMBOL TABLE
27C0 10
27DB 60

27EB C*************************************************************
```

```
27EB          SUBROUTINE FORMAT@AVG@BER
27EB
27EB C*****************************************************************


27EB          SELECT=2
27F0          SCALE=6
27F3          IF (BUS@FORMAT.EQ.80H) GO TO 60
27FB          MASK=BLOCK@MASK@A
2800          CALL SELECT@AUTO@BUF
2803          INLINE /0CDH,ADDRESS(10)/
2806          CALL QUE@AUTO
2809          RETURN
280A

280A 10       CALL FORMAT@THRESH@PAIR (BERMES,
2810      &    P@MEAS@3A,P@MEAS@3A@T7,
281C      &    DENOM@MEAS@A)
2825
2825          RETURN
282E


2826 C    FOR PRINTER
2826 60       CALL SELECT@PRT@BUF
2829          MASK=BLOCK@MASK@P
282F          INLINE /0CDH,ADDRESS(10)/
2832          CALL PRINT
2835          RETURN
2836
283E          END


PROGRAM  STORAGE: 0075

VARIABLE STORAGE: 0000

SYMBOL TABLE
280A 10
2826 60

283E C*****************************************************************

2836          SUBROUTINE FORMAT@ERR@SEC@EFF
2836
2836 C                                      ************************

283E          SELECT=3
283B          SCALE=0
283E
283E          IF ((BUS@FORMAT.AND.80H).NE.0) GO TO 60
2848          MASK=BLOCK@MASK@A
284D          CALL SELECT@AUTO@BUF
2850          INLINE /0CDH,ADDRESS(10)/
2853          CALL QUE@AUTO
2856          RETURN
2857
```

```
2857 10       CALL FORMAT@THRESH@PAIR (EFFMES,
285D       &   P@MEAS@4A,P@MEAS@4A@T7,
2868       &   DENOM@MEAS@A)
2871
2871          RETURN
2872


2872 C    FOR PRINTER
2872 60       CALL SELECT@PRT@BUF
2875          MASK=BLOCK@MASK@P
287B          INLINE /0CDH,ADDRESS(10)/
287E          CALL PRINT
2881          RETURN
2882

2882          END


PROGRAM  STORAGE: 0076

VARIABLE STORAGE: 0000

SYMBOL TABLE
2857 10
2872 60

2882 C*************************************************************

2882          SUBROUTINE FORMAT@PCER@SEC
2882
2882 C*************************************************************

2882          SELECT=2
2887          SCALE=-2
288A
288A          IF (BUS@FORMAT.EQ.80H) GO TO 60
2892          CALL SELECT@AUTO@BUF
2895          MASK=BLOCK@MASK@A
289B          INLINE /0CDH,ADDRESS(10)/
289E          CALL QUE@AUTO
28A1          RETURN
28A2
28A2 10       CALL FORMAT@THRESH@PAIR (PESMES,
28A8       &   P@MEAS@4A,P@MEAS@4A@T7,
28B3       &   DENOM@MEAS@SEC@A)
28BC
28BC          RETURN
28BD

28BD C    FOR PRINTER
28BD 60       CALL SELECT@PRT@BUF
28C0          MASK=BLOCK@MASK@P
28C6          INLINE /0CDH,ADDRESS(10)/
28C9          CALL PRINT
28CC          RETURN
28CD
```

```
28CD          END

PROGRAM  STORAGE: 0075

VARIABLE STORAGE: 0000

SYMBOL TABLE
28A2 10
28BD 60

28CD C*************************************************************

28CD          SUBROUTINE FORMAT@TOT@ERR
28CD
28CD C*************************************************************


28CD          SELECT=1
28D2          SCALE=0
28D5          IF (BUS@FORMAT.EQ.80H) GO TO 60
28DD          CALL SELECT@AUTO@BUF
28E0          MASK=BLOCK@MASK@A
28E6          INLINE /0CDH,ADDRESS(10)/
28E9          CALL QUE@AUTO
28EC          RETURN
28ED

28ED 10       CALL FORMAT@THRESH@PAIR (ERMES,
28F3       &   P@MEAS@1A,P@MEAS@1A@T7,
28FE       &   DENOM@MEAS@A)
2907
2907          RETURN
2908


2908 C    FOR PRINTER
2908 60       CALL SELECT@PRT@BUF
290B
290B          MASK=BLOCK@MASK@P
2911          INLINE /0CDH,ADDRESS(10)/
2914          CALL PRINT
2917          RETURN
2918          END


PROGRAM  STORAGE: 0075

VARIABLE STORAGE: 0000

SYMBOL TABLE
28ED 10
2908 60

2918 C*************************************************************

2918          SUBROUTINE   SEND@MES(MES)
```

```
2918
2918 C*****************************************************************

2918          INTEGER*1 MES(100)
2918
2918          IF(BUS@FORMAT.EQ.80H) GO TO 5
2920          CALL SEND@MESSAGE@A(MES)
2929          RETURN
292A 5        CALL SEND@MESSAGE@P(MES)
2933          RETURN
2934          END
```

PROGRAM STORAGE: 0028

VARIABLE STORAGE: 0002

SYMBOL TABLE
292A 5
3CA2 MES

```
2934 C*****************************************************************

2934          SUBROUTINE FORMAT@BURST
2934
2934 C*****************************************************************
2934          INTEGER*1 MASK1
2934          MASK1=5
2939
2939          IF(BUS@FORMAT.EQ.1) GO TO 20
2941          MASK1=7
2943          CALL SEND@MES(BURST@HEAD)
294C          SEPARATOR1=0
2951          SEPARATOR2=' '
2954          IF(BUS@FORMAT.EQ.80H) GO TO 40
295C 20       CALL SELECT@AUTO@BUF
295F          BURST@FLAG=.TRUE.
2964          CALL INTEGER@TO@ASCII(BST1@TMP@A)
296C          CALL INTEGER@TO@ASCII(BST2@TMP@A)
2975          CALL INTE ER@TO@ASCII(BST3@TMP@A)
297E          CALL FORMAT@3@OF@TYPE(BURST@LETTER,BST1@TMP@A,
298A      &       BST2@TMP@A,BST3@TMP@A)
299D          BURST@FLAG=.FALSE.
29A2          CALL END@MESSAGE(MASK1)
29B1          CALL QUE@AUTO
29B4          RETURN
29B5 40       CALL SELECT@PRT@BUF
29B8          BURST@FLAG=.TRUE.
29BD          CALL INTE ER@TO@ASCII(BST1@TMP@P)
29C5          CALL INTEGER@TO@ASCII(BST2@TMP@P)
29CE          CALL INTEGER@TO@ASCII(BST3@TMP@P)
29D7          CALL FORMAT@3@OF@TYPE(BURST@LETTER,BST1@TMP@P,
29E3      &       BST2@TMP@P,BST3@TMP@P)
29F6          BURST@FLAG=.FALSE.
29FF          CALL END@MESSAGE(MASK1)
2A0A          CALL PRINT
2A0D          RETURN
2A0E          END
```

```
PROGRAM  STORAGE: 0218

VARIABLE STORAGE: 0001

SYMBOL TABLE
29B5 40
295C 20
3CA1 MASK1

2A0E C*********************************************************

2A0E          SUBROUTINE FORMAT@TOTAL@LOG@ALL
2A0E
2A0E C*********************************************************


2A0E          IF (LOG@COUNTER.NE.0) GO TO 30
2A16

2A16          GO TO (5,15,3,10,3,3,3,5),TEMP1

2A22 3        RETURN
2A23 5        CALL SEND@MES(T0@HEAD)
2A2C          GO TO 25
2A2F 10       CALL SEND@MES(T6@HEAD)
2A38          GO TO 25
2A3B 15       CALL SEND@MES(T7@HEAD)
2A44
2A44 25       LOG@COUNTER=1
2A49
2A49 30       GO TO (40,50,60,70,80,40,50,60,70,80,110),LOG@COUNTER

2A55
2A55 40       CALL FORMAT@TOT@ERR
2A58          LOG@COUNTER=LOG@COUNTER+1
2A60          RETURN
2A61
2A61 50       CALL FORM T@ERR@SEC
2A64          LOG@COUNTER=LOG@COUNTER+1
2A6C          RETURN
2A6D
2A6D 60       CALL FORMAT@PCER@SEC
2A70          LOG@COUNTER=LOG@COUNTER+1
2A78          RETURN
2A79
2A79 70       CALL FORMAT@ERR@SEC@EFF
2A7C          LOG@COUNTER=LOG@COUNTER+1
2A84          RETURN
2A85
2A85 80       CALL FORMAT@AVG@BER
2A88
2A88          LOG@COUNTER=LOG@COUNTER+1
2A90          IF((TEMP1.NE.1).OR.(LOG@COUNTER.GT.6)) GO TO 100
2AA8          CALL SEND@MES(T6@T7@HEAD)
2AB1          RETURN
2AB2 100      LOG@COUNTER=11
```

```
2AB7          RETURN
2AB8
2AB8 110      CALL FORMAT@BURST
2ABB          LOG@COUNTER=0
2AC0          RETURN
2AC1          END


PROGRAM  STORAGE: 0217

VARIABLE STORAGE: 0000

SYMBOL TABLE
2AB2 100
2AB8 110
2A85 80
2A79 70
2A6D 60
2A61 50
2A55 40
2A44 25
2A2F 10
2A22 3
2A3B 15
2A23 5
2A49 30

2AC1 C*****************************************************************

2AC1          SUBROUTINE  SELECT@PARTIAL@PRINT
2AC1
2AC1 C*****************************************************************
2AC1          SEPARATOR1=0
2AC6          SEPARATOR2=' '
2AC9          IF(BUS@FORMAT.EQ.80H) GO TO 70
2AD1          TEMP1=T@TEMP@A
2AD7          MASK=BLOCK@MASK@A
2ADD          IF(.NOT.MACHINE@FORMAT) GO TO 15
1@D4          RDO@Q@SNQ0<RDO@Q@SNQ1<&+&
2AEC          IF(A@COUNTER.GT.1) GO TO 20
2AF4 15       CALL TIME@BEGINS@OUTPUT
2AF7
2AF7 20       GO TO (30,40,25,50,25,25,25,60),TEMP1

2B03 25       RETURN
2B04 30       CALL SEND@MES(T0@HEAD@MAC)
2B0D          INLINE/0CDH,ADDRESS(80)/
2B10          CALL FORMAT@TOTAL@LOG(B@MEAS@2A,B@MEAS@1A,B@MEAS@3A,
2B20      &     DENOM@MEAS@A)
2B29          RETURN
2B2A
2B2A 40       CALL SEND@MES(T7@HEAD@MAC)
2B33          INLINE /0CDH,ADDRESS(80)/
2B36          CALL FORMAT@TOTAL@LOG(B@MEAS@2A@T7,B@MEAS@1A@T7,B@MEAS@3A@T7,
2B46      &     DENOM@MEAS@A)
2B4F          RETURN
2B50
2B50 50       CALL SEND@MES(T6@HEAD@MAC)
```

```
2B59          INLINE /0CDH,ADDRESS(80)/
2B5C          CALL FORMAT@TOTAL@LOG(B@MEAS@2A@T6,B@MEAS@1A@T6,B@MEAS@3A@T6,
2B6C       &    DENOM@MEAS@A)
2B75          RETURN
2B76
2B76 60       CALL SEND@MES(T3@HEAD@MAC)
2B7F          INLINE /0CDH,ADDRESS(80)/
2B82          CALL FORMAT@TOTAL@LOG(B@MEAS@1A@T3,B@MEAS@2A@T3,B@MEAS@3A@T3,
2B92       &    DENOM@MEAS@A)
2B9B
2B9B          RETURN
2B9C
2B9C 70       TEMP1=T@TEMP@P
2BA2          MASK=BLOCK@MASK@P
2BA8          CALL PRINT@DATE
2BAB          GO TO 20
2BAE
2BAE 80       IF(BUS@FORMAT.NE.1) CALL SEND@MES(T0@S@HEAD)
2BBF          CALL SELECT@AUTO@BUF
2BC2          IF(BUS@FORMAT.EQ.82H) CALL SELECT@PRT@BUF
2BCA          IF(BUS@FORMAT.EQ.1) CALL CLOSURE
2BD2          RETURN
2BD3          END


PROGRAM  STORAGE: 0290

VARIABLE STORAGE: 0000

SYMBOL TABLE
2BAE 80
2B76 60
2B50 50
2B03 25
2B2A 40
2B04 30
2AF7 20
2AF4 15
2B9C 70

2BD3 C*********************************************************************

2BD3          SUBROUTINE FORMAT@TOTAL@LOG@MAC
2BD3
2BD3 C*********************************************************************

2BD3          IF(A@COUNTER.NE.0) GO TO 10
2BDB          A@COUNTER=1
2BE0
2BE0 10       GO TO (15,20,25,30,40),A@COUNTER

2BEC
2BEC 15       T@TEMP@A=1
2BF1          GO TO 35
2BF4 20       T@TEMP@A=8
2BF9          GO TO 35
2BFC 25       T@TEMP@A=4
2C01          GO TO 35
```

```
2C04 30       T@TEMP@A=2
2C09
2C09 35       CALL SELECT@PARTIAL@PRINT
2C0C          CALL ADD@TO@BUFFER(0)
2C14          CALL QUE@AUTO
2C17          A@COUNTER=A@COUNTER+1
2C1F          RETURN
2C20 40       CALL FORMAT@BURST
2C23          A@COUNTER=0
2C28          RETURN
2C29          END


PROGRAM  STORAGE: 0096

VARIABLE STORAGE: 0000

SYMBOL TABLE
2C09 35
2C20 40
2C04 30
2BFC 25
2BF4 20
2BEC 15
2BE0 10

2C29 C ************************************************************

2C29          SUBROUTINE      MEASUREMENT@COMPLETE?
2C29
2C29 C ************************************************************

2C29          INTEGER*1 MASK1
2C29
2C29 C ** BLOCK OF DATA??
2C29          IF ( .NOT.BLOCK@TEMPS@FULL@A) GO TO 18
2C31          IF((.NOT.BUS@REMOTE@ENABLE).AND.(.NOT.RS@232@THERE)) GO TO 179
2C40          IF(.NOT.TO@AUTO@OK) GO TO 18
2C48          BUS@FORMAT=0
2C4D          IF(MACHINE@FORMAT)BUS@FORMAT=1
2C56          IF(A@COUNTER.NE.0) GO TO 176
2C5E C ..........................................................
2C5E C   SEE IF OK TO SEND TO AUTO DEVICE

2C5E          T@TEMP@A=THRESHOLD
2C64          CALL FLOAT@ACCUM(DENOM@MEAS@A)
2C6D          CALL FLOAT@ACCUM(DENOM@MEAS@SEC@A)
2C76

2C76 C ..........................................................
2C76 C BLOCK OF TOTALS TO RS@232 OR GPIB ??
2C76 17       IF (MACHINE@FORMAT) BLOCK@MASK@A=7
2C82          Q@COMMAND@A=Q@COMMAND@A+16
2C8A          IF (Q@COMMAND@A.LE.16) GO TO 170
2C93          IF(Q@COMMAND@A.NE.17) GO TO 1700
2C99          CALL TIME@BEGINS@OUTPUT
2C9C          CALL FORMAT@BURST
2C9F          GO TO 179
```

```
2CA2 1700     BLOCK@MASK@A=7
2CA7          IF ((BLU@CONDITION.OR.(.NOT.PRBS@OBTAINED)).AND.
2CAF      &   (.NOT.MACHINE@FORMAT)) BLOCK@MASK@A=6
2CBF          IF (Q@COMMAND@A.EQ.24) BLOCK@MASK@A=ERROR@TYPE.AND.BLOCK@MASK@A
2CD1          IF (BLOCK@MASK@A.NE.0) GO TO 171
2CD9          MESSAGE@NUMBER=6
2CDE          CALL OUTPUT@TRAFFIC@LIGHT@STATE
2CE1          MESSAGE@NUMBER=0
2CE6          GO TO 179
2CE9 170      IF(BLOCK@MASK@A.EQ.0) GO TO 179
2CF1          IF ( UNLOGGED@ERRORS@TO@AUTO .EQ. 0) GO TO 179
2CF9          UNLOGGED@ERRORS@TO@AUTO = 0
2CFE          IF (TX@MODE.EQ.0) GO TO 179
2D06
2D06 171      UNLOGGED@ERRORS@TO@AUTO = 0
2D0B          IF(PRINT@CONTROL.EQ.1) GO TO 176
2D13          CALL SELECT@PARTIAL@PRINT
2D16          MASK1=7
2D1B          IF(MACHINE@FORMAT) MASK1=5
2D24          CALL END@MESSAGE(MASK1)
2D33          CALL QUE@AUTO
2D36          GO TO 179
2D39 176      TEMP1=T@TEMP@A
2D3F          IF(MACHINE@FORMAT) GO TO 177
2D46          IF(A@COUNTER.EQ.0) CALL DATE@TO@RS@232
2D4E          LOG@COUNTER=A@COUNTER
2D54          CALL FORMAT@TOTAL@LOG@ALL
2D57          A@COUNTER=LOG@COUNTER
2D5D          GO TO 178
2D60 177      CALL FORMAT@TOTAL@LOG@MAC
2D63 178      IF(A@COUNTER.NE.0) GO TO 18
2D6B 179      BLOCK@TEMPS@FULL@A=.FALSE.
2D70          IF((Q@COMMAND@A.AND.16).NE.0) Q@COMMAND@A=0
2D7F
2D7F C ..........................................................
2D7F C BLOCK OF TOTALS TO PRINTER ??
2D7F 18       IF(.NOT.BLOCK@TEMPS@FULL@P) RETURN
2D85          IF(.NOT.PRINTER@THERE) GO TO 20
2D8D          IF(.NOT.PRINT@OK) RETURN
2D93          BUS@FORMAT=80H
2D98          IF(P@COUNTER.NE.0) GO TO 195
2DA0          IF(Q@COMMAND@P.NE.0) BLOCK@MASK@P=7
2DAD          IF(BLOCK@MASK@P.EQ.0) GO TO 20
2DB5          T@TEMP@P=THRESHOLD
2DBB          CALL FLOAT@ACCUM(DENOM@MEAS@P)
2DC4          CALL FLOAT@ACCUM(DENOM@MEAS@SEC@P)
2DCD          Q@COMMAND@P=Q@COMMAND@P+16
2DD5          IF ( Q@COMMAND@P.EQ.16) GO TO 19
2DDD          IF (COUNTER@15@MIN.NE.0) GO TO 20
2DE9          IF ( UNLOGGED@ERRORS@TO@PRINT.EQ.0) GO TO 20
2DF1 19       UNLOGGED@ERRORS@TO@PRINT = 0
2DF6          IF (.NOT.PRINTER@THERE ) GO TO 20
2DFE          IF(PRINT@CONTROL.EQ.1) GO TO 196
2E06          CALL SELECT@PARTIAL@PRINT
2E09          CALL ADD@TO@BUFFER(0)
2E11          CALL PRINT
2E14          GO TO 20
2E17 196      TEMP1=T@TEMP@P
```

```
2E1D          IF(P@COUNTER.EQ.0) CALL PRINT@DATE
2E25          LOG@COUNTER=P@COUNTER
2E2B          CALL FORMAT@TOTAL@LOG@ALL
2E2E          P@COUNTER=LOG@COUNTER
2E34          IF(P@COUNTER.NE.0) RETURN
2E3A
2E3A 20       BLOCK@TEMPS@FULL@P = .FALSE.
2E3F          IF ((O@COMMAND@P.AND.16).NE.0) Q@COMMAND@P=0
2E4E          RETURN
2E4F C
2E4F          END


PROGRAM  STORAGE: 0550

VARIABLE STORAGE: 0001

SYMBOL TABLE
2DF1 19
2E17 196
2E3A 20
2D63 178
2D60 177
2D06 171
2CA2 1720
2CE9 170
2C76 17
2D39 176
2D6B 179
2D7F 18
3CA0 MASK1

2E4F C*********************************************************

2E4F        SUBROUTINE         CONVERT@RESULT@INTO@ERROR@DISPLAY
2E4F
2E4F C*********************************************************

2E4F C ***  RESULT TASK CAN EQUAL 0, 6, 10H, OR 0F0H
2E4F C ***  ( WITH ADDITIONAL MASK OF 20H, SEE IF RESULT GOES TO PERIPH)
2E4F          INTEGER*1        P@FLAG
2E4F
2E4F C 0F0H IN RESULT@TASK MEANS BLANK DISPLAY
2E4F          IF (RESULT@TASK .NE. 0F0H) GO TO 1025
2E57
2E57 C DISABLE INTERRUPTS BRIEFLY TO CHANGE COMMON FOR DISPLAY UPDATE
2E57          INLINE /0F3H/
2E58 C  BLANK THE ERROR@DISPLAY
2E58          CALL MOVE4(BLANKS, ERROR@DISPLAY)
2E67          INLINE /0FBH/
2E68          UPDATE@FRONT@PANEL = .TRUE.
2E6D          RETURN
2E6E


2E6E C1025    P@FLAG = .FALSE.
2E6E C        IF (( RESULT@TASK .AND. 20H) .NE. 0) P@FLAG = .TRUE.
2E6E C        RESULT@TASK = RESULT@TASK .AND. 1FH
```

```
2E6E 1025     IF ((RESULT@TASK .EQ.6).OR.(RESULT@TASK.EQ.-2))
2E7C      &     CALL FLOAT@ACCUM(RDTMP)
2E8E          CALL INTEGER@BCD(RTMP,RDTMP,RESULT@TASK)
2EA8
2EA8          IF }.NOT.} PRBS@LOSS .AND. BLU@CONDITION .AND.
2EA8      &           (.NOT.SIGNAL@LOSS)
2EAF      &           .AND. (ERROR@TYPE.EQ.1)))  GO TO 1027
2EC6
2EC6          INLINE /0F3H/
2EC7 C *** IF BLU LINE IS  ON AND  BIT ERRORS SELECTED AND NOT PRES
2EC7 C        PUT blu  into the error@DISPLAY
2EC7          CALL MOVE4(BLU@CONFIG, ERROR@DISPLAY)
2ED6          INLINE /0FBH/
2ED7 C        GO TO 1028
2ED7          GO TO 1031
2EDA
2EDA C ** OTHERWISE
2EDA 1027     INLINE / 0F3H/
2EDB          CALL MOVE4(PCDNUM,ERROR@DISPLAY)
2EE9          INLINE /0FBH /
2EEA

2EEA C SET CTR TO WINK THE GATING LIGHT
2EEA 1031     IF (GATING@WINK@CTR .EQ. 0) GO TO 1029
2EF2          GATING@WINK@CTR = 0
2EF7          GO TO 1034
2EFA 1029     GATING@WINK@CTR = 40
2EFF
2EFF 1034     UPDATE@FRONT@PANEL = .TRUE.
2F04          RETURN
2F05          END


PROGRAM  STORAGE: 0182

VARIABLE STORAGE: 0001

SYMBOL TABLE
2EFF 1034
2EFA 1029
2EEA 1031
2EDA 1027
2E6E 1025
3C9F P@FLAG

2F05 C*********************************************************

2F05          SUBROUTINE PERFORM@ANY@CLOCK@READS@OR@SETS(CTASK,NEW@DISPLAY)
2F05

2F05 C*********************************************************
2F05          INTEGER*1 CTASK,NEW@DISPLAY
2F05
2F05          GO TO ( 805,802,803,804,825), CTASK

2F11          RETURN
2F12 802      INLINE /0CDH,15H, 0/
2F15 C        CALL SET@TIME@OF@DAY
```

```
2F15          SAVE@TIME@DISPLAY = .FALSE.
2F1A          RETURN
2F1B
2F1B 803      INLINE /0CDH,18H,0/
2F1E C        CALL SET@MO@DAY
2F1E C /*  TIME@DISPLAY WILL STILL BE PROTECTED WHEN AN INVALID
2F1E C /*             ENTRY IN TIME TO SET
2F1E          IF (SAVE@TIME@DISPLAY) CALL TIME@DISPLAY@ERROR
2F25          RETURN
2F26
2F26 804      INLINE /0CDH, 0FH,0/
2F29 C        CALL READTIME@OF@DAY
2F29          NEW@DISPLAY = .TRUE.
2F2E          RETURN
2F2F
2F2F 805      INLINE /0CDH, 12H,0/
2F32 C        CALL READMO@DAY
2F32          NEW@DISPLAY = .TRUE.
2F37 806      RETURN
2F38          END


PROGRAM  STORAGE: 0061

VARIABLE STORAGE: 0006

SYMBOL TABLE
2F2F 805
2F26 804
2F1B 803
2F12 802
2F37 806
3C99 NEW@DISPLAY
3C9B CTASK

2F38 C*******************************************************

2F38        SUBROUTINE         UPDATE@SYSTEM@STATUS
2F38
2F38 C*******************************************************

2F38 C  ***   CALL STREAD TO SET SIGNAL LOSS,FRAME LOSS,PRBS LOSS,BLUE CONDITION
2F38          INLINE / 0CDH, 21H, 0, 0F3H/
2F3C
2F3C          IF ( RUN@STOP .EQ. 2 ) RETURN
2F42
2F42 C *** IF RUNNING THEN LOOK FOR STATUS LOSSES
2F42          IF ( SIGNAL@LOSS .AND. SIGNAL@OBTAINED) BLINK@SIGNAL@LOSS=.TRUE.
2F52          IF ( FRAME@LOSS .AND. FRAME@OBTAINED) BLINK@FRAME@LOSS=.TRUE.
2F62          IF ( PRBS@LOSS .AND. PRBS@OBTAINED) BLINK@PRBS@LOSS=.TRUE.
2F72          RETURN
2F73
2F73          END


PROGRAM  STORAGE: 0059

VARIABLE STORAGE: 0000
```

```
SYMBOL TABLE

2F73 C*************************************************************

2F73          SUBROUTINE      SEND@POWER@DOWN
2F73
2F73 C*************************************************************

2F73          CALL SELECT@AUTO@BUF
2F76          BUS@FORMAT=0
2F7B          IF(.NOT.MACHINE@FORMAT) GO TO 4
2F83          BUS@FORMAT=1
2F85          CALL CLOSURE
2F88          CALL BUFFER@REAL@TIME('R',3,',')
2F9A          CALL QUE@AUTO
2F9D          CALL SEND@MESSAGE@A(MES@PD@M)
2FA6          GO TO 6
2FA9 4        INLINE /0CDH,ADDRESS(7)/
2FAC          CALL QUE@AUTO
2FAF          CALL SEND@MESSAGE@A}MES@PD~
2FB8 6        CALL SELECT@PRT@BUF
2FBF          BUS@FORMAT=80H
2FC0          INLINE /0CDH,ADDRESS(7)/
2FC3          CALL PRINT
2FC6          CALL SEND@MESSAGE@P(MES@PD)
2FCF          RETURN
2FD0
2FD0 7        CALL CLOSURE
2FD3          CALL BUFFER@REAL@TIME(0,3,' ')
2FE5          RETURN
2FE6          END


PROGRAM  STORAGE: 0115

VARIABLE STORAGE: 0000

SYMBOL TABLE
2FD0 7
2FB8 6
2FA9 4

2FE6 C*************************************************************

2FE6        SUBROUTINE        SAVE@STATUS
2FE6
2FE6 C*************************************************************
2FE6          LAST@SIGNAL@LOSS = SIGNAL@LOSS
2FEC          LAST@PRBS LOSS = PRBS@LOSS
2FF2          LAST@FRAME@LOSS = FRAME@LOSS
2FF8          LAST@BLU@COND = BLU@CONDITION
2FFE          RETURN
2FFF          END


PROGRAM  STORAGE: 0025
```

I. MAIN LINE

VARIABLE STORAGE: 0000

SYMBOL TABLE

```
2FFF
2FFF          COMPILER(2)=7000H
7000

7000          INTEGER*1 CTEMP, FRONTPANEL@CHANGED@BY@MAIN, START@UP
7002
7000 C                S Y S T E M   O P E R A T I O N   B E G I N S

7000 C *** DISABLE INTERRUPTS TO INITIALIZE ALL THE VARIABLES FOR
7000 C        SYSTEM OPERATION
7000 C *** DEVICE CLEAR RESTARTS HERE, THEREFORE REASSIGN STACK POINTER
7000 1005     INLINE /  F3H, 31H, 40H , 3CH /
7004          CALL INITIAL@SYSTEM@STATE
7007          IF(.NOT.FIRST@FLAG) CALL SEND@POWER@DOWN
700F
700F C ***  INITIALIZE THE FIRST STATE OF THE SYSTEM

700F 1006     START@UP = .TRUE.
7014 C *** READ MONTH DAY
7014          INLINE / 0CDH,12H,0 /
7017 C *** CALL READ THE TIME OF DAY ROUTINE
7017          INLINE /  0CDH, 0FH, 0 /
701A C *** POWER-UP MESSAGE TO PRINTER OR RS-232
701A          MESSAGE@NUMBER = 2
701F          CALL SELECT@MESSAGE@TEXT
7022 C ***  ENABLE INTERRUPTS AND GO  !!!!
7022          INLINE /  FBH /
7023


7023 C ************************************************************
7023 C
7023 C                         M A I N   L O O P
7023 C
7023 C ************************************************************



7023 801      FRONTPANEL@CHANGED@BY@MAIN = .FALSE.
7028
7028          CALL UPDATE@SYSTEM@STATUS
702P
702B C *** DISABLE INTERRUPTS TO MANIPULATE RESULT@TASK FLAG  AND
702B C                  CLOCK@TASK FLAG--SO DON'T MISS CLOCK@TASKS OR RESULTS
702B          CTEMP = CLOCK@TASK
7031          CLOCK@TASK = 1
7036          RESULT@TASK = 0
703A          IF ( RESULT@READY .EQ. 0) GO TO 1010
7041
7041          RESULT@TASK = RESULT@READY
7044          RESULT@READY = 0
```

```
7047 C MOVE RESULT AND RESULT@DENOM TO  RTEMP & RDTEMP
7047 C MOVE@ACCUM CAN ONLY BE CALLED HERE BECAUSE INTERRUPTS ARE DISABLED!!!!!
7047         CALL MOVE@ACCUM(RESULT,RTMP)
7054         CALL MOVE@ACCUM(RESULT@DENOM,RDTMP)
7062
7062 C ***  RE ENABLE INTERRUPTS
7062 1010    INLINE /0FBH/
7063



7063 C ************************************************************************
7063 C                C L O C K   T A S K
7063         CALL  PERFORM@ANY@CLOCK@READS@OR@SETS
7063      &          (CTEMP, FRONTPANEL@CHANGED@BY@MAIN)
707E

707E C ** NO STATUS LIGHT UPDATE DURING LIGHT TEST
707E         CALL SET@TRAFFIC@LIGHTS
7081
7081         IF (.NOT.START@UP) GO TO 1015
7089 C ** OUTPUT INITIAL STATUS
7089         START@UP = .FALSE.
708E         MESSAGE@NUMBER=7
7093         CALL SAVE@STATUS
7096
7096 1015    IF (RESULT@TASK .EQ. 0)  GOTO 1020
709E
709E C ************************************************************************
709E C                        R E S U L T   T A S K
709E C ***   RTMP READY TO CONVERT TO BCD AND PUT IN ERROR@DISPLAY
709E          CALL  CONVERT@RESULT@INTO@ERROR@DISPLAY
70A1         FRONTPANEL@CHANGED@BY@MAIN = .TRUE.
70A6

70A6 1020    IF  (.NOT. (SIGNAL@LOSS .AND.
70A6      &  (UPDATE@FRONT@PANEL .OR. FRONTPANEL@CHANGED@BY@MAIN)))
70B0      &          GO TO 1030
70B5
70B5 C ************************************************************************
70B5 C                F R O N T   P A N E L   U P D A T E
70B5 C /*  SINCE NO MILLION BIT INTERRUPS DURING SIGNAL LOSS AND SINCE
70B5 C       FRONT PANEL UPDATE OCCURS DURING INTERRUPT, BETTER UPDATE
70B5 C       PANEL DIRECTLY, MAY BE A WHILE BEFORE ONE SEC TIC

70B5         INLINE / 0F3H,0CDH,1EH,0,0FBH /
70BA C /*  INTERRUPTS MUST BE DISABLED DURING FRONT PANEL UPDATE--CALL THRU JMP
ABLF


70BA C ************************************************************************
70BA C                S E R V I C E   P E R I P H E R A L S
70BA 1030    CALL PSERV
70BD         IF (.NOT.( PRINTER@THERE .OR. RS@232@THERE
70BD      &                  .OR. BUS@REMOTE@ENABLE)) GO TO 801
70CC
70CC         IF ( MESSAGE@NUMBER .EQ.2) GO TO 1031
70D4         IF ( CLOCK@TASK.NE.1) GO TO 1031
```

```
70DC          CALL SELECT@MESSAGE@TEXT
70DF          GO TO 1032
70E2
70E2 1031     CALL MEASUREMENT@COMPLETE?
70E5
70E5          IF ( START@UP) GO TO 1032
70EC          IF ( RUN@STOP .EQ. 2) GO TO 1032
70F4          CALL NOTE@STAT@CHANGES
70F7          CALL SAVE@STATUS
70FA
70FA 1032     CALL PSERV
70FD          GO TO 801
7100
```

