A PROGRAMMING SYSTEM FOR
HYBRID COMPUTER COMPUTATION

by

ROGER BRADFORD FISH

Submitted in Partial Fulfillment
of the Requirements for the
Degree of Bachelor of Science
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
June, 1975

Signature redacted
Signature of Author
Department of Mechanical Engineering, May 9, 1975

Certified by.....
Signature redacted
Thesis Supervisor

Accepted by.....
Signature redacted
Chairman, Departmental Committee on Theses

Archives
MAR 15 1976
ABSTRACT

Simulation of dynamic systems is often limited by the computer hardware it is running on. Digital simulation is often slow and analog computation lacks the ability to do operations such as delays and nonlinear functions. Hybrid computation is a technique to combine digital and analog computers into a system which has the best features of each type of computer. This thesis is an attempt to produce a programming system to reduce the problems presently involved in hybrid computing. A system of digital computer programs was created which automate many operations needed in hybrid computing. Also a hybrid block operation's interpreter was developed to allow high speed parallel operation of the analog and digital computers. Tests of this system show capabilities and features which improve the operation and programming of hybrid computers. The total system gives the user a powerful and simple system for improving his simulation computations.
ACKNOWLEDGEMENTS

I would like to acknowledge my thesis advisor, Richard S. Sidell, for the ideas, suggestions, and guidance he has given me during my undergraduate education and during the preparation of this thesis.

I also want to express my thanks to my parents who have supported and assisted me during my education, and to my mother who suffered through the typing of this thesis.

Thank you very much.
## TABLE OF CONTENTS

Abstract  
Acknowledgements  
Table of Contents  
List of Tables and Figures  
Introduction  
Hybrid Software System  
Hybrid Programming Language  
System Operation  
Hybrid Simulation Test Cases  
Conclusion  
Recommendations  
Bibliography  
Appendixes:  
  A) Simulation  
  B) Gas Turbine Ship  
  C) Program Listings  

page 2.  
page 3.  
page 4.  
page 5.  
pages 6-8.  
pages 11-12.  
pages 16-18.  
page 22.  
page 23.  
page 31.  
page 32.  
page 33.  
pages 34-68.  
pages 35-37.  
pages 38-39.  
pages 40-68.
LIST OF TABLES AND FIGURES

Tables:
Table 1. Roles of Digital Computers in Hybrid Computation and Simulation page 7.
Table 2. Goals of System page 13.
Table 3. Hybrid Operations and Commands page 19.

Figures:
Figure 1. Hybrid Computing System page 9.
Figure 2. Bilateral Computer System page 10.
Figure 3. Hybrid Programming System page 14.
Figure 4. Interpreter Flow Chart page 15.
Figure 5. Example Block Program page 20.
Figure 6. Example Hybrid Program page 21.
Figure 7. Bar Simulation Diagram page 24.
Figure 8. Hybrid Model of Bar page 25.
Figure 9. Bar Program Listing page 26.
Figure 10. Gas Turbine Ship Diagram page 27.
Figure 11. Hybrid Model of Ship page 28.
Figure 12. Ship Program Listings page 29.
13. 
INTRODUCTION

Hybrid computation includes all computing techniques which combine features of digital and analog computation in the solution of a problem. The term hybrid computer is now used to characterize computer systems involving linkages between general-purpose digital computers and electronic analog computers. Digital and analog computers each have characteristics which lend them to certain forms of computation.

Analog computers are excellent systems or simulating continuous systems.

Analog computers:
1) Treat all variables in continuous form.
2) All components operate in parallel.
3) Computation speed is not affected by problem complexity.
4) Operations such as multiplication, addition, integration are easily done.
5) Provisions exist for system modification during simulation.

Digital computers are best suited for logic operations, control signal generation, and data manipulation.

Digital computers:
1) Have the facility to memorize numerical and nonnumerical data indefinitely.
2) They perform logical operations and decision making using numerical or nonnumerical data.
3) Floating point arithmetic eliminates scale-factors.
4) Information can be handled in discrete form.
5) Digital programs can control and modify themselves, or analog systems by control signals.

An outline of hybrid operations is shown in table 1.
ROLES OF DIGITAL COMPUTERS IN HYBRID COMPUTATION AND SIMULATION

I.) Digital supervision by digital system:
   a) Supervisory control of analog-computer studies:
   b) Storage and reduction of results:
   c) Report preparation.

II.) Digital "Housekeeping" functions:
   a) Set coefficient and function-generator settings.
   b) Perform static checking.
   c) Run diagnostic programs.
   d) Do digitally controlled patching.

III.) Combined analog-digital simulation:
   a) Open-loop combinations.
      1) Digital operations on analog-computer outputs without feedback.
      2) Analog operation on digital-computer outputs without feedback.
   b) Digital simulation of a digital computer.
      1) Simulate closed loop digital controller.
   c) Closed-loop combined simulation:
      1) Parallel signal processing with feedback.
      2) Digital signal generation, recording, and control.
      3) Data sampling and coefficient modification.

Table 1.
Hybrid computer techniques represent an effort to combine the best features of each type of computer to produce the optimal computing system. True hybrid computers are those systems which combine appreciable amounts of each type of hardware, and are capable of bilateral operation. (See fig. 2.) A system diagram of the Mechanical Engineering Joint Computer Facility hybrid computer is shown in figure 1.

Hybrid computation is ideally suited for several computation methods:
1) Sampled data system simulation.
2) Random process simulation.
3) Control system optimization.
4) Simulation of distributed parameter systems.
5) General simulation of systems which combine continuous and discrete signals.

Presently hybrid computation is accomplished by programming the analog computer, and then writing a FORTRAN program for the digital computer using a number of data conversion and analog control subroutines. The routines do operations such as send and receive signals, set analog coefficients and control the operation of the analog computer.

The goal of my project is to produce a programming system which will allow parallel operation of the analog and digital computer during hybrid runs. This system must be fast enough so that the digital computer will not add a large time delay to the signals it is processing. The system will use digital operations to replace and complement the analog computational elements, and to control and synchronize total system operation.
HYBRID COMPUTING SYSTEM
BILATERAL HYBRID SYSTEM
HYBRID SOFTWARE SYSTEM

To accomplish goal of this thesis I have developed an interpretive software system which can perform many functions necessary to simplify and automate hybrid computer operations. The capabilities designed into the system are shown in table 2. The system has six basic states of operation.

1) Supervisor state.
In this mode the system processes input from the user and determines the nature of the command. The supervisor handles all overhead and linkage operations required, and handles user interaction and error analysis. If the input is a command to perform an immediate operation the system performs the operation or gives control to another section which performs the operation, such as input/output or set up. If the input is code to be compiled, control is passed to the compile section. The supervisor is itself broken into several sections so that it is able to control linkage between the several main programs which make up the system.

2) Program compiler.
In this mode the system inputs the users hybrid programming language statements and compiles them into an intermediate code. This code is stored in disk files which are referenced by the loader at load time.

3) Analog control.
In this mode the system reads the users command and performs the desired operation on the analog system. Control operations normally done manually at the analog control console are done automatically by the digital computer.
4) Input/Output.
This mode controls the reading, printing, and plotting of data requested by the user. After control is passed from the supervisor to this section, the storage area referenced is located in the symbol table, and the operation performed. Scaled fraction data is used as the data type for the system. The present input output operations include reading, printing, and plotting, and are called by giving the operation, the storage area identifier, and any required parameters.

5) Loader.
This mode controls data base and program loading. The compiled program is loaded from the disk and the storage allocated by the compiler is initialized to the required form. Buffers, tables, and delay lines are cleared and special pointers and constants loaded. The system can now perform input/output or enter the hybrid interpreter section.

6) Hybrid Interpreter.
This mode controls the actual parallel operation of the analog and digital computers. The users hybrid program is translated from the intermediate form, generated by the compiler, into core addresses in the digital computer. The interpreter then synchronizes itself with an external real time clock and begins execution. The users program is executed line by line for the specified time. Automatic A/D and D/A conversions are performed each cycle. After the cycle is completed, the interpreter waits for the clock time step to finish and then restarts the cycle. If time steps specified are too small to complete a program cycle, error messages are given. (See fig. 4.)

A system structure and information flow diagram of the complete system is shown in figure 3.
GOALS OF SYSTEM

I.) Control capabilities:
   a) Program compilation.
   b) Program loader.
   c) Time synchronization.
   d) Analog computer control.
   e) Input/output controller.

II.) Block oriented programming:
   a) Mathematical operations.
   b) Automatic buffer input/output.
   c) Function table lookup and interpolate.
   d) Time delay line simulation.
   e) Data manipulation. (scaled fraction).
   f) Storage allocation.
   g) A/D, D/A automatic conversion.

III.) Analog setup and control:
   a) Servo pot setting.
   b) Analog mode control.
   c) Time Scale control.
   d) Analog device measurement.

IV.) Automatic digital input/output:
   a) Plot buffers. (scaled fraction data vs. time)
   b) Print. (scaled fraction data)
      1) Buffers
      2) Tables
      3) Variables
   c) Read. (scaled fraction data)
      1) Variables
      2) Function tables
      3) Input buffers, delay line buffers

Table 2.
Figure 3.
ENTRY

SET POINTERS + FLAGS
TRANSLATE ADDRESS AND OP CODES TO BYTE COUNTS
SYNCHRONIZE WITH CLOCK

READ A/D CHANNELS
EXECUTE PROGRAM STEPS
WRITE D/A CHANNELS

DONE

WAIT FOR CLOCK DONE
SET COUNT, LOOP TO READ

DATA

YES

RETURN

NO

FIGURE 4. INTERPRETER FLOW CHART
HYBRID PROGRAMMING LANGUAGE

The features included in the programming language are listed in table 2, section II. The hybrid language is a simple block-oriented system with a format similar to popular digital simulation systems such as EAI'S "HYTRAN", or IBM'S "CSMP". The major difference between these systems and the hybrid block system is the internal linkage system. The former are pure digital simulation systems, while the hybrid system allows a simulation program using analog and digital operation blocks which are linked together through the A/D, D/A interface. This allows operations such as long time delays, data storage, multi-input functions, and complex logic control, which can all be done best digitally, to become integral parts of an analog simulation.