```
7100 C ************************************************************************

7100 C                J U M P   T A B L E

7100 C ************************************************************************

7100 C JUMP TABLE ENTRIES FOR FORTRAN ROUTINES

7100          COMPILER(2) = 0006H
0006
0006          INLINE/0C3H,ADDRESS(ONCE@PER@SEC)/
0009 C JMP ONCE@PER@SEC                6
0009          INLINE/0C3H,ADDRESS(TEN@6@BITS)/
000C C JMP TEN@6@BITS                  9
000C          INLINE/0C3H,ADDRESS(BRANCH@ON@INPUT)/
000F C JMP BRANCH@ON@INPUT             0CH
000F C JMP READ@TIME@OF@DAY            0FH
000F C JMP READ@MO@DAY                 12H
000F C JMP SET@TIME@OF@DAY             15H
000F C JMP SET@MO@DAY                  18H
000F C JMP LEDTEST                     1BH
000F C JMP DISPLA                      1EH
000F C JMP STREAD                      21H
000F C JMP CLEAR DEVICE STATE          24H
000F          COMPILER(2) = 24H
0024          INLINE/0C3H,ADDRESS(1005)/
0027 C JMP GSERV                       27h
0027 C JMP BURST@DATA
0027          COMPILER(2)=2AH
002A          INLINE /0C3H,ADDRESS(BURST@DATA)/
002D C JMP RX@PARSE
002D          COMPILER(2) = 2DH
002D          INLINE/0C3H,ADDRESS(RX@PARSE)/
0030 C JMP FPBCD                       30H     ;RESTART 7
0030 C JMP PSERV                       33H
0030 C JMP INTERRUPT                   38H
```

```
0030 C ************************************************************************

0030 C                          C O M P I L E R   ADD ONS

0030          COMPILER(2) = 7101H
7101
7101                  END
```

PROGRAM STORAGE: 0274

VARIABLE STORAGE: 0003

TOTAL PROGRAM STORAGE: 012797

TOTAL VARIABLE STORAGE: 2637

SYMBOL TABLE

```
70FA 1032
70E2 1031
70BA 1030
70A6 1020
7096 1015
7062 1010
7023 801
700F 1006
7002 1005
3C96 START@UP
3C97 FRONTPANEL@CHANGED@BY@MAIN
3C98 CTEMP
2FE6 SAVE@STATUS
2F73 SEND@POWER@DOWN
2F38 UPDATE@SYSTEM@STATUS
2F05 PERFORM@ANY@CLOCK@READS@OR@SETS
2E4F CONVERT@RESULT@INTO@ERROR@DISPLAY
2C29 MEASUREMENT@COMPLETE?
2BD3 FORMAT@TOTAL@LOG@MAC
2AC1 SELECT@PARTIAL@PRINT
2A0E FORMAT@TOTAL@LOG@ALL
2934 FORMAT@BURST
2918 SEND@MES
28CD FORMAT@TOT@ERR
2882 FORMAT@PCER@SEC
2836 FORMAT@ERR@SEC@EFF
27FB FORMAT@AVG@BER
27A1 FORMAT@ERR@SEC
2572 FORMAT@THRESH@PAIR
2291 CONVERT@ER@SEC@EFF
21BE FORMAT@TOTAL@LOG
2196 SCALE@INTEGER@ASCII
2178 INTEGER@TO@ASCII
211B INTEGER@BCD
203F FLOAT@ACCUM
```

```
201D GO@FLOAT@TEMP
1EEE BCDNUM@TO@ASCII
1DD7 SELECT@MESSAGE@TEXT
1CE7 SET@TRAFFIC@LIGHTS
1C14 NOTE@STAT@CHANGES
1B38 FORMAT@3@OF@TYPE
1AD9 TIME@BEGINS@OUTPUT
1A0F FORMAT@1@LOG
19BA OUTPUT@TRAFFIC@LIGHT@STATE
196E LOG@MES
195E SEND@MESSAGE@P
194E SEND@MESSAGE@A
1924 PRINT@DATE
18FA DATE@TO@RS@232
18C9 TO@AUTO@OK
189A PRINT@OK
1872 BUFFER@ELAPSED@TIME
1819 BUFFER@REAL@TIME
17D3 MOVE@ASCII@TIME
17A6 END@MESSAGE
1795 CLOSURE
16F2 FORMAT@TIME
16E2 ADD@TO@BUFFER
16AD QUE@AUTO
16A6 G@SERVICE
1671 PRINT
166E PSERV
1602 BURST@DATA
15CD TEN@6@BITS
1505 GET@ERROR@COUNTS
14F0 MOVE@CTR@TEMP@TO@ACCUM
145F SERVICE@TEN@7@8@ACCUM
1399 SERVICE@BLOCK@ACCUMS
12F1 UPDATE@BLOCK
10F7 ONCE@PER@SEC
10D0 SET@HORN
1038 ELAPSED@TEST@DATA@TO@PERIPH
0FE0 UPDATE@TOTAL@ERRORS
0F6F UPDATE@ERROR@SECONDS
0F54 ACCUM@GE@THR?
0F2E UPDATE@THIS@INT@ACCUMS
0ECB DUMMY@ALL@ERRORS@DURING@SIGNAL@LOSS
0EAF PERCENT@ERROR@SECONDS
0EA1 ERROR@SECONDS
0E1E CUMULATIVE@RATE
0E10 TOTAL@ERRORS
0DE3 SET@RESULT
0D85 UPDATE@TEST
0D04 UPDATE@TIME
0CE8 BUMP@ASCII@TIMER
0CC4 COUNT@DOWN@TIMER
0C9D BUMP@ELAPSED@TIME
0C34 BUMP@CLOCK
0AD9 RX@PARSE
0AD2 SYNTAX@ERROR
06B8 BRANCH@ON@INPUT
0600 GET@SET@TO@SET
0538 CATCH@JUNK@IN@TIME
```

```
0525 TIME@DISPLAY@ERROR
051F COMMAND@ERROR
0500 ROTATE@DIGIT@INTO@TIME
04?B ADJUST@NEW@ERROR@MASK
0404 SAVE@TOTAL@BLOCK@W@INTERVAL@RATE
03CE SAVE@INTERVAL@BLOCK
0372 SAVE@TOTAL@BLOCK@W@CUMULATIVE@RATE@P
031E SAVE@TOTAL@BLOCK@W@CUMULATIVE@RATE@A
02EA SAVE@TOTAL@BLOCK
02DE CHECK@MIL@INT
01C5 INITIAL@SYSTEM@STATE
01B9 SELECT@AUTO@BUF
01AD SELECT@PRT@BUF
01A7 RESET@SYNTAX@FLAGS
0197 ZERO@INTERNAL@ACCUMS
0187 ZERO@ASCII@TIMER
015D CLEAR@THIS@INT@ACCUMS@PRT
0133 CLEAR@THIS@INT@ACCUMS@AUTO
011A RESET@?@S
00F7 STRING@TO@BUFFER
00DE TIME@ZERO?
00D1 MOVE4
00BB ZERO@TIME@REG
00A8 MOVE@TIME@TO@DIS
0094 PUT@TIME
0076 ACCUM@GE@LIM?
0066 ZERO@ACCUM
0051 ADD@ACCUM
0044 MOVE@ACCUM
003B MOVE@MEMORY
3DC3 UNLOGGED@ERRORS@TO@AUTO
3DC4 UNLOGGED@ERRORS@TO@PRINT
3DC5 SFLAG
3DC6 LOSS@REPORT@CTR
3DC7 INT@CTR
3DC8 Q@COMMAND@P
3DC9 Q@COMMAND@A
3DCA REAL@TIME@A
3E2E APUF1
3E92 REAL@TIME@P
3EAE PBUF1
3EFE BUFFER@INSERT@PTR
3F00 LOG@COUNTER
3F01 TEMP1
3F02 FAULT@FLAG
3F03 BUS@FORMAT
3F04 MASK
3F05 SELECT
3F06 SCALE
3F07 SEPARATOR2
3F08 SEPARATOR1
3F09 FORMAT@SEPARATOR
3F0A FORMAT@INDEX
3F0B COUNTER@15@MIN
3F0D NUMBER@SEQUENTIAL@LOGS
3F0E LOG@MADE@IN@LAST@HOUR
3F0F LETTER
3F10 BURST@FLAG
```

```
3F11 LAST@BLU@COND
3F12 LAST@PRBS@LOSS
3F13 LAST@FRAME@LOSS
3F14 LAST@SIGNAL@LOSS
3F15 NEW@ERROR@MASK
3F16 FPERROR@TO@PERIPH
3F17 MESSAGE@NUMBER
3F18 T@TEMP@P
3F19 T@TEMP@A
3F1A A@COUNTER
3F1B P@COUNTER
3F1C BST3@TMP@P
3F23 BST2@TMP@P
3F2A BST1@TMP@P
3F31 BST3@TMP@A
3F38 BST2@TMP@A
3F3F BST1@TMP@A
3F46 DENOM@MEAS@SEC@P
3F4B DENOM@MEAS@P
3F50 DENOM@MEAS@SEC@A
3F55 DENOM@MEAS@A
3F5A V@MEAS@4P@T7
3F61 V@MEAS@4P@T6
3F68 V@MEAS@4P@T3
3F6F V@MEAS@4P
3F76 V@MEAS@3P@T7
3F7D V@MEAS@3P@T6
3F84 V@MEAS@3P@T3
3F8B V@MEAS@3P
3F92 P@MEAS@4P@T7
3F99 P@MEAS@4P@T6
3FA0 P@MEAS@4P@T3
3FA7 P@MEAS@4P
3FAE P@MEAS@3P@T7
3FB5 P@MEAS@3P@T6
3FBC P@MEAS@3P@T3
3FC3 P@MEAS@3P
3FCA B@MEAS@4P@T7
3FD1 B@MEAS@4P@T6
3FD8 B@MEAS@4P@T3
3FDF B@MEAS@4P
3FE6 B@MEAS@3P@T7
3FED B@MEAS@3P@T6
3FF4 B@MEAS@3P@T3
3FFB B@MEAS@3P
4002 V@MEAS@2P@T7
4009 V@MEAS@2P@T6
4010 V@MEAS@2P@T3
4017 V@MEAS@2P
401E V@MEAS@1P@T7
4025 V@MEAS@1P@T6
402C V@MEAS@1P@T3
4033 V@MEAS@1P
403A P@MEAS@2P@T7
4041 P@MEAS@2P@T6
4048 P@MEAS@2P@T3
404F P@MEAS@2P
4056 P@MEAS@1P@T7
```

```
405D P@MEAS@1P@T6
4064 P@MEAS@1P@T3
406B P@MEAS@1P
4072 B@MEAS@2P@T7
4079 B@MEAS@2P@T6
4087 B@MEAS@2P@T3
4087 B@MEAS@2P
408E B@MEAS@1P@T7
4095 B@MEAS@1P@T6
409C B@MEAS@1P@T3
40A3 B@MEAS@1P
40AA V@MEAS@4A@T7
40B1 V@MEAS@4A@T6
40B8 V@MEAS@4A@T3
40BF V@MEAS@4A
40C6 V@MEAS@3A@T7
40CD V@MEAS@3A@T6
40D4 V@MEAS@3A@T3
40DB V@MEAS@3A
40E2 P@MEAS@4A@T7
40E9 P@MEAS@4A@T6
40F7 P@MEAS@4A@T3
40F7 P@MEAS@4A
40FE P@MEAS@3A@T7
4105 P@MEAS@3A@T6
410C P@MEAS@3A@T3
4113 P@MEAS@3A
411A B@MEAS@4A@T7
4121 B@MEAS@4A@T6
4128 B@MEAS@4A@T3
412F B@MEAS@4A
4136 B@MEAS@3A@T7
413D B@MEAS@3A@T6
4144 B@MEAS@3A@T3
414B B@MEAS@3A
4152 V@MEAS@2A@T7
4159 V@MEAS@2A@T6
4160 V@MEAS@2A@T3
4167 V@MEAS@2A
416E V@MEAS@1A@T7
4175 V@MEAS@1A@T6
417C V@MEAS@1A@T3
4183 V@MEAS@1A
418A P@MEAS@2A@T7
4191 P@MEAS@2A@T6
4198 P@MEAS@2A@T3
419F P@MEAS@2A
41A6 P@MEAS@1A@T7
41AD P@MEAS@1A@T6
41B4 P@MEAS@1A@T3
41BB P@MEAS@1A
41C2 B@MEAS@2A@T7
41C9 B@MEAS@2A@T6
41D0 B@MEAS@2A@T3
41D7 B@MEAS@2A
41DE B@MEAS@1A@T7
41E5 B@MEAS@1A@T6
41EC B@MEAS@1A@T3
```

41F3 B@MEAS@1A
41FA BLOCK@MASK@P
41FB BLOCK@MASK@A
41FC BLOCK@TEMPS@FULL@P
41FD BLOCK@TEMPS@FULL@A
41FE ALPHA1
41FF RX@NEED@ALPHA1
4200 MACHINE@FORMAT
4201 TX@TIME
4202 TX@MODE
4203 TEST@TYPE
4204 GATING@WINK@CTR
4205 AT@LEAST@ONE@TO@BLINK
4206 BLINKER
4207 BLINK@CTR
4208 LAST@TIME@MODE
4209 MO@DISPLAY@CTR
420A TEN@7@8@CTR
420B ASCII@TIMER
4212 TIME@TEMP
421A ELAPSED@TIME
4222 TIMER@START@TIME
422A TIMER
4232 CLOCK@TASK
4233 PRBS@OBTAINED
4234 RESULT@TASK
4235 RESULT@READY
4236 BLINK@SIGNAL@LOSS
4237 BLINK@PRBS@LOSS
4238 BLINK@FRAME@LOSS
4239 FRAME@OBTAINED
423A SIGNAL@OBTAINED
423B RESET@TIMER@NEXT@ UN
423C RESET@TEST@ON@TICK
423D BURST@LETTER
423E RTMP
4245 RDTMP
424A RESULT
424F RESULT@DENOM
4254 ERRORS@TEN@7@8
4259 TOTAL@THIS@SEC@ACC
425E TOTAL@THIS@SECOND
425F TOTAL@SECONDS
4264 TOTAL@BITS
4269 BIPOLAR@VIOLATION@SECS@T7
426F BIPOLAR@VIOLATION@SECS@T6
4273 BIPOLAR@VIOLATION@SECS@T3
4278 BIPOLAR@VIOLATION@SECS
427D TOTAL@BIPOLAR@VIOLATIONS@T7
4282 TOTAL@BIPOLAR@VIOLATIONS@T6
4287 TOTAL@BIPOLAR@VIOLATIONS@T3
428C TOTAL@BIPOLAR@VIOLATIONS
4291 PARITY@ERROR@SECS@T7
4296 PARITY@ERROR@SECS@T6
429B PARITY@ERROR@SECS@T3
42A0 PARITY@ERROR@SECS
42A5 TOTAL@PARITY@ERRORS@T7
42AA TOTAL@PARITY@ERRORS@T6

```
42AF TOTAL@PARITY@ERRORS@T3
42B4 TOTAL@PARITY@ERRORS
42B9 BIT@ERROR@SECS@T7
42BE BIT@ERROR@SECS@T6
42C3 BIT@ERROR@SECS@T3
42C8 BIT@ERROR@SECS
42CD TOTAL@BIT@ERRORS@T7
42D2 TOTAL@BIT@ERRORS@T6
42D7 TOTAL@BIT@ERRORS@T3
42DC TOTAL@BIT@ERRORS
42E1 THIS@INT@BIPOLAR@VIOLATIONS@A
42E6 THIS@INT@BIPOLAR@VIOLATIONS@P
42EB THIS@SEC@BIPOLAR@VIOLATIONS
42F0 BIPOLAR@VIOLATIONS
42F5 BP@INT@P
42F6 BP@INT@A
42F7 BP@SEC@FLAG@T7
42F8 BP@SEC@FLAG@T6
42F9 BP@SEC@FLAG@T3
42FA BP@SEC@FLAG
42FB THIS@INT@PARITY@ERRORS@A
4300 THIS@INT@PARITY@ERRORS@P
4305 THIS@SEC@PARITY@ERRORS
430A PARITY@ERRORS
430F PAR@INT@P
4310 PAR@INT@A
4311 PAR@SEC@FLAG@T7
4312 PAR@SEC@FLAG@T6
4313 PAR@SEC@FLAG@T3
4314 PAR@SEC@FLAG
4315 THIS@INT@BIT@ERRORS@A
431A THIS@INT@BIT@ERRORS@P
431F THIS@SEC@BIT@ERRORS
4324 BIT@ERRORS
4329 BER@INT@P
432A BER@INT@A
432B BER@SEC@FLAG@T7
432C BER@SEC@FLAG@T6
432D BER@SEC@FLAG@T3
432E BER@SEC@FLAG
432F BURST@GT@100
4334 BURST@11@99
4339 BURST@1@10
433E BP@SYNC@CTR
433F BP@SEC@TEMP
4344 PAR@SYNC@CTR
4345 PAR@SEC@TEMP
434A BER@SYNC@CTR
434B BER@SEC@TEMP
4350 CTR@TEMP
4352 LST@BER@CTR
4354 LST@PAR@CTR
4356 LST@BP@CTR
4358 BEFORE@FLOAT@TEMP
4359 FLOAT@TEMP
0003 FLTPT@PACK
436B LAST@BURST@COUNT
436C BURST@COUNT
```

```
436D BURST@LENGTH
436E TRANSMIT@ENABLE
436F TEST@CTR
4370 RS@DELAY@CTR
4371 RS@DELAY@FLAG
4372 FIRST@FLAG
4373 NV@RAM@TEST2
4374 NV@RAM@TEST1
4375 FPBCD@ASM@TEMP
437C DISPLA@ASM@TEMP
4384 PRNG@BUFFER
4398 RBUFFER@OUT@P
439A RBUFFER@OFF@P
439C RBUFFER@IN@P
439E SPC@COUNTER@P
439F NO@POSITIONS@P
43A0 RBUFFER@STRT@P
43A2 BUFFER@TOTAL@P
43A3 BCDNUM
43A7 PRINTER@THERE
43A8 UNFRAMED
43A9 ASYNC
43AA PS@LOCAL@LOCKOUT
43AB RS@232@THERE
43AC ARNG@BUFFER
43C0 RBUFFER@OUT@A
43C2 RBUFFER@OFF@A
43C4 RBUFFER@IN@A
43C6 SPC@COUNTER@A
43C7 NO@POSITIONS@A
43C8 RBUFFER@STRT@A
43CA BUFFER@TOTAL@A
43CB PARALLEL@POLL@MASK
43CC BUS@REMOTE@ENABLE
43CD DATE
43D5 CLOCK
43DD RXCHAR
43DE GPIB@THERE
43DF RUN@ON@TICK
43E0 RUN@STOP
43E1 BLU@CONDITION
43E2 UPDATE@FRONT@PANEL
43E3 SAVE@TIME@DISPLAY
43E4 BER@CTR
43E6 PAR@CTR
43E8 BP@CTR
43EA PRBS@LOSS
43EB FRAME@LOSS
43EC SIGNAL@LOSS
43ED DECODED@INPUT
43EE HORN
43EF HORN@ENABLE
43F0 PRINT@CONTROL
43F1 THRESHOLD
43F2 TRAFFIC@LIGHTS
43F3 RUN@STOP@LIGHT
43F4 ERROR@TYPE
43F5 INPUT@TYPE
```

```
43F6 TIME@MODE
43F7 ERROR@MODE
43F8 ERROR@DISPLAY
43FC TIME@DISPLAY
7D1D ERROR@DESIGNATOR
7D21 T7@HEAD@MAC
7D25 T6@HEAD@MAC
7D29 T3@HEAD@MAC
7D2D T0@HEAD@MAC
7D31 BURST@HEAD
7D58 T6@T7@HEAD
7D6B T7@HEAD
7D7C T6@HEAD
7D8D T0@HEAD
7D9E MES@PD@M
7DA6 T0@S@HEAD
7DBA BERMES
7DBD PESMES
7DC0 ESMES
7DC3 EFFMES
7DC6 ERMES
7DC9 MES@PD
7DD9 MES@UN
7DDF MES@OV
7DE5 MES@CE
7DF1 MES@PW
7DFF MES@T
7E1A MES@R
7E2E MES@S
7E42 MES@L
7E57 MES@NP
7E64 MES@NL
7E71 MES@PK
7E87 MES@FK
7E90 MES@GS
7EA1 MES@BS
7EB2 MES@PL
7EC1 MES@FL
7ED1 MES@SL
7EE2 BIT@MASKS
7EEA J@HFX@SEQUENCE
7EF2 COMMAND@TABLE
7F32 FLOAT
7F33 FIX
7F34 TIME@ERROR
7F38 MAX@CLOCK@DISPLAY
7F3E MAX@TIME@DISPLAY
7F46 TTABLE
7F4E ETABLE
7F52 BLU@CONFIG
7F56 BLANKS
7F5A T3@CONS@2
7F5F T6@CONS@2
7F64 T7@CONS@2
7F69 T3@CONS@1
7F6E T6@CONS@1
7F73 T7@CONS@1
7F78 ZEROS
```

```
7F7D I@HUNDRED
7F82 ONE
7F87 I@TEN
7F8C DUMMY@ALL@PARITY@ERRORS
7F91 DUMMY@ALL@BIT@ERRORS
7F96 DUMMY@PARITY@ERRORS
7F9B DUMMY@BIT@ERRORS
7FA0 FPTEN@NEG11
7FA4 FPTEN@NEG10
7FA8 FPTEN@NEG9
7FAC FPTEN@NEG8
7FB0 FPTEN@NEG7
7FB4 FPTEN@NEG6
7FB8 FPTEN@NEG5
7FBC FPTEN@NEG4
7FC0 FPTEN@NEG3
7FC4 FPTEN@NEG2
7FC8 FPTENTH
7FCC FP1
7FD0 FP10
7FD4 FP100
7FD8 FP1000
7FDC FPTEN4
7FE0 FPTEN5
7FE4 FPTEN6
7FE8 FP@TEN7
7FEC FP@TEN8
7FF0 FPTEN9
7FF4 FPTEN10
7FF8 FPTEN11
7FFC FPTEN12
```

SECTION II. DRIVER LISTINGS

A. CLOCK DRIVERS

```
ASM80.OV4 :F1:RMOD1.ASM PRINT(:LP:) PAGEWIDTH(80) DEBUG


ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0         RMOD     PAGE    1

  LOC  OBJ          LINE        SOURCE STATEMENT

                      1          NAME    RMOD
                      2          PUBLIC  RMOD
  00FF                3 TRU      EQU     0FFH
  0000                4 FAL      EQU     0
                      5 ;
                      6 $NOLIST
                     57 $LIST
                     58 ;
                     59 ;*****************************************************
                     60 ;
                     61 ;    RMOD-
                     62 ;      THIS ROUTINE READS THE NATIONAL REAL TIME CLOCK
                     63 ;  CHIP FOR THE DATE AND MONTH COUNT.IT TRANSFERS THIS
                     64 ;  DATA INTO THE DISPAY RAM AND INTO A SOFTWARE CLOCK.
                     65 ;  INTERRUPTS ARE DISABLED WHILE THE NON VOLATILE RAM
                     66 ;  IS BEING LOADED.
                     67 ;
                     68 ;*****************************************************
                     69 ;
                     70          CSEG
                     71 RMOD:
  0000 21FC43        72          LXI     H,TODRS ;ADDRESS OF DISPLAY RAM
  0003 11CD43        73          LXI     D,DATE  ;DATE ADDRESS OF SW CLOCK
  0006 DB46          74 RM01:    IN      DATEC   ;INPUT DATE COUNT
  0008 47            75          MOV     B,A     ;SAVE IN B
  0009 DB54          76          IN      STATBT  ;WAS IT A ROLL OVER
  000B E601          77          ANI     01
  000D C20600    C   78          JNZ     RM01    ;IF SO JUMP BACK
  0010 F3            79          DI              ;DISABLE INTERRUPTS
  0011 78            8           MOV     A,B
  0012 77            81          MOV     M,A     ;STORE IN DISPLAY CLOCK
  0013 23            82          INX     H
  0014 12            83          STAX    D       ;STORE IN SW CLOCK
  0015 11CF43        84          LXI     D,MONTH ;MONTH ADDRESS OF SW CLOCK
  0018 DB47          85          IN      MONC    ;INPUT MONTHS COUNT
  001A 47            86          MOV     B,A     ;SAME AS DATE
  001B 77            87          MOV     M,A
  001C 23            88          INX     H
  001D 12            89          STAX    D
  001E AF            90          XRA     A       ;BLANK REST OF DISPLAY
  001F 77            91          MOV     M,A
  0020 23            92          INX     H
  0021 77            93          MOV     M,A
  0022 FB            94          EI              ;ENABLE INTERRUPT
  0023 3EFF          95          MVI     A,TRU
  0025 32E243        96          STA     DSFLG
  0028 C9            97          RET             ;RETURN
                     98          END


PUBLIC SYMBOLS
RMOD   C 0000

EXTERNAL SYMBOLS
```

USER SYMBOLS

CBASE	A 0040	CLOCK	A 43D5	CRESET	A 0052	DATE	A 43CD
DATEC	A 0046	DATEL	A 004E	DOWC	A 0045	DOWL	A 004D
DSFLG	A 43E2	FAL	A 0000	FLGAD	A 43E3	GOCOM	A 0055
HRC	A 0044	HRL	A 004C	HUTENC	A 0041	HUTENL	A 0049
INTCON	A 0051	INTST	A 0050	LRESET	A 0053	MINC	A 0043
MINL	A 004B	MONC	A 0047	MONL	A 004F	MONTH	A 43CF
PM01	C 0006	RMOD	C 0000	SDRYIN	A 0056	SECC	A 0042
SECL	A 004A	STATBT	A 0054	TESTMO	A 005F	THOUC	A 0040
THOUL	A 0048	TODRS	A 43FC	TRU	A 00FF		

ASSEMBLY COMPLETE, NO ERRORS

```
ASM80.CV4 :F1:CLOCK.ASM PRINT(:LP:) PAGEWIDTH(80) DEBUG


ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0         MODULE   PAGE    1

  LOC  OBJ          LINE        SOURCE STATEMENT

                       1 $INCLUDE (:F1:CLOCK.EQU)
                    =  2 ;***********************************************
                    =  3 ;
                    =  4 ;          EQUATES FOR THE NATIONAL MM58167
                    =  5 ;             REAL TIME CLOCK CHIP
                    =  6 ;
                    =  7 ;***********************************************
                    =  8 ;
  0040              =  9 CBASE    EQU     40H ;BASE ADDRESS OF CLOCK
  0040              = 10 THOUC    EQU     CBASE   ;THOUSANDTHS OF SEC. CNTR.
  0041              = 11 HUTENC   EQU     CBASE+1 ;HUNDRETHS & TENTHS SEC. CNTR.
  0042              = 12 SECC     EQU     CBASE+2 ;SECONDS COUNTER
  0043              = 13 MINC     EQU     CBASE+3 ;MINUTES COUNTER
  0044              = 14 HRC      EQU     CBASE+4 ;HOURS COUNTER
  0045              = 15 DOWC     EQU     CBASE+5 ;DAY OF WEEK COUNTER
  0046              = 16 DATEC    EQU     CBASE+6 ;DATE COUNTER
  0047              = 17 MONC     EQU     CBASE+7 ;MONTH COUNTER
  0048              = 18 THOUL    EQU     CBASE+8 ;THOU. OF SEC. LATCH
  0049              = 19 HUTENL   EQU     CBASE+9 ;HUN. AND TENTHS OF SEC. LATCH
  004A              = 20 SECL     EQU     CBASE+10 ;SECONDS LATCH
  004B              = 21 MINL     EQU     CBASE+11 ;MINUTES LATCH
  004C              = 22 HRL      EQU     CBASE+12 ;HOURS LATCH
  004D              = 23 DOWL     EQU     CBASE+13 ;DAY OF WEEK LATCH
  004E              = 24 DATEL    EQU     CBASE+14 ;DATE LATCH
  004F              = 25 MONL     EQU     CBASE+15 ;MONTH LATCH
  0050              = 26 INTST    EQU     CBASE+16 ;INTERRUPT STATUS
  0051              = 27 INTCON   EQU     CBASE+17 ;INTERRUPT CONTROL
  0052              = 28 CRESET   EQU     CBASE+18 ;COUNTER RESET
  0053              = 29 LRESET   EQU     CBASE+19 ;LATCH RESET
  0054              = 30 STATBT   EQU     CBASE+20 ;STATUS BIT
  0055              = 31 GOCOM    EQU     CBASE+21 ;GO COMMAND
  0056              = 32 SDBYIN   EQU     CBASE+22 ;STANDBY INTERRUPT
  005F              = 33 TESTMO   EQU     CBASE+31 ;TEST MODE
                    = 34 ;
                    = 35 ;
                    = 36 ;
                    = 37 ;***************************************************
                    = 38 ;
                    = 39 ;  COMMON MEMORY EQUATES FOR THE FORTRAN
                    = 40 ;
                    = 41 ;***************************************************
                    = 42 ;
  43E2              = 43 DSFLG    EQU     43E2H   ;FLAG TO INDICATE DISPLAY NEEDS
                    = 44                          ;UPDATE
  43FC              = 45 TODRS    EQU     43FCH   ;DISPLAY BUFFER- WHAT IS IN HERE
                    = 46                          ;GOES OUT TO DISPLAY
  43CD              = 47 DATE     EQU     43CDH   ;SOFTWARE CLOCK- THE DATE
  43CF              = 48 MONTH    EQU     43CFH   ;SOFTWARE CLOCK- THE MONTH
  43D5              = 49 CLOCK    EQU     43D5H   ;SOFTWARE CLOCK- THE TIME OF DAY
  43E3              = 50 FLGAD    EQU     43E3H   ;FLAG TO INDICATE VALID SETTING
                       51         END

PUBLIC SYMBOLS
```

EXTERNAL SYMBOLS

USER SYMBOLS

CBASE	A 0040	CLOCK	A 43D5	CRESET	A 0052	DATE	A 43CD
DATEC	A 0046	DATEL	A 004E	DOWC	A 0045	DOWL	A 004D
DSFLG	A 43E2	FLGAD	A 43E3	GOCOM	A 0055	HRC	A 0044
HRL	A 004C	HUTENC	A 0041	HUTENL	A 0049	INTCON	A 0051
INTST	A 0050	LRESET	A 0053	MINC	A 0043	MINL	A 004B
MONC	A 0047	MONL	A 004F	MONTH	A 43CF	SDBYIN	A 0056
SECC	A 0042	SECL	A 004A	STATBT	A 0054	TESTMO	A 005F
THOUC	A 0040	THOUL	A 0048	TODRS	A 43FC		

ASSEMBLY COMPLETE, NO ERRORS

```
ASM80.OV4 :F1:RTOD1.ASM PRINT(:LP:) PAGEWIDTH(80) DEBUG


ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0         RTOD     PAGE    1

  LOC  OBJ          LINE        SOURCE STATEMENT

                      1          NAME    RTOD
                      2          PUBLIC  RTOD
  00FF                3 TRU      EQU     0FFH
  0000                4 FAL      EQU     00H
                        ;
                      6 $NOLIST
                     57 $LIST
                     58 ;
                     59 ;*****************************************************
                     60 ;
                     61 ;  RTOD-
                     62 ;    THIS ROUTINE READS SECONDS,MINUTES,AND HOURS
                     63 ; FROM THE NATIONAL CLOCK CHIP. THE STATUS BIT IS
                     64 ; CHECKED EVERY READ AND IF IS TRUE, THE READS UP
                     65 ; TO THAT POINT ARE REPEATED. INTERRUPTS ARE DISABLED
                     66 ; WHILE LOADING THE NON-VOLATILE RAM WITH THE CLOCK
                     67 ; INFORMATION. ALL VALUES ARE BCD, PACKED TWO PER
                     68 ; BYTE.
                     69 ;
                     70 ;*****************************************************
                     71 ;
                     72          CSEG
                     73 RTOD:
  0000 21FC43        74          LXI     H,TODRS ;TIME OF DAY REG. FOR DISPLAY
  0003 11D543        75          LXI     D,CLOCK ;SOFTWARE CLOCK RAM
  0006 DB42          76 RT01:    IN      SECC    ;INPUT SECONDS COUNTER
  0008 47            77          MOV     B,A     ;SAVE IN B
  0009 DB54          78          IN      STATBT  ;SEE IF NO ROLL OVER
  000B E601          79          ANI     01
  000D C20600    C   80          JNZ     RT01    ;IF THERE IS ROLL OVER,JUMP BACK
  0010 78            81          MOV     A,B
  0011 F3            82          DI              ;DISABLE INTERRUPTS
  0012 77            83          MOV     M,A     ;STORE IN RAM
  0013 23            84          INX     H
  0014 12            85          STAX    D
  0015 13            86          INX     D
  0016 13            87          INX     D       ;TWO BYTES PER FORTRAN VAR.
  0017 DB43          88 RT02:    IN      MINC    ;INPUT MINUTES COUNTER
  0019 47            89          MOV     B,A
  001A DB54          9           IN      STATBT  ;SAME AS SECONDS
  001C E601          91          ANI     01
  001E C20000    C   92          JNZ     RTOD
  0021 78            93          MOV     A,B
  0022 77            94          MOV     M,A
  0023 23            95          INX     H
  0024 12            96          STAX    D
  0025 13            97          INX     D
  0026 13            98          INX     D
  0027 DB44          99 RT03:    IN      HRC     ;INPUT HOURS COUNTER
  0029 47           100          MOV     B,A
  002A DB54         101          IN      STATBT  ;SAME AS SECONDS
  002C E601         102          ANI     01
  002E C20000    C  103          JNZ     RTOD
  0031 78           104          MOV     A,B
```