The block nature of the system allows the user to set up a system of defined operations using simple commands and automatically link them to the analog system. The operations available to the user are listed in table 3. An example of a hybrid block program is shown in figure 5, and the hybrid program needed to create this system of blocks is shown in figure 6.

The programmer need only know a few simple rules of argument syntax and the operation codes to program any block system. The user has several data types and data structures available to him.

1) Buffers for getting data from or putting data to.
2) Delay lines of specified length which shift one entry per cycle.
3) Scaled fraction constants between ± 1.0.
4) Internal variables referenced by name.
5) Two-dimensional tables for function table lookup.
6) Analog to digital and digital to analog I/O channels.
7) Double precision variables for integrals.
One of the most important features of the system is the automatic input-output operations and data conversions done each time step. The interpreter reads all the available A/D converters into a special buffer at the start of each interpretive cycle. The program then interprets the program, taking values of the referenced A/D channels from the buffer. References to D/A channels also refer to a special buffer. After the interpreter has executed the program for the cycle the D/A block is written out from the D/A buffer. The interpreter then waits for the external clock to finish and the process begins again. While the program is executing, the interpreter keeps a pointer that allows sequential access to buffer entries. This allows the user to record data into or send data from a buffer with a PUT or GET statement.

The function generator capabilities of the system are very useful in hybrid computation. These functions allow the user to store functions digitally and use them during hybrid simulations. Single input functions with any number of breakpoints can be used to generate nonlinear functions that are difficult to generate on an analog computer. Multiple input functions with 441 values over the range of the two inputs allow nonlinear functions of two parameters to be generated during simulations by referencing the data table with a function operator.

Two function types are presently available. The first is a single input, piecewise linear table look up function. The user gives a table of input break points and output values and then references the table by name in a function operation. The value is computed by linear interpolation of the input to the break points and the output values. The second type of function available is the two input function. This function is a fixed break point, two dimensional table look up system. The user creates a table of 21 by 21 locations and inputs the function values to the table in reference to an X Y plane.
The function calculates a location in the input arguments plane and interpolates between four points to find the output.

Delay line simulation is the final operation available that is difficult or impossible to do on an analog computer. Since lumped parameter systems often require transport delays in their simulation, the digital computer gives us the ability to delay signals any amount of time required. The delay operator works by shifting the input signal through a serial shift register of length $N$. ($N = \text{DELAY TIME/CYCLE TIME}$.) As the interpreter executes the program, a pointer is recirculated through the delay line. When the pointer reaches a storage slot the delayed value is sent to the output argument and the new input value is loaded into the slot. This operation continues, slots being loaded, delayed, and sent out as long as the program is executing.

Mathematical operations are also available to do arithmetic operations with constants, variables, or A/D channels. (See table 2.) This feature frees analog amplifiers from operations such as addition, subtraction and inversion to be used as integrators or in other analog operations.

The last feature of the hybrid programming language is storage allocation and referencing. The allocation of storage, to be identified by a symbol name, is controlled by DEFINE statements. This command is followed by a specification as to the type of storage, its identifier, and any dimensioning data necessary. The system presently supports five data structures.

1) TABLE - 2 dimensional data tables for functions.
2) BUFFER - 1 dimensional data arrays for I/O.
3) DELAY - delay line buffers of length $N$.
4) VARIABLE - internal variable storage locations. (16 bits.)
5) DOUBLE - double precision variable for integrations. (32 bits.)

All the data handled by the system is in 16 bit scaled fractions for input or output to or from the user.
HYBRID OPERATIONS AND COMMANDS

I.) Operations:

ADD (add two inputs)
SUBTRACT (subtract two inputs)
MULTIPLY (multiply two inputs)
DIVIDE (divide two inputs)
FUNCF1 (one input function)
FUNCF2 (two input function)
DELAY (delay input N cycles)
PUT (input into buffer)
GET (output from buffer)
INTEGRAT (integrate input, store in output)
MOVE (move input to output location)
INVERT (invert input)
DEFINE (allocate storage for processing)

II.) Commands:

*PROGRAM (start new compilation)
*RUN (execute program)
*LOAD (load program from disk files)
*LIST (print program code to printer)
*SYMBOL (print program symbol table)
*EXEC (link to EXECUTIVE program)
*READ (input data to specified locations)
*PRINT (print contents of specified arg.)
*PLOT (plot buffer on CRT)
*SETPOT (set servo pot)
*TIMBOP (set external clock for cycles)
*TSCL (set analog time scale)
*ANVAL (read and print analog component)
*EOJ (end of job, exit hybrid system)

Table 3.
EXAMPLE BLOCK PROGRAM

[Diagram with blocks and connections including:
- A/D-1
- A/D-3
- A/D-4
- A/D-2
- A/D-5
- Delay (DL1)
- Add
- Subtract
- Func F2 (Func)
- Multiply
- Integrate (Int1)
- Put Buffer (Buf)

Connections and arrows indicate flow of data.

FIGURE 5.]
EXAMPLE HYBOL PROGRAM

DEFINE VARIABLE AB1;
DEFINE VARIABLE DTP;
DEFINE VARIABLE X1;
DEFINE VARIABLE
DEFINE DOUBLE INT;
DEFINE TABLE FUNC(20,2);
DEFINE BUFFER BUF(200);
DEFINE DELAY DL1 10;

/ DONE STORAGE ALLOCATION, START PROGRAM CODE.
DELAY DL1,@AD01,AB1;
ADD AB1,@AD03,@DA01;
PUT @DA01,BUF;
SUBTRACT @AD04,@AD02,DTP;
FUNC F2 DTP,@AD03,@DA04;
MULTIPLY @AD05,#0.030,X1;
INTEGRATE X1,INT;
ADD @AD02,INT,@DA02;
END;

NOTE: @ADXX = A/D CHANNEL XX
        @DAXX = D/A CHANNEL XX
        # = SCALED FRACTION CONSTANT IDENTIFIER

FIGURE 6.
SYSTEM OPERATION

Once the user has entered the hybrid system he controls the operation of the system by a sequence of commands and program statements from a deck of computer cards. A typical run would consist of a *PROGRAM command followed by the hybrid program statements terminated with an END. The user would then input commands to set the external clock, read any required input data, and then load and run the program. After the system had finished the computation requested the user could print or plot data saved during the run, rerun the program, input a new program or any other combination of operations.

The size of a hybrid program is limited by two factors. During a simulation, the user can specify a run time of any length he requires but the number of cycles of program execution is limited to 32767 maximum. The time step of this cycle is limited to the time required to execute one cycle of the program. The program cycle requires 600 microseconds plus the time required to execute one program statement, typically 15 to 50 microseconds. The second constraint is storage space required. 10000 storage locations are available to the user for tables, buffers, program, etc. Programs require 5 locations per line of code plus 32 locations for A/D and D/A buffers. The remainder of core is available for the users DEFINE allocations.

The format of commands and program statements are standardized and must be input in the expected forms.

1) *COMMAND ARG1, ARG2,...;
2) OPCODE ARG1, ARG2,...;
3) DEFINE (TYPE), ARG1, ARG2,...;
4) END;

Example of programs and command input streams are shown in figures 6,9,12.
HYBRID SIMULATION TEST CASES

Two hybrid simulations test cases were run to capabilities of the hybrid programming system. The first test case is the simulation of a bar of material, steel is assumed, and its response to the impact of a force, assumed to be a hammer. The lumped parameter model of the system is shown in figure 7. The hybrid simulation model is shown in figure 8, and the hybrid program is shown in figure 9. By controlling the time scale of the analog computer and the input potentionmeter of the integrator, simulations of different bar to hammer mass ratios and bar lengths were run. Results for several cases are shown in appendix A.

The second test was of a simplified model of a gas turbine powered ship. The system model of the ship was developed from a report on a pure digital simulation of the system run on an IBM 360. (See fig. 10.) The hybrid system model included four non linear functions and two integrations to simulate the open loop performance of the ship to fuel inputs. A block diagram of the hybrid simulation model is shown in figure 11., and the hybrid program is listed in figure 12 and 13. One test case simulation a speed change from idling to full fuel flow was run simulating 100 seconds of real time. The hybrid simulation required 10 seconds and produced results equal to the digital simulation at 30 seconds and 20% below at 100 seconds. A plot of the simulation output is shown in appendix B.
SIMULATION OF A BAR

LUMPED PARAMETER SIMULATION MODEL
OF AN ELASTIC BAR

FIGURE 7.
HYBRID BAR SIMULATION

FIGURE 8.
*PROGRAM;
/ BAR SIMULATION PROGRAM
DEFINE BUFFER VOUT(550);
DEFINE BUFFER AOUT(550);
DEFINE DELAY DEL 60;
/ DELAY SIGNAL 60 CYCLES
DELAY DEL @AD00,DA00;
PUT @AD01,VOUT;
PUT @AD02,AOUT;
END;
/ CONTROL CARDS AND DATA
*EXEC;
*TIMEOP 10.0,0.020;
*LOAD;
*RUN;
*PLOT VOUT 200;
*PLOT AOUT 200;
*EOJ;

**FIGURE 9.**
GAS TURBINE
SHIP BLOCK DIAGRAM

FIGURE 10.
Figure 11.
*PROGRAM*

/ GAS TURBINE SHIP SIMULATION PROGRAM
DEFINE TABLE TORQE(21,21);
DEFINE TABLE TPROP(21,21);
DEFINE TABLE FPROP(21,21);
DEFINE TABLE FRIC(8,2);
DEFINE VARIABLE X;
DEFINE VARIABLE Z;
DEFINE BUFFER VSHIP(1000);
DEFINE BUFFER RPM(1000);
DEFINE BUFFER BUF2(1000);
DEFINE VARIABLE Y;
DEFINE VARIABLE R;

/ FUNCTIONS FOR SIMULATION
FUNCF2 TORQE, @AD01, @AD00, @DA00;
FUNCF2 TPROP, @AD01, @AD04, @DA01;
FUNCF2 FPROP, @AD01, @AD04, @DA03;
FUNCF1 FRIC, @AD04, @DA04;
PUT @AD01, RPM;
PUT @AD04, VSHIP;
PUT @AD00, BUF2;
END;

FIGURE 12.
/* CONTROL CARDS AND DATA */
*EXEC;
*READ FRIC 7,2;
DATA CARDS FOR TABLE FRIC
  .
  .
*READ TORQE(21,21);
DATA CARDS FOR TABLE TORQE
  .
  .
*READ FPROP(21,21);
DATA CARDS FOR TABLE FPROP
  .
  .
*READ TPROP(21,21);
DATA CARDS FOR TABLE TPROP
  .
  .
*TIMEOP 10.0,0.0010;
*LOAD;
*RUN;
*PLOT VSHIP 200,5,1;
*PLOT BUF2 200,5,1;
*PLOT RPM 200,5;
*EDJ;

FIGURE 13.
CONCLUSION

From the results of tests run on system operation and of the two test cases run, I must conclude that the system fulfills the goals set for it, and provides a useful system for doing hybrid computation. The operations available to the user remove most of the difficulties presently associated with hybrid computation. Input/output, time synchronization, delay simulation, and nonlinear function generation are all available to the user without his needing to be an accomplished programmer.

The practicality of doing hybrid computation in simulation of complex systems is apparent from the second test case. An extremely simplified model, generated from a small amount of data on ship component response was able to produce results in good agreement with a much more complicated model of a large digital computer. Using a hybrid computer system such as this can greatly increase the speed and involvement of many types of computer simulation. As digital simulation of dynamic systems becomes faster and requires less programmer experience analog simulation must move toward hybrid computer system. Hybrid systems have the flexibility and power to automate parameter optimization and statistical studies, which are very time consuming simulations to run digitally. As hybrid computer hardware and software improve simulations will be able to benefit from a system which provides the best features of digital and analog computation.
RECOMMENDATIONS

The system now exists in a form which is flexible and provides the user with many commands. But some features which would allow better utilization of available hardware should be added. There are presently no features for testing or setting sense lines from the analog computer. The system also requires repetition of commands for multiple executions of the hybrid program. Another nonoptimal feature is the line by line non alterable execution of the program. Several features added to the compiler/interpreter system could solve some of these problems. The addition of statement labels and logical decision making operations would allow the program to control its own execution during run time. Several new function types would also improve the system. A function with hysteresis would be useful in some systems. Also the addition of a fixed break point single input function, and a variable bread point multiple input function would simplify function generation for the user.

In general a number of new operations such as transcendental functions and more complex integrator operators would be useful. The input/output operations could also be improved by providing an option for indexed operation so that data could be referenced on integral numbers of time steps. The commands available could also be increased to provide operations such as load and run, repeat operations, and disk storage of data for later use. There are many areas where the system can be expanded and improved and in the future I plan to include most of the above recommendations.
BIBLIOGRAPHY


Korn, Granino A., and Theresa M. Korn.  
*Electronic Analog and Hybrid Computers*, New York:  

Rubis, C.J.  
"Acceleration and Steady-State Propulsion Dynamics of a  
Gas Turbine Ship with Controllable Pitch Propeller",  
New York: The Society of Naval Architects and Marine  
Engineers., 1972.
APPENDIX

A) Bar Simulation Response Plots  
B) Ship Simulation Response Plots  
C) Hybrid Programming System
   Fortran and Assembler Source Listings

pages 35-37.
pages 38-39.
pages 40-68.
SCALE - $T = 40 \, \text{sec.}$

$M = 0.8589$

$V = V_c \, \text{ft/sec.}$
\[ \text{SCALE - } T = 4. \text{ sec.} \]
\[ M = 0.8589 \]
\[ V = V_c \text{ ft/sec.} \]
SCALE - $T = 40 \text{ sec}$,

$M = 8.589$

$V = V_c \text{ ft/sec}$
C MAIN COMMAND PROCESSOR AND COMPILER PROGRAM

REAL*8 LVAR
REAL*8 DPTES
REAL*8 RVSYMP,OPCODE
REAL*8 LTAB,LRUF
REAL*8 ARG(12)
REAL*8 SYMTAB,CLEAR
REAL*8 POT
REAL*8 VAL
INTEGER*2 IREC1,IREC2,LITAL,LITZ,LOAD,ERRORS
INTEGER*2 ORPCRD
INTEGER*2 COMMA,SEMICL
INTEGER*2 EXEC,SETUP
INTEGER*2 READ,PRINT,PLOT,RUN,EOJ,PROGRM
REAL*8 PRGS(3)
REAL*8 LOAL,LODEL
INTEGER*2 LPCUND
INTEGER*2 CARDI,IPPOS,LOCNNT
INTEGER*2 LITADC,LITDAC
INTEGER*2 BUFFER,CARD,ERROR,OPVAL
INTEGER*2 CORE,SYMBOL,DUMP
INTEGER*2 PCOUNT,PSSTART
INTEGER*2 PEND
INTEGER*2 LIST
INTEGER*2 DATATAB
INTEGER*2 TOPCOR
INTEGER*2 TIMEOP,ITYPF,IPCODE
INTEGER*2 SYMTYP
INTEGER*2 NFILE1,NFILE2,NINP,NTYP,NSYMAX
INTEGER*2 OPFLAG,PFLAG,ERROR,NSYMBOL,NSTMT,NOPTAB,I,JK,KK,JJ,
1 NAGS,NN,LM,SAVE,FILE
INTEGER*2 SYMVAL,ATSIGN
INTEGER*2 ARGVAL(10)
INTEGER*2 CTEST,NAME
INTEGER*2 ASTRST,BLANK,DOLESN
INTEGER*2 PNMM
INTEGER*2 ONE,THREE,ZERO
INTEGER*2 IFLAG,NDFCS,IDUMP
INTEGER*2 ARGTYP(13)
INTEGER*2 SETPT,P,TSCALE,ANAVAL
LOGICAL EQUAL
DIMENSION CARD(80),X(10),OPCODE(40),OPVAL(40)
DIMENSION SYMTAB(100),SYMVAL(100),SYMTYP(100)

C **********************************************************

C COMMON DATA BASE AREA
COMMON/CORES/HEAD,PROGRAM,PRINT,PLOT,RUN,EOJ,CORE,SYMBOL,DUMP,LIST,
1 TIMEOP,LOAD,EXEC,SETPT,MODE,ANAVAL,TSCALE
COMMON/CTRL/PFLAG,NFILE1,NFILE2,OPFLAG,MAP,OPVAL,NTYP,NINP,NSYMAX,
1 NOPTAB,LOCNNT,NSYM
COMMON/OPCOD /OPCODE,OPVAL,OPERC,RVSYMP,ARG
COMMON/PRgRM/PSTART,PCOUNT,PEND,NSTMT,PNUM,NDFCS,ERRORS
COMMON/SYmBOL/SYMTAB,SYMVAL,SYMTYP
COMMON PRGS

C ********** ***************************************************************
C FILE SPECIFICATION STATEMENTS
DEFINe FILE 1(100,3,1,IREC1)
DEFINe FILE 2(60,2,1,IREC2)
DEFINe FILE 3(10000,1,1,IREC3)

C ********** ***************************************************************
C MARK FILE 1 WITH PROGRAM ID
IPCODE=1
CALL PUTMES(' $MAIN ;')
READ(NFILE1'1)IPCODE,ERROR
IF(IPCODE.EQ.2) GO TO 10
IF(IPCODE.EQ.5) GO TO 11
IPCODE=1
WRITE(NFILE1'1)IPCODE