```
 LOC  OBJ          LINE        SOURCE STATEMENT

 0032 E6F0          105          ANI     0F0H     ;BLANK 10'S HOURS IF CAN
 0034 C23B00    C   106          JNZ     RT04
 0037 78            107          MOV     A,B
 0038 F6F0          108          ORI     0F0H
 003A 47            109          MOV     B,A
 003B 70            110 RT04:    MOV     M,B
 003C 78            111          MOV     A,B
 003D 12            112          STAX    D
 003E 23            113          INX     H
 003F 3EFF          114          MVI     A,0FFH
 0041 77            115          MOV     M,A
 0042 13            116          INX     D
 0043 13            117          INX     D        ;SETTING 100'S HOURS TO ZERO
 0044 12            118          STAX    D
 0045 FB            119          EI               ;ENABLE INTERRUPTS
 0046 3EFF          120          MVI     A,TRU    ;SET DISPLAY UPDATE FLAG
 0048 32E243        121          STA     DSFLG
 004B C9            122          RET              ;RETURN
                    123          END

PUBLIC SYMBOLS
RTOD   C 0000

EXTERNAL SYMBOLS

USER SYMBOLS
CBASE  A 0040    CLOCK  A 43D5    CRESET A 0052    DATE   A 43CD
DATEC  A 0046    DATEL  A 004E    DOWC   A 0045    DOWL   A 004D
DSFLG  A 43E2    FAL    A 0000    FLGAD  A 43E3    GOCOM  A 0055
HRC    A 0044    HRL    A 004C    HUTENC A 0041    HUTENL A 0049
INTCON A 0051    INTST  A 0050    LRESET A 0053    MINC   A 0043
MINL   A 004B    MONC   A 0047    MONL   A 004F    MONTH  A 43CF
RT01   C 0006    RT02   C 0017    RT03   C 0027    RT04   C 003B
RTOD   C 0000    SDBYIN A 0056    SECC   A 0042    SECL   A 004A
STATBT A 0054    TESTMO A 005F    THOUC  A 0040    THOUL  A 0048
TODRS  A 43FC    TRU    A 00FF

ASSEMBLY COMPLETE,  NO ERRORS
```

```
ASM80.OV4 :F1:STOD1.ASM PRINT(:LP:) PAGEWIDTH(80) DEBUG


ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        STOD     PAGE    1

  LOC  OBJ          LINE        SOURCE STATEMENT

                      1          NAME    STOD
                      2          PUBLIC  STOD
                      3          EXTRN   RTOD
                      4 ;
                      5 $NOLIST
                     56 $LIST
                     57 ;
                     58 ;*****************************************************
                     59 ;
                     60 ;    STOD-
                     61 ;      THIS ROUTINE SETS THE NATIONAL CLOCK CHIP FROM
                     62 ;  THE DISPLAY RAM.
                     63 ;
                     64 ;*****************************************************
                     65 ;
                     66          CSEG
                     67 STOD:
  0000 21FC43        68          LXI     H,TODRS ;SET DISPAY RAM ADDRESS
  0003 7E            69          MOV     A,M
  0004 D342          70          OUT     SECC    ;SET SECONDS
  0006 23            71          INX     H
  0007 7E            72          MOV     A,M
  0008 D343          73          OUT     MINC    ;SET MINUTES
  000A 23            74          INX     H
  000B 7E            75          MOV     A,M
  000C D344          76          OUT     HRC     ;SET HOURS
  000E C30000    E   77          JMP     RTOD    ;READ IT BACK TO SET SOFTWARE
                     78                          ;CLOCK
                     79          END


PUBLIC SYMBOLS
STOD   C 0000

EXTERNAL SYMBOLS
RTOD   E 0000

USER SYMBOLS
CBASE  A 0040    CLOCK  A 43D5    CRESET A 0052    DATE   A 43CD
DATEC  A 0046    DATEL  A 004E    DOWC   A 0045    DOWL   A 004D
DSFLG  A 43E2    FLGAD  A 43E3    GOCOM  A 0055    HRC    A 0044
HRL    A 004C    HUTENC A 0041    HUTENL A 0049    INTCON A 0051
INTST  A 0050    LRESET A 0053    MINC   A 0043    MINL   A 004B
MONC   A 0047    MONL   A 004F    MONTH  A 43CF    RTOD   E 0000
SDBYIN A 0056    SECC   A 0042    SECL   A 004A    STATBT A 0054
STOD   C 0000    TESTMO A 005F    THOUC  A 0040    THOUL  A 0048
TODRS  A 43FC

ASSEMBLY COMPLETE,  NO ERRORS
```

```
ASM80.OV4 :F1:SMOD1.ASM PRINT(:LP:) PAGEWIDTH(80) DEBUG


ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0         SMOD     PAGE    1


  LOC  OBJ          LINE        SOURCE STATEMENT

                      1          NAME    SMOD
                      2          PUBLIC  SMOD
                      3          EXTRN   RMOD
                      4 ;
                      5 $NOLIST
                     56 $LIST
                     57 ;
                     58 ;*******************************************************
                     59 ;
                     60 ;   SMOD-
                     61 ;     THIS ROUTINE SETS THE MONTH AND DATE FROM THE
                     62 ;  DISPLAY RAM.
                     63 ;
                     64 ;*******************************************************
                     65 ;
                     66          CSEG
                     67 SMOD:
  0000 21FC43        68          LXI     H,TODRS ;DISPLAY RAM ADDRESS
  0003 7E            69          MOV     A,M     ;GET DATE MONTH
  0004 47            70          MOV     B,A
  0005 A7            71          ANA     A
  0006 C8            72          RZ              ;RETURN IF BAD
  0007 23            73          INX     H       ;GO TO MONTH COUNT
  0008 7E            74          MOV     A,M
  0009 5F            75          MOV     E,A     ;SAVE IN REG. E
  000A A7            76          ANA     A       ;IF ZERO ERROR
  000B C8            77          RZ
  000C FE13          78          CPI     13H
  000E D0            79          RNC             ;IF OVER 12 THEN RETURN
  000F 212500    C   80          LXI     H,DTAB  ;SET TABLE FOR DATE LIMITS
  0012 1600          81          MVI     D,0     ;SET D=0
  0014 19            82          DAD     D       ;ADD MONTH COUNT TO TABLE
  0015 7E            83          MOV     A,M     ;GET MAX DATE FOR THAT MONTH
  0016 B8            84          CMP     B       ;COMPARE TO DATE
  0017 D8            85          RC              ;RETURN IF OVER LIMIT
  0018 78            86          MOV     A,B
  0019 D346          87          OUT     DATEC   ;OTHERWISE SET DATE
  001B 7B            88          MOV     A,E
  001C D347          89          OUT     MONC    ;AND SET MONTH
  001E AF            90          XRA     A       ;RESET VALID MONTH FLAG
  001F 32E343        91          STA     FLGAD
  0022 C30000    E   92          JMP     RMOD    ;READ MONTH/DATE TO SET
                     93                          ;SOFTWARE CLOCK
                     94 DTAB:
  0025 00            9           DB      0       ;0 COUNT ERROR
  0026 31            96          DB      31H     ;JANUARY
  0027 28            97          DB      28H     ;FEBRUARY
  0028 31            98          DB      31H     ;MARCH
  0029 30            99          DB      30H     ;APRIL
  002A 31           100          DB      31H     ;MAY
  002B 30           101          DB      30H     ;JUNE
  002C 31           102          DB      31H     ;JULY
  002D 31           103          DB      31H     ;AUGUST
  002E 30           104          DB      30H     ;SEPTEMBER
```

```
  LOC  OBJ          LIN          SOURCE STATEMENT

  002F 00            105          DB      0        ;SINCE BCD-MAKE A-F ZERO
  0030 00            106          DB      0
  0031 00            107          DB      0
  0032 00            108          DB      0
  0033 00            109          DB      0
  0034 00            110          DB      0
  0035 31            111          DB      31H      ;OCTOBER
  0036 30            112          DB      30H      ;NOVEMBER
  0037 31            11           DB      31H      ;DECEMBER
                     114 ;
                     115 ;
                     116          END


PUBLIC SYMBOLS
SMOD    C 0000

EXTERNAL SYMBOLS
RMOD    E 0000

USER SYMBOLS
CBASE  A 0040    CLOCK  A 43D5    CRESET A 0052    DATE   A 43CD
DATEC  A 0046    DATEL  A 004E    DOWC   A 0045    DOWL   A 004D
DSFLG  A 43E2    DTAB   C 0025    FLGAD  A 43E3    GOCOM  A 0055
HRC    A 0044    HRL    A 004C    HUTENC A 0041    HUTENL A 0049
INTCON A 0051    INTST  A 0050    LRESET A 0053    MINC   A 0043
MINL   A 004B    MONC   A 0047    MONL   A 004F    MONTH  A 43CF
RMOD   E 0000    SDBYIN A 0056    SECC   A 0042    SECL   A 004A
SMOD   C 0000    STATBT A 0054    TESTMO A 005F    THOUC  A 0040
THOUL  A 0048    TODRS  A 43FC

ASSEMBLY COMPLETE,   NO ERRORS
```

B. PANEL DRIVERS

```
ASM80.OV4 :F1:LEDTES.ASM PRINT(:LP:) PAGEWIDTH(80) DEBUG


ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0         LEDTES   PAGE    1

  LOC  OBJ          LINE        SOURCE STATEMENT

                      1          NAME    LEDTES
                      2          PUBLIC  LEDTES
                      3 ;
  436F                  LEDCNT   EQU     436FH
  0070                5 DSPLY    EQU     70H
  0071                6 CNTRL    EQU     71H
                      7 ;
                      8 ;************************************************
                      9 ;
                     10 ;   LEDTES-
                     11 ;      THIS ROUTINE SETS THE DISPLAY RAM TO CAUSE
                     12 ; ALL LEDS TO BE TURNED ON. IT THEN DISABLES UPDATE
                     1  ; OF THESE RAM LOCATIONS OF THE 8279. THE ROUTINE
                     14 ; SETS A COMMON RAM COUNTER (LOADED FOR 3 SECONDS)
                     15 ; WHICH IS DECREMENTED IN THE ONE SECOND INTERRUPT
                     16 ; ROUTINE. ONCE THE COUNTER REACHES ZERO IN THE ONE
                     17 ; SECOND INTERRUPT ROUTINE, THE DISPLAY RAM IS THEN
                     18 ; ALLOWED TO UPDATE PROPERLY.
                     19 ;
                     20 ;   INTERRUPTS ARE NOT DISABLED BECAUSE IT IS ASSUMED
                     21 ; THAT INTERRUPTS ARE ALREADY DISABLED WHEN THIS ROU-
                     22 ; TINE IS USED.
                     23 ;
                     24 ;************************************************
                     25 ;
                     26          CSEG
                     27 LEDTES:
  0000 3E03          28          MVI     A,3       ;LOAD COMMON RAM COUNTER
  0002 326F43        29          STA     LEDCNT
  0005 0E0D          30          MVI     C,13      ;COUNTER FOR NUMBER OF DISPLAY
                     31                            ;RAM LOCATIONS
  0007 3E90          32          MVI     A,90H     ;SETS 8279 TO RECIEVE DISPLAY
                     33                            ;DATA
  0009 D371          34          OUT     CNTRL
  000B 3EFF          35          MVI     A,0FFH    ;DATA THAT WILL TURN ON ALL LEDS
  000D D370          36 LT1:     OUT     DSPLY     ;OUTPUT TO 8279
  000F 0D            37          DCR     C
  0010 C20D00    C   38          JNZ     LT1
  0013 3EF7          39          MVI     A,0F7H    ;SO HORN WONT COME ON
  0015 D370          40          OUT     DSPLY
  0017 D370          41          OUT     DSPLY
  0019 3EAC          42          MVI     A,0ACH    ;THIS COMMAND INHIBITS UPDATE TO

                     43                            ;DISPLAY RAM
  001B D371          44          OUT     CNTRL
  001D C9            45          RET               ;RETURN
                     46          END


PUBLIC SYMBOLS
LEDTES C 0000

EXTERNAL SYMBOLS
```

```
USER SYMBOLS
CNTRL  A 0071    DSPLY  A 0070    LEDCNT A 436F    LEDTES C 0000
LT1    C 000D

ASSEMBLY COMPLETE,  NO ERRORS
```

```
ASM80.OV4 :F1:SWTCH1.ASM PRINT(:LP:) PAGEWIDTH(80) DEBUG


ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0         SWITCH   PAGE    1
8279 INPUT ROUTINE

  LOC  OBJ          LINE        SOURCE STATEMENT

                      1 $TITLE ('8279 INPUT ROUTINE')
                      2
                      3         NAME    SWITCH
                      4
                      5         PUBLIC  SWITCH
                      6
  43DE                  REMFL   EQU     43DEH   ;REMOTE ENABLE FOR 68488
  8000                  8 BASE  EQU     8000H   ;BASE ADDRESS FOR 68488
  43AA                  9 RSLL  EQU     43AAH   ;RS232 LOCAL LOCKOUT FLAG
  43AB                 10 RSTHER EQU    43ABH   ;RS232 PRESENT FLAG
  0071                 11 CNTRSW EQU    071H
  0070                 12 SWVAL  EQU    070H
  43ED                 13 SWREG  EQU    043EDH
  000C                  1 INADD  EQU    000CH
                       15         CSEG
                       16
                       17 ;*****************************************************
                        1 ;
                       19 ;    SWITCH
                       20 ;      THIS INTERRUPT DRIVEN ROUTINE ACCESSES THE
                        2 ;   8279, DETERMINES WHICH SWITCH WAS OPERATED,
                       22 ;   DECODES THE RESULT, STORES IT, AND INITIALIZES
                       23 ;   THE INPUT ROUTINE OF THE MAIN PROGRAM.
                       24 ;
                       25 ;*****************************************************
                       26 ;
                       27
  0000 3E50            28 SWITCH: MVI    A,050H  ;LOAD 8279 CONTROL WORD
  0002 D371            29         OUT    CNTRSW  ;SEND CONTROL WORD
  0004 DB70            30         IN     SWVAL   ;INPUT ENCODED SWITCH VALUE
  0006 47              31         MOV    B,A
                       32
                       33 ;------
  0007 3ADE43          34         LDA    REMFL   ;SEE IF GPIB HAS LOCAL LOCKOUT
  000A A7              35         ANA    A
  000B CA1700    C     36         JZ     SKP5    ;JUMP OVER IF NOT REN
  000E 3A0180          37         LDA    BASE+1  ;GET LOK BIT IN 68488
  0011 E620            38         ANI    20H
  0013 C0              39         RNZ            ;IF LLO THEN RETURN
  0014 C32300    C     40         JMP    DECOD   ;JUMP OVER RS232 TEST
  0017 3AAB43          41 SKP5:   LDA    RSTHER  ;LOAD FLAG
  001A A7              42         ANA    A
  001B CA2300    C     43         JZ     DECOD   ;IF NOT THERE JUMP TO DECODE
  001E 3AAA43          44         LDA    RSLL    ;CHECK LOCKOUT FLAG
  0021 A7              45         ANA    A
  0022 C0              46         RNZ            ;RETRUN IF LOCKED OUT
                       47 ;-------
                       48
                       49 ;*****************************************************
                       50 ;
                       51 ;  DECOD
                       52 ;    THIS ROUTINE DECODES THE SWITCH POSITON VALUE OF
                       53 ;  THE 8279, TRANSLATES THE VALUE FOR THE MAIN PROG-
                        5 ;  GRAM, AND STORES THIS VALUE IN A COMMON RAM LOCA-
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

                    55 ;  TION.THE ENCODED SWITCH POSITION IS ENTERED INTO
                    56 ;  THE ACCUMULATOR.
                    57 ;
                    58 ;*************************************************
0023 78             59 DECOD:  MOV     A,B      ;RESTORE INPUT BYTE
0024 07             60         RLC              ;ROTATE SET BIT TO PROPER
                    61                          ;POSITION FOR INPUT ROUTINE
0025 2F             62         CMA              ;COMPLEMENT THE SET BIT
0026 E680           63         ANI     80H      ;MASK OUT ALL OTHER BITS
0028 4F             64         MOV     C,A      ;STORE SET BIT IN REG. C
0029 78             65         MOV     A,B      ;GET SWITCH VALUE
002A E61F           66         ANI     1FH      ;MASK OUT SET SWITCH
002C FE0B           67         CPI     0BH      ;COMPARE TO "B" BECAUSE ALL
                    68                          ;VALUES UNDER B CORRESPOND
                    69                          ;EXACTLY WITH INPUT ROUTINE.
002E DA3E00    C    70         JC      FIN      ;JUMP TO FINISH IF UNDER "B"
0031 B7             71         ORA     A        ;CLEAR CARRY
0032 DE0B           72         SBI     0BH      ;SUBTRACT ELEVEN FROM ACCUMU-
                    73                          ;LATOR SO THAT TABLE ADDRESSES
                    74                          ;WILL BE COMPATIBLE
0034 214700    C    75         LXI     H,TEXT1  ;LOAD BEGINNING TABLE
                    76                          ;ADDRESS
0037 85             77         ADD     L        ;ADD TEXT ADDRESS TO IT
0038 6F             78         MOV     L,A      ;RESTORE TEXT ADDRESS
0039 3E00           79         MVI     A,0      ;CLEAR ACCUM WITHOUT
                    80                          ;DISTURBING FLAGS
003B 8C             81         ADC     H        ;ADD THE UPPER PORTION OF
                    82                          ;THE TEXT ADDRESS
003C 67             83         MOV     H,A      ;RESTORE UPPER PORTION-TEXT
                    84                          ;ADDRESS IS NOW COMPLETE
003D 7E             85         MOV     A,M      ;STORE TEXT VALUE IN ACCUM
003E B1             86 FIN:    ORA     C
003F 21ED43         87         LXI     H,SWREG  ;LOAD COMMON RAM LOCATION
0042 77             88         MOV     M,A      ;STORE DECODED VALUE
0043 CD0C00         89         CALL    INADD    ;CALL INPUT ROUTINE
0046 C9             90         RET              ;RETURN
                    91 ;*************************************************
                    92 ;
                    93 ;  TABLE OF CONVERSION VALUES
                    94 ;
                    95 ;*************************************************
                    96 ;
                    97 TEXT1:                   ;TABLE FOR TRANSLATING SWITCH
                    98                          ;VALUES
0000                99 MODA    EQU     $ - TEXT1
0047 0E            100         DB      0EH      ;MONTH-DAY SWITCH VALUE
0001               101 TOD     EQU     $ - TEXT1
0048 0F            102         DB      0FH      ;TIME OF DAY VALUE
0002                10 ELAPS   EQU     $ - TEXT1
0049 10            104         DB      10H      ;ELAPS TIME
0003               105 SING    EQU     $ - TEXT1
004A 11            106         DB      11H      ;SINGLE TIME VALUE
0004               107 REP     EQU     $ - TEXT1
004B 12            108         DB      12H      ;REPEATED TIME
0005               109 LOHI    EQU     $ - TEXT1
```

```
  LOC  OBJ          LINE        SOURCE STATEMENT

  004C 0B            110          DB      0BH      ;ERROR INPUT TYPE LOW
  0006               111 DSX3     EQU     $ - TEXT1
  004D 0C            112          DB      0CH      ;DSX3 INPUT TYPE
  0007               113 HORNE    EQU     $ - TEXT1
  004E 0D            114          DB      0DH      ;HORN ENABLE SWITCH
                     115          END
```

PUBLIC SYMBOLS
SWITCH C 0000

EXTERNAL SYMBOLS

USER SYMBOLS

BASE	A	8000	CNTRSW	A	0071	DECOD	C	0023	DSX3	A	0006
ELAPS	A	0002	FIN	C	003E	HORNE	A	0007	INADD	A	000C
LOHI	A	0005	MODA	A	0000	REMFL	A	43DE	REP	A	0004
RSLL	A	43AA	RSTHER	A	43AB	SING	A	0003	SKP5	C	0017
SWITCH	C	0000	SWREG	A	43ED	SWVAL	A	0070	TEXT1	C	0047
TOD	A	0001									

ASSEMBLY COMPLETE, NO ERRORS

```
ASM80.OV4 :F1:DISPL1.ASM PRINT(:LP:) PAGEWIDTH(80) DEBUG


ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        DISPLA   PAGE    1

  LOC  OBJ          LINE        SOURCE STATEMENT

                      1          NAME    DISPLA
                      2          PUBLIC  DISPLA
  43FF                3 TODRH    EQU     043FFH
  4383                4 RMVAL    EQU     04383H
  4382                5 RMVAL1   EQU     04382H
  43FB                6 ERREG    EQU     043FBH
  0070                7 DSPLY    EQU     70H
  0071                8 CNTRL    EQU     71H
  43E0                9 RUNSTP   EQU     43E0H
  43F2               10 TRFLT    EQU     43F2H
  43E2                1  DSFLG   EQU     43E2H
  0000               12 FAL      EQU     00H
  0094               13 INOUT    EQU     94H
  43A8               14 UNFRAM   EQU     43A8H
  43EF               15 HORNEN   EQU     43EFH
  43EE               16 HORN     EQU     43EEH
  43F1               17 THRESH   EQU     43F1H
  43F0               18 PRINTC   EQU     43F0H
                     19          CSEG
                     20 ;***********************************************
                     21 ;
                     22 ;   DISPLA
                     23 ;     THIS ROUTINE EXTRACTS DATA FROM THE COMMON
                     24 ; RAM LOCATIONS, DECODES AND FORMATS THE DATA,
                     25 ; AND SENDS IT OUT TO THE 8279 DISPLAY RAM.
                     26 ;   THE COMMON RAM LOCATIONS AND THEIR CONTENTS
                     27 ; ARE AS FOLLOWS:
                     28 ;      TIME DISPLAY
                     29 ;    100 HRS. (BCD)  - 3FFFH
                     30 ;    HOURS (PBCD)     - 3FFEH
                     31 ;    MINUTES (PBCD)  - 3FFDH
                     32 ;    SECONDS (PBCD)  - 3FFCH
                     33 ;      ERROR DISPLAY
                     34 ;    ERROR ONES (BCD)   - 3FFBH
                     35 ;    ERROR TENTHS (BCD) - 3FFAH
                     36 ;    ERROR HUNDTHS.(BCD)- 3FF9H
                     37 ;    ERROR EXPONENT(BCD)- 3FF8H
                     38 ;
                     39 ;    TRAFFIC LIGHTS   - 3FF2H
                     40 ;      BIT 0   - AUTO CONTROL
                     41 ;          1   - SIGNAL LOSS
                     42 ;          2   - FRAME LOSS
                     43 ;          3   - PRBS LOSS
                     44 ;          4   - GATING
                     45 ;    RUN/STOP          - 3FF3H
                     46 ;      BIT 0   - RUN
                     47 ;          1   - STOP
                     48 ;    ERROR TYPE        - 3FF4H
                     4  ;      BIT 0   - BIT
                     50 ;          1   - PARITY
                     51 ;          2   - BIPOLAR VIOL.
                     52 ;    INPUT TYPE        - 3FF5H
                     53 ;      BIT 0   - LOW
                     54 ;          1   - DSX-3
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

                    55 ;          2   - HIGH
                    56 ;     TIME MODE          - 3FF6H
                    57 ;       BIT 0   - TIME OF DAY
                    58 ;            1   - ELAPSED
                    59 ;            2   - SINGLE
                    60 ;            3   - REPEAT
                    61 ;     ERROR MODE         - 3FF7H
                    62 ;       BIT 0   - ERROR SEC
                    63 ;            1   - TOTAL
                    64 ;            2   - CUM. RATE
                    65 ;            3   - 10**7
                    66 ;            4   - 10**8
                    67 ;            5   - % ERROR SECONDS
                    68 ;     ***************************************
                    69 ;          NEW VARIABLES ADDED
                    70 ;     ***************************************
                    71 ;      THRESHOLD                -3FF1H
                    72 ;        BIT 0  - 0 THRESHOLD
                    73 ;        BIT 1  - 10E-7  THRESHOLD
                    74 ;        BIT 2  - 10E-6  THRESHOLD
                    75 ;        BIT 3  - 10E-3  THRESHOLD
                    76 ;
                    77 ;      HORN                     -3FEEH
                    78 ;        BIT 0  - HORN ON/OFF
                    79 ;
                    80 ;      HORN ENABLE              -3FEFH
                    8  ;        BIT 0  - HORN ENABLE ON/OFF
                    82 ;
                    83 ;      PRINT CONTROL            -3FF0H
                    84 ;        BIT 0  - ALL
                    85 ;        BIT 1  - PARTIAL
                    86 ;
                    87 ;
                    88 ;  THE OUTPUT OF THE DISCRETE LED SIGNALS ARE
                    89 ; AS FOLLOWS:
                    90 ;
                    91 ;   DISPLAY RAM ADD. OF 8279- BH
                    92 ;    BIT 0- LOW INPUT
                    93 ;         1- DSX-3 INPUT
                    94 ;         2- HIGH INPUT
                    95 ;         3- AUTO CONTROL
                    96 ;         4- TIME OF DAY
                    97 ;         5- ELAPSED
                    98 ;         6- SINGLE
                    99 ;         7- REPEAT
                   100 ;   DISPLAY RAM ADD. OF 8279- CH
                   101 ;    BIT 0- BIPOLAR VIOLATION
                   102 ;         1- SIGNAL LOSS
                   103 ;         2- FRAME LOSS
                   104 ;         3- PRBS SYNC LOSS
                   105 ;         4- RUN
                   106 ;         5- STOP
                   107 ;         6- BIT
                   108 ;         7- PARITY
                   109 ;   DISPLAY RAM ADD. OF 8279- DH
```

```
 LOC  OBJ          LINE        SOURCE STATEMENT

                    110 ;    BIT 0- ERROR SECONDS
                    111 ;        1- % ERROR SECONDS
                    112 ;        2- GATING
                    113 ;        3- HORN ON/OFF
                    114 ;        4- 0 THRESHOLD
                    115 ;        5- 10E-7 TH.
                    116 ;        6- 10E-6 TH.
                    117 ;        7- 10E-3 TH.
                    118 ;
                    119 ;   DISPLAY RAM ADD. OF 8279- EH
                    120 ;   BIT 0- ALL PRINT
                    121 ;        1- PARTIAL PRINT
                    122 ;        2- HORN ENABLE ON/OFF
                    123 ;        3- SPARE
                    12  ;        4- TOTAL ERRORS
                    125 ;        5- AVG BER
                    126 ;        6- ERROR  10E-7
                    127 ;        7- ERROR  10E-8
                    128 ;
                    129 ;
                    130 ;*****************************************
                    131 ;
0000 21FF43         132 DISPLA: LXI     H,TODRH ;LOAD POINTER WITH HUNDRED
                    133                         ;HOURS ADDRESS
0003 118343         134         LXI     D,RMVAL ;LOAD D&E WITH TEMP. RAM
                    135                         ;FOR STORING THE UNPACKED
                    136                         ;TIME DATA
0026 0604           137         MVI     B,04H   ;STORE COUNT IN REG. B
0008 7E             138 LOOP:   MOV     A,M     ;MOVE DISPLAY DATA INTO
                    139                         ;ACCUM.
0009 EB             140         XCHG            ;EXCHANGE POINTERS
000A CD7200    C    141         CALL    HXDSP   ;CALL ROUTINE TO UNPACK THE
                    142                         ;PACKED TIME DATA
000D EB             143         XCHG            ;EXCHANGE POINTERS
000E 2B             144         DCX     H       ;DECREMENT COMMON RAM POINTER
000F 1B             145         DCX     D       ;INCREMENT TEMP. RAM POINTER
0010 05             146         DCR     B       ;DECREMENT COUNT
0011 C20800    C    147         JNZ     LOOP    ;REPEAT TILL FINISHED
                    148 ;NOW CHECK TO SEE IF CAN BLANK OUT ANY LEADING ZEROS
0014 218243         149         LXI     H,RMVAL1;LOAD MEMORY POINTER TO
                    150                         ;RMVAL-1 LOCATION SINCE
                    151                         ;ONLY ONE NIBBLE USED IN
                    152                         ;MOST SIGNIFICANT BYTE
0017 0E05           153         MVI     C,5     ;STORE COUNT OF HOW MANY
                    154                         ;BYTES TO CHECK
0019 7E             155 FIR:    MOV     A,M     ;MOVE DISPLAY CONTENT INTO
                    156                         ;ACCUM.
001A FE00           157         CPI     00H     ;COMPARE TO ZERO
001C C22600    C    158         JNZ     OT4     ;JUMP OUT IF NOT ZERO
001F 362F           159         MVI     M,0FH   ;IF ZERO REPLACE WITH
                    160                         ;BLANK CHARACTER
0021 2B             161         DCX     H       ;DECREMENT MEMORY POINTER
0022 0D             162         DCR     C       ;DECREMENT COUNT
0023 C21900    C    163         JNZ     FIR     ;JUMP BACK IF COUNT NOT ZERO
0026 0E07           164 OT4:    MVI     C,07H   ;MOVE COUNT INTO REG. C
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

0028 3E90          165          MVI     A,90H    ;LOAD DISPLAY CONTROL DATA FOR
                   166                           ;8279 INTO ACCUM.
002A D371          167          OUT     CNTRL    ;OUTPUT TO 8279
002C 218243        168          LXI     H,RMVAL1;LOAD TEMP RAM VALUE(RMVAL1=
                   169                           ;RMVAL-1 SINCE HUNDRED HRS.
                   170                           ;CONTAINS ONLY ONE BCD NIBBLE
002F CD4300    C   171          CALL    OUTPUT   ;CALL 8279 DISPLAY OUTPUT
                   172                           ;ROUTINE
0032 21FB43        173          LXI     H,ERREG  ;LOAD POINTER WITH ERROR DATA
                   174                           ;ADDRESS
0035 0E04          175          MVI     C,04     ;STORE COUNT INTO REG. C
0037 CD4300    C   176          CALL    OUTPUT   ;CALL OUTPUT ROUTINE
003A CD8000    C   177          CALL    LED      ;CALL DISCRETE LED OUTPUT
                   178                           ;ROUTINE
003D 21E243        179          LXI     H,DSFLG  ;LOAD POINTER WITH DISPLAY
                   180                           ;FLAG ADDRESS
0040 3600          181          MVI     M,FAL    ;LOAD FLAG WITH FALSE DATA
0042 C9            182          RET              ;RETURN
                   183 ;
                   184 ;*************************************************
                   185 ;
                   186 ;  OUTPUT
                   187 ;     THIS ROUTINE TAKES UNPACKED BCD WITH OR WITHOUT
                   188 ; DECIMAL POINT OR SIGN, DECODES THEIR VALUE AND OUT-
                   189 ; PUTS THE DATA TO THE 8279 DISPLAY RAM WHICH DRIVES
                   190 ; THE SEVEN-SEGMENT DISPLAYS (NAMELY THE TIME AND
                   191 ; ERROR DISPLAYS)
                   192 ;
                   193 ; THE STARTING ADDRESS OF THE DISPLAY DATA IS LOCATED
                   194 ; IN H&L AND THE NUMBER OF DISPLAYS TO BE OUTPUTTED
                   195 ; IS ENTERED IN REG. C
                   196 ;
                   197 ;*************************************************
                   198 ;
0043 7E            199 OUTPUT: MOV     A,M      ;GET BYTE TO DISPLAY
0044 47            200         MOV     B,A      ;SAVE IN REG. B
0045 EB            201         XCHG             ;SAVE H&L REG.
0046 216200    C   202         LXI     H,DSPTB  ;LOAD POINTER WITH CONVERSION
                   203                          ;TABLE ADDRESS
0049 E60F          204         ANI     0FH      ;MASK OFF DECIMAL POINT OR
                   205                          ;SIGN BIT AND UNNECESSARY DATA
004B 85            206         ADD     L        ;ADD TABLE ADDRESS TO VALUE
004C 6F            207         MOV     L,A      ;RESTORE IN REG. L
004D 3E00          208         MVI     A,00     ;CLEAR ACCUM.
004F 8C            209         ADC     H        ;ADD UPPER PORTION OF TABLE
                   210                          ;ADDRESS
                   211
0050 67            212         MOV     H,A      ;RESORE IN REG. H - NOW HAVE
                   213                          ;COMPLETE ADDRESS OF THE CONVERT
                   214                          ;ED VALUE
0051 78            215         MOV     A,B      ;GET ORIGINAL VALUE
0052 E680          216         ANI     080H     ;MASK OUT ALL BUT D.P. OR SIGN
                   217                          ;BIT
0054 07            218         RLC              ;ROTATE 4 BITS TO CORRESPOND TO
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