C ********** ***************************************************************
C *PROGRAM OP INIITAL ENTRY, INITIAlIZE FLAGS AND CLEAR TABLES
P
CONTINUE
NFILe=3
NDFCS=0
LOCCNT=33
ERRORS=0
SYMVAL=0
NSTMT=1
ERROR=0
PFLAG=0
JERROR=0
PNUM=0
PEND=0
PCOUNT=0
PSTART=0
C CLEAR SYMTAB
DO 700 I=1,NSYMAX
   SYMTAB(I)='CLEAR'
700 CONTINUE
SYMVAL(I)=0
SYMTYP(I)=0
CONTINUE
9150 FORMT(5X,4A2,2X,15)
GO TO 10
C PRINT ERROR MESSAGE
1 WRITE(NTyp,i9120)ERROR,BUFSYM
9120 FORMT(1X,***ERROR #",I5,'SYMBOL ERROR => ',1A8,'SKIP CARD')
C ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **
    IF(IERROR.NE.10)ERRORS=ERRORS+1
10 CONTINUE
ITYpe=7
KK=0
BUFSYM=CLEAR
IERROR=0
ONE=1
THREE=3
ZERO=0
ERROR=0
Lcount=LOCNT
IF(LOCNT.GE.TOPCOR) GO TO 6000
C READ IN A CARD AND PROCESS
CALL CARDTN(ANGPFL AGsOPFLAGIERROR)
11 CONTINUE
IF(IERROR.NE.10)GO TO 1
CALL MOVE(ARG(1),0,OPTEST,0,8)
C LOOK IN OPERATOR TABLE
DO 550 K=NOPTAF
   IF(OPTEST.EQ.OPCODE(K)) GO TO 600
550 CONTINUE
C IF OP CODE ILLEGAL DO ERROR ROUTINE
WRITE(NTyp,i9102)OPTEST
9102 FORMAT(1X,'ILLEGAL OPERATION CODE => ',1A8)
IERROR=4
BUFSYM=OPTEST
GO TO 1
600 OPEDCN=OPVAL(K)
C TEST ARG S FOR TYPE
C ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **
C GET ARG #
C TEST FOR IMMEDIATE
   IF(OPFLAG.EQ.1) GO TO 3000
C TEST FOR DEFINE,FUNCTION,DELAY STATEMENTS
   OPHLFL=REAL(OPTEST)
   IF(OPHALF.EQ.UPSYM) GO TO 1000
   IF(OPHALF.EQ.UPSYM) GO TO 1100
   IF(OPHALF.EQ.UPSYM) GO TO 1200
C NOT TYPE STATEMENT C CHECK ARG S FOR TYPE
C @ IS AN I/O CHANNEL
C # IS A SCALED FRACTION CONSTANT
C OTHERS MUST BE DEFINED NAMES
800 CONTINUE
NARGS=10
C PROCESS EACH ARGUMENT
DO 801 I=1,NARGS
ARGTYP(I)=8
801 ARGV AL(I)=0
C CHECK FOR ARG TYPE IN FIRST CHARACTER
DO 800 IJ=2,NARGS
CALL MOVE(ARG(IJ),0,TEST,0,3)
IF(EQUAL(TEST,0,ATEST,0,3)) GO TO 840
IF(EQUAL(TEST,0,DTEST,0,3)) GO TO 845
IF(EQUAL(TEST,0,STEST,0,1)) GO TO 870
IF(EQUAL(TEST,0,ETEST,0,1)) GO TO 895
C TEST SYMBOL TABLE
IF(NSYMBL.EQ.0) GO TO 810
C LOOK IN SYMBOL TABLE
DO 810 JJ=1,NSYMBL
IF(EQUAL(ARG(IJ),0,SYMTAB(JJ),0,8)) GO TO 820
810 CONTINUE
C NOT IN SYMTAB
WRITE(NTYP,9105) ARG(IJ)
IERROR=5
BUFFSYM=ARG(IJ)
GO TO 1
C SET ARG VALUE TO SYMBOL LOCATION
820 ARGVAL(IJ)=SYMVAL(JJ)
ARGTYP(IJ)=SYMTYP(JJ)
GO TO 880
C ********************************************************************************
C @ PROCESS A/D D/A CHANNEL
C GET NUMBER OF CHANNEL
840 ARGVAL(IJ)=NCFD(ARG(IJ),THREE)+1
ARGTYP(IJ)=6
IF((ARGVAL(IJ).LT.1).OR.(ARGVAL(IJ).GT.16)) GO TO 846
GO TO 880
845 ARGVAL(IJ)=NCFD(ARG(IJ),THREE)+17
ARGTYP(IJ)=7
IF((ARGVAL(IJ).LT.17).OR.(ARGVAL(IJ).GT.32)) GO TO 846
GO TO 880
846 IERROR=25
WRITE(NTYP,9501)
9501 FORMAT(1X,'ILLEGAL CHANNEL CODE')
BUFFSYM=ARGVAL(IJ)
GO TO 1
C ********************************************************************************
C SCALED FRACTION CONSTANT
870 ARGVAL(IJ)=LOCCNT
ARGTYP(IJ)=8
C LOAD SCALED FRACTION INTO STORAGE
   NDFCS=NDFFCS+1
   ITYPE=4
   MM=ISFRX(FPIN(ARG(IJ),ONE))
C WRITE SCALED FRACTION ONTO STORAGE FILE?
   WRITE(NFILE2'NDFCS)ITYPE,LOCCNT,MM,i
   LOCCNT=LOCCNT+1
880 CONTINUE
890 CONTINUE
895 CONTINUE
900 CONTINUE
   PNUM=PNUM+6
   NSTMT=NSTMT+1
C WRITE HYBOL PROGRAM LINE ONTO STORAGE FILE
   WRITE(NFILE1'NSTMT)OPERCD,ARGVAL(2),ARGVAL(3),ARGVAL(4)
   ARGVAL(5)=0
   ARGVAL(6)=0
   WRITE(NTYP9998)OPERCD,ARGVAL(2),ARGVAL(3),ARGVAL(4),ARGVAL(5),IJ
9988 FORMAT(1x,8(I6,4X))
   CONTINUE
   NSTMT=NSTMT+1
C DEFINE STATEMENT PROCESSOR
1000 BUFSYM=ARG(3)
   IFS(NSYMBL.EQ.0) GO TO 1020
C TEST FOR CONFLICT
C LOOK IN SYMBOL TABLE FOR LOCATION
   DO 1005 I=1,NSYMBL
      IF(BUFSYM.EQ.SYMTAB(I)) GO TO 1010
1005 CONTINUE
   GO TO 1020
1010 WRITE(NTYP9107,BUFYSYM)
9107 FORMAT(1x,'NAME CONFLICT => ',1A8,' PREVIOUSLY DEFINED NAME')
   IERROR=6
   GO TO 1
1020 CTEST=BLANK
   CALL BYTE(BUFSYM,0,CTEST,0)
   IF(CTEST.LT.LITA) GO TO 1025
   IF(CTEST.GT.LITZ) GO TO 1025
C PUT NEW NAME IN SYMTAB
   NSYMBL=NSYMBL+1
   IF(NSYMBL.GT.NSYM) GO TO 1021
   SYMTAB(NSYMBL)=BUFSYM
C SYMBOL VALUE IS LOCATION COUNTER
   SYMVAL(NSYMBL)=LOCCNT
   SYMTYP(I)=1
   LSAVE=LOCCNT
   LOCCNT=LOCCNT+1
   BUFSYM=ARG(2)
C TEST FOR TYPE OF DEFINE STATEMENT
IF(RUFSYM, EQ, LVAR) GO TO 10
IF(RUFSYM, EQ, LTAR) GO TO 1030
IF(RUFSYM, EQ, LBUF) GO TO 1040
IF(RUFSYM, EQ, LDBL) GO TO 1060
IF(RUFSYM, EQ, LDEL) GO TO 1070
WRITE(NTYP, 9110) RUFSYM

9110 FORMAT(1X, 'ILLEGAL TYPE SPECIFICATION IN A DEFINE =>', 1A8)
ERROR=7
GO TO 1

1021 IFERROR=29
GO TO 1

C ***************************************************************
C TABLE DEFINED ALLOCATE SPACE FOR N*M
C CALCULATE TABLE SIZE
1030 MM=NCFD(ARG(4), ZERO)
SYMTYP(I)=2
ITYPE=1
ZERO=0
MM=NCFD(ARG(5), ZERO)
NN=MM*MM
GO TO 1050

C *****************************************************************
C BUFFER DEFINED ALLOCATE LENGTH N+1
C CALCULATE BUFFER LENGTH
1040 NN=NCFD(ARG(4), ZERO)
ITYPE=2
SYMTYP(I)=3
C SAVE CODE =NN
1050 LOCCNT= LOCCNT+NN
1051 NDCCS=NDCCS+1
C WRITE TYPE, LENGTH LOCATION AND CODE INTO STORAGE FILE 2 FOR LOADER
WRITE(NFILE2, 'NDCCS') ITYPE, LSAVE, NN, I
GO TO 19
C IF DOUBLE PRECISION SCALED FRACTION SAVE ONE MORE HALFWORD
1060 LOCCNT= LOCCNT+1
SYMTYP(I)=4
GO TO 19

C ***************************************************************
C DELAY STATEMENT ALLOCATE LENGTH AND POINTERS
1070 NN=NCFD(ARG(4), ZERO)
SYMTYP(I)=NN
ITYPE=3
ZERO=0
I=0
IF(EQUAL(ARG(5), F, ETEN, F, 1)) GO TO 1071
I=ISFIX(FPON(ARG(5), ZERO))
1071 LOCCNT= LOCCNT+NN+2
GO TO 1051
1075 WRITE(NTYP, 9110) RUFSYM
9110 FORMAT(1X,'ILLEGAL NAME IN A DEFINE =>',1A8)
       ERROR=8
       GO TO 1
C       *       *       *       *       *       *       *       *       *       *
C      DELAY STATEMENT PROCESSOR
1100   CONTINUE
       BUFSYM=ARG(2)
C      LOOK IN SYMBOL TABLE
   DO 1101 I=1,NSYMRL
       IF(BUFSYM,EQ.*SYMTAB(I)) GO TO 1105
1101   CONTINUE
       ERROR=20
       GO TO 1
1105   LSAYV=SYMVAL(I)
       IF(SYMTYP(I),LT.0) GO TO 1106
       ERROR=26
       GO TO 1
1106   ARG(2)=ARG(3)
       ARG(3)=ARG(4)
C      SHIFT DELAY NAME TO 4TH ARGUMENT SLOT
       ARG(4)=BUFSYM
       GO TO 900
C       *       *       *       *       *       *       *       *       *       *
C      FUNCTION STATEMENT PROCESSOR
1200   BUFSYM=ARG(2)
   DO 1210 I=1,NSYMRL
       IF(BUFSYM,EQ.*SYMTAB(I)) GO TO 1215
1210   CONTINUE
       WRITE(NTYP,9115)BUFSYM
9115   FORMAT(1X,'UNDEFINED FUNCTION NAME =>',1A8)
       ERROR=9
       GO TO 1
1215   ARG(2)=ARG(3)
       ARG(3)=ARG(4)
C      SHIFT FUNCTION NAME TO LAST ARG LOCATION
       IF(EQUAL(ARG(5),0,ETEST,0,1)) GO TO 1220
       ARG(4)=ARG(5)
       ARG(5)=BUFSYM
       GO TO 900
1220   ARG(4)=BUFSYM
       GO TO 800
C       *       *       *       *       *       *       *       *       *       *
C      CARD IMMEDIATE COMMAND
3000   CONTINUE
       IDUMP='1
C      TEST FOR OPERATION BLOCK CODE AND BRANCH
       IF(OPERCD,EQ.PROGPM) GO TO 2
       IF(OPERCD,EQ.Symbol) GO TO 3600
       IF(OPERCD,EQ.EQJ) GO TO 5000
       IF(OPERCD,EQ.TSCALE) GO TO 7000
IF(OPERCD,EQ,ANAVAL) GO TO 8000
IF(OPERCD,EQ,SETPOT) GO TO 9000
C IF OP IN EXEC LINK TO EXEC
IF(OPERCD,EQ,EXEC) GO TO 3001
IF(OPERCD,EQ,LOAD) GO TO 3001
IF(OPERCD,EQ,LIST) GO TO 3001
IF(OPERCD,EQ,PRINT) GO TO 3001
IF(OPERCD,EQ, PLOT) GO TO 3001
IF (OPERCD, EQ, READ) GO TO 3001
IF (OPERCD, EQ, RUN) GO TO 3001
IF(OPERCD, EQ, TIMEOP) GO TO 3001
C ILLEGAL OPERATION
ERROR=10
GO TO 1
C LINK TO EXEC MAIN
3001 CALL CHAIN(PROGS,2)
C ********************************************
C SYMBOL TABLE LIST BLOCK
3690 CONTINUE
WRITE(NTYP,9371)LOCCNT, TOPCOR
9371 FORMAT(' LOCATION COUNTER =',I6,' OF ',I6,'/
WRITE(NTYP,9372)
9372 FORMAT(I6,'SYMBOL TABLE'/)
WRITE(NTYP,9373)(SYMTAB(I),SYMVAL(I),SYMTYP(I),I=1,NSMBL)
9373 FORMAT(16X,'SYMBOL TABLE',/
GO TO 19
C ********************************************
C MEMORY FULL FATAL ERROR
600 CONTINUE
WRITE(NTYP,9395)LOCCNT, TOPCOR
9395 FORMAT('LOCATION COUNTER =',I6,' AVAILABLE CORE=1,I6,' FATAL ERROR',/)
GO TO 5000
C ********************************************
C SET ANALOG TIME SCALE
700 CONTINUE
NN=VCFD(ARG(2),ZERO)
CALL HINT
CALL TSCL(NN)
GO TO 19
C ********************************************
C SET ANALOG VOLTAGE FROM SPECIFIED CHANNEL
800 CONTINUE
CALL MOVE(ARG(3),0,POT,0,4)
CALL HINT
VAL=ANVAL(POT)
PUT, VAL
GO TO 19
C ********************************************
C SET SERVO POT SETTING AND DISPLAY ERROR
900V CONTINUE
CALL MOVE(ARG(2),0,POT,0,4)
VAL=FPIN(ARG(3),ZERO)
CALL HINT
CALL SPOT(POT,VAL,IER)
PUT,IER
GO TO 10
C END OF JOB EXIT TO OPERATING SYSTEM
500V CONTINUE
WRITE(NTYP,9400)
940V FORMAT(1X,'** ** ** END OF JOB ** ** ** **')
C END OF PROGRAM
END
C EXEC PROGRAM LOADER INPUT/OUTPUT, CLOCK CONTROLLER

REAL*8 PROGS(3)
REAL*8 LOAD, LDEL
REAL*8 LVAR
REAL*8 OPTEST
REAL*8 BUFSYM,_OPCODE
REAL*8 LTAB, LBUF
REAL*8 ARG(10)
REAL*8 SYMTAB, CLEAR

INTEGER*2 LOC, LOAD, ITYPE
INTEGER*2 OPERCD, LBOUND, CARDI, IPOS, LOCCNT, LITADC, LITDAC, BUFFER, CARD.
INTEGER*2 ERROR, OPCODE, LITZ
INTEGER*2 CORE, SYMBL, DUMP, PCOUNT, PSTART, PEND, LIST, DATTAB, TOPCOR
INTEGER*2 TIMEOP, ITYPE, IPCODE, SYMTYP
INTEGER*2 COMMA, SEMICL
INTEGER*2 READ, PRINT, PLOT, RUN, EOJ, PROGRAM
INTEGER*2 OPFLAG, PFLAG, IERROR, NSYMBL, NSTMT, NOPTAB, I, J, K, IJ, KK, JJ,
1 NARGS, NN, MM, LSAVE, NFILE
INTEGER*2 SYMVAL, ATRNS
INTEGER*2 ARGVAL(10)
INTEGER*2 CTSTART, NAME
INTEGER*2 BUFFER(40), BFIELD(10)
INTEGER*2 ATRSK, BLANK, DOLSN
INTEGER*2 PNUM
INTEGER*2 ERRORS
INTEGER*2 EXEC
INTEGER*2 ONE, THREE, ZERO
INTEGER*2 IFLAG, NDFCS, IDUMP
INTEGER*2 TSTEP, NCYCL, IREC1, IREC2
INTEGER*2 LENGTH, LSAVE, ERROR, NFILE1, NFILE2, NTYP, NINP, NSYMAX
INTEGER*2 SETPOT, MODE, ANVAL, TSCL
REAL XSCL(4), XSCALE

LOGICAL EQUAL
DIMENSION XSCALE(200), XSCALE(4)
DIMENSION CARD(80), X(25), OPCODE(40), OPCODE(40)
DIMENSION SYMTAB(100), SYMVAL(100), SYMVAL(100)
DIMENSION DATTAB(1000)

C COMMON DATA AREA

COMMON/CODES/READ, PROGRAM, PRINT, PLOT, RUN, EOJ, CORE, SYMBOL, DUMP, LIST,
1 TIMEOP, LOAD, EXEC, SETPOT, MODE, ANVAL, TSCL
COMMON/CONTRL/PFLAG, NFILE1, NFILE2, TOPCOR, OPFLAG, NTYP, NINP, NSYMAX,
1 NOPTAB, LOCNT, NSYMBL
COMMON/OPCODE/_OPCODE, OPCODE, OPVAL, OPCODE, BUFSYM, ARG
COMMON/PROGRAM/PSTART, PCOUNT, PEND, NSTMT, PNUM, NDFCS, ERRORS
COMMON/SYMBL/SYMTAB, SYMVAL, SYMVAL, SYMTYP
COMMON/PROGRAM

C DATA INITIALIZATION

DATA CLEAR/1
DATA FTEST/1!

C FILE SPECIFICATION STATEMENTS
DEFINE FILE 1(100,3,U,IREC1)
DEFINE FILE 2(60,2,U,IREC2)
DEFINE FILE 3(10000,1,U,IREC3)

C MARK FILE 1 WITH PROGRAM ID
NFILE3=3
IPCODE=2
WRITE(NFILE1'1)IPCODE
CALL OUTMEM('EXEC ;')

C READ CODE FROM DAFILE
READ(NFILE3'1)(DATTAB(I),I=1,TOCPOR)
GO TO 1

1 CONTINUE
C WRITE ERROR MESSAGE AND PROBABLE SYMBOL
WRITE(NTYp,9001)ERROR,BUSYM
9001 FORMAT(1X,'***** ERROR #',I5,'BUSYM =',1A8)
GO TO 1

C INITIALIZE FLAGS AND BUFS
BUSYM=CLEAR
ERRORS=0
ZERO=0
IRFLAG=0
ERROR=0
OPFLAG=0

C READ A CARD AND PROCESS INTO ARGS AND FLAGS
CALL CARDIN(ARGPFLG,OPFLAG,ERROR)
IF(TEPROR,NE,0) GO TO 1

C IF NOT IMMEDIATE OPERATION GOTO MAIN
IF(OPFLG,NE,1) GO TO 15
OPTEST=ARG(1)

C LOOK IN OP CODE TABLE FOR VALUE OF OP
DO 16 I=1,NOPTAB
IF(OPTEST.EQ.OPCODE(I)) GO TO 17
16 CONTINUE
ERROR=4
GO TO 1

C ITEM Mandatory field
17 OPERCD=OPVAL(I)
9510 FORMAT(1X,5(1A8,2X))
C TEST AND GO TO PROPER SUBBLOCK
IF(OPERC,=TIMEOP) GO TO 3800
IF(OPERC,=HEAD) GO TO 3100
IF(OPERC,=PRINT) GO TO 3200
IF(OPERC,=RUN) GO TO 3400
IF(OPERC,=LOAD) GO TO 2000
IF(OPERC,=LIST) GO TO 2500
IF(OPERC,=DUMP) GO TO 3500
IF(OPERCODE.EQ.PLOT) GO TO 4000
C RETURN TO MAIN IF NOT A * OPERCODE
15 IPCODE=5
C WRITE CORE LOAD TO DAFILE
    WRITE(NFILE31)(DATTAB(I),I=TOPCOR) WRITE(NFILE31)(IPCODE,ERROR)
    CALL CHATN(PROGS1)
C ******************************************** LOAD PROGRAM AND INITIALIZE DATA BASE
2000 PSTART=LCCNT
    PCOUNT=PSTART
    DO 2010 I=2,NSTMT
       READ(NFILE11)DATTAB(PCOUNT),DATTAB(PCOUNT+1),DATTAB(PCOUNT+2),DAT,
       ITAB(PCOUNT+3),DATTAB(PCOUNT+4),JJ
       PCOUNT=PCOUNT+5
    2010 CONTINUE
    PCOUNT=PCOUNT-5
    PEND=PCOUNT
    WRITE(NFILE9200,PSTART,PEND)
9200 FORMAT(1X,'PROGRAM STARTS AT ',I5,5X,'PROGRAM ENDS AT ',I5,/) DO 1000 I=1,NDFCS
    READ(NFILE21)ITYPE,LOC,LENGTH,NN
    PUT,ITYPE,LOC,LENGTH,NN
C TABLE,BUFFER,DELAY,DONE
    IF(ITYPE.EQ.0) GO TO 1100
    IF(ITYPE.EQ.2) GO TO 1200
    IF(ITYPE.EQ.3) GO TO 1300
    IF(ITYPE.EQ.4) GO TO 1400
    IF(ITYPE.EQ.5) GO TO 1500
    ERROR=21
    GO TO 1
1100 DATTAB(LOC)=0
    GO TO 999
1200 DATTAB(LOC)=NN
    GO TO 999
1300 DATTAB(LC)=LENGTH
    DATTAB(LC+1)=1
    DATTAB(LC+2)=NN
    GO TO 999
1400 DATTAB(LC)=LENGTH
999 CONTINUE
1000 CONTINUE
1500 CONTINUE
IF(PEND.LT.TOPCOR) GO TO 10
C IF CORE LOAD TOO LARGE ERROR
    WRITE(NFILE9500,PEND,ERROR,TOPCOR)
9500 FORMAT(1X,'PROGRAM TOO LARGE =',I6,'AVAILABLE CORE =',I6,/) ERROR=36
    GO TO 1
C ********************************************
C **LIST PROGRAM CODE ON PRINTER**

2500 CONTINUE
DO 2000 I=IEND+5,IPSTART,5
WRITE(NTYP,9210)IDATTAB(I),DATTAB(I+1),DATTAB(I+2),DATTAB(I+3)
9210 FORMAT(5X,5I5,5X,5I6,5X,4I6,4X,4I6,4X,4I6)
2000 CONTINUE
IF(ERRORS.EQ.0) GO TO 90
WRITE(NTYP,9385) ERROR
9385 FORMAT(1X,15,'ERRORS IN PROGRAM RUN WILL BE SUPPRESSED')
ERROR=12
IRFLAG=1
GO TO 1