                   219                          ;THE 8279 D.P. (OR SIGN) LOCATIO
                       N
0055 07            220          RLC
0056 07            221          RLC
0057 07            222          RLC
0058 B6            223          ORA      M       ;OR WITH THE TABLE VALUE
0059 D370          224          OUT      DSPLY   ;OUTPUT DATA TO 8279
005B EB            225          XCHG             ;RESTORE BUFFER ADDRESS
005C 2B            226          DCX      H       ;INCREMENT BUFFER ADDRESS
005D 0D            227          DCR      C       ;DECREMENT COUNT
005E C24300    C   228          JNZ      OUTPUT  ;LOOP TILL THROUGH
0061 C9            229          RET              ;RETURN
                   23  ;*****************************************************
                   231 ;
                   232 ;  DSPTB
                   233 ;     THE FOLLOWING CODE CONTAINS THE CONVERSION VALUES
                   234 ; FOR THE 8279 DISPLAY DRIVER.
                   235 ;
                   236 ;*****************************************************
                   237 ;
                   238 DSPTB:
0000               239 ZERO     EQU      $-DSPTB
0062 F3            240          DB       0F3H
0001               241 ONE      EQU      $-DSPTB
0063 60            242          DB       060H
0002               243 TWO      EQU      $-DSPTB
0064 B5            244          DB       0B5H
0003               245 THREE    EQU      $-DSPTB
0065 F4            246          DB       0F4H
0004               247 FOUR     EQU      $-DSPTB
0066 66            248          DB       066H
0005               249 FIVE     EQU      $-DSPTB
0067 D6            250          DB       0D6H
0006               251 SIX      EQU      $-DSPTB
0068 D7            25           DB       0D7H
0007               253 SEVEN    EQU      $-DSPTB
0069 70            254          DB       070H
0008               255 EIGHT    EQU      $-DSPTB
006A F7            256          DB       0F7H
0009               257 NINE     EQU      $-DSPTB
006B F6            258          DB       0F6H
000A               259 OVAL     EQU      $-DSPTB
006C C5            260          DB       0C5H
000B               261 RVAL     EQU      $-DSPTB
006D 05            262          DB       05H
000C               26  BVAL     EQU      $-DSPTB
006E 83            264          DB       083H
000D               26  UVAL     EQU      $-DSPTB
006F E3            266          DB       0E3H
000E               267 EVAL     EQU      $-DSPTB
0070 97            268          DB       097H
000F               269 BLANK    EQU      $-DSPTB
0071 00            270          DB       00H
                   271 ;*****************************************
                   272 ;
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

                   273 ;   HXDSP
                   274 ;     THIS ROUTINE TAKES THE PACKED DATA STORED
                   275 ; IN THE ACCUMULATOR, UNPACKS IT AND STORES
                   276 ; IT IN MEMORY .
                   277 ;
                   278 ;*********************************************
                   279 ;
0072 4F            280 HXDSP:  MOV     C,A     ;SAVE THE ACCUM.
0073 0F            281         RRC             ;ROTATE TO GET THE UPPER
0074 0F            282         RRC             ; NIBBLE
0075 0F            28          RRC
0076 0F            284         RRC
0077 E60F          285         ANI     0FH     ;MASK OFF UPPER NIBBLE
0079 77            286         MOV     M,A     ;STORE IN MEMORY
007A 2B            287         DCX     H       ;INCREMENT POINTER
007B 79            288         MOV     A,C     ;RESTORE VALUE IN ACCUM.
007C E60F          289         ANI     0FH     ;MASK OFF LOWER NIBBLE
007E 77            290         MOV     M,A     ;STORE IN MEMORY
007F C9            291         RET             ;RETURN
                   292 ;*************************************************
                   29  ;
                   294 ;   LED
                   295 ;     THIS ROUTINE READS THE COMMON RAM LOCATIONS
                   296 ; CONTAINING THE DISCRETE LED VALUES, REARRANGES
                   297 ; THE DATA IN ORDER TO BE COMPATIBLE WITH THE 8279
                   298 ; DISPLAY RAM AND THE OUTPUTS THE DATA TO THE 8279,
                   299 ; WHICH DRIVES THE LEDS.
                   300 ;     THIS ROUTINE ALSO DRIVES THE ERROR OUTPUT GATE
                   301 ; BY DETERMINING WHICH TYPE OF ERROR IS TO BE OUT-
                   302 ; PUTTED AND THE INPUT TYPE RELAY (DSX-3).
                   303 ;
                   304 ;*************************************************
                   305 ;
0080 21F243        306 LED:    LXI     H,TRFLT ;LOAD POINTER WITH TRAFFIC
                   307                         ;LIGHT SIGNALS ADDRESS
0083 7E            308         MOV     A,M     ;MOVE INTO ACCUM.
0084 07            309         RLC             ;ROTATE 3 BITS LEFT
0085 07            31          RLC
0086 07            311         RLC
0087 E608          312         ANI     08H     ;MASK OUT ALL BUT AUTO
                   313                         ;CONTROL BIT
0089 47            314         MOV     B,A     ;STORE IN REG. B
008A 7E            315         MOV     A,M     ;GET VALUE AGAIN
008B E60E          316         ANI     0EH     ;MASK OUT ALL BUT LOSS
                   317                         ;SIGNALS
008D 4F            318         MOV     C,A     ;STORE IN REG. C
008E 7E            319         MOV     A,M     ;GET VALUE AGAIN
008F 0F            320         RRC             ;ROTATE 2 BITS RIGHT
0090 0F            321         RRC
0091 E604          322         ANI     04H     ;MASK OUT ALL BUT GATING
                   323                         ;BIT
0093 57            324         MOV     D,A     ;STORE IN REG. D
0094 23            325         INX     H       ;INCREMENT POINTER TO RUN/
                   326                         ;STOP REG.
0095 7E            327         MOV     A,M     ;STORE VALUE IN ACCUM.
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

0096 07           328         RLC             ;ROTATE 4 BITS LEFT
0097 07           329         RLC
0098 07           33          RLC
0099 07           33          RLC
009A E630         332         ANI     30H     ;MASK OUT ALL INVALID DATA
009C B1           333         ORA     C       ;OR WITH REG. C
009D 4F           334         MOV     C,A     ;RESTORE IN REG. C
009E 23           335         INX     H       ;INCREMENT POINTER TO ERROR
                  336                         ;TYPE DATA REG.
009F 7E           337         MOV     A,M     ;STORE IN ACCUM.
00A0 0F           338         RRC             ;ROTATE 2 BITS RIGHT
00A1 0F           339         RRC
00A2 E6C1         340         ANI     0C1H    ;GET VALID DATA
00A4 B1           341         ORA     C       ;OR WITH REG. C
00A5 4F           342         MOV     C,A     ;RESTORE IN REG. C
00A6 0F           343         RRC             ;ROTATE ANOTHER BIT RIGHT
                  344                         ;TO  OUTPUT TO ERROR GATE
                  345                         ;ON FRONT PANEL
00A7 E6E0         346         ANI     0E0H    ;MASK OUT ALL BUT OUTPUT
                  347                         ;TYPE DATA
00A9 5F           348         MOV     E,A     ;STORE TEMPORARILY IN
                  349                         ;REG E
00AA 23           350         INX     H       ;INCREMENT POINTER TO INPUT
                  35                          ;TYPE
00AB 7E           352         MOV     A,M     ;STORE IN ACCUM.
00AC E607         353         ANI     07H     ;GET VALID DATA
00AE B0           354         ORA     B       ;OR WITH REG. B
00AF 47           355         MOV     B,A     ;RESTORE IN REG. B
00B0 07           356         RLC             ;ROTATE DS-3 INPUT BIT 3
                  357                         ;PLACES IN ORDER TO OUTPUT
                  358                         ;TO FRONT PANEL ALONG WITH
                  359                         ;ERROR TYPE
00B1 07           36          RLC
00B2 07           361         RLC
00B3 E610         362         ANI     10H     ;MASK OUT ALL OTHER BITS
00B5 B3           363         ORA     E       ;APPEND TO ERROR TYPE BITS
00B6 C5           364         PUSH    B
00B7 47           365         MOV     B,A
00B8 3AE043       36          LDA     RUNSTP  ;IF RUNNING THEN DONT SET THE
                  367                         ; ERROR OUTPUT CIRCUIT DO TO
                  368                         ; BURST CONTROL
00BB E601         369         ANI     01H
00BD C2C800   C   370         JNZ     NOCH
00C0 3AA843       371         LDA     UNFRAM
00C3 E608         372         ANI     08H
00C5 B0           373         ORA     B
00C6 D394         374         OUT     INOUT   ;OUTPUT TO FRONT PANEL
00C8 C1           375 NOCH:   POP     B
00C9 23           376         INX     H       ;INCREMENT POINTER TO TIME
                  377                         ;MODE
00CA 7E           378         MOV     A,M     ;STORE VALUE IN ACCUM.
00CB E60F         379         ANI     0FH     ;GET VALID DATA
00CD 07           380         RLC             ;ROTATE 4 BITS LEFT
00CE 07           381         RLC
00CF 07           382         RLC
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

00D0 07            383          RLC
00D1 B0            384          ORA     B       ;OR WITH REG. B
                   385 ;REG. B & C NOW COMPLETE AND READY FOR OUTPUT
00D2 F5            386          PUSH    PSW     ;SAVE ON STACK--LAST ONE
00D3 23            387          INX     H       ;INCREMENT POINTER TO
                   388                          ;ERROR MODE DATA ADDRESS
00D4 7E            389          MOV     A,M     ;STORE VALUE IN ACCUM.
00D5 F5            390          PUSH    PSW     ;SAVE ON STACK TWICE
00D6 F5            391          PUSH    PSW
00D7 E601          392          ANI     01H     ;GET ERROR SEC.
00D9 B2            393          ORA     D
00DA 57            394          MOV     D,A
00DB F1             39          POP     PSW     ;RETRIEVE ERROR MODE DATA
00DC E620           39          ANI     20H     ;GET %ERROR SEC.
00DE 0F            397          RRC
00DF 0F            398          RRC
00E0 0F            399          RRC             ;MOVE INTO BIT 1 POS.
00E1 0F            400          RRC
00E2 B2            401          ORA     D       ;OR WITH OTHER
00E3 57            402          MOV     D,A     ;RESTORE
00E4 3AF143        403          LDA     THRESH  ;GET THRESHOLD INFORM.
00E7 E60F          404          ANI     0FH     ;GET THE WHOLE REG.
00E9 07             40          RLC
00EA 07            406          RLC
00EB 07            407          RLC
00EC 07            408          RLC             ;MOV TO TOP 4 BITS
00ED B2            409          ORA     D       ;COMBINE WITH OTHERS
00EE 57            410          MOV     D,A
00EF 3AEE43        411          LDA     HORN    ;FETCH HORN DATA
00F2 E601          412          ANI     01H     ;MASK OUT ALL ELSE
00F4 07            413          RLC
00F5 07            414          RLC
00F6 07            415          RLC             ;MOVE TO BIT POS. 3
00F7 B2            416          ORA     D       ;COMBINE WITH OTHERS
00F8 57            417          MOV     D,A     ;STORE IN D
00F9 F1            418          POP     PSW     ;RETRIEVE ERROR MODE DATA
00FA E61E          419          ANI     1EH     ;GET REST OF ERROR INFO.
00FC 07            420          RLC
00FD 07            421          RLC
00FE 07            422          RLC             ;SHIFT OVER TO RIGHT PLACE
00FF 47            423          MOV     B,A     ;STORE IN B
0100 3AF043        424          LDA     PRINTC  ;GET PRINT CONTROL DATA
0103 E603          425          ANI     03H
0105 B0            426          ORA     B       ;COMBINE AND STORE
0106 47             42          MOV     B,A
0107 3AEF43        428          LDA     HORNEN  ;FETCH HORN ENABLE DATA
010A E601          429          ANI     01H
010C 07            430          RLC
010D 07            431          RLC
010E B0            432          ORA     B
010F D370          433          OUT     DSPLY   ;OUTPUT
0111 79            434          MOV     A,C     ;GET OTHERS
0112 D370          435          OUT     DSPLY
0114 7A            436          MOV     A,D
0115 D370          437          OUT     DSPLY
```

```
  LOC  OBJ          LINE        SOURCE STATEMENT

  0117 F1            438          POP     PSW
  0118 D370          439          OUT     DSPLY
  011A C9            440          RET
                     441          END


PUBLIC SYMBOLS
DISPLA C 0000

EXTERNAL SYMBOLS


USER SYMBOLS
BLANK  A 000F    BVAL   A 000C    CNTRL  A 0071    DISPLA C 0000
DSFLG  A 43E2    DSPLY  A 0070    DSPTB  C 0062    EIGHT  A 0008
ERREG  A 43FB    EVAL   A 000E    FAL    A 0000    FIR    C 0019
FIVE   A 0005    FOUR   A 0004    HORN   A 43EE    HORNEN A 43EF
HXDSP  C 0072    INOUT  A 0094    LED    C 0080    LOOP   C 0008
NINE   A 0009    NOCH   C 00C8    ONE    A 0001    OT4    C 0026
OUTPUT C 0043    OVAL   A 000A    PRINTC A 43F0    RMVAL  A 4383
RMVAL1 A 4382    RUNSTP A 43E0    RVAL   A 000B    SEVEN  A 0007
SIX    A 0006    THREE  A 0003    THRESH A 43F1    TODRH  A 43FF
TRFLT  A 43F2    TWO    A 0002    UNFRAM A 43A8    UVAL   A 000D
ZERO   A 0000

ASSEMBLY COMPLETE,  NO ERRORS
```

C. REMOTE COMMUNICATIONS DRIVERS

```
ASM80.OV4 :F1:RINIT1.ASM PRINT(:LP:) PAGEWIDTH(80) DEBUG


ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0         RINIT    PAGE    1

 LOC  OBJ          LINE        SOURCE STATEMENT

                      1          NAME    RINIT
                      2          PUBLIC  RINIT
 43AB                 3 RSTHER   EQU     43ABH;
 436E                 4 TRANSM   EQU     436EH
                      5 ;*****************************************************
                      6 ;
                      ;          RINIT
                      8 ;  THIS ROUTINE IS USED TO INITIALIZE THE 8251 AND
                      9 ; 8253 ON THE SERIAL I/O BOARD. DURING THE INITIALIZATIO
                        N
                     10 ; OF THE 8253 ,TIMER 2 IS LOADED AND THEN READ TO SEE
                     11 ; IF THE BOARD IS ACTUALLY PRESENT.
                     12 ;
                     13 ;*****************************************************
                     14 ;
                     15 ;
                      1  ;  TABLE OF DIVIDE-BY-N FOR SETTING THE TIMER
                     17 ;
                     18          CSEG
                     19 ;
                     20 ;
 0000 4710           21 DIVN:    DW      1047H   ;110 BPS,  BDSWIT=0
 0002 8403            2          DW      384H    ;300 BPS,  BDSWIT=1
 0004 9201           23          DW      192H    ;600 BPS,  BDSWIT=2
 0006 6000           24          DW      96H     ;1200 BPS, BDSWIT=3
 0008 3000           25          DW      48H     ;2400 BPS, BDSWIT=4
 000A 1800           26          DW      24H     ;4800 BPS, BDSWIT=5
 000C 0C00           27          DW      12H     ;9600 BPS, BDSWIT=6
 000E 0600           28          DW      6H      ;19200 BPR,BDSWIT=7
                     29 ;
                     30 ; PORTS
                     31 ;
 0093                32 BDSWIT   EQU     93H     ;BAUD SELECTOR
 0087                33 TMODE    EQU     87H     ;MODE REGISTER ADDRESS FOR 8253
 0084                34 TCONT0   EQU     84H     ;TIMER TO CONTROL THE BAUD RATE
 0086                 3 TCONT2   EQU     86H     ;TIMER USED TO SEE IF BOARD IS
 0088                36 TCLOCK   EQU     88H     ;INPUT TO TIMER 2 CLOCK
                     37                          ;THERE
                     38 ;
                     39 ;   BAUD SWITCH MASK
 00E0                40 BDMASK   EQU     11100000B
                     41 ;
                     42 ;  TIMER CONTROL MASKS
                     43 ;
 0000                44 SC0      EQU     00000000B ;SELECT COUNTER ZERO
 0080                45 SC2      EQU     10000000B ;SELECT COUNTER TWO
 0030                46 RL       EQU     00110000B ;READ/LOAD MODE
 0006                47 MODE     EQU     00000110B ;SQUARE WAVE MODE
 0001                48 BCDON    EQU     00000001B ;ENABLE BCD
 0000                49 BCDOFF   EQU     00000000B ;DISABLE BCD
                      5 ;
                     51 ;
                     52 ;   FIRST SEE IF TIMER IS THERE
                     53 ;
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

0010 3EB6           54 RINIT:  MVI     A,(SC2+RL+MODE+BCDOFF)
0012 D387           55         OUT     TMODE   ;OUTPUT TIMER MODE
0014 3E55           56         MVI     A,55H   ;LOAD TEST DATA
0016 D386           57         OUT     TCONT2  ;OUTPUT TO TEST TIMER
0018 3EAA           58         MVI     A,0AAH
001A D386           59         OUT     TCONT2
001C DB88           60         IN      TCLOCK  ;BUMP THE TIMER 2 CLOCK
001E DB88           61         IN      TCLOCK
                    62 ;  NOW READ BACK DATA TO SEE IF THERE
0020 3EB6           63         MVI     A,(SC2+RL+MODE+BCDOFF)
0022 D387           64         OUT     TMODE   ;SELECT TIMER TWO
0024 DB86           65         IN      TCONT2  ;READ LSB
0026 DB86           66         IN      TCONT2  ;COMPARE ONLY MSB SINCE
                    67                         ;LOWER ONE COULD HAVE CHANGED
0028 FEAA           68         CPI     0AAH
002A CA3300    C    69         JZ      HER1    ;JUMP IF COMPARE
002D 3E00           70         MVI     A,00    ;OTHERWISE SET FLAG
002F 32AB43         71         STA     RSTHER
0032 C9             72         RET
                    73 ;
                    74 ;  BOARD MUST BE PRESENT
                    75 ;
0033 3EFF           76 HER1:   MVI     A,0FFH  ;STORE FLAG
0035 32AB43         77         STA     RSTHER
                    78 ;
                    79 TSET:
0038 210000    C     8         LXI     H,DIVN  ;READ BAUD SETTING
003B DB93           81         IN      BDSWIT
003D E6E0           82         ANI     BDMASK
003F 07             83         RLC
0040 07              8         RLC
0041 07             85         RLC             ;GET INTO 3 LSBS
0042 5F             86         MOV     E,A
                    87 TSET1:
0043 CA4C00    C    88         JZ      TSET2
0046 23             89         INX     H
0047 23             90         INX     H       ;POSITION POINTER INTO TABLE
0048 1D              9         DCR     E
0049 C34300    C    92         JMP     TSET1
                    93 TSET2:
004C 3E37           94         MVI     A,(SC0+RL+MODE+BCDON)
004E D387           95         OUT     TMODE   ;SET UP TIMER
0050 7E             96         MOV     A,M     ;GET BAUD SETTING
0051 D384           97         OUT     TCONT0  ;LSB
0053 23             98         INX     H
0054 7E             99         MOV     A,M
0055 D384          100         OUT     TCONT0
                   101 ;
                    10 ;
                   103 ;  INITIALIZE USART
                   104 ;
                   105 ;  PORTS
                   106 ;
0080               107 SDATIO  EQU     80H     ;SERAIAL DATA
0081               108 SCNTRL  EQU     81H     ;CONTROL PORT
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

0081               109 SSTAT   EQU     81H      ;USART STATUS IN
                   110 ;
                   111 ;  MODE
                   112 ;
0040               113 STPBIT  EQU     01000000B ;1 STOP BIT
00C0               114 TWSBIT  EQU     11000000B ;2 STOP BITS
0020               115 EP      EQU     00100000B ;EVEN PARITY
0010               116 PEN     EQU     00010000B ;PARITY ENABLE
0008               117 CHL7    EQU     00001000B ;7 BIT CHAR.
0002               118 BRF     EQU     00000010B ;BAUD RATE FACTOR 16X
                   119 ;
                   120 ;  COMMAND
                   121 ;
0040               122 IR      EQU     01000000B ;INTERNAL RESET
0020               123 RTS     EQU     00100000B ;REQUEST TO SEND
0010               124 ER      EQU     00010000B ;ERROR RESET
0004               125 RXE     EQU     00000100B ;RECEIVE ENABLE
0002               126 DTR     EQU     00000010B ;DATA TERMINAL READY
0001               127 TXEN    EQU     00000001B ;TRANSMIT ENABLE
                   128 ;
                   129 ;
                   130 USET:
0057 AF            131         XRA     A        ;IN CASE, POWER UP IN
0058 D381          132         OUT     SCNTRL   ; SYNC MODE, THE USART
005A D381          133         OUT     SCNTRL   ;WOULD BE WAITING FOR
005C D381          134         OUT     SCNTRL   ; 3 ADDITIONAL CHAR.
                   135                          ; SO DUMMY THEM
005E 3E40          136         MVI     A,IR     ;INTERNAL RESET
0060 D381          137         OUT     SCNTRL
                   138 ;
0062 067A          139         MVI     B,(PEN+CHL7+EP+BRF+STPBIT)
0064 DB93          140         IN      BDSWIT   ;SET AUTO 110/2 STOP BIT
0066 E6E0          141         ANI     0E0H     ;MASK OUT PROPER BITS
0068 C26D00    C   142         JNZ     ONEB
006B 06FA          143         MVI     B,(PEN+CHL7+EP+TWSBIT+BRF)
006D 78            144 ONEB:   MOV     A,B
006E D381          145         OUT     SCNTRL   ;SET MODE
                   146 ;
0070 3E36          147         MVI     A,(RTS+ER+RXE+DTR)
0072 D381          148         OUT     SCNTRL   ;SET COMMAND
                   149 ;
0074 DB80          150         IN      SDATIO   ;DUMMY READ
                   151 ;
0076 3EFF          15          MVI     A,0FFH   ; OK TO TRANSMIT
0078 326E43        15          STA     TRANSM
                   154 ;
007B C9            155         RET
                   156         END


PUBLIC SYMBOLS
PINIT  C 0010

EXTERNAL SYMBOLS
```

USER SYMBOLS

BCDOFF	A	0000	BCDON	A	0001	BDMASK	A	00E0	BDSWIT	A	0093
BRF	A	0002	CHL7	A	0008	DIVN	C	0000	DTR	A	0002
EP	A	0020	ER	A	0010	HER1	C	0033	IR	A	0040
MODE	A	0006	ONEB	C	006D	PEN	A	0010	RINIT	C	0010
RL	A	0030	RSTHER	A	43AB	RTS	A	0020	RXE	A	0004
SC0	A	0000	SC2	A	0080	SCNTRL	A	0081	SDATIO	A	0080
SSTAT	A	0081	STPBIT	A	0040	TCLOCK	A	0088	TCONT0	A	0084
TCONT2	A	0086	TMODE	A	0087	TRANSM	A	436E	TSET	C	0038
TSET1	C	0043	TSET2	C	004C	TWSBIT	A	00C0	TXEN	A	0001
USET	C	0057									

ASSEMBLY COMPLETE, NO ERRORS

```
ASM80.OV4 :F1:RINTR1.ASM PRINT(:LP:) PAGEWIDTH(80) DEBUG


ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        RINTR    PAGE    1

 LOC  OBJ          LINE        SOURCE STATEMENT

                      1          NAME    RINTR
                      2          PUBLIC  RINTR
                      3          EXTRN   GETBYT
 43DD                 4 BYTEIN   EQU     43DDH
 002D                 5 RSIN     EQU     002DH
 000C                 6 INADD    EQU     000CH
 0011                 7 XON      EQU     11H
 0013                 8 XOFF     EQU     13H
 000D                 9 CR       EQU     0DH
 4370                10 NCNTR    EQU     4370H
 4371                11 DELAY    EQU     4371H
 0081                12 SCNTRL   EQU     81H
 436E                13 TRANSM   EQU     436EH
 0080                1  SDATIO   EQU     80H
 43ED                15 SWREG    EQU     43EDH
 0003                16 ETX      EQU     03H
 43C0                17 ABLOCK   EQU     43C0H
                     18 ;
                     19 ;.....
                     20 ;
                     21 ;       RINTR--
                     22 ;  THIS IS THE INTERRUPT SERVICE ROUTINE FOR THE
                     23 ; RS232 OPTION. THIS ROUTINE CHECKS FOR BOTH THE
                     2  ; THE TXRDY AND RXRDY INTERRUPTS. THE TRANSMIT
                     2  ; ROUTINE USES THE SAME QUEUE FORMAT AS THE GPIB.
                     26 ; THAT IS WHY THIS ROUITNE CALLS THE BYTOUT,WHICH
                     27 ; IS ALSO IS USED BY THE GPIB ROUTINE.
                     28 ;
                     29 ;*************************************************
                     30 ;
                     31          CSEG
                     32 ;
                     33 RINTR:
 0000 E602           34          ANI     02H       ;CHECK FOR RXRDY
 0002 CA3200    C    35          JZ      TXRD      ;JUMP TO SEE IF TXRDY HAS
                     36                            ;OCCURED
 0005 DB80           37          IN      SDATIO    ;OTHERWISE INPUT THE DATA
                     38 ;  NOW CHECK XON XOFF STATUS
 0007 FE13           3           CPI     XOFF      ;IS IT XOFF
 0009 C21300    C    4           JNZ     SKP00     ;SKIP DOWN IF NOT
 000C AF             41          XRA     A
 000D 326E43         42          STA     TRANSM    ;SET FLAG NOT TO TRANSMIT
 0010 C33200    C    43          JMP     TXRD
 0013 FE11           44 SKP00:   CPI     XON       ;IS IT XON
 0015 C22400    C    45          JNZ     SBYTE     ;JUMP DOWN IF NOT
 0018 3E37           46          MVI     A,37H     ; ENABLE TX
 001A D381           47          OUT     SCNTRL
 001C 3EFF           48          MVI     A,0FFH
 001E 326E43         49          STA     TRANSM    ;OK TO TRANSMIT
 0021 C33200    C    5           JMP     TXRD
 0024 32DD43         5  SBYTE:   STA     BYTEIN    ;STORE AWAY
 0027 A7             52          ANA     A         ;IF IT IS ZERO THEN IGNORE
 0028 CA3200    C    53          JZ      TXRD      ;JUMP ON DOWN IF ZERO
 002B 3E37           54 OVTHIS:  MVI     A,37H     ;ENABLE TRANSMIT
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

002D D381           55          OUT     SCNTRL
002F CD2D00         56          CALL    RSIN    ;CALL ASCII INPUT ROUTINE
                    57 ; TRANSMIT READY
                    58 ;
                    59 TXRD:
0032 DB81           60          IN      SCNTRL  ;CHECK TXRDY
0034 E601           61          ANI     01H     ;CHECK TRANSMIT READY BIT
0036 C8             62          RZ              ;RETURN IF NOT READY
0037 3A6E43         63          LDA     TRANSM  ;SEE IF OK OR ANYTHING TO GO
003A A7             6           ANA     A
003B C24300    C    65          JNZ     OKTX
003E 3E36           66          MVI     A,36H
0040 D381           67          OUT     SCNTRL  ;DISABLE INTERRUPT
0042 C9             68          RET             ;RETURN
0043 3A7043         69 OKTX:    LDA     NCNTR   ;LOAD NULL COUNTER
0046 FE01           70          CPI     1       ;COMPARE TO 1 TO SEE IF
                    7                           ;WE CAN SEND OUT ECHO
0048 CA6500    C    72          JZ      TXRD1   ;IF SO THEN SEND OUT NEXT
                    7                           ;BYTE -NOT THE "BYTEIN"
004B A7             74          ANA     A       ;IF ZERO THEN SKIP OVER
004C CA5800    C    75          JZ      NOCR
004F 3D             7           DCR     A       ;OTHERWISE DECREMENT THE COUNT
                    7                           ;AND SEND A NULL
0050 327043         78          STA     NCNTR
0053 3E00           79          MVI     A,00H
0055 D380           8           OUT     SDATIO  ;OUTPUT NULL
0057 C9             81          RET             ;RETURN
0058 21DD43         82 NOCR:    LXI     H,BYTEIN ;SEE IF NEED TO ECHO A CHAR.
005F 7E             83          MOV     A,M
005C A7             84          ANA     A       ;IF ZERO JUMP TO TXRD1
005D CA6500    C    8           JZ      TXRD1
0060 D380           8           OUT     SDATIO  ;OTHERWISE ECHO
0062 AF             8           XRA     A       ;ZERO OUT "BYTEIN"
0063 77             88          MOV     M,A
0064 C9             89          RET             ;RETURN
                    90 TXRD1:
0065 21C043         91          LXI     H,ABLOCK ;LOAD UP PARAMETER BLOCK FOR
                    9                           ;GETBYT
0068 CD0000    E    9           CALL    GETBYT  ;CALL ROUTINE TO DECIDE
                    9                           ;WHICH CHAR TO GO OUT
                    95 ;
                    96 ;  WILL RETURN WITH OUTPUT CHAR IN A, AND IF THIS IS
                    9  ; THE LAST CHAR. TO GO, THEN REG. C WILL BE NON-ZERO.
                    98 ; OTHERWISE REG. C WILL BE ZERO.
                    99 ;
006B A7            100          ANA     A
006C CA7400    C   101          JZ      NOTX
006F D380          102          OUT     SDATIO
0071 C37900    C   10           JMP     OUTPT   ;JUMP OVER
                   104 NOTX:
0074 3E36          105          MVI     A,36H   ;DISABLE TRANSM.
0076 D381          106          OUT     SCNTRL
0078 C9            107          RET
                   108 OUTPT:
0079 FE0D          109          CPI     CR
```

```
  LOC  OBJ          LINE        SOURCE STATEMENT

  007B C28D00    C   110          JNZ     NODEL   ;IF NOT JUMP ON DOWN
  007E 3E01          111          MVI     A,1     ;SET DEFAULT VALUE FOR CR
  0080 3A7143        112          LDA     DELAY   ;SEE IF WE SHOULD DELAY LONGER
  0083 A7            113          ANA     A
  0084 CA8900    C   114          JZ      OV2     ;JUMP OVER IF NOT
  0087 3E07          115          MVI     A,7     ;SET NUMBER OF NULLS TO 7-1=6
                     116 OV2:
  0089 327043        117          STA     NCNTR
  008C C9            118          RET
  008D AF            119 NODEL:   XRA     A       ;SET NCNTR=0
  008E 327043        120          STA     NCNTR
  0091 C9            121          RET             ;RETURN
                     122 ;
                     123 ;
                     124          END


PUBLIC SYMBOLS
RINTR  C 0000

EXTERNAL SYMBOLS
GETBYT E 0000

USER SYMBOLS
ABLOCK A 43C0    BYTEIN A 43DD    CR     A 000D    DELAY  A 4371
ETX    A 0003    GETBYT E 0000    INADD  A 002C    NCNTR  A 4370
NOCR   C 0058    NODEL  C 008D    NOTX   C 0074    OKTX   C 0043
OUTPT  C 0079    OV2    C 2289    OVTHIS C 002B    RINTR  C 0000
RSIN   A 202D    SBYTE  C 2024    SCNTRL A 0081    SDATIO A 0080
SKP90  C 0013    SWREG  A 43ED    TRANSM A 436E    TXRD   C 0032
TXRD1  C 0065    XOFF   A 0013    XON    A 0011

ASSEMBLY COMPLETE,  NO ERRORS
```

```
ASM80.OV4 :F1:GINIT1.ASM PRINT}:LP:~ PAGEWIDTH}80~ DEBUG


ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0         GINIT    PAGE    1

  LOC  OBJ          LINE        SOURCE STATEMENT

                       1          NAME    GINIT
                       2          PUBLIC  GINIT
  43DE                 3 GBTHER   EQU     43DEH   ;GPIB PRESENT FLAG
  8000                 4 BASE     EQU     8200H   ;BASE ADDRESS FOR 68488
  43CC                 5 REMFL    EQU     43CCH   ;REMOTE ENABLE FLAG
                       6 ;*************************************************
                         ;
                         ;     GINIT
                       9 ;  THIS ROUTINE IS THE INITIALIZATION PROCESS FOR THE
                      10 ; MOTOROLA 68488 GPIA CHIP.
                      11 ;
                      12 ;*************************************************
                      13 ;
                      14          CSEG
                      15 ;
  0200 3E80           16 GINIT:   MVI     A,80H   ;RESET CHIP FIRST AND THEN DETER
                                  -
                      17                          ;MINE IF IT IS THERE
  0202 210380         18          LXI     H,BASE+3
  0205 77             19          MOV     M,A
  0206 AF             20          XRA     A
  0207 77             21          MOV     M,A
  0208 2B             22          DCX     H       ;DECREMENT DOWN TO ADDRESS
                      23                          ;STATUS REG.
  0209 77             24          MOV     M,A     ;ZERO OUT THE REG.
  020A 32DE43         25          STA     GBTHER  ;INITIALIZE THERE FLAG
  020D 3EAA           26          MVI     A,0AAH  ;WRITE INTO SERIAL POLL AND
  020F 23             27          INX     H       ;INCREMENT TO SER. POLL REG.
  0210 23             28          INX     H
  0211 23             29          INX     H
  0212 77             30          MOV     M,A
  0213 7E             31          MOV     A,M
  0214 FEAA           32          CPI     0AAH    ;SEE IF YOU READ IT BACK
  0216 C0             33          RNZ             ;RETURN IF NOT ON LINE
  0217 3EFF           34          MVI     A,0FFH  ;SET FLAG IF ON LINE
  0219 32DE43         35          STA     GBTHER
  021C 2B             36          DCX     H       ;GET TO BASE+4 TO SET ADDRESS
  021D 7E             37          MOV     A,M
  021E 4F             38          MOV     C,A     ;STORE TEMP. IN C
  021F E61F           39          ANI     1FH
  0221 77             40          MOV     M,A
  0222 79             41          MOV     A,C     ;RESTORE REG.
  0223 E6E0           42          ANI     0E0H    ;MASK OUT ALL BUT T.O. MODE
  0225 FEE0           43          CPI     0E0H    ;IF TO THEN SET BIT IN
                      44                          ;ADD. MODE REG. AND SET
                      45                          ;REMOTE EN. FLAG TO REMOTE
  0227 C23402    C    46          JNZ     SKPU
  022A 3E40           47          MVI     A,40H   ;SET T.O.
  022C 320282         48          STA     BASE+2
  022F 3EFF           49          MVI     A,0FFH  ;SET TO REN
  0231 32CC43         50          STA     REMFL
  0234 AF             51 SKPU:    XRA     A
  0235 2B             52          DCX     H       ;GET TO BASE+3
  0236 77             53          MOV     M,A     ;RESET AUX. COMMAND REG.
```

```
  LOC  OBJ          LIN          SOURCE STATEMENT

  0037 23             54          INX     H
  0038 23             55          INX     H
  0039 77             56          MOV     M,A       ;RESET SERIAL POLL REG.
  003A 23             57          INX     H
  003B 77             58          MOV     M,A       ;RESET PARALLEL POLL REG.
  003C 3E84           59          MVI     A,84H     ;ENABLE INTERRUPT CMD
  003E 320080         60          STA     BASE
  0041 C9             61          RET               ;RETURN
                      62          END


PUBLIC SYMBOLS
GINIT  C 0000

EXTERNAL SYMBOLS


USER SYMBOLS
BASE   A 8000    GBTHER A 43DE    GINIT  C 0000    REMFL  A 43CC
SKPU   C 0034

ASSEMBLY COMPLETE,   NO ERRORS
```

```
ASM80.OV4 :F1:GINTR1.ASM PRINT(:LP:) PAGEWIDTH(80) DEBUG


ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        GINTR    PAGE    1

  LOC  OBJ          LINE        SOURCE STATEMENT

                      1          NAME    GINTR
                      2          PUBLIC  GINTR
                      3          EXTRN   GETBYT,DCSTRT
                      4 ;
                      5 ;
  002D                6 GPIBIN   EQU     002DH
  0000                7 NULL     EQU     00H
  8000                8 BASE     EQU     8000H   ;BASE FOR 68488
  43ED                9 SWREG    EQU     43EDH
  43DD               10 BYTEIN   EQU     43DDH
  000C               11 INADD    EQU     000CH
  43CC               12 REMFL    EQU     43CCH
  4372               13 FIRSFL   EQU     4372H
  0003               14 ETX      EQU     03H
  43C0               15 ABLOCK   EQU     43C0H
                     16 ;*********************************************
                     17 ;
                     1  ;       GPIB INTERRUPT SEVICE ROUTINE
                     1  ;
                     2  ;
                     21 ;  THIS ROUTINE HANDLES ALL INTERRUPTS CAUSED
                     22 ; BY THE 68488 CHIP. THE ROUTINE EXPECTS THE
                     23 ; FOLLOWING INTERRUPTS ONLY:
                     2  ;                CMD- COMMAND INTERRUPT FROM
                     2  ;                     SPAS, DCAS, & RLC
                     26 ;                BI - BYTE INPUT
                     27 ;                BO - BYTE OUTPUT
                     28 ;                GET- GROUP EXECUTE TRIGGER
                     29 ;
                     30 ;**********************************************
                     3  ;
                     32          CSEG
                     33 ;
  0000 3A0080        34 GINTR:   LDA     BASE    ;FETCH INTR. STATUS REG.
  0003 4F            35 G1:      MOV     C,A
  0004 E604          36          ANI     04H     ;IS IT A COMMAND?
  0006 C44F00    C   37          CNZ     CMD
  0009 79            38          MOV     A,C
  000A E620          39          ANI     20H     ;IS IT GROUP EXECUTE TRIGGER
  000C C44200    C   40          CNZ     GET
  000F 79            41          MOV     A,C
  0010 E640          42          ANI     40H     ;IS IT BYTE OUTPUT?
  0012 C46800    C   43          CNZ     OUTB
  0015 79            44          MOV     A,C
  0016 E601          45          ANI     01H     ;IS IT BYTE INPUT?
  0018 C48700    C   46          CNZ     INB
                     47 ; NOW MUST CLEAR AND REPROGRAM INTERRUPT STATUS REG.
                     4  ;
  001B 3E04          49          MVI     A,04H   ;CLEAR INTERRUPT MASK REG.
                     50                          ;EXCEPT FOR COMMAND BECAUSE
                     51                          ;OF THE SPAS STATE IF ENABLED
                     52                          ;(SPE) WILL CAUSE CONTINUOUS
                     53                          ;INTERRUPTS
  001D 320080        54          STA     BASE
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

0020 3A0580         55          LDA     BASE+5
0023 47             56          MOV     B,A
0024 E640           57          ANI     40H
0026 C23B00    C    58          JNZ     OVT
0029 3A0180         59          LDA     BASE+1
002C E604           60          ANI     04H
002E C23B00    C    61          JNZ     OVT         ;CHECK BOTH SRQS AND SPAS BEFORE
                    62                              ;CHANGE ANYTHING
0031 78             63          MOV     A,B         ;RESTORE BYTE
0032 E617           64          ANI     17H
0034 320580         65          STA     BASE+5      ;TAKE OUT MEAS. COMPLETE
0037 AF             66          XRA     A
0038 320680         67          STA     BASE+6      ;TAKE OUT PAR. POLL
003B 06E5           68 OVT:     MVI     B,0E5H      ;MUST BE TRANSMITTING, SO SET
                    69                              ;BO INTERRUPT
003D 78             70 END1:    MOV     A,B
003E 320080         71          STA     BASE        ;RESTORE INTERRUPT MASK
0041 C9             72          RET                 ;RETURN
                    73 ;
                    74 ;
                    75 ;
                    76 ;
                    77 ;   GROUP EXECUTE TRIGGER
                    78 ;
                    79 GET:
0042 3E10           80          MVI     A,10H       ;RELEASE DAC
0044 320380         81          STA     BASE+3
0047 21ED43         82          LXI     H,SWREG     ;STORE CODE FOR RESET KEY
004A 360A           83          MVI     M,0AH
004C C30C00         84          JMP     INADD       ;CALL BRANCH ON INP. ROUTINE
                    85 ;
                    86 ;
                    87 ;
                    88 ;   COMMAND
                    89 ;
004F 3A0180         90 CMD:     LDA     BASE+1      ;FETCH COMMAND REG.
0052 57             91          MOV     D,A
0053 E602           92          ANI     02H         ;IS IT DEVICE CLEAR?
0055 C49200    C    93          CNZ     DEVC
0058 7A             94          MOV     A,D
0059 E608           95          ANI     08H         ;REMOTE/LOCAL CHANGE?
005B C4A500    C    96          CNZ     REML
005E 7A             97          MOV     A,D
005F E681           98          ANI     81H         ;IS IT UUCG OR UACG
0061 C8             99          RZ                  ;RETURN IF NOT
0062 3E10          100          MVI     A,10H       ;IF IT IS RELEASE DAC
0064 320380        101          STA     BASE+3
0067 C9            102          RET                 ;RETURN
                   103 ;
                   10  ;
                   10  ;
                   126 ;    BYTE OUTPUT
                   107 ;
                   108 ;
                   109 OUTB:
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

0068 21C043       110          LXI     H,ABLOCK ;GET PARAMETER BLOCK POINTER
006B CD0000   E   111          CALL    GETBYT  ;CALL ROUTINE TO DETERMINE
006E FE00         112          CPI     0       ;IF ZERO DISABLE INT.
0070 C27900   C   113          JNZ     OUTB1
0073 3EA5         114          MVI     A,0A5H
0075 320080       115          STA     BASE
0078 C9           116          RET             ;RETURN
0079 F5           117 OUTB1:   PUSH    PSW     ;SAVE OUTPUT BYTE
007A 79           118          MOV     A,C     ;SEE IF THIS IS LAST ONE TO GO
007B A7           119          ANA     A
007C CA8200   C   120          JZ      SKIP08  ;IF NOT THEN SKIP DOWN
                  121 ; OTHERWISE FORCE EOI AND DISABLE INTERRUPT
007F 320380       122          STA     BASE+3  ;FORCE EOI
0082 F1           123 SKIP08:  POP     PSW     ;RESTORE OUTPUT BYTE
0083 320780       124          STA     BASE+7  ;TRANSMIT BYTE
0086 C9           125          RET             ;RETURN
                  126 ;
                  12  ;
                  128 ;
                  129 ;      BYTE INPUT
                  130 ;
                  131 ;
0087 3A0780       132 INB:     LDA     BASE+7  ;FETCH INPUT DATA
008A E67F         133          ANI     7FH     ;MASK OFF MOST SIG. BIT
008C 32DD43       134          STA     BYTEIN
008F C32D00       135          JMP     GPIBIN
                  136 ;
                  137 ;
                  138 ;
                  139 ;     DEVICE CLEAR STATE
                  140 ;
0092 3E10         141 DEVC:    MVI     A,10H   ;RELEASE DAC
0094 320380       14           STA     BASE+3
0097 AF           143          XRA     A
0098 320680       144          STA     BASE+6  ;CLEAR PARALLEL POLL
009B 320580       145          STA     BASE+5  ;CLEAR SERIAL POLL
009E 2F           146          CMA
009F 327243       147          STA     FIRSFL  ;REINT. EVERYTHING
00A2 C30000   E   148          JMP     DCSTRT  ;JUMP TO INSTRUMENT RESET STATE
                  149                          ; BUT DO NOT RESET 68458!!!
                  150 ;
                  151 ;
                  152 ;  REMOTE/LOCAL CHANGE
                  153 ;
00A5 7A           154 REML:    MOV     A,D     ;GET THE COMMAND REG.
00A6 E640         155          ANI     40H     ;IS IT NOW REMOTE ENABLED?
00A8 CAB600   C   156          JZ      SKP6    ;IF NOT THEN SKIP
00AB 3EFF         157          MVI     A,0FFH  ;SET REMFL TO INDICATE REM
00AD 32CC43       158          STA     REMFL
00B0 3EE5         159          MVI     A,0E5H  ;RE-ENBLE INTERUPTS
00B2 320080       160          STA     BASE
00B5 C9           161          RET
00B6 AF           162 SKP6:    XRA     A       ;SET REMFL TO INDICATE LOCAL
00B7 32CC43       163          STA     REMFL
00BA 3E84         164          MVI     A,84H   ;DISABLE ALL BUT CMD. INT.
```

```
 LOC  OBJ          LINE        SOURCE STATEMENT

 00BC 320080        165          STA     BASE
 00BF C9            166          RET             ;RETURN
                    167 ;
                    168 ;
                    169 ;
                    170 ;
                    17           END

PUBLIC SYMBOLS
GINTR  C 0000

EXTERNAL SYMBOLS
DCSTRT E 0000    GETBY  E 0000

USER SYMBOLS
ABLOCK A 43C0    BASE   A 8002    BYTEIN A 43DD    CMD    C 004F
DCSTRT E 0000    DEVC   C 0092    END1   C 003D    ETX    A 0003
FIRSFL A 4372    G1     C 0003    GET    C 0042    GETBYT E 0000
GINTR  C 0000    GPIBIN A 002D    INADD  A 000C    INB    C 0087
NULL   A 0000    OUTB   C 0068    OUTB1  C 0079    OVT    C 003B
REMFL  A 43CC    REML   C 00A5    SKIP08 C 0082    SKP8   C 00B6
SWREG  A 43ED

ASSEMBLY COMPLETE,   NO ERRORS
```

```
ASM80.OV4 :F1:GSER1.ASM PRINT(:LP:) PAGEWIDTH(80) DEBUG


ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        GSER     PAGE    1

  LOC  OBJ          LINE        SOURCE STATEMENT

                       1          NAME    GSER
                       2          PUBLIC  GSER
  43CB                 3 PPMASK   EQU     43CBH   ;PARALLEL POLL MASK
  43C0                 4 ABLOCK   EQU     43C0H
  8000                 5 BASE     EQU     8000H   ;BASE ADDRESS OF 68488
  436E                 6 TRANSM   EQU     436EH
  43DE                 7 GBTHER   EQU     43DEH   ;GPIB PRESENT FLAG
  43AB                 8 RSTHER   EQU     43ABH   ;RS232 FLAG
  0081                 9 SCNTRL   EQU     81H
                      10 ;*************************************************
                      11 ;
                      12 ;    GSER--
                      13 ;  THIS ROUTINE EXECUTES A SERVICE REQUEST FOR
                      14 ; THE FOLLOWING CONDITIONS:
                      15 ;
                      16 ;      1. CHANGE IN SIGNAL STATUS
                      17 ;      2. A BAD COMMAND HAS OCCURRED
                      18 ;      3. A MEASUREMENT IS COMPLETE
                      19 ;
                      20 ;   THE CONDITION THAT OCCURRED IS PASSED IN THE
                      21 ;  ACCUMULATOR IN THE FORM OF THE PROPER MASK
                      22 ;  TO THE SERIAL POLL REGISTER. THE ROUTINE
                      23 ;  READS IN THE SERIAL POLL REG. AND "ANDS" THE
                      24 ;  THE ACCUMULATOR VALUE AND THEN OUTPUTS IT BACK.
                      25 ;
                      26 ;   THE MASKS SHOULD BE OF THE FOLLOWING FORM:
                      27 ;      1. CHANGE IN STATUS  A=40H
                      28 ;      2. BAD COMMAND       A=60H
                      29 ;      3. MEAS. COMPLETE    A=08H
                      30 ;      4. SIMPLE TRANS-
                      31 ;         MISSION           A=00H
                      32 ;  (IF MORE THAN ONE, OR THEM TOGETHER)
                      33 ;
                      34 ;
                      35 ;  IF THE GPIA IS ALREADY ADDRESSED AS A TALKER THEN
                      36 ;  IT WILL GO AHEAD AND SPIT OUT THE MESSAGE AND NOT
                      37 ;  ISSUE A SERVICE REQUEST FOR THAT PARTICULAR MEAS-
                      38 ;  UREMENT COMPLETE.
                      39 ;
                      40 ;  IF THE GPIA IS NOT THERE THEN THE ROUTINE WILL
                      41 ;  TRY TO SEND THE MESSAGE TO THE RS232. IF NEITHER
                      42 ;  THE RS232 NOR THE GPIA IS THERE THEN IT WILL RETURN
                      43 ;  AUTOMATICALLY TO THE CALLING PROG.
                      44 ;
                      45 ;*************************************************
                      46 ;
                      47          CSEG
                      48 ;
  0000 4F             49 GSER:    MOV     C,A     ;STORE MASK IN C
  0001 3ADE43         50          LDA     GBTHER  ;SEE IF 68488 IS THERE
  0004 A7             51          ANA     A
  0005 C21200    C    52          JNZ     SKOV    ;TRY RS232 IF GPIA IS NOT THERE
  0008 3AAB43         53          LDA     RSTHER
  000B A7             54          ANA     A       ;IF IT IS ZERO, RETURN
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

000C C8             55          RZ
000D 3E37           56          MVI     A,37H
000F D381           57          OUT     SCNTRL
0011 C9             58          RET
0012 3A0280         59 SKOV:    LDA     BASE+2  ;SEE IF IN TALK ONLY MODE
0015 E640           60          ANI     40H
0017 CA2400    C    61          JZ      SKPE    ;JUMP OVER IF NOT T.O.
001A 79             62          MOV     A,C     ;GET MASK
001B E608           63          ANI     08H     ;IS IT A MEAS. COMPLETE
001D C0             64          RNZ
001E 79             65 SKPA:    MOV     A,C     ;GET MASK
001F E6B7           66          ANI     0B7H    ;AND OUT MEAS. COM. & SER.
                    67                          ;REQUEST
0021 C35300    C    68          JMP     OVG     ;JUMP TO END
0024 3ACB43         69 SKPE:    LDA     PPMASK  ;FETCH PARALLEL POLL MASK
0027 320680         7           STA     BASE+6
002A 79             7           MOV     A,C     ;CHECK TO SEE IF SIMPLE
                    72                          ;TRANSMISSION
002B E608           73          ANI     08H     ;FIND OUT IF MEAS. COMPLETE
002D C24A00    C    74          JNZ     MEAS
0030 3A0580         75 STSP:    LDA     BASE+5  ;IF NOT, FETCH SERIAL POLL
                    76                          ;REG. AND OR IN MASK
0033 47             77          MOV     B,A
0034 E640           7           ANI     40H
0036 78             79          MOV     A,B
0037 CA3E00    C    8           JZ      OV9     ;IF NOT SET THEN GO ABOUT SETTIN
                       G
                    81                          ;SERIAL POLL NORMALLY
003A B1             82          ORA     C       ;OTHERWISE OR IN NEW STUFF
003B C34600    C    83          JMP     DOWN3
003E B1             84 OV9:     ORA     C
003F F5             8           PUSH    PSW     ;SAVE ON STACK
0040 E63F           8           ANI     3FH     ;MASK OUT EVERYTHING EXCEPT
                    87                          ;SERVICE REQUEST
0042 320580         88          STA     BASE+5  ;ZERO OUT SERVICE REQUEST
0045 F1             89          POP     PSW     ;RESTORE SERVICE REQUEST
0046 320580         90 DOWN3:   STA     BASE+5  ;THIS IS DONE SO THAT SERVICE
                    9                           ;REQUEST WILL HAVE A RISING EDGE
0049 C9             92          RET             ;RETURN
                    93 MEAS:
004A 79             94          MOV     A,C
004B F640           95          ORI     40H     ;OR IN SERVICE REQUEST
004D C35300    C    96          JMP     OVG     ;JUMP OVER
0050 79             97 SKPR:    MOV     A,C
0051 E6F7           98          ANI     0F7H    ;AND OUT THE MEAS. COMPLETE
                    99                          ;BIT
0053 4F             100 OVG:    MOV     C,A     ;SAVE POLL INFORMATION
0054 CD3000    C    101         CALL    STSP    ;SET SERVICE REQUEST
                    102 OV31:
0057 3EE5           10          MVI     A,0E5H  ;ENABLE BO INTERPUPT
0059 F3             10          DI
005A 320280         105         STA     BASE
005D FB             106         EI
005E C9             107         RET
                    108         END
```

LOC OBJ LIN SOURCE STATEMENT

PUBLIC SYMBOLS
GSER C 0000

EXTERNAL SYMBOLS

USER SYMBOLS

ABLOCK	A	43C0	BASE	A	8000	DOWN3	C	0046	GBTHER	A	43DE
GSER	C	0000	MEAS	C	004A	OV9	C	003E	OVG	C	0053
OVG1	C	0057	PPMASK	A	43CB	RSTHER	A	43AB	SCNTRL	A	0081
SKOV	C	0212	SKPA	C	021E	SKPE	C	0024	SKPR	C	0050
STSP	C	0030	TRANSM	A	436E						

ASSEMBLY COMPLETE, NO ERRORS

```
ASM80.OV4 :F1:PSERV1.ASM PRINT(:LP:) PAGEWIDTH(80) DEBUG


ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0         PSERV    PAGE    1

  LOC  OBJ          LINE        SOURCE STATEMENT

                      1
                      2          NAME PSERV
                      3          PUBLIC PSERV
                      4
                      5          EXTRN   PRNGIN,GETBYT
  3E92                6 PDAT1    EQU     3E92H
  3EAE                7 PBUF1    EQU     3EAEH
                      8 ; ***********************************************
                      9 ; PORTS:
  00E3               1          CNTLWD  EQU     0E3H
  00E1               11         DATAB   EQU     0E1H
  00E2               12         STACTL  EQU     0E2H
  00E0               13         EXTRA   EQU     0E0H
                     14
                     15 ; ***********************************************
                     16 ; PORT E2 BIT ASSIGNMENT
                     1  ;       BIT 0 -- OUTPUT NOT(WRITE)
                     1  ;       BIT 1 -- OUTPUT NOT(RESET)
                     1  ;       BIT 4 -- INPUT NOT(ERROR)
                     20 ;       BIT 5 -- INPUT NOT(BUSY) -- 20H
                     21 ;       BIT 6 -- INPUT NOT(PRTON) -- 40H
                     22 ;       BIT 7 -- INPUT NOT(PAPOK) -- 80H
                     23
                     24
                     25 ; ***********************************************
                     2  ; VARIABLES:
  43A7               27         PRTFLG  EQU     43A7H
  4398               2          PBLOCK  EQU     4398H
  439C               29         PINPTR  EQU     439CH
  4398               3          POUPTR  EQU     4398H
                     31
                     32
                     33 ; ***********************************************
                     3  ; JUMP TABLE ENTRY
                     3          ORG     33H
  0033 C30000    C   36         JMP PSERV
                     37
                     38
                     39         CSEG
                     40
                     41 ; ***********************************************
                     4  ; ENTRY POINT FOR PRINTER SERVICE -- PSERV
                     4  ; IF PRINTER@THERE FLAG IS FALSE, BRANCH TO
                     44 ;       INITIALIZATION ROUTINE(SEE IF PRT THERE)
                     45 ; IF PRINTER@THERE=TRUE, BRANCH TO SEE IF A
                     46 ;       MESSAGE TO PROCESS/ OR IN PROCESS ALREADY
                     47 ; ***********************************************
                     48
                     49 PSERV:
  0000 21A743        5          LXI     H,PRTFLG
  0003 7E            51         MOV     A,M
  0004 B7            52         ORA     A
  0005 CA3000    C   53         JZ      INITPR
                     54
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

                    55 ; ***********************************************
                    56 ; CHECK TO SEE IF MESSAGE IN PROGRESS -- STAT=1
                    57 ;       AND CONTINUE
                    58 ; ELSE SEE IF NEW MESSAGE TO START -- STAT=FFH
                    59 ; ***********************************************
                     6  PRTCH:
0008 3A9C43         61          LDA     PINPTR  ;GET INPUT POINTER
000B 4F             62          MOV     C,A     ;STORE
000C 3A9843         63          LDA     POUPTR  ;GET OUTPUT POINTER
000F B9             64          CMP     C
0010 C8             65          RZ              ;RETURN IF EQUAL
                    66
                    67
                     6  ; ***************************************************
                    69 ; TRY TO OUTPUT CHARACTER, IF SO MOVE POINTER TO NEXT
                    70 ;       SEE IF NEXT IS END OF MESSAGE(0) OR
                     7  ;       IF NEXT IS A CARRIAGE RETURN(0DH) WHICH
                     7  ;       IS IGNORED!!
                     7  ; AT MESSAGE END, LOOK AT OTHER STAT TO SEE IF ANOTHER
                     7  ;       MESSAGE ON QUE
                    75 ; ***************************************************
                    76 PRPROC:
0011 DBE2           77          IN      STACTL
0013 E6C0           78          ANI     0C0H    ;OR IS PRT OFF?
                    79                          ;OR NO PAPER??
0015 CA2000    C    80          JZ      CON
0018 AF             81          XRA     A
0019 32A743         82          STA     PRTFLG  ;SET PRINTER@THERE
001C 32C43D         83          STA     3DC4H   ;RESET UNLOGGEDERRORS TO PRINT
001F C9             8           RET             ;  = FALSE
                    85
                    86 CON:
0020 DBE2           87          IN      STACTL  ;SEE IF BUSY
0022 E620           88          ANI     20H
0024 C8             89          RZ              ;RETURN IF BUSY
0025 219843         90 GETIT:   LXI     H,PBLOCK
0028 CD0000    E    91          CALL    GETBYT
002B A7             92          ANA     A
002C C8             9           RZ              ;RETURN IF ZERO
                    94
002D C35800    C    95          JMP     SENDCH
                    96
                    9
                    98
                    99
                   100
                   101
                   102 ; ***************************************************
                   103 ;INITIALZE THE PRINTER INTERFACE, SET PRINTER@THERE FLAG
                   104 ;       IF THE PRINTER IS ON
                   105 ; ***************************************************
                    1  INITPR:
0030 3E98          107          MVI     A,98H   ;CONFIGURE 8255 CHIP
                   108                          ;PORTB DATA OUTPUT
                   109                          ;PORTC HI NIB STAT, LO NIB CNTL
```

```
  LOC  OBJ          LINE     SOURCE STATEMENT

                     110                          ;MODE 0 OPERATION, STRAIGHT
                     111                          ; I/O
  0032 D3E3          112          OUT     CNTLWD
                     113
                     114 CHKON:
  0034 DBE2          115          IN      STACTL  ;SEE IF PRINTER IS ON
  0036 E6C0          116          ANI     0C0H
  0038 C21100    C   117          JNZ     PRPROC  ;GO BACK TO SET "UNLOGGED ERRORS

  003B 35            118          DCR     M       ;SET PRINTER@THERE
                     119                          ;   = TRUE(FFH)
                     120
  003C CD0000    E   121          CALL    PRNGIN  ;SET RING BUFFER UP
  003F 3E01          122          MVI     A,01H
  0041 D3E2          123          OUT     STACTL  ;RESET BIT1 ACTIVE LO,
  0043 32923E        124          STA     PDAT1   ;INIT. BUFFERS
  0046 32AF3E        125          STA     PBUF1
                     126                          ;WRITE BIT0 PASSIVE HI
                     127
  0049 210005        128          LXI     H,500H  ;HOLD RESET FOR >=50M.
                     129                          ;  SECONDS
  004C 23            130 DELAY1:  INX     H
  004D 7C            131          MOV     A,H
  004E B5            132          ORA     L
  004F C24C00    C   133          JNZ     DELAY1
                     134
  0052 3E03          135          MVI     A,3     ;RETURN RESET LINE TO PASSIVE HI
  0054 D3E2          136          OUT     STACTL
  0056 3E0A          137          MVI     A,0AH   ;SEND A LF TO PRINTER
                     138
                     139
                     140 SENDCH:
  0058 D3E1          141          OUT     DATAB   ;CHARACTER OUT
  005A AF            142          XRA     A       ;PULSE WRITE TO PRINTER
  005B F3            143          DI              ;NO INTERRUPTS DURING STROBE!
  005C D3E3          144          OUT     CNTLWD
  005E 3C            145          INR     A       ;SET STROBE LOW, THEN HI
  005F D3E3          146          OUT     CNTLWD
  0061 FB            147          EI
                     148
  0062 C9            149          RET
                     150
                     151
                     152          END
```

```
PUBLIC SYMBOLS
PSERV  C 2000

EXTERNAL SYMBOLS
GETBYT E 0000    PRNGIN E 0000

USER SYMBOLS
CHKON  C 0034    CNTLWD A 00E3    CON    C 0020    DATAB  A 00E1
DELAY1 C 004C    EXTRA  A 02E0    GETBYT E 0000    GETIT  C 0025
```

```
INITPR C 0030    PBLOCK A 4398    PBUF1  A 3EAE    PDAT1  A 3E92
PINPTR A 439C    POUPTR A 4398    PRNGIN E 0000    PRPROC C 0011
PRTCH  C 0008    PRTFLG A 43A7    PSERV  C 0000    SENDCH C 0058
STACTL A 00E2
```

ASSEMBLY COMPLETE, N ERRORS

```
ASM80.OV4 :F1:GETBYT.ASM PRINT(:LP:) PAGEWIDTH(80) DEBUG


ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        GETBYT   PAGE    1

  LOC  OBJ          LINE        SOURCE STATEMENT

                      1          NAME    GETBYT
                      2          PUBLIC  GETBYT
                      3 ;
0020                  4 SPC      EQU     20H      ;ASCII SPACE
0003                  5 ETX      EQU     03H
0030                  6 RBLK1    EQU     30H
                      7 ;
                      8 ;*****************************************************
                      9 ;
                     10 ;   GETBYT-
                     11 ;
                     12 ;      THIS ROUTINE DETERMINES THE NEXT CHARACTER THAT
                     13 ; IT IS SCHEDULED TO GO OUT OVER THE PRINTER OR RS232/
                      1 ; GPIB INTERFACES. THE ONLY INPUT IS THE POINTER TO THE
                     15 ; APPRORIATE RING BUFFER PARAMETER BLOCK. SOMEWHERE IN
                     16 ; MEMORY THERE WILL BE SUCH A BLOCK FOR THE PRINTER AND
                     17 ; ONE FOR THE RS232/GPIB. THE ROUTINE IS DESIGNED AROUND
                     18 ; THE FOLLOWING STRUCTURE:
                     19 ;
                     20 ;  INPUT POINTER---> [ RING BUFFER OUTPUT POINTER ]
                     21 ;                    [     (2 BYTES)               ]
                     22 ;                    [ OFFSET COUNTER FOR OUTPUT   ]
                     23 ;                    [     (2 BYTES)               ]
                      2 ;                    [ RING BUFFER INPUT POINTER   ]
                     25 ;                    [     (2 BYTES)               ]
                     26 ;                    [ SPACE COUNTER               ]
                     27 ;                    [ NO. OF BUFFER POSIT. AVAIL. ]
                     28 ;                    [ STARTING ADD. OF RING BUFF. ]
                     29 ;                    [     (2 BYTES)               ]
                     30 ;                    [ TOT. COUNT OF BUFF. POSITS. ]
                      3 ;
                     32 ;  THE RING BUFFER ITSELF LOOKS LIKE THIS:
                     33 ;
                     34 ;                    [                             ]
                     35 ;                    [                             ]
                     36 ;                    [   2 BYTE POINTERS TO        ]
                     37 ;                    [  FRAGMENTS OF MESSAGES      ]
                     38 ;   OUTPUT PTR-----> [                             ]
                     39 ;                          .    .     .    .
                     40 ;                          .    .     .    .
                     41 ;                    [                             ]
                     42 ;                    [                             ]
                     43 ;                    [                             ]
                     44 ;   INPUT PTR.-----> [                             ]
                     45 ;                    [                             ]
                     46 ;
                     47 ;   THE OUTPUT POINTER POINTS TO THE MESSAGE THAT
                     48 ; IS CURRENTLY GOING OUT. THE INPUT POINTER POINTS
                     49 ; TO THE NEXT AVAILABLE SPOT ON THE RING BUFFER.
                     50 ;  (THE POINTERS WRAP AROUND)
                     51 ;
                     52 ;
                     53 ;
                     54 ;  THE POINTER TO THE PARAMETER BLOCK IS PASSED IN
```