C READ OPERATION PROCESS ARG'S
C NAME LENGTH WIDTH
3100 NN=NCFD(ARG(3),ZERO)
ZERO=0
MM=NCFD(ARG(4),ZERO)
BUFSYM=ARG(2)
C LOOK IN SYMBOL TABLE FOR LOCATION
DO 3110 I=1,NSYMPL
IF(=BUFSYM.EQ.SYMTAB(I)) GO TO 3120
3110 CONTINUE
WRITE(NTYP,9230)
9230 FORMAT(1X,15,'SYMBOL NOT FOUND')
BUFSYM=ARG(2)
ERROR=11
GO TO 1
3120 LSAVE=SYMVAL(I)
ITYPE=SYMTyp(I)
IF(ITYPE.LT.0)LSAVE=LSAVE+2
IF(ITYPE.GT.0)LSAVE=LSAVE-1
PUT,NN,MM
DO 3130 I=1,NN
READ(NIND,9050)(X(JJ),JJ=1,MM)
9050 FORMAT(10F10.5)
DO 3125 J=1,MM
K=LSAVE+J
DATTAB(K)=ISFIX(X(J))
3125 CONTINUE
LSAVE=LSAVE+MM
3130 CONTINUE
GO TO 10

C **PRINT OPERATION PROCESS ARG'S**
3200 BUFSYM=ARG(2)
C LOOK IN SYMBOL TABLE FOR LOCATION
DO 3210 I=1,NSYMPL
IF(=BUFSYM.EQ.SYMTAB(I)) GO TO 3220
3210 CONTINUE
WRITE(NTYP,9230)
BUFFSYM=ARG(2)
go to 1
3220 LSAVE=SYMVAL(I)
NN=NCFD(ARG(3),ZER)
MM=NN/10+1
PUT,NN,MM
DO 3230 I=1,MM
DO 3225 J=1,10
K=LSAVE+j
3225 X(J)=FLOTS(DATBAR(K))
WRITE(NTYP,9240)(X(J),J=1,10)
9240 FORMAT(1X,1F12.5)
LSAVE=LSAVE+10
3230 CONTINUE
GO TO 10
C ***********************************************
C RUN OPERATION, INITIALIZE AND CALL INTERP
3402 CONTINUE
IF(PSTART.EQ.PEND) GO TO 1
CALL WINT
CALL CVTO(0)
CALL TC
PAUSE 10
C LINK TO ASM INTERPRETER PHASE
CALL INTERP(DATBAR,PSTART,NSTMT,TSTEP,NCYCL)
CALL SP
CALL CVTO(1)
C DONE SIMULATION, RETURN
GO TO 10
C ***********************************************
C CLEAR STORAGE
3510 DO 3519 I=1,ITOPCOR
3519 DATBAR(I)=0
WRITE(NFILE,9301)(DATBAR(I),I=1,ITOPCOR)
go to 10
C ***********************************************
C CLOCK SET UP RLOCK
C WRITE OUT CLOCK VALUES AND SET CLOK
3820 DT=PIN(ARG(3),ZER)
ZER=0
TIME=PIN(ARG(2),ZER)
NCYCL=1+IX(TIME/DT)
TSTEP=ISFIX(DT)
PUT,TIME,NSTMT,NCYCL,TSTEP
CALL WINT
CALL CTSTEP(DT)
go to 10
C ***********************************************
C PLOT BLOCK, PROCESS ARGUMENTS
C NN= # OF POINTS TO BE PLOTTED, MAX=200
4000 CONTINUE
NN=NCFD(ARG(3),ZERO)
   IF(NN,GT,200) NN=200
ZERO=0
4005 BUTFSYM=ARG(2)
C LOOK IN SYMBOL TABLE FOR LOCATION
DO 4010 I=NSYMBL
   IF(EQUAL(BUTFSYM,O,SYMTAB(I),0,8)) GO TO 4020
4010 CONTINUE
IERROR=41
GO TO 1
4020 K=SYMVAL(I)
MM=1
IBOX=2
IPLNT=1
   IF(EQUAL(FTEST,0,ARG(4),0,1)) GO TO 4021
ZERO=0
   MM=NCFD(ARG(4),ZERO)
   IF(EQUAL(FTEST,0,ARG(5),0,1)) GO TO 4021
IBOX=2
   IF(EQUAL(FTEST,0,ARG(6),0,1)) GO TO 4021
IPLNT=2
4021 CONTINUE
DO 4030 I=1,NN
   J=K+I*MM
   XPLT(I)=FLTS(DATTAR(J))
4030 CONTINUE
C SET SCALE FACTORS
XSCL(1)=0.0
XSCL(2)=1.0
XSCL(3)=1.0
XSCL(4)=1.0
C PAUSE BEFORE PLOT
PAUSE 30
   CALL PICTR(XPLT,1,XLAB,XSCL,1,NN,0,0,IBOX,-2,10,1,IPLNT)
   CALL HINT
GO TO 10
C END OF PROGRAM
END
C HYBOL SETUP MAIN=INITIALIZE FILES AND PARAMETERS
REAL*8 PNAME(3)
REAL*8 PROGS(3)
REAL*8 OPNAME(40)
REAL*8 OPC(40)
REAL*8 ARG(10), BUFSYM, SYMTAB(100)
INTEGER*2 ZERO
INTEGER*2 OPVAL(40)
INTEGER*2 SYMV(100), SYMTYP(100)
IMPLICIT INTEGER*2 (A-Z)
INTEGER*2 PFLAG, NFILE1, NFILE2, TOPCOR, IFLAG, NYTP, NINP, NSYM, NOPTAB
C COMMON DATA BASE AREA
COMMON/CORES/READ, PROG, PRINT, PLOT, RUN, EOJ, CORE, SYMB, DUMP, LIST,
1 TIMEOP, LOAD, EXEC, SETPOT, MODE, ANVAL, TSC
COMMON/CTRL/PFLAG, NFILE1, NFILE2, TOPCOR, OPNAME, NYTP, NINP, NSYM,
1 NOPTAB, LOCNT, NSYMBAL
COMMON/OPCODE/OPCODE, OPVAL, OPERCD, BUFSYM, ARG
COMMON/PROG/PSTART, PCOUNT, PEND, NSTMT, PNUM, NDFC, ERRORS
COMMON/SYMB/SMTAB, SYMV, SYMTYP
COMMON PROGS
C ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **
DATA PNAME/'HYBOLMAN', 'HYBOLRUN', 'HYBOLSET/'
DATA OPNAME/'DEFINE ', 'FUNCF1 ', 'DELAY ', 'SETPOT ', 'ADD ', 'SUB
1 'SUM', 'MUL', 'COS', 'EXP ', 'GET
2 'SET ', 'PUT ', 'SENSE ', 'READ ', 'PRINT ', 'PLOT ', 'RU
3 'RUN ', 'END ', 'IN ', 'INTEGRAT ', 'CONVOL ', 'DELA ', 'FU
4 'FUNCF2 ', 'MODE ', 'SQR ', 'EOJ ', 'PROGRAM', 'DUMP ',
5 'MOVE ', 'SYMB ', 'ANVAL ', 'LIST ', 'LOAD ', 'RETURN ',
6 'EXEC ', 'TSC ', 'TIMEOP ', 'FUNCHYST/','/
C ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **
C FILE SPECIFICATION STATEMENTS
C CALL DFI TO OPEN FILES
C
CALL DFI('HYBOLFL1 ',ICART,1,IER)
IF(IER.NE.0) GO TO 9876
ICART=0
CALL DFI('HYBOLFL2 ',ICART,2,IER)
IF(IER.NE.0) GO TO 9876
ICART=0
CALL DFI('DAFILE ',ICART,3,IER)
IF(IER.NE.0) GO TO 9876
DEFINE FILE 1(100,3,1,IREC1)
DEFINE FILE 2(60,2,1,IREC2)
DEFINE FILE 3(10000,1,1,IREC3)
C ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **
C DATA INITI TIALIZATION
TOPCOR=10000
ZERO=0
IPCRDE=3
NFILER3=3
NFILER2=2
NFILER1=1
C CLEAR FILES 1,2,3 TO ZERO
DO 5 I=1,100
5 WRITE(NFILER1'I)ZERO,ZERO,ZERO,ZERO,ZERO
DO 10 I=1,60
10 WRITE(NFILER2'I)ZERO,ZERO,ZERO,ZERO
DO 15 I=1,TOPCR
WRITE(NFILER3'I)ZERO,ZERO
15 CONTINUE
C MARK FILE 1 WITH PROGRAM ID
WRITE(NFILER1'I)IPCODE
DO 3 I=1,3
3 PROGS(I)=P NAMES(I)
C SET UP CONTROL INFORMATION
NTYP=5
NINP=8
NOPTAB=40
NSYMAX=100
NSYMRL=0
Loan=34
TIMFOR=38
EXEC=34
READ=15
PROGRAM=28
PRINT=16
PLOT=17
RUN=18
EOJ=27
CORE=32
SYMBOL=31
DUMP=29
LIST=33
SETDOT=4
MODE=25
TSCL=37
ANVAL=32
C ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **
C TRANSFER DATA INTO OPCODE TABLES
DO 2 I=1,NOPTAB
OPCODE(I)=OPC(I)
OPVAL(I)=1
2 CONTINUE
OPVAL(40)=0
C SET UP PROGRAM CHAINING LINKAGE
CALL SETCN(PROGS,3)
C CHAIN TO W YBOL MAIN
CALL CHAIN(PROGS,1)
C END OF PROGRAM
CALL EXIT
END
SUBROUTINE CARDIN(ARG,PFLAG,OPFLAG,ERROR)
C SUBROUTINE TO READ,PRINT AND PROCESS A CARD.
REAL*8 ARG(10)
INTEGER*2 BUFFER(40),FIELD(40)
INTEGER*2 CARDI,CARDI,CARDLIT,CARDLITZ,OPFLAG,I,J,JJ,JK,KK,ASTRSK,BUFFER,
1 FIELD,CTEST,SEMICL,COMMA,ERROR,BLANK
INTEGER*2 NINP,NTYP,KS,IPSO,J,PFLAG
INTEGER*2 PARR,PARA
INTEGER*2 SLASH
LOGICAL EQUAL
DIMENSION CARD(80)
C DATA FOR CHARACTER TESTING
DATA COMMA,SEMICL/' ',',','/
DATA PARR,PARA/' ',','
DATA LITA/'A ',',LITZ/','/
DATA BLANK/' '/
DATA CTEST/,' '/
DATA NTYP/N5/5/
DATA ASTRSK/S '/
DATA SLASH/,' /
C ***********************************************************************
10 CONTINUE
   ERROR=0
READ(NINP,9001)(CARD(I),I=1,80)
WRITE(NTYP,9160) CARD
91A6 FORMAT(/,IX,'INPUT CARD = ',80A1,1,15)
PFLAG=0
OPFLAG=0
9061 FORMAT(70A1)
C IF / COMMENT CARD JUST PRINT
   IF(EQUAL(CARD(1),0,SLASH,0,1)) GO TO 10
   DO 15 I=1,80
      CARDI=CARD(I)
   C LOOK FOR A ;
   IF(EQUAL(CARDI,0,TEST,0,1)) GO TO 105
   IF(EQUAL(CARDI,0,PARR,0,1)) CARD(I)=COMMA
   IF(EQUAL(CARDI,0,PARA,0,1)) CARD(I)=COMMA
   15 CONTINUE
WRITE(NTYP,9390)
9390 FORMAT(1X,'NO SEMICOLEN TERMINATOR CHARACTER IN CARD INPUT!')
   ERROR=1
   RETURN
105 CARD(I)=COMMA
   CARD(I+1)=SEMICL
   I=0
100 I=I+1
   CARDI=BLANK
   CALL WRITE(CARD(I),0,CARDI,0)
   IF(I,N'T+1) ERROR=2
   IF(I,N'T+1) RETURN
C TEST FOR IMMEDIATE
C SKIP BLANKS
C CHECK FOR ALPHABETIC
    IF(EQUAL(CARDI, ASTRSK, 0, 1)) GO TO 101
    IF(EQUAL(CARDI, BLANK, 0, 1)) GO TO 100
    IF(CARDI*LT*LITA) GO TO 200
    IF(CARDI*GT*LITZ) GO TO 200
    GO TO 500
C ***************************************************
C SET * FLAG
101  OPFLAG=1
    GO TO 100
200  WRITE(INTYP, 9100) I, CARDI
C ILLEGAL INPUT IN FIRST ARG
9100  FORMAT(1X,'ILLEGAL CHARACTER IN CARD COLUMN', I2, '=> ', A2)
      IERROR=3
      RETURN
C ***************************************************
C HAVE PROGRAM HYBUL PROGRAM CARD
C PACK OUT JUNK BYTES
500  IF(I.FC+1) GO TO 510
       I=I+1
C CLEAR LEADING BLANKS
   DO 505 JJ=1, IJ
   DO 501 KK=1, 79
501  CARD(KK)=CARD(KK+1)
505  CONTINUE
510  CONTINUE
   CALL PACK(CARD, 80, BUFFER)
C GET FIELDS OUT
C MOVE ARGUMENT FIELDS FROM CARD TO ARRAY
   IPOS=0
   DO 520 K=1, 8
      CALL FIELD(BUFFER, BFIELD, IPOS)
      CALL MOVE(BFIELD, 0, ARG(K), 0, 8)
   CONTINUE
520  CONTINUE
C CLEAR BLANK ARGS
   DO 530 K=1, 8
      CTEST=BLANK
      CALL BYTE(ARG(K), 0, CTEST, 0)
      IF(CTEST*NE*BLANK) GO TO 530
      DO 540 J=K+1, 7
         ARG(J)=ARG(J+1)
      CONTINUE
530  CONTINUE
540  CONTINUE
C DUNE RETURN TO CALLING PROGRAM
      RETURN
END
EXTERN OP, SP
ENTRY INTERP
* HYBOL ASM INTERPRETER PHASE 2 REAL TIME PROGRAM EXEC.
* CALL INTERP(DATTab, PSTART, NSTMt, TSTEP, NCyCt, ...)
* INTERPRETER ENTRY
R0 EQU 0
R1 EQU 1
R2 EQU 2
R3 EQU 3
R4 EQU 4
R5 EQU 5
R6 EQU 6
R7 EQU 7
R8 EQU 8
R9 EQU 9
R10 EQU 10
R11 EQU 11
R12 EQU 12
R13 EQU 13
R14 EQU 14
R15 EQU 15
ONE EQU 1
ZERO EQU 0
TWO EQU 2
OPNUM EQU 12
ADNUM EQU 16
DATNUM EQU 12
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* SAVE REGISTERS FOR FORTRAN
INTERP STM 0, SAVEAR
LH R14, R15 LOAD LINKAGE ADDRESS INTO 14
SVC 2, MESSIN
SVC 2, SVC 204 SET STATUS FOR NC
* GET ARGUMENTS INTO REGISTERS
LH R1, 2(R14)
LH R2, 4(R14)
LH R3, 6(R14)
LH R4, 8(R14)
LH R5, 10(R14)
* STORE ARGS IN SUB
* GET VALUES OF ARGUMENTS
LH R2, P(R2)
LH R3, P(R3)
LH R4, P(R4)
LH R5, P(R5)
STH R1, TABLE
STH R2, PSTART
STH R3, NSTMt
STH R4, TSTEP
STH R5, NCyCt
* CONVERT ARGUMENT ARRAY INDEX VALUES INTO BYTE ADDRESSES
* CODE TO TRANSLATE INDEXES INTO ACTUAL BYTE ADDRESSES
* GET PROGRAM OFFSET
LHR R5, R1
LHR R6, R3  REG 6 = LINES OF CODE IN PROG
* BYTE OFFSET = (I-1)*2
SIS R2, R1
AHR R2, R2
AHR R1, R2
STH R1, PROGAD
* PSTART = PSTART OFFSET PLUS ADDRESS OF DATTAB
* DO ALL CONVERSION ON 3 ARGS AT ONCE
GETADD LH R2, R1
LH R3, A(R1)
LH R4, B(R1)
LH R9, C(R1)  LOAD OP CODE
LH R10, D(R1)
* CHANGE TO BYTES
* ADD=((I-1)*2+A(DATTAB)
SIS R2, R1
SIS R3, R1
SIS R4, R1
SIS R10, R1
AHR R2, R2
AHR R3, R3
AHR R4, R4
AHR R10, R10
* ADD A(DATTAB)
AHR R2, R5
AHR R3, R5
AHR R4, R5
AHR R10, R5
* STORE IN OLD LOC
STH R2, R1
STH R3, R1
STH R4, R1
STH R10, R1
XHR RR, RR LOAD A 0 INTO R
COMPAR LH R7, OPCODE+2(R8)  LOAD ADDRESS OF OP BLOCK
CLH R9, OPCODE(R8) COMPARE OP CODE TO TABLE
BE FOUND R8 4 INCREMENT TO NEXT OP CODE
BS COMPARE TEXT NEXT
FOUND R0 TH R7, 0(R1)  SAVE OP ADD IN OP CODE LOCATION
* TEST IF DONE
SIS R6, R1
R2S CONTINUE GO TO INTERPRETER SECTION IF DONE ADDRESS CONVERSION
* INCREMENT AND DO NEXT LINE
AIS R1, R1
B GFTADD