```
LOC  OBJ          LINE       SOURCE STATEMENT

                    55 ; H&L AND THE CHARACTER IS RETURNED IN THE ACCUMULATOR.
                    56 ; IF THE RETURNED CHAR. IS ZERO, IT MEANS THAT THERE
                    57 ; ARE NO MORE BYTES TO SEND.
                    58 ;   IF AN ETX CHARACTER GOES OUT THE "C" REGISTER HAS A
                    59 ; 20H IN IT TO INDICATE AN EOI SHOULD GO OUT IF TRANS-
                    6  ; MITTING ON THE BUS. OTHERWISE IT IS ZERO.
                    6  ;  SPACES CAN BE PACKED TOGETHER IN THE MESSAGE BUFFERS
                    62 ; BY SETTING THE MOST SIG. BIT AND PUTTING THE NUMBER OF
                    63 ; SPACES IN THE LOWER 7 BITS.
                    64 ;
                    65 ;*****************************************************
                    66 ;
                    67 ;
                    6          CSEG
                    69 GETBYT:
0000 E5             70         PUSH    H        ;POINTER IS PASSED IN H- SAVE IT
                    71                          ;ON THE STACK
0001 5E             72         MOV     E,M      ;PUT OUTPUT POINTER IN D&E
0002 23             73         INX     H
0003 56             74         MOV     D,M
0004 23             75         INX     H
0005 4E             76         MOV     C,M      ;PUT OFFSET IN B&C
0006 23             77         INX     H
0007 46             78         MOV     B,M
0008 23             79         INX     H        ;IF INPUT POINTER IS EQUAL TO
                    80                          ;OUTPUT POINTER THEN NO BYTES
                    81                          ;GO OUT AND MUST RETURN A 0 IN
                    82                          ;BOTH ACCUMULATOR AND "C"
0009 7E             83         MOV     A,M
000A BB             84         CMP     E        ;COMPARE LEAST SIGNIFICANT
000B CA1D00    C    85         JZ      BADINT
000E 23             86         INX     H
000F 23             87         INX     H
0010 7E             88         MOV     A,M      ;GET SPACE COUNTER
0011 A7             89         ANA     A        ;PUT OUT SPACES IF COUNTER IS
                    90                          ;NON-ZERO
0012 CA2200    C    91         JZ      BYT1     ;OTHERWISE JUMP
0015 3D             92         DCR     A        ;DECREMENT COUNT OF SPACES
0016 77             93         MOV     M,A      ;RESTORE BACK IN PAR. BLOCK
0017 E1             94         POP     H
0018 3E20           95         MVI     A,SPC    ;SEND A SPACE BACK
001A 0E00           96         MVI     C,0      ;ZERO OUT C
001C C9             97         RET              ;RETURN
                    98 ;
                    99 BADINT:
001D E1            100         POP     H        ;DUMMY POP
001E AF            101         XRA     A
001F 0E00          102         MVI     C,0      ;ZERO OUT A & C
0021 C9            103         RET              ;RETURN
                   104 ;
                   105 BYT1:
0022 EB            106         XCHG             ;PUT OUTPUT POINTER IN H&L
0023 5E            107         MOV     E,M      ;FETCH CURRENT FRAGMENT POINTER
0024 23            108         INX     H
0025 56            109         MOV     D,M
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

0026 62            110         MOV     H,D     ;TRANSFER FRAG. POINTER INTO
                   111                         ;H & L
0027 6B            112         MOV     L,E
0028 09            113         DAD     B       ;ADD OFFSET TO FRAG. POINTER
0029 7E            114         MOV     A,M     ;GET CHARACTER TO SEND
002A A7            115         ANA     A       ;SEE IF IT IS ZERO--IF IT IS
                   116                         ;THEN END OF THAT MESSAGE-
                   117                         ;MUST GO TO NEXT
002B CA4400    C   118         JZ      GOTONX
002E 03            119         INX     B       ;OTHERWISE WE JUST HAVE A REG.
                   120                         ;BYTE TRANSMIT
                   12                          ;MUST INCREMENT OFFSET
002F E1            122         POP     H       ;NOW RESTORE IT IN THE PARAM.
                   123                         ;BLOCK
0030 23            12          INX     H
0031 23            125         INX     H
0032 71            126         MOV     M,C
0033 23            127         INX     H
0034 70            128         MOV     M,B
0035 F5            129         PUSH    PSW     ;SAVE CHAR.
0036 E680          130         ANI     80H     ;IS IT SPACES
0038 C28600    C   131         JNZ     SPCSET  ;NOW JUMP TO SET SPACE COUNTER
003B F1            132         POP     PSW     ;RESTORE CHAR.
003C 0E00          133         MVI     C,0     ;ZERO OUT C
003E FE03          134         CPI     ETX     ;COMPARE TO ETX CHAR.
0040 C0            135         RNZ             ;RETURN IF NOT
0041 0E20          136         MVI     C,20H   ;OTHERWISE SET EOI FLAG
0043 C9            137         RET             ;RETURN
                   138 ;
                   139 ;
                   140 GOTONX:
0044 EB            141         XCHG            ;PUT FRAGMENT POINTER BACK IN P&
                       L
0045 7C            142         MOV     A,H     ;PUT MSB IN H&L
0046 E6F0          143         ANI     0F0H
0048 FE30          144         CPI     RBLK1   ;IF IN RAM THEN PUT FLAG IN FIRS
                       T
                   145                         ;BYTE
004A C24F00    C   146         JNZ     FINSET
004D 3601          147 TSET:   MVI     M,01H
004F E1            148 FINSET: POP     H       ;RETRIEVE PARAM. BLOCK POINTER
0050 E5            149         PUSH    H       ;SAVE IT AGAIN
0051 5E            150         MOV     E,M
0052 23            151         INX     H       ;PUT OUTPUT PTR IN D&E
0053 56            152         MOV     D,M
0054 010700        153         LXI     B,7     ;GET STARTING ADDRESS PARAMETER
                   154                         ;OF THE RING BUFFER AND CAL-
                   155                         ;CULATE IF THERE IS WRAP AROUND
0057 09            156         DAD     B
0058 4E            157         MOV     C,M     ;GET THE STARTING ADDRESS
0059 23            158         INX     H
005A 46            159         MOV     B,M     ;PUT IT IN B&C
005B 23            160         INX     H       ;
005C 6E            161         MOV     L,M     ;NOW GET TOTAL COUNT OF BYTES
                   162                         ;ALLOWED IN THE RING BUFFER
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

005D 2600          163          MVI     H,0      ;ZERO OUT H
005F 2B            164          DCX     H        ;SUBTRACT TWO FROM IT IN ORDER
                   165                           ;TO ALLIGN IT WITH THE LAST
                   166                           ;POINTER
0060 29            167          DAD     H
0061 09            168          DAD     B        ;ADD TO STARTING ADDRESS
0062 7D            169          MOV     A,L      ;PUT LEAST SIG. BYTE IN ACC.
0063 BB            170          CMP     E        ;COMPARE WITH CURRENT OUTPUT
                   171                           ;POINTER
0064 CA6B00    C   172          JZ      BYT2     ;IF EQUAL THEN JUMP
0067 13            17           INX     D        ;OTHERWISE INCREMENT OUTPUT
                   174                           ;POINTER
0068 13            175          INX     D
0069 4B            176          MOV     C,E
006A 42            177          MOV     B,D      ;PUT NEW OUTPUT POINTER IN B&C
                   178                           ;IF WRAPPED AROUND, THE NEW
                   179                           ;OUTPUT POINTER IS ALREADY IN
                   180                           ;B&C (STARTING ADDRESS)
006B E1            181 BYT2:    POP     H        ;RETRIEVE PARAMETER BLOCK PTR.
006C E5            182          PUSH    H        ;SAVE IT AGAIN
006D 71            183          MOV     M,C      ;RESTORE OUTPUT POINTER BACK
                   184                           ;INTO PARAM. BLOCK
006E 23            185          INX     H
006F 70            186          MOV     M,B
0070 23            187          INX     H        ;RESET OFFSET
0071 3600          188          MVI     M,0
0073 23            189          INX     H
0074 3600          190          MVI     M,0
0076 23            191          INX     H
0077 7E            192          MOV     A,M      ;GET INPUT POINTER LEAST
                   193                           ;SIGNIFICANT BYTE
0078 B9            194          CMP     C        ;COMPARE TO NEW OUTPUT POINTER
                   195                           ; IF THEY ARE THE SAME THEN NO
                   196                           ; MESSAGES ARE READY TO GO AND
                   197                           ; A ZERO WILL BE RETURNED
0079 F5            198          PUSH    PSW      ;SAVE FLAGS
007A 23            199          INX     H        ;INCREMENT TO "SPACES LEFT"
007B 23            200          INX     H        ; BYTE IN PARAM. BLOCK
007C 23            201          INX     H
007D 34            202          INR     M        ;INCREMENT IT BECAUSE WE HAVE
007E F1            203          POP     PSW      ;RESTORE FLAGS
007F E1            204          POP     H        ;RESTORE ORIGINAL POINTER
0080 C20000    C   20           JNZ     GETBYT   ;IF THE INPUT POINTER IS NOT
                   206                           ;THE SAME AS THE OUTPUT POINTER
                   207                           ;THEN ANOTHER FRAGMENT IS READY
                   208                           ; SO JUMP BACK TO THE START TO
                   209                           ; FETCH THE NEXT BYTE
0083 AF            210          XRA     A        ;LOAD ZERO IF LAST BYTE TO GO
0084 4F            211          MOV     C,A
0085 C9            212          RET              ; AND RETURN
                   213 ;
                   214 ;
                   215 SPCSET:
0086 F1            216          POP     PSW
0087 23            217          INX     H
```

```
  LOC  OBJ          LINE        SOURCE STATEMENT

  0088 23            218          INX     H
  0089 23            219          INX     H
  008A E67F          220          ANI     7FH     ;TAKE SPACE COUNT AND PUT IT IN
                     221                          ;MEMORY
  008C 3D            222          DCR     A       ;DECREMENT COUNTER
  008D 77            223          MOV     M,A
  008E 3E20          224          MVI     A,SPC
  0090 0E00          225          MVI     C,0
  0092 C9            226          RET             ;RETURN
                     227          END

PUBLIC SYMBOLS
GETBYT C 0000

EXTERNAL SYMBOLS


USER SYMBOLS
BADINT C 001D    BYT1   C 0022    BYT2   C 006B    ETX    A 0003
FINSET C 004F    GETBYT C 0000    GOTONX C 0044    RBLK1  A 0030
SPC    A 0020    SPCSET C 0086    TSET   C 004D

ASSEMBLY COMPLETE,  NO ERRORS
```

D. ERROR COUNTER DRIVERS

```
ASM80.OV4 :F1:EREAD1.A M PRINT(:LP:) PAGEWIDTH(80) DEBUG


ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0         ERREAD   PAGE    1

 LOC  OBJ          LINE        SOURCE STATEMENT

                     1          NAME    ERREAD
                     2          PUBLIC  ERREAD
                     3          PUBLIC  STREAD
                                EXTRN   DISPLA
 43E4                5 ERBUF    EQU     043E4H
 43EC                6 FLADD    EQU     043ECH
 43E1                7 BLUFLG   EQU     043E1H
 00FF                8 TRU      EQU     0FFH
 0000                9 FAL      EQU     00H
 0009               10 TEN6     EQU     0009H
 0093               11 FLGDTA   EQU     93H
 43E0               12 RUNSTP   EQU     43E0H
 0020                1  LOBE    EQU     20H
 0021                1  HIBE    EQU     21H
 0092               15 PARER    EQU     92H
 0090                1  LOBVE   EQU     90H
 8000               17 PASE     EQU     8000H   ;BASE ADDRESS FOR 68488
 43DE                1  GBTHER  EQU     43DEH
 0091               19 HIBVE    EQU     91H
 43EA               20 PRBSL    EQU     43EAH
 43A8               21 UNFRAM   EQU     43A8H
                    22          CSEG
                    23 ;
                    24 ;*************************************************
                    25 ;
                    26 ;   ERREAD
                    27 ;      THIS ROUTINE READS ALL THE ERROR COUNTERS
                    28 ;   (BIT,PARITY, AND BIPOLAR VIOLATIONS), PLACES
                    29 ;   THEIR CONTENTS INTO A COMMON RAM LOCATION AND
                    3  ;   CALLS THE 10**6 ERROR ROUTINE.
                    3  ;
                    32 ;      THE FOLLOWING INPUT ADDRESSES ARE LISTED
                    33 ;   BELOW:
                    34 ;            BIT ERRORS (8 LSB'S)- ADDRESS 20H
                    3  ;            BIT ERRORS (4 MSB'S)- ADDRESS 21H
                    36 ;            PARITY ERRORS (8 BITS)- ADDRESS 52H
                    3  ;            BIPOLAR VIOL. ERRORS-
                    3  ;                        (8 LSB'S)- ADDRESS 50H
                    39 ;                        (4 MSB'S)- ADDRESS 51H
                    40 ;      THESE INPUT ADDRESSES ARE MAPPED TO THE
                    41 ;   FOLLOWING COMMON RAM LOCATIONS:
                    42 ;            BIT ERRORS (LS)    -ADDRESS 3FE8H
                    4  ;            BIT ERRORS (MS)    -ADDRESS 3FE9H
                    44 ;            PARITY ERRORS (LS) -ADDRESS 3FEAH
                    45 ;            PARITY ERRORS (MS) -ADDRESS 3FEBH
                    46 ;            BIPOLAR VIOL. (LS) -ADDRESS 3FECH
                    47 ;            BIPOLAR VIOL. (MS) -ADDRESS 3FEDH
                    48 ;*************************************************
                    49 ;
                    5  ERREAD:
 0000 DB58          51          IN      58H     ;PREVENT DEAD MAN RESET
 0002 21E443        5           LXI     H,ERBUF ;SET POINTER TO COMMON RAM
                    53                          ;ADDRESS
 0005 DB20          54          IN      LOBE    ;INPUT BIT ERROR COUNT (8
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

                    55                          ;LSB'S)
0007 77             56          MOV     M,A     ;STORE IT
0008 23             57          INX     H       ;INCREMENT MEM. POINTER
0009 DB21           58          IN      HIBE    ;INPUT BIT ERROR COUNT (4
                    59                          ;MSB'S)
000B E60F           60          ANI     0FH     ;MASK OUT INVALID DATA
000D 77             61          MOV     M,A     ;STORE IT
000E 23             62          INX     H       ;INCREMENT MEM. POINTER
000F DB92           6           IN      PARER   ;INPUT PARITY ERROR COUNT
                    64                          ;(LEAST SIG. 8 BITS)
0011 77             65          MOV     M,A     ;STORE IT
0012 23             66          INX     H       ;INCREMENT MEM. POINTER
0013 3600           67          MVI     M,0     ;SET UPPER 8 BITS OF
                    68                          ;PARITY ERROR COUNT
                    69                          ;TO ZERO
0015 23             7           INX     H       ;INCREMENT MEM. POINTER
0016 DB90           71          IN      LOBVE   ;INPUT BIPOLAR VIOLATION
                    7                           ;ERROR COUNT (8 LSB'S)
0018 77             7           MOV     M,A     ;STORE IT
0019 23             74          INX     H       ;INCREMENT MEM. POINTER
001A DB91           75          IN      HIBVE   ;INPUT BIPOLAR VIOLATION
                    76                          ;ERROR COUNT (4 MSB'S)
001C E60F           77          ANI     0FH     ;MASK OUT INVALID DATA
001E 77             78          MOV     M,A     ;STORE IT
001F CD0900         79          CALL    TEN6    ;CALL TEN**6 INTERRUPT
                    80                          ;ROUTINE
0022 C9             81          RET
                    82 ;************************************************
                    8  ; THE FOLLOWING
                    8  ; ROUTINE SETS THE SIGNAL LOSS, FRAME LOSS, AND
                    8  ; PRBS LOSS AND BLU-CONDITION FLAGS.
                    86 ;
                    87 ;  TO READ THE FLAG DATA AN "IN 53H" INSTRUCTION IS
                    88 ; EXECUTED.THE FOLLOWING DATA IS PLACED INTO THE
                    89 ; ACCUMULATOR:
                    90 ;               BIT 0- SIGNAL LOSS
                    91 ;               BIT 1- F-FRAME
                    92 ;               BIT 2- M-FRAME
                    93 ;               BIT 3- PRBS SYNC
                    94 ;                BIT 4 - BLU-CONDITION
                    95 ; THESE BITS ARE MAPPED TO THE COMMON RAM FLAGS IN THE
                    96 ; FOLLOWING MANNER:
                    9  ;               SIGNAL LOSS FLAG= (BIT 0)'
                    98 ;               FRAME LOSS FLAG=(BIT 1)'+ (BIT 2)'
                    99 ;               PRBS LOSS FLAG=BIT 3
                   100 ;                BLU-CONDITION= (BIT 4)'
                   101 ; THE RAM LOCATIONS TO STORE THE FLAG DATA ARE AS
                   102 ; FOLLOWS:
                   103 ;               PRBS LOSS   - 3FEEH
                   104 ;               FRAME LOSS  - 3FEFH
                   105 ;               SIGNAL LOSS - 3FF0H
                   106 ;                BLU-CONDITION  - 3FE5H
                   107 ;************************************************
                   108 ;
                   109 STREAD:
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

0023 1669          110         MVI     D,69H   ;SET REG. D TO ACCUMULATE THE
                   111                         ;PROPER STATUS INFO. FOR THE
                   112                         ;SERIAL POLL IN THE 68468
0025 21EC43        113         LXI     H,FLADD ;SETS MEM. POINTER TO BE-
                   114                         ;GINNING FLAG ADDRESS
0028 E5            115         PUSH    H
0029 DB93          116         IN      FLGDTA  ;READS FLAG DATA
002B 0EFF          117         MVI     C,TRU   ;MOVE TRUE DATA INTO REG. C
002D 0F            118         RRC             ;PLACES SIGNAL LOSS BIT
                   119                         ;INTO CARRY
002E 47            120         MOV     B,A     ;STORES OTHER BITS
002F DA3800    C   121         JC      F1      ;JUMP IF TRUE
0032 0E00          122         MVI     C,FAL   ;LOAD REG. C WITH FALSE
                   123                         ;DATA
0034 7A            124         MOV     A,D     ;MOVE SER. POLL TO ACCUM.
0035 E6FE          125         ANI     0FEH    ;MASK OUT SIGNAL LOSS BIT
0037 57            126         MOV     D,A     ;RESTORE
0038 71            127 F1:     MOV     M,C     ;STORE SIGNAL LOSS DATA
0039 59            128         MOV     E,C     ;SAVE IN E REG.
003A 2B            129         DCX     H       ;DECREMENT MEM. POINTER
003B 7B            130         MOV     A,E     ;SET FRAME LOSS IF SIGNAL LOSS
003C A7            131         ANA     A
003D C24A00    C   132         JNZ     F11
0040 0E00          133         MVI     C,FAL   ;MOVE FALSE DATA INTO REG. C
0042 78            134         MOV     A,B     ;MOVE FLAG DATA INTO
                   135                         ;ACCUM.
0043 2F            136         CMA             ;COMPLEMENT FRAME DATA
0044 E603          137         ANI     03H     ;MASK OUT ALL BUT FRAME DATA
0046 B7            138         ORA     A       ;GENERATE THE ZERO FLAG
0047 CA5000    C   139         JZ      F2      ;JUMP IF FALSE
004A 0EFF          140 F11:    MVI     C,TRU   ;STORE TRUE DATA
004C 7A            141         MOV     A,D     ;GET SERIAL POLL REG.
004D F602          142         ORI     02H     ;SET FRAME LOSS TO TRUE
004F 57            143         MOV     D,A     ;RESTORE
0050 71            144 F2:     MOV     M,C     ;STORE FRAME LOSS FLAG
0051 0E00          14          MVI     C,FAL   ;MOVE FALSE DATA INTO REG. C
0053 78            146         MOV     A,B     ;MOVE FLAG DATA INTO
                   147                         ;ACCUM
0054 E604          148         ANI     04H     ;MASK OUT ALL BUT THE
                   149                         ;PRBS SYNC SIGNAL
0056 C25E00    C   150         JNZ     F25     ;JUMP IF TRUE
0059 7B            151         MOV     A,E     ;CHECK IF SIGNAL LOSS-IF SO
                   152                         ;THEN AUTOMATICALLY SET
                   153                         ;PRBS LOSS
005A A7            154         ANA     A
005B CA6400    C   155         JZ      F3      ;IF NOT SET THEN SET PRBS TO
                   156                         ;FALSE
005E 0EFF          157 F25:    MVI     C,TRU   ;STORE TRUE DATA INTO
                   158                         ;ACCUM.
0060 7A            159         MOV     A,D     ;GET SERIAL POLL REG.
0061 F604          160         ORI     04H     ;SET PRBS LOSS TO TRUE
0063 57            161         MOV     D,A     ;RESTORE
0064 2B            162 F3:     DCX     H       ;DECREMENT POINTER
0065 71            16          MOV     M,C     ;STORE FLAG DATA INTO
                   164                         ;MEMORY
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

0066 0E00          165          MVI     C,FAL   ;MOVE FALSE DATA INTO C
0068 78            166          MOV     A,B     ;MOVE FLAG DATA INTO A
0069 E608          167          ANI     8       ;MASK ALL BUT BLU-COND
006B C27400    C   168          JNZ     F4      ;JUMP IF  HI,NOT ACTIVE
006E 0EFF          169          MVI     C,TRU   ;MOVE TRUE DATA INTO C
                   170                          ;IF BIT LOW = ACTIVE
0070 7A            171          MOV     A,D     ;GET SERIAL POLL INFO.
0071 F610          172          ORI     10H     ;OR IN BLUE SIGNAL
0073 57            173          MOV     D,A     ;RESTORE
0074 21E143        174 F4:      LXI     H,BLUFLG;STORE FLAG DATA INTO MEM.
0077 3AEA43        175          LDA     PRBSL   ;GET PRBS LOSS FLAG
007A A1            176          ANA     C       ;AND WITH BLUE FLAG
007B 77            177          MOV     M,A
007C F6EF          178          ORI     0EFH    ;SET UP SERIAL POLL REG.
007E A2            179          ANA     D       ;AND PRBS LOSS WITH BLUE
                   180                          ;TO GET ACTUAL BLUE
007F 57            181          MOV     D,A     ;RESTORE IN D
0080 3ADE43        182          LDA     GBTHER  ;SEE IF 68488 IS THERE
0083 A7            183          ANA     A
0084 CA9800    C   184          JZ      SFRA
0087 3A0180        185          LDA     BASE+1  ;DONT DO ANYTHING IF IN SPAS
008A E604          186          ANI     04H
008C C29800    C   187          JNZ     SFRA
008F 3A0580        188          LDA     BASE+5  ;LOAD SERIAL POLL INFO.
0092 F617          189          ORI     17H     ;OR IN MASK TO CHANGE
0094 A2            190          ANA     D       ;AND IN DATA
0095 320580        191          STA     BASE+5  ;STORE SERIAL POLL INFO.
                   192 ;************************************************
                   193 ;
                   194 ;
                   195 ;   NOW SET THE UNFRAMED/FRAMED FLAG
                   196 ;
                   197 SFRA:
0098 E1            198          POP     H       ;RESTORE FLAG POINTER
                   199 ; IF RUNNING THE DONT CHANGE
0099 3AE043        200          LDA     RUNSTP
009C FE01          201          CPI     1
009E C8            202          RZ
                   203 ;
009F 7E            204          MOV     A,M     ;FETCH SIGNAL LOSS
00A0 A7            205          ANA     A
00A1 C0            206          RNZ             ;RETURN IF SIGNAL LOSS
00A2 2B            207          DCX     H
00A3 7E            208          MOV     A,M     ;FETCH FRAME LOSS
00A4 A7            209          ANA     A
00A5 C8            210          RZ              ;RETURN IF FRAMED
00A6 2B            211          DCX     H
00A7 7E            212          MOV     A,M     ;FETCH PRBS LOSS
00A8 A7            213          ANA     A
00A9 C8            21           RZ              ;RETURN IF NO LOSS
00AA 3AE143        215          LDA     BLUFLG
00AD A7            216          ANA     A
00AE C0            217          RNZ             ;RETURN IF BLUE
00AF 21A843        218          LXI     H,UNFRAM;NOW COMPLEMENT THE UNFRAMED/FRA
                   MED
```

```
  LOC  OBJ          LINE        SOURCE STATEMENT

  00B2 7E            219         MOV     A,M
  00B3 2F            220         CMA
  00B4 77            221         MOV     M,A
  00B5 C9            222         RET             ;RETURN
                     223         END


PUBLIC SYMBOLS
ERREAD C 0000    STREAD C 0023

EXTERNAL SYMBOLS
DISPLA E 0000

USER SYMBOLS
BASE   A 8000    BLUFL  A 43E1    DISPLA E 0000    ERBUF  A 43E4
ERREAD C 0000    F1     C 0038    F11    C 004A    F2     C 0050
F25    C 005E    F3     C 0064    F4     C 0074    FAL    A 0000
FLADD  A 43EC    FLGDTA A 0093    GBTHER A 43DE    HIBE   A 0021
HIBVE  A 0091    LOBE   A 0020    LOBVE  A 0090    PARER  A 0092
PRBSL  A 43EA    RUNSTP A 43E0    SFRA   C 0098    STREAD C 0023
TEN6   A 0009    TRU    A 00FF    UNFRAM A 43A8

ASSEMBLY COMPLETE,   NO ERRORS
```

E. JUMP TABLE AND INTERRUPT SERVICE

```
ASM80.OV4 :F1:JTAB1.ASM PRINT(:LP:) PAGEWIDTH(80) DEBUG


ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        JTAB     PAGE    1

  LOC  OBJ          LINE        SOURCE STATEMENT

                                NAME    JTAB
                      2         EXTRN   RTOD,RMOD,STOD,SMOD,ERREAD
                      3         EXTRN   SWITCH,DISPLA
                      4         EXTRN   STREAD
                      5         EXTRN   GINTR,RINTR
                      6         EXTRN   GSER
                      7         EXTRN   LEDTES,PSERV
                      8         EXTRN   RINIT,GINIT
                      9         PUBLIC  PRNGIN,DCSTRT
  43E2                10 DSFLG  EQU     43E2H
  00FF                11 TRU    EQU     0FFH
  0071                1  CNTRL  EQU     71H
  0070                1  DSPLY  EQU     70H
  0006                14 OPSA   EQU     0006H
  0024                15 FINITA EQU     0024H
  43C0                1  ABLOCK EQU     43C0H
  4398                17 PBLOCK EQU     4398H
  43AC                1  ARNGB  EQU     43ACH
  4384                19 PRNGB  EQU     4384H
  0091                20 HIPVE  EQU     91H
  436B                21 LBCNT  EQU     436BH
  436C                22 BCNT   EQU     436CH
  436D                23 BLNG   EQU     436DH
  43A8                24 UNFRAM EQU     43A8H
  43A9                25 ASYNC  EQU     43A9H
  4372                26 FIRSFL EQU     4372H
  43E0                2  RUNSTP EQU     43E0H
  4374                28 RAMTS1 EQU     4374H
  4373                29 RAMTS2 EQU     4373H
  0050                30 INTST  EQU     40H+16
  436F                31 LEDCNT EQU     436FH
  43A7                32 PTHER  EQU     43A7H
  0051                33 INTCON EQU     40H+17
  0021                34 HIBE   EQU     21H
  002A                35 BURAD  EQU     002AH   ;JUMP TABLE ADDRESS FOR BURST
  43DE                36 GBTHER EQU     43DEH
  43AB                37 RSTEER EQU     43ABH
  8000                38 PASE   EQU     8000H
  0080                39 SBASE  EQU     80H
  005A                40 BURLI  EQU     5AH
  0058                41 BURCI  EQU     58H
                      42 ;*************************************************
                      43 ;
                      44 ;   JUMP TABLE
                      4  ; THE FOLOWING JUMP TABLE IS USED TO LINK THE
                      46 ; FORTRAN WRITTEN PROGRAM WITH THE ASSEMELY
                      47 ; LANGUAGE PROGRAM.
                      48 ;
                      49 ;*************************************************
  0000                5         ORG     00H
  0000 C37B00    C    51 MAINA: JMP     DINIT   ;JUMP TO DRIVER INITIALIZATION
                      52                        ; 0000H
                      53 ;FLPA:  JMP     FLP     ;JUMP TO FLOATING POINT
                      54                        ;ROUTINE (FORTRAN) 0003H
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

                    55 ;OPSA:  JMP     OPS      ;JUMP TO ONCE PER SECOND
                    56                          ;ROUTINE (FORTRAN) 000EH
                    57 ;TEN6A: JMP     TEN6     ;JUMP TO TEN**6 ROUTINE
                    58                          ;(FORTRAN) 0009H
                    59 ;INPUTA:JMP     INPUT    ;JUMP TO FRONT PANEL INPUT
                    60                          ;ROUTINE (FORTRAN) 000CH
000F                61         ORG     000FH    ;START AT LOCATION 00FH
000F C30000   E     62 RTODA:  JMP     RTOD     ;JUMP TO READ TIME OF DAY
                    63                          ;ROUTINE (ASSEMBLY)
0012 C30000   E     64 RMODA:  JMP     RMOD     ;JUMP TO READ MONTH & DATE
                    65                          ;ROUTINE (ASSEMBLY)
0015 C30000   E     66 STODA:  JMP     STOD     ;JUMP TO SET TIME OF DAY
                    6                           ;ROUTINE (ASSEMBLY)
0018 C30000   E     68 SMODA:  JMP     SMOD     ;JUMP TO SET MONTH & DATE
                    69                          ;ROUTINE (ASSEMBLY)
001B C30000   E     70 LEDA:   JMP     LEDTES   ;JUMP TO LED TEST-
001E C30000   E     71 DSPLAA: JMP     DISPLA   ;JUMP TO DISPLAY
                    72                          ;ROUTINE (ASSEMBLY)
0021 C30000   E     73 STATUS: JMP     STREAD   ;JUMP TO READ STATUS
                    74                          ;LINES ROUTINE(ASSEMBLY)
                    75 ;DEVCL  JMP     GBCLEAR  ;JUMP TO DEVICE CLEAR STATUS
                    76                          ;(FORTRAN) 0024H
0027                77         ORG     27H
0027 C30000   E     78 GSERA:  JMP     GSER     ;JUMP TO GPIB SERVICE REQUEST
                    79                          ;}ASSEMBLY
                    80                          ;OR TRANSMIT ROUTINE
                    81 ;CHAR   JMP     CHARIN   ;JUMP TO RECIEVE CHARACTER
                    82                          ;ROUTINE (FORTRAN) 002DH
0038                83         ORG     0038H    ;RESTART 7 LOCATION FOR
                    84                          ;INTERRUPT
0038 C30000   C     85         JMP     INT      ;JUMP TO INTERRUPT SEVICE
                    86                          ;ROUTINE
                    87 ;************************************************
                    88 ;
                    89 ;    INTERRUPT SERVICE ROUTINE
                    90 ;      THE INTERRUPTS ARE CHECKED UNDER THE FOL-
                    91 ; LOWING PRIORITY.
                    92 ;     TEN**6  - HIGHEST PRIORITY
                    9  ;     ONCE A SECOND  -SECOND HIGEEST PRIORITY
                    94 ;     FRONT PNL. INPUT  -LOWEST PRIORITY
                    95 ;
                    96 ;************************************************
                    97 ;
                    98          CSEG
                    99 INT:
0020 F5            100         PUSH    PSW      ;SAVE PROCESSOR STATUS
0021 E5            101         PUSH    H
0022 D5            102         PUSH    D
0023 C5            103         PUSH    B
                   104 ; FIRST CHECK POWER DOWN INTERRUPT
                   105 ;       IN      BURCI
                   106 ;       ANI     08H
                   107 ;       JNZ     DEAD
0024 DB50          108         IN      INTST    ;INPUT BIT TO DETERMINE IF
                   109                          ;ONCE A SECOND INTERRUPT HAS
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

                   110                          ;OCCURRED
0006 F5            111          PUSH    PSW     ;SAVE THE REGISTER
0007 DB5A          112          IN      BURLI   ;READ BURST LENGTH
0009 D35A          113          OUT     BURLI   ;RESET INTERRUPT
000B E67F          114          ANI     7FH
000D 326D43        115          STA     BLNG    ;STORE BURST LENGTH
0010 3A6B43        116          LDA     LBCNT   ;GET LAST COUNT
0013 47            117          MOV     B,A
0014 DB58          118          IN      BURCI   ;GET BURST COUNT
0016 4F            119          MOV     C,A
0017 90            120          SUB     B       ;SUBTRACT LAST COUNT
0018 E607          121          ANI     7       ;JUST 3 BITS
001A 326C43        122          STA     BCNT
001D 79            123          MOV     A,C
001E 326B43        124          STA     LBCNT
0021 C42A00        125          CNZ     BURAD   ;CALL BURST IF ANY
0024 DB21          126          IN      HIBE    ;INPUT BIT TO DETERMINE IF
                   127                          ;TEN**6 INTERRURT
0026 E610          128          ANI     10H     ;MASK OUT ALL OTHER BITS
0028 CC0000    E   129          CZ      ERREAD  ;IF BIT=0 GO TO ERROR READ
                   130                          ;ROUTINE
002B 3ADE43        131          LDA     GBTHER  ;SEE IF 68488 IS THERE
002E A7            132          ANA     A
002F CA3D00    C   133          JZ      SKP11   ;JUMP TO SEE IF RS232 IS PRES.
                   134 ; GPIB ON LINE
0032 3A0080        13           LDA     BASE    ;GET INT. STATUS REG.
0035 E680          136          ANI     80H     ;IS INT. FROM GPIB
0037 C40000    E   137          CNZ     GINTR   ;IF SO CALL SERV. ROUTINE
003A C34B00    C   138          JMP     SKP12   ;JUMP ON DOWN
                   139 ;
003D 3AAB43        140 SKP11:   LDA     RSTHER  ;IS RS232 ON LINE?
0040 A7            141          ANA     A
0041 CA4B00    C   142          JZ      SKP12   ;IF NOT JUMP DOWN
0044 DB81          143          IN      SBASE+1 ;SEE IF IT INTERRUPTED
0046 E603          144          ANI     03H     ;MASK OUT ALL BUT
                   145                          ;RXRDY AND TXRDY
0048 C40000    E   146          CNZ     RINTR   ;CALL RS232 INT. SERV. ROUTINE
                   147 ;
                   148 ;
004B F1            149 SKP12:   POP     PSW     ;RECALL THE ONCE A SECOND
                   150                          ;INTERRUPT BIT
004C E604          151          ANI     04H     ;MASK OUT ALL OTHER BITS
004E CA6600    C   152          JZ      FRTPNL  ; ACTIVE HIGH LINE
0051 3A6F43        153          LDA     LEDCNT  ;CHECK LED TEST
0054 A7            154          ANA     A
0055 C25F00    C   155          JNZ     OVOPS   ;IF NON-ZERO CONTINUE
0058 3EA2          156          MVI     A,0A2H  ;IF ZERO ALLOW CHANGE IN DISPLAY
005A D371          157          OUT     CNTRL
005C C36300    C   158          JMP     DOPS
005F 3D            159 OVOPS:   DCR     A
0060 326F43        160          STA     LEDCNT  ;OTHERWISE DECREMENT AND STORE C
                       NT.
                   161 DOPS:
0063 CD0600        162          CALL    OPSA    ;IF BIT=1 JUMP TO ONCE PER
                   16                           ; SECOND ROUTINE
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