* START INTERPRETER

CONTINUE EQU *
LH R12,NCYCL
LIS R10,1 LOAD FIRST TIME STEP VALUE INTO 10
* STOP AND WAIT FOR CLOCK TO RESET
LHI R1,X'6A' LOAD CLOCK ADDRESS INTO 1

CWAIT SSR R1,R2 SENSE CLOCK STATUS
BNCS CWAIT_LOOP UNTIL CLOCK RESTETS

LHI 3,X'047'
LHI 2,X'089'
LHI 1,X'06A'
OCR 1,3
WWW 1,2

CYCLD EQU *
LH R13,PROGRAD LOAD PROGRAD START ADDRESS
SIS R13,10

* READ IN DATA FROM A/D BLOCK
* CODE TO READ A BLOCK OF A/D CONVERTERS
READAD LH R1,TABLE LOAD A(DATATAB) INTO R1
LHI R2,NUM LOAD NUM OF A/Ds
LHI R3,X'04' LOAD OPCODE TO READ IN ADC AND INCREMENT MUX
LHI R5,X'64' LOAD A HEX A4
LHI R6,X'6A' LOAD A HEX A6
XWR 7,7 ZERO REG 7

READY OCR R6,R5 SEND AN A4 TO DEVICE A6
WWW R6,R7 SEND CHANNEL NUMBER
LIS R9,10 LOAD A 10 AND COUNT DOWN TO WAIT 10 MICROSECONDS
WWW R9,10 WAIT FOR 20 MICROSECONDS

BAMS RWWAIT

READ OCR R6,R3 SEND AN 84 TO THE INTERFACE
WWW R6,R8 READ SCALED FRACTION INTO R8
NWR R8,X'FFC0'

STH R8,(R1) SAVE VALUE IN DATATAB
SIS R1,2 INCREMENT ADDRESS
SIS R7,1 INCREMENT CHANNEL #
SIS R2,1 SUBTRACT FROM LOOP COUNTER

WWW READ

* DONE READING BLOCK OF A/D IN

* TEST EQU *
* GET OP CODE,ARG1,ARG2,ARG3
* 9 INSTRUCTION OVERHEAD PER SET
SIS R13,10
LH R1,0(R13)
LH R2,2(R13)
LH R3,4(R13)
LH R4,6(R13)
LH R5,8(R13)
STM R2,ARG2 SAVE ADDRESSES
STM R3,ARG3 BECAUSE SOME OPS NEED THEM
LH R2,0(R2) GET VALUES OF ARGS FOR IMMEDIATEs
LH R3,0(R3)
BR R1 BRANCH TO OP CODE BLOCK

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* INTERNAL ARGUMENT STORAGE LOCATIONS *
A3VAL DC X'1A3000'
SVC204 DC X'10004'
DC X'140000'
TABLE DC X'0'
PSTART DC X'10'
NSTMT DC X'10'
PROGAD DC X'10'
TSTEP DC X'10'
SVCHNE DC X'10001'
ERROR DC X'100'
ERRORS DC X'100'
LINE DC X'10'
NCYCL DC X'10'
ARG2 DC X'10'
ARG3 DC X'10'
A2VAL DC X'142000'
MASKAD DC X'FFC0'
LOST DC X'100'
SAVEAR DS 32