0066 DB71          164 FRTPNL: IN      CNTRL   ;INPUT 8279 STATUS WORD
                   165                         ;TO DETERMINE IF FRONT
                   166                         ;PANEL OPERATED
0068 E607          167         ANI     07H     ;MASK OUT ALL BUT NO.
                   168                         ;OF SWITCH ENTRIES
006A C40000    E   169         CNZ     SWITCH  ;CALL SWITCH ROUTINE IF
                   170                         ;ONE OR MORE SWITCHES
                   171                         ;ENTERED
006D 3AE243        172         LDA     DSFLG   ;LOAD DISPLAY ENABLE FLAG
                   173                         ;INTO ACCUM.
0070 FEFF          174         CPI     TRU     ;COMPARE TO TRUE VALUE
0072 CC0000    E   175         CZ      DISPLA  ;CALL DISPLAY IF FLAG IS
                   176                         ;TRUE
0075 C1            177         POP     B       ;RESTORE PROCESSOR STATUS
0076 D1            178         POP     D
0077 E1            179         POP     H
0078 F1            180         POP     PSW
0079 FB            181         EI              ;ENABLE INTERRUPTS
007A C9            182         RET
                    18 ;
                   184 ;
                   185 ;DEAD:  JMP     DEAD    ;I'M MELTING
                   186 ;
                   187 ;*********************************************
                   188 ;
                   189 ;
                   190 ;   DINIT-
                   191 ;  THIS ROUTINE IS THE DRIVER INITIALIZATION
                   192 ; ROUTINE. IT MUST SET THOSE VARIABLES WHICH
                   193 ; ARE DESTOYED IN POWER DOWN.
                   194 ;
                   195 ;*********************************************
                   196 ;
007B 31403C        197 DINIT:  LXI     SP,3C40H
007E 3E2A          198         MVI     A,0AH   ;INITIALIZE 8279
0080 D371          199         OUT     CNTRL
0082 3E34          200         MVI     A,34H
0084 D371          201         OUT     CNTRL
                   202 ;
                   203 ;   RESET CONTROL TO ENABLE NV RAM
                   204 ;
0086 CD0000    E   205         CALL    GINIT   ;GPIB INIT.
0089 CD0000    E   206         CALL    RINIT   ;RS232 INIT.
008C CD0000    E   207         CALL    LEDTES
                   208 PDCHK:
008F 0EFF          209         MVI     C,0FFH
0091 3A7443        210         LDA     RAMTS1  ;CHECK POWER DOWN PATTERN
0094 FEAA          211         CPI     0AAH
0096 C2A300    C   212         JNZ     SETIT   ;IF NOT SET IT
0099 3A7343        213         LDA     RAMTS2
009C FE55          214         CPI     055H
009F C2A300    C   215         JNZ     SETIT
00A1 0E00          216         MVI     C,0
00A3 3EAA          217 SETIT:  MVI     A,0AAH
00A5 327443        218         STA     RAMTS1  ;SET UP RAM
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

00A8 2F            219          CMA
00A9 327343        220          STA     RAMTS2
00AC 79            221          MOV     A,C
00AD 327243        22           STA     FIRSFL  ;SET FIRST FLAG
00B0 CDF000    C   223 DCSTRT:  CALL    ARNGIN  ;INIT. OF AUTO RING BUFF.
00B3 3E04          224          MVI     A,04H   ;INITIALIZE ONE SECOND INT.
00B5 D351          22           OUT     INTCON
00B7 3A7243        226          LDA     FIRSFL  ;SEE IF WE NEED TO INIT. UNFRAM
00BA A7            227          ANA     A
00BB CAC200    C   228          JZ      M2
00BE AF            229          XRA     A
00BF 32A843        230          STA     UNFRAM
00C2 21E043        231 M2:      LXI     H,RUNSTP ;SET RUN/STOP TO RUN TO FOOL
                   232                          ;DISPLA TO SET HARDWARE
00C5 7E            233          MOV     A,M
00C6 E5            234          PUSH    H       ;SAVE POINTER
00C7 F5            235          PUSH    PSW     ;SAVE RUN/STOP STATUS
00C8 3602          236          MVI     M,2     ;SET TO STOP
00CA CD0000    E   237          CALL    DISPLA  ;CALL DISPLA TO SET FRAMING
00CD F1            238          POP     PSW     ;RECALL THE FLAG
00CE E1            239          POP     H       ;RECALL POINTER
00CF 77            240          MOV     M,A     ;RESTORE ORIGINAL STATUS
00D0 AF            241          XRA     A
00D1 32A743        242          STA     PTHER   ;INIT. PRINTER
00D4 CD0000    E   243          CALL    PSERV
00D7 DB58          244          IN      BURCI   ;GET INIT. BURST COUNT
00D9 E607          245          ANI     7
00DB 326F43        246          STA     LBCNT
00DE 0E00          247          MVI     C,0
00E0 DB91          248          IN      HIBVE   ;INPUT HIGH BYTE OF BIPOLAR
                   249                          ;VIOLATION COUNTER TO DETERMINE
                   250                          ;SYNC OR ASYNC
00E2 E610          251          ANI     10H     ;BIT 4
00E4 CAE900    C   252          JZ      T4      ;IF ZERO THEN SYNC,OTHERWISE ASY
                       NC
00E7 0EFF          253          MVI     C,0FFH
00E9 79            254 T4:      MOV     A,C
00EA 32A943        255          STA     ASYNC   ;STORE BYTE
00ED C32400        256          JMP     FINITA  ;JUMP TO FORTRAN INIT.
                   257 ;
                   258 ;
                   259 ;
                   260 ;
00F0 21AC43        261 ARNGIN:  LXI     H,ARNGB
00F3 22C043        262          SHLD    ABLOCK
00F6 22C443        263          SHLD    ABLOCK+4;INIT. INPUT AND OUTPUT
00F9 22C843        264          SHLD    ABLOCK+8
00FC 210000        265          LXI     H,0
00FF 22C243        266          SHLD    ABLOCK+2 ;INIT. OFFSET
0102 AF            267          XRA     A
0103 32C643        268          STA     ABLOCK+6 ;INIT. SPACE COUNTER
0106 3E0A          26           MVI     A,10
0108 32C743        270          STA     ABLOCK+7 ;NUMBER OF POSITONS
010B 32CA43        271          STA     ABLOCK+10
                   272
```

```
 LOC  OBJ          LINE        SOURCE STATEMENT

                    273
 010E 218443        274 PRNGIN: LXI     H,PRNGB
 0111 229843        275         SHLD    PBLOCK  ;INIT INPUT & OUTPUT
 0114 229C43        276         SHLD    PBLOCK+4
 0117 22A043        277         SHLD    PBLOCK+8
 011A 210000        278         LXI     H,0
 011D 229A43        27          SHLD    PBLOCK+2 ;OFFSET
 0120 AF            280         XRA     A
 0121 329E43        281         STA     PBLOCK+6
 0124 3E0A          282         MVI     A,10
 0126 329F43        283         STA     PBLOCK+7
 0129 32A243        284         STA     PBLOCK+10
 012C C9            285         RET
                    286 ;
                    287 ;
                    288         END
```

```
PUBLIC SYMBOLS
DCSTRT C 00B0    PRNGIN C 010E

EXTERNAL SYMBOLS
DISPLA E 0000    ERREAD E 0000    GINIT  E 0000    GINTR  E 0000
GSER   E 0000    LEDTES E 0000    PSERV  E 0000    RINIT  E 0000
RINTR  E 0000    RMOD   E 0000    RTOD   E 0000    SMOD   E 0000
STOD   E 0000    STREAD E 0000    SWITCH E 0000

USER SYMBOLS
ABLOCK A 43C0    ARNGB  A 43AC    ARNGIN C 00F0    ASYNC  A 43A9
BASE   A 8000    BCNT   A 436C    BLNG   A 436D    BURAD  A 002A
BURCI  A 0058    BURLI  A 005A    CNTRL  A 0071    DCSTRT C 00B0
DINIT  C 007B    DISPLA E 0000    DOPS   C 0063    DSFLG  A 43E2
DSPLAA A 001E    DSPLY  A 0070    ERREAD E 0000    FINITA A 0024
FIRSFL A 4372    FRTPNL C 0066    GPTHER A 43DE    GINIT  E 0000
GINTR  E 0000    GSER   E 0000    GSERA  A 0027    HIBE   A 0021
HIPVE  A 0091    INT    C 0000    INTCON A 0051    INTST  A 0050
LBCNT  A 436B    LEDA   A 001B    LEDCNT A 436F    LEDTES E 0000
M2     C 00C2    MAINA  A 0000    OPSA   A 0006    OVOPS  C 005F
PBLOCK A 4398    PDCHK  C 008F    PRNGB  A 4384    PRNGIN C 010E
PSERV  E 0000    PTHER  A 43A7    RAMTS1 A 4374    RAMTS2 A 4373
RINIT  E 0000    RINTR  E 0000    RMOD   E 0000    RMODA  A 0012
PSTHER A 43AE    RTOD   E 0000    RTODA  A 000F    RUNSTP A 43E0
SBASE  A 00B0    SETIT  C 00A3    SKP11  C 003D    SKP12  C 004E
SMOD   E 0000    SMODA  A 0018    STATUS A 0021    STOD   E 0000
STODA  A 0015    STREAD E 0000    SWITCH E 0000    T4     C 00E9
TRU    A 00FF    UNFRAM A 43A8

ASSEMBLY COMPLETE,  NO ERRORS
```

F. FLOATING POINT UTILITIES

```
ASM80.OV4 :F1:FPBCD1.ASM PRINT(:LP:) PAGEWIDTH(80) DEBUG


ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0         FPBCD    PAGE    1

  LOC  OBJ          LINE        SOURCE STATEMENT

                      1          NAME FPBCD
                      2
                      3          EXTRN  FADD, FDIV, FIX, FPCMPR
                      4
                      5 ; ***********************************************
                      6 ;
                      7 ;        ROUTINES TO CONVERT THE FLOATING POINT
                      8 ;        VALUE IN B REG, D PAIR TO BCD IN ERROR@DISPLAY
                      9 ;        ENTER THROUGH JUMP TABLE ENTRY AT 30H
                     10 ;
                     11 ; ***********************************************
                     12
                     13 ; 7 TEMPORARIES FOR FPBCD:
  4375               1          EXPNPT  EQU     4375H
  4377               15         DP      EQU     EXPNPT+2
  4378               16         ADJUST  EQU     EXPNPT+3
  4379               17         P10PTR  EQU     EXPNPT+4
  437B               18         FPEXP   EQU     EXPNPT+6
                     19
                     2  ;BCDNUM IN COMMON
  43A3               21         BCDNUM  EQU     43A3H
                     22
                     23
                     24 ;JUMP TABLE ENTRY:
                     25         ASEG
  0030               26         ORG 30H
  0030 C32700    C   27         JMP FPBCD
                     28
                     29         CSEG
                     3
                     31
                     32 ;CONFIGURATIONS FOR OVERFLOW AND UNDERFLOW VALUES IN ERR
                        OR@DISPLAY
                     33 OVRFLW:
  0000 09            34         DB      9,89H,89H,89H
  0001 89
  0002 89
  0003 89
                     35 UNDFLW:
  0004 89            36         DB      89H,80H,80H,81H
  0005 80
  0006 80
  0007 81
                     37 ZERO:
  0008 0F            38         DB      0FH,0,0,0
  0009 00
  000A 00
  000B 00
                     39
                     40
                     41
                     42 $EJECT
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

                    43 ;*****************************************************
                       ***
                    44 ;CONVERSION ROUTINE FOR BINARY TO BCD -- SA'S DUPDAB
                    45 ;        CONVERT 16 BIT UNSIGNED BIN # IN H AND L REG
                    46          ; TO 5 DIG BCD # W/ LSD'S IN H AND L REG
                    47          ; AD MSD IN A REG
                    48 ;*****************************************************
                       ****
                    49 BINBCD:
0000 011000         50          LXI     B,16
0003 110000         51          LXI     D,0
                    52
                    53 BCDNXT:
0006 29             54          DAD     H
0007 7B             55          MOV     A,E
0008 8B             56          ADC     E
0009 27             57          DAA
000A 5F             58          MOV     E,A
000B 7A             59          MOV     A,D
000C 8A             60          ADC     D
000D 27             61          DAA
000E 57             62          MOV     D,A
000F 78             63          MOV     A,B
0010 88             64          ADC     B
0011 27             65          DAA
0012 47             66          MOV     B,A
0013 0D             67          DCR     C
0014 C20600     C    6          JNZ     BCDNXT
0017 EB             69          XCHG
0018 78             70          MOV     A,B
0019 B7             71          ORA     A
001A C9             72          RET
                    73
                    74
                    75
                    76
                    77 $EJECT
```

```
LOC  OBJ          LIN          SOURCE STATEMENT

                    78 ;*****************************************************
                    79 ;                    F P B C D
                    80 ;OBJECTIVE:
                    81 ;        CONVERT 3 BYTE FLOATING POINT NUMBER
                    82 ;        INTO 3 DIGIT BCD DISPLAY
                    83 ;        W/ EXPONENT OF POWER OF TEN.
                    84 ;        4 BYTE BCD RESULT IS PLACED AT 3FFCH
                    8  ;        (ERROR@DISPLAY) IN THE COMMON RAM
                    86 ;        FOR UPDATE TO THE FRONT PANEL OR BUS INTERFACE
                    87 ;CALLING SEQUENCE:
                    88 ;        D PAIR,
                    89 ;          CONTAINS MANTISSA OF FLOATING PT. NUMBER
                    9  ;        B REG,
                    91 ;          CONTAINS EXPONENT OF FLOATING PT. NUMBER
                    9  ;        A REG,
                    9  ;           CONTAINS VALUE TO SUBTRACT FROM DECIMAL
                    94 ;           EXPONENT TO PUT IN DISPLAY
                    95 ;          ( 0 OR 6, 6 IS IMPLIED DIVISION BY 1 MILLION)
                    96 ;        IN H PAIR,
                    97 ;          POINTER TO BASE OF THE TABLE OF POWERS OF TEN
                    98 ;          COMPILED IN THE FORTRAN FILE MUST BE PASSED
                    99 ;DISCUSSION:
                   100 ;        THE VALUE TO BE CONVERTED IS CHECKED TO SEE IF
                   101 ;          IT IS WITH IN DISPLAYABLE RANGE -- IF NOT A
                   102 ;          SPECIAL DISPLAY CONFIGURATION IS MADE.
                   103 ;        ESTIMATE OF SIZE OF THE NUMBER CAN BE OBTAINED
                   10  ;          BY USING EXPONENT.  THE CLOSEST POWER OF TEN
                   105 ;          EQUAL OR LESS THAN THE NUMBER IS FOUND BY
                   106 ;          COMPARISION(S) TO THE POWER OF TEN TABLE.
                   107 ;          THE NUMBER IS ADJUSTED BY DIVISION BY POWER
                   108 ;          OLWDaDATAB`EXTRA<?PPTR1:?PPTR2D?PRTFLG??PSTAT1
                       >?PSTAT2bSTACTL;zhCHKCRrCHKON@CHKSTAQCONDELAY
                   109 ;          1.00 - 9.99 WHEN IN THE DISPLAY).  SCALED
                   110 ;          NUMBER CAN BE FIXED & BINBCD. AN EXPONENT
                   111 ;          ( 10 ** N) IS IN THE POWER T1nINITPR3NEWMES[NX
                       TCHPROC1&PROC2?PRPROCPRTCHPSERVO
STAT2RvqIN A REG.  IF BC
                       D NUMBER GREATER THAN 10**9
                   112 ;          & LESS THAN 10**12, DECIMAL POINT IN DISPLAY
                   113 ;          SHIFTS, IE. 100. X 10**9  IS  1.02 X 10**11
                   114
                   115 $EJECT
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

                   116 ;*****************************************************
                   117
                   118 FPBCD:
0027 327843        119         STA     ADJUST  ;SAVE ANY EXP ADJUST.
                   120                         ;6 IF DIVIDE BY 10**6
                   121                         ;IS HIDDEN
                   122                         ;IE. TOTAL BER/TOTAL BITS
                   123                         ;EITHER 2 OR 6
002A 227943        124         SHLD    P10PTR  ;SAVE PWR10 TABLE PTR
                   125
002D 3E02          126         MVI     A,2     ;DEFAULT DEC.PT. FOR DISPLAY
                   127                         ;TO ADJACENT FIRST DIGIT
002F 327743        128         STA     DP
0032 78            129         MOV     A,B
0033 327B43        130         STA     FPEXP   ;SAVE EXP OF NUMBER
                   131
                   132 ;I S  N U M B E R  E Q U A L  T O  Z E R O  ??
0036 FE40          133         CPI     40H     ;CHECK IF # = FP0
0038 C24600    C   134         JNZ     LIMITS  ;FP0 = 40H,0,0
003B 7A            135         MOV     A,D
003C B3            136         ORA     E
003D C24600    C   137         JNZ     LIMITS
                   138
0040 210000    C   139         LXI     H,ZERO  ;YES, IT DOES = FP0,
                   140                         ;NO NEED TO SEARCH TABLE
0043 C36501    C   141         JMP     MOVE    ;PUT ZERO INTO
                   142                         ; ERROR@DISPLAY
                   143
                   144
                   145
                   146 ;****
                   147 LIMITS:
                   148 ;I S  N U M B E R  I N  R A N G E  T O  D I S P L A Y  ?
                       ?
                   14                          ;EXPONENT NO# FROM 24H-67H
                   150
                   151                         ;PTR TO BASE OF TABLE IS IN H PA
                       IR
0046 78            152         MOV     A,B     ;SEE IF EXPONENT
                   153                         ;# IS OUT OF  RANGE
0047 FE68          154         CPI     68H
0049 D25C01    C   155         JNC     TOOBIG  ;# IS BIGGER THAN 999. X 10**9
                   156
004C FE24          157         CPI     24H
004E DA6201    C   158         JC      TOOSML  ;# IS LESS THAN 1.00 X 10**-9
                   159
                   160
                   161 $EJECT
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

                   162 ;****
                   163 INRANG:
                   164 ;SEE IF NUMBER SHOULD BE TREATED AS AN INTEGER OR W/ AN
                       E FORMAT
0051 3A7843        165          LDA     ADJUST  ;CHECK THE  ADJUST
0054 B7            166          ORA     A       ;IF NOT ZERO,
0055 C27C00    C   167          JNZ     EFORMT  ;MUST USE E-FORMAT FOR DISPLAY
                   168
                   169 ;SEE IF NUMBER IS IN THE RANGE OF  1-999
                   170 ;        TREAT  SUCH NUMBERS WITH INTEGER FORMAT
                   171 ;        FOR THE DISPLAY INSETAD OF A E-FORMAT
                   17  ;        WHICH IS USUALLY DONE
                   173
0058 0E41          174          MVI C,41H       ;COMPARE NUMBER TO 1
005A 210080        175          LXI H,8000H     ;LOAD REGISTERS FOR CALL
005D CD0000    E   176          CALL    FPCMPR  ;IS THE NUMBER GT 1??
0060 DA7C00    C   177          JC      EFORMT  ;JUMP IF NUMBER <1
0063 0E4A          178          MVI     C,4AH   ;COMPARE NUMBER TO 1000
0065 2102FA        179          LXI     H,0FA00H        ;LOAD REGISTERS
0068 CD0000    E   180          CALL FPCMPR
006B D27C00    C   181          JNC     EFORMT  ;JUMP IF NUMBER > 999
                   182
                   183 ;NUMBER IS BETWEEN  1 AND 999
                   184 ;        THEREFORE NO SCALING NEEDED, USE AS IS
                   185 ;        AS AN INTEGER  WITH NO DISPLAY EXPONENT
006E EB            186          XCHG
006F 48            187          MOV C,B         ;GET NUMBER IN H PR AND C REG
                   188
0070 CD0000    E   189          CALL FIX        ;FIX THE NUMBER
0073 CD0C00    C   190          CALL BINBCD     ;CHANGE TO BCD
0076 3E0F          191          MVI     A,0FH   ;EXPONENT SHOULD BE A
                   192                          ;  BLANK}0FH CODE
0078 CD4401    C   193          CALL    PUTDIS  ;GO PUT DISPLAY
                   194
007B C9            195          RET             ;THAT'S ALL  !!!
                   196
                   197 $EJECT
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

                   198 ;****
                   199 EFORMT:
                   200 ;                   E - F O R M A T
                   201 ;E T I M A T E  A  P T R  I N T O  P W R 10  T A B L E
                   202
007C 2A7943        203           LHLD    P10PTR  ;PTR TO PWR10 TABLE IN H PR
                   20
007F 3A7B43        205           LDA     FPEXP   ;RETRIEVE THE FLTPT EXP
0082 FE48          206           CPI     48H     ;USE EXPONENT TO APPROXIMATE AN
                       ENTRY
                   207                           ;INTO THE POWERS OF TEN TABLE
                   208                           ;WANT TO MINIMIZE
                   209                           ; # OF COMPARISIONS
0084 D28C00    C   210           JNC     LARGE   ;TO VALUES IN THE TABLE
                   211
                   212 SMALL:
0087 D61C          213           SUI     1CH     ;SCALE THE EXPONENT
                   214                           ;NEED TO SCALE BIG
                   215                           ;EXPONENT LESS THAN SMALL
0089 C38E00    C   216           JMP     GETPTR  ;TO GET THE OFFSET
                   217 LARGE:
008C D614          218           SUI     14H
                   219
                   220
                   221
                   222 ;****
                   223 GETPTR:
008E E6FC          224           ANI     0FCH    ;ENTRIES ARE 4 BYTES—
                   225                           ; GET A OFFSET TO IT
                   226
0090 D5            227           PUSH    D       ;SAVE MANTISSA OF RESULT
                   228
0091 5F            229           MOV     E,A     ;USE SCALED EXP. AS OFFSET
0092 1600          230           MVI     D,0
0094 19            231           DAD     D       ;ADD OFFSET TO BASE
                   232                           ;   OF TABLE TO GET PTR
                   233
                   234
                   235
                   236
                   237 $EJECT
```

```
LOC  OBJ          LIN         SOURCE STATEMENT

                  238 ;****
                  239 LOOP:
                  240 ;C O M P A R E  N U M B E R  T O  P O W E R  OF  T E N
                  241
0095 D1           242         POP     D       ;RESTORE MANTISSA OF RESULT
0096 D5           243         PUSH    D
0097 E5           244         PUSH    H       ;SAVE POINTER TO PWR10 TABLE
                  245
0098 4E           246         MOV     C,M     ;PUT VALUE2(PWR10)
                  247                         ;  IN H PAIR AND C REG
0099 23           248         INX     H
009A 7E           249         MOV     A,M
009B 23           250         INX     H
009C 66           251         MOV     H,M
009D 6F           252         MOV     L,A
                  253
009E CD0000   E   254         CALL    FPCMPR  ;COMPARE A POWER OF
                  255                         ;   TEN AND THE RESULT
                  256
00A1 E1           257         POP     H
                  258
00A2 110400       259         LXI     D,4     ;INCREMENT FOR
                  260                         ;NEXT TABLE ENTRY
                  261
00A5 CAE400   C   262         JZ      MATCH   ;GOT A POWER OF TEN
                  263                         ;  EXACTLY
                  264
00A8 DAAF00   C   265         JC      FOUND   ;RESULT IS LESS THAN
                  266                         ; TABLE ENTRY--GOOD
                  267
00AB 19           268         DAD     D       ;ELSE TRY NEXT LARGER ENTRY
00AC C39500   C   269         JMP     LOOP
                  270
                  271
                  272
                  273
                  274
                  275
                  276 $EJECT
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

                   277 ;****
                   278 FOUND:
                   27  ;S A V E  E X P O N E N T,  S C A L E  N U M B E R
                   280
00AF 2B            281          DCX     H          ;GET NEXT SMALLER EXPONENT
00B0 227543        282          SHLD    EXPNPT     ;  WHICH IS THE RIGHT ONE
                   283
                   28  ;**** SCALE RESULT BY DIVISION BY POWER OF 10
00B3 11F5FF        285          LXI     D, -11
00B6 19            286          DAD     D          ;USE THE POWER OF TEN
                   287                             ;  WHICH IS TWO
                   288                             ;  POWERS LOWER THAN EXPONENT
00B7 D1            289          POP     D          ;RESTORE MANTISSA OF RESULT
00B8 4E            290          MOV     C,M        ;PUT VALUE2(PWR10)
                   291                             ;  IN H PAIR AND C REG
00B9 23            292          INX     H          ;VAL2=DIVISOR,
                   293                             ;I.E 10**-6 / 10**-8
                   294
00BA 7E            295          MOV     A,M
00BB 23            296          INX     H
00BC 66            297          MOV     H,M
00BD 6F            298          MOV     L,A
                   299
00BE CD0000    E   300          CALL    FDIV
                   301
                   302 ;ROUND THE FLOATING POINT NUMBER BY ADDING .5
00C1 110080        303          LXI     D, 8000H           ;ADD 1/2 TO NUM
                   304                                     ;(ROUND IT, NOT TRUNCATE
                       )
                   305
00C4 0640          306          MVI     B, 40H             ;1/2 = 40H,0,80H
00C6 CD0000    E   307          CALL    FADD
00C9 1100FA        308          LXI     D, 0FA00H          ;COMPARE NUM TO 1000
00CC 064A          309          MVI     B,4AH              ;1000 = 4AH,0,FAH
00CE CD0000    E   310          CALL    FPCMPR
00D1 CAE900    C   311          JZ      GE1000             ;BRANCH ON FLAGS
00D4 DAE900    C   312          JC      GE1000             ;GO TREAT NUM >= 1000
                   313
                   314
00D7 CD0000    E   315          CALL    FIX                ;CHANGE FLOATING PT
                   316                                     ; NUM TO FIXED PT
                   317
                   318
00DA CD0C00    C   319          CALL    BINBCD             ;CHANGE FIXED PT TO BCD
                   320                                     ;IN H&L
                   321
                   322
00DD E5            323          PUSH    H          ;SAVE BCD DIGITS
                   324
                   325
00DE 2A7543        326          LHLD    EXPNPT     ;INSERT EXPONENT
00E1 C3F400    C   327          JMP EXP
                   328
                   329
                   330 $EJECT
```

LOC OBJ LIN SOURCE STATEMENT

```
LOC  OBJ          LIN         SOURCE STATEMENT

                   33 ;N U M B E R  >=   1 . 0 0  X  1 0 ** N
                  332 ;****
                  333 MATCH:
00E4 2B           334          DCX     H
00E5 227543       335          SHLD EXPNPT
00E8 E1           336          POP     H        ;RESET STACK
                  337
                  338
                  339 ;ENTER HERE IF NUMBER > 1000 AFTER ROUNDING
                  340 ;****
                  341 GE1000:
                  342                   ;NEED NEXT BIGGER EXP
                  343                   ;THE BCD DIGITS HAVE TO
                  344                   ;BE 1.00, SO DUMMY THEM
                  345                   ;AND SKIP FIX
00E9 210001       346          LXI     H,100H
00EC E5           347          PUSH    H        ;SAVE ON STACK
                  348                   ;INCREASE EXP PTR BY 4
00ED 2A7543       349          LHLD    EXPNPT
00F0 110400       350          LXI     D,4
00F3 19           351          DAD     D
                  352
                  353 $EJECT
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

                   354 ;****
                   355 EXP:
                   356 ;A D J U S T  E X P O N E N T
                   357 ; FROM HERE ON JUST ADJUSTING EXPONENT, AND PUTTING
                   358 ;       DIGITS IN TO ERROR@DISPLAY
                   359
                   360
00F4 7E            361         MOV     A,M
00F5 217843        362         LXI     H, ADJUST
00F8 96            363         SUB     M
00F9 F20C01    C   364         JP      CHKDP
                   365
                   366
                   367
                   368
                   369 ;HANDLE NEGATIVE EXPONENTS
                   370 ;****
                   371 NEGEXP:
00FC 2F            372         CMA
00FD 3C            373         INR     A       ;- ( - EXPONENT)
00FE FE0A          374         CPI     10      ;-9 IS SMALLEST EXP
                   375
0100 DA0701    C   376         JC      NEG1
0103 E1            377         POP     H       ;RESET STACK
0104 C36201    C   378         JMP     TOOSML
                   379
                   38  NEG1:
0107 F680          381         ORI     80H     ;TURN ON MINUS SIGN
0109 C32F01    C   382         JMP     RDYPUT
                   383
                   384
                   385
                   386 ;HANDLE POSITIVE EXPONENTS
                   387 ;****
                   388 CHKDP:
                   389                 ;BE SURE DEC. PT. IS IN RIGHT PLACE
                   390                 ;(DEC. PT. SLIDES FOR NUM * 10**-10 (OR
                       -11))
                   391
010C FE0A          392         CPI     0AH     ;CHECK POS EXP
010E DA2001    C   393         JC      PERC    ;DONE IF EXP<10
                   394
0111 D609          395         SUI     9H      ;ELSE MOVE DEC PT
0113 47            396         MOV     B,A     ;SAVE AMOUNT TO MOVE D P
0114 3A7743        397         LDA     DP      ;CHANGE D P POSITION
0117 90            398         SUB     B       ;POS.= MID(1) OR LSD(0)
0118 327743        399         STA     DP
011B 3E09          400         MVI     A, 9    ;EXPONENT = 9
011D C32F01    C   401         JMP     RDYPUT  ;JUMP TO PUT INTO DISPLAY
0120 FE03          402 PERC:   CPI     3       ;SEE IF LESS THAN 3
0122 D22F01    C   403         JNC     RDYPUT  ;IF NOT JUMP
0125 47            404         MOV     B,A     ;STORE IN B
0126 3A7743        405         LDA     DP      ;FETCH DECIMAL POINT
0129 90            406         SUB     B       ;SUBTRACT EXP.
012A 327743        407         STA     DP
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

012D 3E1F          408          MVI     A,1FH    ;STORE EXPONENT - THE 1 IS TO
                   409                           ;DIFFERENTIATE FROM INTEGERS
                   410
                   411
                   412 $EJECT
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

                   413 RDYPUT:
                   414 ; G E T  T H E  D I S P L A Y  R E A D Y
012F E1            415          POP     H          ;BCD DIGITS INTO H PR.
0130 CD4401    C   41           CALL PUTDIS        ;PUT DIGITS INTO DISPLAY
                   417 ;**** ADD A DECIMAL POINT TO ONE OF THE DIGITS
0133 217743        418          LXI     H, DP      ;TURN ON THE DP BIT
0136 6E            419          MOV     L,M
0137 2600          420          MVI     H,0
0139 11A443        421          LXI     D, BCDNUM + 1
013C 19            422          DAD     D          ;GET PTR TO RIGHT DIGIT
013D 7E            423          MOV     A,M
013E F680          424          ORI     80H
0140 F3            425          DI                 ;DISABLE TO PUT EXPONENT
0141 77            426          MOV     M,A
0142 FB            427          EI                 ;REENABLE
0143 C9            428          RET
                   429 $EJECT
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

                   430 ;*****************************************************
                   43
                   43  PUTDIS:
                   433 ;C H A N G E   E R R O R @ D I S P L A Y
                   43  ; UNPACK THE DIGITS(BCD) PASSED IN THE  H REG. PAIR INTO
                   435 ;       ERROR@DISPLAY(1)-(3)
                   436 ;       THE EXPONENT PASSED IN THE A REG. IS PLACED INTO
                   437 ;       ERROR@DISPLAY(0)
                   438
                   439 ;*****************************************************
                   440                         ;BCD DIGITS IN H&L
0144 11A343        441         LXI     D,BCDNUM
                   442                         ;CHANGE ALL 4 BYTES
                   443                         ;OF ERROR DISPLAY #/0 INT
0147 12            44          STAX    D       ;MOVE NEW EXPONENT IN
                   445
0148 13            44          INX     D
                   447
                   448
                   449 ;**** UNPACK B C D  INTO ERROR@DISPLAY
0149 7D            450         MOV     A,L
014A E60F          451         ANI     0FF
014C 12            452         STAX    D       ;LSD DIGIT
                   453
014D 13            454         INX     D
014E 7D            455         MOV     A,L
014F 1F            456         RAR
0150 1F            457         RAR
0151 1F            458         RAR
0152 1F            459         RAR
0153 E60F          460         ANI     0FH
0155 12            46          STAX    D       ;MID DIGIT
                   462
0156 13            463         INX     D
0157 7C            464         MOV     A,H
0158 E60F          465         ANI     0FH
015A 12            466         STAX    D       ;MSG DIGIT
                   467
015B C9            468         RET
                   469
                   470 $EJECT
```

```
 LOC  OBJ          LINE        SOURCE STATEMENT

                    471 ;S P E C I A L   C A S E   N U M B E R S
                    472
                    473
                    474 ;T O O  B I G  !!!!
                    475 TOOBIG:
 015C 210000    C   476          LXI     H,OVRFLW
 015F C36501    C   477          JMP     MOVE
                    478
                    479 ;T O O   S M A L L  !!!!!
                    480 TOOSML:
 0162 210400    C   481          LXI     H,UNDFLW
                    482
                    483
                    484
                    485 MOVE:
 0165 11A343        486          LXI     D,BCDNUM
 0168 7E            487          MOV     A,M
 0169 12            488          STAX    D
                    489
 016A 23            490          INX     H
 016B 13            491          INX     D
 016C 7E            492          MOV     A,M
 016D 12            493          STAX    D
                    494
 016E 23            495          INX     H
 016F 13            496          INX     D
 0170 7E            497          MOV     A,M
 0171 12            498          STAX    D
                    499
 0172 23            500          INX     H
 0173 13            501          INX     D
 0174 7E            502          MOV     A,M
 0175 12            503          STAX    D
                    504
 0176 C9            505          RET             ;RETURN TO MAINLINE
                    506
                    507
                    508
                    509
                    510          END
```

PUBLIC SYMBOLS

EXTERNAL SYMBOLS

```
FADD    E 0000    FDIV    E 0000    FIX     E 0000    FPCMPR E 0000
```

USER SYMBOLS

```
ADJUST A 4376    BCDNUM A 43A3    BCDNXT C 0012    BINBCD C 000C
CHKDP  C 010C    DP     A 4377    EPOPMT C 007C    EXP    C 00F4
EXPNPT A 4375    FADD   E 0000    FDIV   E 0000    FIX    E 0000
FOUND  C 00AF    FPBCD  C 0027    FPCMPR E 0000    FPEXP  A 437E
GE1000 C 00E9    GETPTR C 008E    INRANG C 0051    LARGE  C 008C
LIMITS C 004C    LOOP   C 0095    MATCH  C 00E4    MOVE   C 0165
```

```
NEG1    C 0107    NEGEXP C 00FC    OVRFLW C 0000    P10PTR A 4379
PERC    C 0120    PUTDIS C 0144    RDYPUT C 012F    SMALL  C 0087
TOOBIG  C 015C    TOOSML C 0162    UNDFLW C 0004    ZERO   C 0008

ASSEMBLY COMPLETE,  NO ERRORS
```

```
ASM80.V4 :F1:CALFP1.ASM PRINT(:LP:) PAGEWIDTH(80) DEBUG


ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0         CALLFP   PAGE    1

  LOC  OBJ          LINE        SOURCE STATEMENT

                       1          NAME    CALLFP
                                  EXTRN   FADD,FDIV,FLOAT

                       4 ; ***************************************************
                       5 ;       ROUTINE TO CALL FLOATING POINT PACKAGE
                         ;       FROM REALISTIC CONTROL FORTRAN:  FORT-80
                       7 ;       SCIENTIFIC ATLANTA TEST RECEIVER
                       8 ;       10-15-79
                       9 ; ***************************************************
                        *
                      10
                      11
                      12 ;*** N O T E :  SECTIONS ARE COMMENTED OUT THAT WERE NOT
                      13 ;       NEEDED, I.E. MULTIPLY.  THESE SECTIONS HAVE BEEN
                      14 ;       DEBUGGED AND CAN BE USED BY REMOVING ';'.
                      15 ;       ONLY DIVIDE, ADD, AND FLOAT ARE IMPLEMENTED.
                      16
                      17
                      18
                      19 ;PARAMETERS PASSED BY POINTER IN RAM IN BLOCK BEGINNING
                      20 ;       AT 3FC1H EXCEPT FOR OP WHICH IS ACTUALLY IN THE
                      21 ;       BLOCK
                      22 ;ALSO SEE SUBROUTINE FLTPTOPACK IN FORTRAN SOURCE FOR
                      23 ;       THE FORTRAN CALL SEQUENCE
                      24
                      25
  4367                26          PTRV1 EQU 4367H
  4363                27          OP    EQU PTRV1 - 4
  435F                2           PTRV2 EQU OP - 4
  435B                29          PTRSLT EQU PTRV2  -  4
                      30
                      31
  00FF                32          TRUE EQU 0FFH
  0000                33          FALSE EQU 0
                      34
                      35 ; COMPARE OPERATORS:
  0046                36          FLT     EQU 'F'
  0058                3           FIXIT EQU 'X'
  004C                38          LESS    EQU 'L'
                      39 ;        GREAT   EQU 'G'
  0045                4           GRTEQU  EQU 'E'
                      41
                      42
                      43 ; JUMP TABLE ENTRY FOR CALLFP
                      44          ASEG
                      45
  0003                46          ORG 0003
                      47
  0003 C33000   C     48          JMP FPPACK
                       4
                       5
                      51
                       5 ; * * * *    FLOATING POINT LINK   * * * * *
                       5
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

                    54          CSEG            ; RELOCATABLE CODE SEGMENT
                    5
                    56 ;* * * *      LOAD REGISTERS FOR CALLS   * * * * *
                    57 ;* * * *        TO THE F.P.PACKAGE       * * * * *
                    58
0000 2A6743         59 FPPACK: LHLD PTRV1       ;VALUE 1 TO B, D PAIR
0003 23             6          INX H            ;INR FORT PTR TO FIRST ELEMENT
0004 46             6          MOV B,M
0005 23             6          INX H
0006 5F             6          MOV E,M
0007 23             6          INX H
0008 56             65         MOV D,M
                    66
0009 2A5F43         67         LHLD PTRV2       ;VALUE 2 TO C, H PAIR
000C 23             68         INX H            ;INR FORT PTR TO ELEMENT 1
000D 4E             69         MOV C,M
000E 23             70         INX H
000F 7E             7          MOV A,M
0010 23             72         INX H
0011 66             7          MOV H,M
0012 6F             7          MOV L,A
                    75
                    76
                    77 $EJECT
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

                    78 ;* * * *   BRANCH ACCORDING TO THE OP   * * * * *
                    79
0013 3A6343         80          LDA OP              ;* * * *   ENTRY,
                    81                              ; BRANCH TO FP ROUTINES
0016 FE2B           82          CPI '+'             ;* * * *   ADD  V1 + V2
0018 C22100    C    83          JNZ NXT1
001B CD0000    E    84          CALL FADD
001E C33700    C    85          JMP EXIT1
                    86
0021 FE2F           87 NXT1:    CPI '/'             ;* * * *  DIVIDE        V1 / V2
0023 C22C00    C    8           JNZ     NXT2
0026 CD0000    E    89          CALL FDIV
0029 C33700    C    9           JMP EXIT1
                    9
                    9  ;NXT2:   CPI '*'             ;* * * *  MULTIPLY      V1 * V2
                    9  ;        JNZ NXT3
                    9  ;        CALL FMULT
                    95 ;        JMP EXIT1
                    96
002C FE46           97 NXT2:    CPI FLT             ;* * * *  FLOAT         V2
002E C24200    C    98          JNZ NXT5
0031 CD0000    E    99          CALL FLOAT
0034 C33700    C   100          JMP EXIT1
                   101
                   102 ;NXT4:   CPI FIXIT                 ;* * * *  FIX   V2
                   103 ;        JNZ NXT5
                   104 ;        CALL FIX
                   10
                   106
                   107 ;* * * *  ARITMETIC EXIT  * * * * *
                   108
0037 EB            109 EXIT1:   XCHG                ;C, H PAIR TO RESULT
0038 2A5B43        110          LHLD PTRSLT
003B 23            111          INX H               ;INR FORT PTR TO ELEMENT ONE
                   112
003C 71            11           MOV M,C
003D 23            11           INX H
003E 73            115          MOV M,E
003F 23            116          INX H
0040 72            11           MOV M,D
0041 C9            118          RET                 ; RET TO FORTRAN
                   119
                   120
                   121 $EJECT
```

```
 LOC  OBJ          LINE        SOURCE STATEMENT

                    122 ; * * * *    FLOATING POINT COMPARES     * * * * *
                    123
 0042 C9            124 NXT5:   RET             ;COMPARE IS NOW COMMENTED OUT
                    12  ;NXT5:  CALL FPCMPR
                    126                         ;FLAGS RETURN FROM FPCMPR W/ RES
                        ULT:
                    127                         ;       CY=0    VAL1 >= VAL2
                    128                         ;       CY=1    VAL1  < VAL2
                    129                         ;       Z=1     VAL1  = VAL2
                    130 ;       LHLD PTRSLT
                    131 ;       INX H           ;GET POINTER TO RESULT ARRAY
                    132 ;
                    133 ;
                    134 ;;* * * DECIPHER FLAGS ON RETURN ACCORDING TO OP * * * *
                    135 ;
                    136 ;EXIT2: PUSH PSW
                    137 ;       LDA OP
                    138 ;;      CPI GREAT
                    139 ;;      JNZ FPLT
                    140 ;;      POP PSW         ;* * * *  V1  GREATER THAN  V2
                    141 ;;      JZ XFALSE       ; IF =, RET FALSE
                    142 ;       JNC XTRUE
                    143 ;       JMP XFALSE
                    144
                    145 ;FPLT:  CPI LESS
                    146 ;       JNZ FPGE
                    147 ;       POP PSW         ;* * * *  V1 LESS THAN  V2
                    148 ;       JC XTRUE
                    149 ;       JMP XFALSE
                    150 ;
                    151 ;FPGE:  CPI GRTEQU      ;         V1 >=  V2
                    152 ;       JNZ XFALSE
                    153 ;       POP PSW
                    154 ;       JNC XTRUE
                    155 ;
                    156 ;XFALSE:       MVI M, FALSE
                    157 ;       RET
                    158 ;
                    159 ;
                    160 ;XTRUE: MVI M, TRUE
                    161 ;       RET
                    16
                    163
                    164         END

PUBLIC SYMBOLS


EXTERNAL SYMBOLS
FADD   E 0000    FDIV   E 0000    FLOAT  E 0000

USER SYMBOLS
EXIT1  C 0037    FADD   E 0000    FALSE  A 0000    FDIV   E 0000
FIXIT  A 0058    FLOAT  E 0000    FLT    A 0046    FPPACK C 0000
```

```
GRTEQU A 0045    LESS   A 004C    NXT1   C 0021    NXT2   C 002C
NXT5   C 0042    OP     A 4363    PTRSLT A 435B    PTRV1  A 4367
PTRV2  A 435F    TRUE   A 00FF

ASSEMBLY COMPLETE,  NO ERRORS
```

```
ISIS-II OBJECT LINKER V3.0 INVOKED BY:
-LINK :F1:RTOD1.OBJ,:F1:RMOD1.OBJ,:F1:STOD1.OBJ,:F1:SMOD1.OBJ, &
**:F1:DISPL1.OBJ,:F1:SWTCH1.OBJ,:F1:EREAD1.OBJ,:F1:GETBYT.OBJ, &
**:F1:LEDTES.OBJ,:F1:GINIT1.OBJ,:F1:GINTR1.OBJ,:F1:RINIT1.OBJ, &
**:F1:RINTR1.OBJ,:F1:GSER1.OBJ,:F1:PSERV1.OBJ,:F1:JTAB1.OBJ, &
**:F1:CALFP1.OBJ,:F1:FPBCD1.OBJ,:F1:FPASY.LIB TO :F1:SADRIV.LNK  &
**MAP PRINT(:LP:)

LINK MAP OF MODULE SADRIV
WRITTEN TO FILE :F1:SADRIV.LNK
MODULE IS NOT A MAIN MODULE

SEGMENT INFORMATION:
START   STOP LENGTH REL NAME

                AREA  B  CODE
0000H  0002H     3H   A  ABSOLUTE
0003H  0005H     3H   A  ABSOLUTE
000FH  0023H    15H   A  ABSOLUTE
0027H  0029H     3H   A  ABSOLUTE
0030H  0032H     3H   A  ABSOLUTE
0033H  0035H     3H   A  ABSOLUTE
0038H  003AH     3H   A  ABSOLUTE

INPUT MODULES INCLUDED
 :F1:RTOD1.OBJ(RTOD)
 :F1:RMOD1.OBJ(RMOD)
 :F1:STOD1.OBJ(STOD)
 :F1:SMOD1.OBJ(SMOD)
 :F1:DISPL1.OBJ(DISPLA)
 :F1:SWTCH1.OBJ(SWITCH)
 :F1:EREAD1.OBJ(ERREAD)
 :F1:GETBYT.OBJ(GETBYT)
 :F1:LEDTES.OBJ(LEDTES)
 :F1:GINIT1.OBJ(GINIT)
 :F1:GINTR1.OBJ(GINTR)
 :F1:RINIT1.OBJ(RINIT)
 :F1:RINTR1.OBJ(RINTR)
 :F1:GSER1.OBJ(GSER)
 :F1:PSERV1.OBJ(PSERV)
 :F1:JTAB1.OBJ(JTAB)
 :F1:CALFP1.OBJ(CALLFP)
 :F1:FPBCD1.OBJ(FPBCD)
 :F1:FPASY.LIB(FADD)
 :F1:FPASY.LIB(FLOAT)
 :F1:FPASY.LIB(FIX)
 :F1:FPASY.LIB(FPCMPR)
 :F1:FPASY.LIB(FDIV)
 :F1:FPASY.LIB(FPZERO)
 :F1:FPASY.LIB(NORM)
 :F1:FPASY.LIB(IDIV)
 :F1:FPASY.LIB(ZEROD)
 :F1:FPASY.LIB(NEGD)
 :F1:FPASY.LIB(ABSD)
 :F1:FPASY.LIB(RBUMP)
 :F1:FPASY.LIB(BUMP2)
 :F1:FPASY.LIB(BUMP3)
```

```
ISIS-II OBJECT LOCATER V3.0 INVOKED BY:
-LOCATE :F1:SADRIV.LNK CODE(7231H) MAP SYMBOLS COLUMNS(2)  &
**STACKSIZE(0) PRINT(:LP:)

SYMBOL TABLE OF MODULE SADRIV
READ FROM FILE :F1:SADRIV.LNK
WRITTEN TO FILE :F1:SA RIV

VALUE TYPE SYMBOL                          VALUE TYPE SYMBOL

      MOD  RTOD
0047H SYM  CBASE                           43D5H SYM  CLOCK
0052H SYM  CRESET                          43CDH SYM  DATE
0046H SYM  DATEC                           004EH SYM  DATEL
0045H SYM  DOWC                            004DH SYM  DOWL
43E2H SYM  DSFLG                           0000H SYM  FAL
43E3H SYM  FLGAD                           0055H SYM  GOCOM
0044H SYM  HRC                             004CH SYM  HRL
0041H SYM  HUTENC                          0049H SYM  HUTENL
0051H SYM  INTCON                          0050H SYM  INTST
0053H SYM  LRESET                          0043H SYM  MINC
004BH SYM  MINL                            0047H SYM  MONC
004FH SYM  MONL                            43CFH SYM  MONTH
0056H SYM  SDBYIN                          0042H SYM  SECC
004AH SYM  SECL                            0054H SYM  STATBT
005FH SYM  TESTMO                          0040H RXL  SGNT
0048H SYM  THOUL                           43FCH SYM  TODRS
00FFH SYM  TRU                             7237H SYM  RT01
7248H SYM  RT02                            7258H SYM  RT03
726CH SYM  RT04                            7231H SYM  RTOD
      MOD  RMOD
0040H SYM  CBASE                           43D5H SYM  CLOCK
0052H SYM  CRESET                          43CDH SYM  DATE
0046H SYM  DATEC                           004EH SYM  DATEL
0045H SYM  DOWC                            004DH SYM  DOWL
43E2H SYM  DSFLG                           0000H SYM  FAL
43E3H SYM  FLGAD                           0055H SYM  GOCOM
0044H SYM  HRC                             004CH SYM  HRL
0041H SYM  HUTENC                          0049H SYM  HUTENL
0051H SYM  INTCON                          0050H SYM  INTST
0053H SYM  LRESET                          0043H SYM  MINC
004BH SYM  MINL                            0047H SYM  MONC
004FH SYM  MONL                            43CFH SYM  MONTH
0056H SYM  SDBYIN                          0042H SYM  SECC
004AH SYM  SECL                            0054H SYM  STATBT
005FH SYM  TESTMO                          0040H SYM  TFOUC
0048H SYM  THOUL                           43FCH SYM  TODRS
00FFH SYM  TRU                             7283H SYM  RM01
727DH SYM  RMOD
      MOD  STOD
0040H SYM  CBASE                           43D5H SYM  CLOCK
0052H SYM  CRESET                          43CDH SYM  DATE
0046H SYM  DATEC                           004EH SYM  DATEL
0045H SYM  DOWC                            004DH SYM  DOWL
43E2H SYM  DSFLG                           43E3H SYM  FLGAD
0055H SYM  GOCOM                           0044H SYM  HRC
004CH SYM  HRL                             0041H SYM  HUTENC
0049H SYM  HUTENL                          0051H SYM  INTCON
0050H SYM  INTST                           0053H SYM  LRESET
0043H SYM  MINC                            004BH SYM  MINL
0047H SYM  MONC                            004FH SYM  MONL
43CFH SYM  MONTH                           0056H SYM  SDBYIN
0042H SYM  SECC                            004AH SYM  SECL
0054H SYM  STATBT                          005FH SYM  TESTMO
```

```
0040H SYM  THOUC
43FCH SYM  TODRS
       MOD  SMOD
0040H SYM  CBASE
0052H SYM  CRESET
0046H SYM  DATEC
0045H SYM  DOWC
43E2H SYM  DSFLG
0055H SYM  GOCOM
004CH SYM  HRL
0049H SYM  HUTENL
0050H SYM  INTST
0043H SYM  MINC
0047H SYM  MONC
43CFH SYM  MONTH
0042H SYM  SECC
0054H SYM  STATBT
0040H SYM  THOUC
43FCH SYM  TODRS
72B7H SYM  SMOD
       MOD  DISPLA
000FH SYM  BLANK
0071H SYM  CNTRL
0070H SYM  DSPLY
43FBH SYM  ERREG
0000H SYM  FAL
0004H SYM  FOUR
43EFH SYM  HORNEN
0009H SYM  NINE
000AH SYM  OVAL
4383H SYM  RMVAL
43E0H SYM  RUNSTP
0007H SYM  SEVEN
0003H SYM  THREE
43FFH SYM  TODRH
0002H SYM  TWO
000DH SYM  UVAL
72EFH SYM  DISPLA
7308H SYM  FIR
736FH SYM  LED
73B7H SYM  NOCH
7332H SYM  OUTPUT
       MOD  SWITCH
8000H SYM  BASE
0006H SYM  DSX3
0007H SYM  HORNE
0005H SYM  LOHI
43DEH SYM  REMFL
43AAH SYM  RSLL
0003H SYM  SING
0070H SYM  SWVAL
742DH SYM  DECOD
7421H SYM  SKP5
7451H SYM  TEXT1
       MOD  ERREAD
8000H SYM  BASE
43E4H SYM  ERBUF
43ECH SYM  FLADD
43DEH SYM  GPTHER
0091H SYM  HIBVE
0090H SYM  LOBVE
43EAH SYM  PRBSL
0009H SYM  TEN6
43A8H SYM  UNFRAM
7491H SYM  F1
74A9H SYM  F2
0048H SYM  THOUL
72A6H SYM  STOD

43D5H SYM  CLOCK
43CDH SYM  DATE
004EH SYM  DATEL
004DH SYM  DOWL
43E3H SYM  FLGAD
0044H SYM  HRC
0041H SYM  HUTENC
0051H SYM  INTCON
0053H SYM  LRESET
004BH SYM  MINL
004FH SYM  MONL
0056H SYM  SDBYIN
004AH SYM  SECL
005FH SYM  TESTMO
0048H SYM  THOUL
72DCH SYM  DTAB

000CH SYM  BVAL
43E2H SYM  DSFLG
0008H SYM  EIGHT
000EH SYM  EVAL
0005H SYM  FIVE
43EEH SYM  HORN
0094H SYM  INOUT
0001H SYM  ONE
43F0H SYM  PRINTC
4382H SYM  RMVAL1
000BH SYM  RVAL
0006H SYM  SIX
43F1H SYM  TRESH
43F2H SYM  TRFLT
43A8H SYM  UNFRAM
0000H SYM  ZERO
7351H SYM  DSPTB
7361H SYM  HXDSP
72F7H SYM  LOOP
7315H SYM  OT4

0071H SYM  CNTRSW
0002H SYM  ELAPS
000CH SYM  INADD
0000H SYM  MODA
0004H SYM  REP
43ABH SYM  RSTHER
43EDH SYM  SWREG
0001H SYM  TOD
7449H SYM  FIN
740AH SYM  SWITCH

43E1H SYM  BLUFLG
0000H SYM  FAL
0093H SYM  FLGDTA
0021H SYM  HIBE
0020H SYM  LOBE
0092H SYM  PARER
43E0H SYM  RUNSTP
00FFH SYM  TRU
7459H SYM  ERREAD
74A3H SYM  F11
74B7H SYM  F25
```

```
74BDH SYM  F3                    74CDH SYM  F4
74F1H SYM  SFRA                  747CH SYM  STREAD
      MOD  GETBYT
0003H SYM  ETX                   0030H SYM  RBLK1
0220H SYM  SPC                   752CH SYM  BADINT
7531H SYM  BYT1                  757AH SYM  BYT2
755EH SYM  FINSET                750FH SYM  GETBYT
7553H SYM  GOTONX                7595H SYM  SPCSET
755CH SYM  TSET
      MOD  LEDTES
0071H SYM  CNTRL                 0070H SYM  DSPLY
436FH SYM  LEDCNT                75A2H SYM  LEDTES
75AFH SYM  LT1
      MOD  GINIT
8000H SYM  BASE                  43DEH SYM  GBTHER
43CCH SYM  REMFL                 75C0H SYM  GINIT
75F4H SYM  SKPU
      MOD  GINTR
43C0H SYM  ABLOCK                8000H SYM  BASE
43DDH SYM  BYTEIN                0003H SYM  ETX
4372H SYM  FIRSFL                002DH SYM  GPIBIN
020CF SYM  INADD                 0000H SYM  NULL
43CCH SYM  REMFL                 43EDH SYM  SWREG
7651H SYM  CMD                   7694H SYM  DEVC
763FH SYM  END1                  7605H SYM  G1
7644H SYM  GET                   7502H SYM  GINTR
7689H SYM  INB                   766AH SYM  OUTB
767BH SYM  OUTB1                 753DH SYM  OVT
76A7H SYM  REML                  7684H SYM  SKIP08
76B8H SYM  SKP6
      MOD  RINIT
0002H SYM  BCDOFF                0001H SYM  BCDON
00E0H SYM  BDMASK                0093H SYM  BDSWIT
0002H SYM  BRF                   0008H SYM  CHL7
0002H SYM  DTR                   0020H SYM  EP
0210H SYM  ER                    0040H SYM  IR
0006H SYM  MODE                  0010H SYM  PEN
0030H SYM  RL                    43ABH SYM  RSTBER
0220H SYM  RTS                   0004H SYM  RXE
0002H SYM  SC0                   0080H SYM  SC2
0081H SYM  SCNTRL                0080H SYM  SDATIO
0281H SYM  SSTAT                 0240H SYM  STPBIT
0088H SYM  TCLOCK                0084H SYM  TCONT0
0086H SYM  TCONT2                0087H SYM  TMODE
436EH SYM  TRANSM                00C0H SYM  TWSBIT
0201H SYM  TXEN                  75C2H SYM  DIVN
76F5H SYM  HER1                  772FH SYM  ONEB
76D2H SYM  RINIT                 75FAH SYM  TSET
7705H SYM  TSET1                 773EH SYM  TSET2
7719H SYM  USET
      MOD  RINTR
43C0H SYM  APLOCK                43DDH SYM  BYTEIN
000DH SYM  CR                    4371H SYM  DELAY
0003H SYM  ETX                   000CH SYM  INADD
4370H SYM  NCNTR                 002DH SYM  RSIN
0081H SYM  SCNTRL                0080H SYM  SDATIO
43EDH SYM  SWREG                 436EH SYM  TRANSM
0013H SYM  XOFF                  0011H SYM  XON
7796H SYM  NOCR                  77CBH SYM  NODEL
77B2H SYM  NOTX                  7781H SYM  OKTX
77B7H SYM  OUTPT                 77C7H SYM  OV2
7769H SYM  OVTHIS                773EH SYM  RINTR
7762H SYM  SBYTE                 7751H SYM  SKP90
7770H SYM  TXRD                  77A3H SYM  TXRD1
      MOD  GSER
43C0H SYM  APLOCK                8020H SYM  BASE
```

```
43DEH SYM  GBTHER
43ABH SYM  RSTHER
436EH SYM  TRANSM
77D0H SYM  GSER
780EH SYM  OV9
7827H SYM  OVG1
77EEH SYM  SKPA
7820H SYM  SKPR
      MOD  PSERV
00E3H SYM  CNTLWD
00E0H SYM  EXTRA
3EAEH SYM  PBUF1
439CH SYM  PINPTR
43A7H SYM  PRTFLG
7863H SYM  CHKON
787BH SYM  DELAY1
785FH SYM  INITPR
7837H SYM  PRTCH
7887H SYM  SENDCH
      MOD  JTAB
43C0H SYM  ABLOCK
43A9H SYM  ASYNC
436CH SYM  BCNT
002AH SYM  BURAD
005AH SYM  BURLI
43E2H SYM  DSFLG
0070H SYM  DSPLY
4372H SYM  FIRSFL
0027H SYM  GSERA
0091H SYM  HIBYE
0050H SYM  INTST
001BH SYM  LEDA
0000H SYM  MAINA
4398H SYM  PBLOCK
43A7H SYM  PTHER
4373H SYM  RAMTS2
43APH SYM  RSTHER
43E0H SYM  RUNSTP
0018H SYM  SMODA
0015H SYM  STODA
43A8H SYM  UNFRAM
7942H SYM  DOSTRT
78F5H SYM  DOPS
7892H SYM  INT
78F1H SYM  OVOPS
79A0H SYM  PRNGIN
78CFH SYM  SKP11
797BH SYM  T4
      MOD  CALLFP
0000H SYM  FALSE
0046H SYM  FLT
004CH SYM  LESS
435BH SYM  PTRSLT
435FH SYM  PTRV2
79F6H SYM  EXIT1
79E0H SYM  NXT1
7A21H SYM  NXT5
      MOD  FPBCD
4378H SYM  ADJUST
4377H SYM  DP
437BH SYM  FPEXP
7A14H SYM  BCDNXT
7B20H SYM  CHKDP
7AF6H SYM  EXP
7A29H SYM  FPBCD
7A92H SYM  GETPTR
43CBH SYM  PPMASK
0081H SYM  SCNTRL
7816H SYM  DOWN3
781AH SYM  MEAS
7823H SYM  OVG
77E2H SYM  SKOV
77F4H SYM  SKPE
7800H SYM  STSP

00E1H SYM  DATAB
4398H SYM  PBLOCK
3E92H SYM  PDAT1
4398H SYM  POUPTR
00E2H SYM  STACTL
784FH SYM  CON
7854H SYM  GETIT
7840H SYM  PRPROC
782FH SYM  PSERV

43ACH SYM  ARNGB
8000H SYM  BASE
436DH SYM  BLNG
0058H SYM  BURCI
0071H SYM  CNTRL
001EH SYM  DSPLAA
0024H SYM  FINITA
43DEH SYM  GETHER
0021H SYM  HIBE
0051H SYM  INTCON
436BH SYM  LBCNT
436FH SYM  LEDCNT
0006H SYM  OPSA
4384H SYM  PRNGB
4374H SYM  RAMTS1
0012H SYM  RMODA
000FH SYM  RTODA
0080H SYM  SBASE
0021H SYM  STATUS
00FFH SYM  TRU
7982H SYM  ARNGIN
790DH SYM  DINIT
78F8H SYM  FRTPNL
7954H SYM  M2
7921H SYM  PDCHK
7935H SYM  SETIT
78DDH SYM  SKP12

0058H SYM  FIXIT
0045H SYM  GRTEQU
4363H SYM  OP
4367H SYM  PTRV1
00FFH SYM  TRUE
79BFH SYM  FPPACK
79EBH SYM  NXT2

43A3H SYM  BCDNUM
4375H SYM  EXPNPT
4379H SYM  P10PTR
7A0EH SYM  BINBCD
7A7EH SYM  ERRMT
7AB1H SYM  FOUND
7AEBH SYM  GE1000
7A53H SYM  INRANG
```

```
7A8EH SYM  LARGE                    7A48H SYM  LIMITS
7A97H SYM  LOOP                     7AE6H SYM  MATCH
7B67H SYM  MOVE                     7B09H SYM  NEG1
7AFEH SYM  NEGEXP                   7A02H SYM  OVRFLW
7B22H SYM  PERC                     7B46H SYM  PUTDIS
7B31H SYM  RDYPUT                   7A89H SYM  SMALL
7B5EH SYM  TOOBIG                   7B64H SYM  TOOSML
7A06H SYM  UNDFLW                   7A0AH SYM  ZERO
     MOD  FADD
007FH SYM  EXP                      0040H SYM  EXP0
0080H SYM  SIGN                     7B79H SYM  FADD
7B92H SYM  FADD1                    7B94H SYM  FADD2
7B9BH SYM  FADD3                    7BA2H SYM  FADD4
7BAFH SYM  FADD5                    7BB9H SYM  FADD6
7B7AH SYM  FADDX                    7B79H SYM  SCALE
     MOD  FLOAT
0040H SYM  EXP0                     7BCEH SYM  F1
7BBEH SYM  FLOAT                    7BD2H SYM  RETN
     MOD  FIX
007FH SYM  EXP                      0040H SYM  EXP0
0050H SYM  EXP16                    7FFFH SYM  MAX
0080H SYM  SIGN                     7BD3H SYM  FIX
7BEBH SYM  FIX0                     7BF1H SYM  FIX1
7BFEH SYM  OVFL                     7BF8H SYM  RETN
7BFCH SYM  RETN1
     MOD  FPCMPR
007FH SYM  EXPMSK                   7C13H SYM  CMPEXP
7C2BH SYM  CMPG01                   7C3BH SYM  CMPG02
7C38H SYM  CMPG03                   7C13H SYM  FPCMP
7C04H SYM  FPCMPR
     MOD  FDIV
007FH SYM  EXP                      0040H SYM  EXP0
0080H SYM  SIGN                     7C3CH SYM  FDIV
7C6AH SYM  FDIV1                    7C4EH SYM  FDIV2
7C5CH SYM  FDIV3                    7C74H SYM  MDRETN
     MOD  FPZERO
0040H SYM  EXP                      7C7BH SYM  FPZERO
7C7BH SYM  FPZR1                    7C82H SYM  FPZR2
7C87H SYM  FPZRX
     MOD  NORM
7C8EH SYM  NORM                     7C98H SYM  RETN
     MOD  IDIV
7CABH SYM  DIV1                     7CBDH SYM  DIV2
7CCDH SYM  DIV3                     7CBAH SYM  DIVX
7C99H SYM  IDIV
     MOD  ZEROD
7CD9H SYM  RETN                     7CD2H SYM  ZEROD
     MOD  NEGD
7CDAH SYM  NEGD
     MOD  ABSD
7CE4H SYM  ABSD                     7CE9H SYM  RETN
     MOD  RBUMP
7CEBH SYM  RBUMP                    7CEAH SYM  RBUMP1
     MOD  BUMP2
7CFDH SYM  B1                       7CF4H SYM  BUMP2
     MOD  BUMP3
7D18H SYM  B1                       7CFFH SYM  BUMP3

MEMORY MAP OF MODULE S_DRIV
READ FROM FILE :F1:SADRIV.LNK
WRITTEN TO FILE :F1:SADRIV
MODULE IS NOT A MAIN MODULE

START   STOP LENGTH REL NAME

0000H  0005H     6H  A  ABSOLUTE
```

```
000FH  0023H     15H   A  ABSOLUTE
0027H  0029H      3H   A  ABSOLUTE
0030H  0035H      6H   A  ABSOLUTE
0038H  003AH      3H   A  ABSOLUTE
7231H  7D1AH    AEAH   B  CODE
7D1BH  F6BFH  79A5H   B  MEMORY
```