* SVC 2 MESSAGE BLOCKS *
MESSIN DC 7
DC 10
DC C' ENTER INT'
MESSS DC 7
DC 6
DC C' DO OP'
MESSI DC 7
DC 10
DC C' READ ADGs'
MESSR DC 7
DC 10
DC C' WRITE DAs'
MESSCL DC 7
DC 10
DC C' CHECK CLK'

* OPCODE AND BLOCK ADDRESS TABLE *
OPCODE DC 5
DC A(ADDOP)
DC 6
DC A(SUBnp)
DC 7
DC A(MULTOP)
DC 12
DC A(GETOP)
DC 13
DC A(PUTnOP)
DC 8
DC A(DIVnOP)
DC 3
DC A(DELAY)
DC 19
DC A(ENDnPS)
DC 21
DC A(INTGRT)
DC 23
DC A(DELAYS)
DC 22
DC A(CONVLT)
DC 2
DC A(FUNC1)
DC 24
DC A(FUNCf2)
DC 30
DC A(MOVEOP)
DC 0
DC A(ENDnPS)
DC 40
DC A(INVERT)

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

* ADD TWO SCALED FRACtIONS

ADDOP AHR R2, R3
STH R2, Z(4)
B TEST

* SUBTRACT TWO SCALED FRACtIONS

SUBOP SHR R2, R3
STH R2, Z(4)
B TEST

* MULTIPLY TWO SCALED FRACtIONS

MULTOP LHR R9, R9
MHR R8, R9 MULTIPLY BY SF
SLA R8, 1
STH R8, Z(4) SAVE RESULT IN OUTPUT ARG LOC
B TEST

* DIVIDE TWO SCALED FRACtIONS

DIVDOP LHR RR, RR
SHR RR, RR
SRA RR, 1
MHR RR, RR DIVIDE BY SF
STH RR, Z(4)
B TEST

INVERT LHR R3, ARG3 GET LOC BACK
XHR R1, R1 CLEAR REG 1 TO 0
SLR R1, R2  SUBTRACT VAL FROM ZERO
STH R1, 0(R3)  SEND VAL OUT
B TEST

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

* PUT VALUE INTO BUFFER(NCYCL)
PTOP LH R3, ARG3
LHR R4, P10  GET TSTEP #
AHR R4, R4
AHR R4, R3
STH R2, P(4)  PUT VALUE IN BUFFER
B TEST

* PUT VALUE FROM BUFFER(NCYCL)
GETOP LH R3, ARG3
LH R2, ARG2
LHR R4, P10
AHR R4, R4
AHR R4, R3
LH R3, P(4)
STH R3, P(2)
B TEST

MOVEOP LH R3, ARG3  GET ADDRESS OF 3 BACK
STH R2, P(R3)  MOVE VAL TO LOC OF 3
B TEST

DELAYS EQ1 *
B TEST
CONVLT EQ1 *
B TEST

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

ERROR1 LH P7, 1
SVC 2, MESSAG
B TEST
MESSAG DC 7
DC 10
DC 'ILLEGAL OP'

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

* DELAY LINE = SIMPLE TYPE
* DELAY SUBROUTINE
DELAY LH R5, P(R4)  REG 5 IS LENGTH NN
LH R6, 2(R4)
LH R3, ARG3  GET LOC BACK
CLIHR R6, R5  COMPARE PNT TO NN
SNPS PNTROK  IF PNT SMALL ENOUGH GO TO OK
LIS R6, ONE
PNTROK LH R7, R6  SAVE PNT IN REG 7
AHR R6, R4
AHR R6, R4  ADD DISPLACEMENT FOR DATAR
AWI R6, 4  ADD 4 FOR OFFSET
LH R8, 0(R6)  GET DELAYED VALUE
STH R8, P(R3)  GET VAL DELAYED THEN PUT AT OP LOC
STH R2, P(R6)  PUT NEW VALUE IN OLD SLOT
AIS R7,1
STH R7,2(R4) SAVE PNT+1 IN PNT
B TEST

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

* ONE DIMENSIONAL TABLE LOOKUP

FUNC1 LH R3, ARG
AIS R4, 2 INCREMENT PNT TO DATA
FCOMP CH R2, P(R4) COMPARE X WITH X(I) IN TABLE
BE FEQUAL X HIT TABLE EXACT
BSM FCOMP IF X LT POINT OK
AIS R4, 4 INCREMENT TO NEXT ADDRESS
RS FCOMP

FUNCTION S1q R4, 4 SET 4 FOR X(I)
LH R5, 0(R4) LOAD X(I) IN 5
LH R6, 2(R4) R6 = Y(I)
LH R7, 4(R4) R7 = X(I+1)
LH R9, 6(R4) R9 = Y(I+1)
SHR R9, R4 R9 = R9* Y(I+1) - Y(I)
SHR R7, R5 R7 = X(I+1) - X(I)
SHR R2, R5 R2 = X(I)
MUL R8, R3 R9 = R9*(X - X(I)) RESULT IN R8
DIV R8, R7 R8 = R8 / X(I+1) - X(I) RESULT IN R9
AIS R9, R4 R9 = R9 + Y(2)
AIS R7, 0(R3) PUT AT OUT LOC
A TEST DONE SO RETURN
FEQUAL LH R2, 2(R4) GET Y(I)
STH R2, 0(R3) PUT AT OUT LOC
B TEST

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

V3276 DC 3276
FORTY2 DC 42
TEN DC 10
TWENTY DC 20

* 2 INPUT FUNCTION TABLE LOOK UP

FUNCTION LH R4, R5 SAVE TABLE ADDRESS IN 1
LH R5, R6 SAVE OUTPUT LOC IN 0
LH R5, R2 PUT X IN 5
LH R7, R3 PUT Y IN 7
MUL R4, TWENTY
MUL R6, TWENTY
LH R8, R4
LH R9, R4
AIS R4, 1 I = I+1
AUR R4, R4 I = I+2

* R5 NOW HAS COLUMN BYTE OFFSET
AIS R6, 17 J = J+10
LHR R7, R6 PUT OVER FOR *
MUL R6, FORTY2 J = J*42

* R7 NOW HAS ROW OFFSET BYTES
* XI YJ = LOC + ROW + COL
AHK R4, R1 ADD TABLE LOC
LHR R1, R2 PUT OUTPUT LOC IN 1
AHK R4, R7 ADD ROW OFFSET
* R5 NOW POINTS TO XIYJ VALUE LOCATION
LHR R1, R4 SAVE LOC IN 11
LHR R7, R8 OUT # INTO 7
MH R6, V3276 *RY +1 BUT NOT A SCALED FRAC
MH R8, V3276 *RY +1
SHR R2, R7 R2 = DELTA X
SHR R3, R9 R3 = DELTA Y
* CALCULATE FAKE END PNTS AND CENTER
LH R5, Z11 LOAD XI+1, YJ
SH R5, Z11 = XI, YJ
MH R4, R2 S.F. * DELTA X DIFF
DH R4, V3276 S.F. / BY +1
AH R5, Z11 + XI, YJ
LHR R8, R9 SAVE IN 8
LH R5, 44(11) LOAD XI+1, YJ+1
SH R5, 42(11) = XI, YJ+1
MH R4, R2 * BY DELTA X S.F.
DH R4, V3276 S.F. / +1
AH R5, 42(11) + XI, YJ+1
SH R5, R9 DELTA Z = Z(I+1) - Z(I)
MH R4, R3 S.F. * DELTA Y
DL R4, V3276 S.F. / +1
AH R6, R7 Z(I) + DELTA Z
STH R6, 0(R1) PUT AT OUTPUT LOC
B TEST
* INTEGRATOR BLOCK
INTRT LH R3, ARG3 GET ADDRESS
LHR R9, R2 SET UP FOR MULTIPLY
SHR R8, R8 CLEAR R8 TO ZERO
MH R8, TSTEP INPUT * DT
SLA R8, 1 SHIFT OUT EXTRA BIT
AH R9, 2(3)
ACH R8, 0(3) AND TEMP TO OLD INTEGRAL
STH R8, 0(3)
STH R9, 2(3) STORE NEW INTEGRAL IN OLD LOCATION
B TEST DONE RETURN
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 
ENDUPS EDU *
* WRITE OUT DATA INTO D/A BLOCK
* OUTPUT D/A BLOCK
LHI R1, X1401 LOAD A HEX 40
LH R2, TABLE
AHI R2, 32 SET STARTING ADDRESS OF INPUT BUFFER
XHM R3, R3 ZERO 3
LHI R4, DANI M
XHM R5, R5 ZERO 5
LHI R6, X18F LOAD A INC AND SEND COMMAND CODE
OC R1,A2
SEND AN A2
WHR R1,R6
PUT IN RESET JAM MODE
OCR R1,R1
SET CHANNEL VALUE
WHR R1,R6
WRITE FIRST ADDRESS
WRITDA OCR R1,R6
SEND SET COMM
WH R1,R(R2)
SEND VALUE
AIS R2 X 2 NEW ADD
SIS R4+1
DECREMENT LOOP
BNZS WRITDA
OC R1,A3
SEND AN A3 TO INTERFACE FOR TRANSFER COMMAND
WHR R1,R6
SEND JUNK
* DONT WRITNING BLOCK
AIS R10+1
ADD A ONE TO 10 TO COUNT STEPS DONE
SIS R12+1
SUBTRACT 1 FROM 12 TO SET CC FOR DONE
RZ DONE +F CC ZERO FINISHED RUN
LW1 R2,X '68' LOAD CLOCK ADDRESS INTO 2

* CODE TO SENSE CLOCK STATUS
SSR R2,R4
SENSE STATUS OF CLOCK PUT IN 4
RCS CHIT
MISSING PULSE, TOO LONG A PROGRAM
RIPS CHIT
CLOCK NOT RUNNING
CLOOP SSR R2,R4
TEST CLOCK STATUS
RCS CLOOP LOOP IF NOT DONE COUNTING DOWN
B CYCLON
RETURN AND RESTART PROGRAM IF NOT DONE
CHIT SVC P, MESSP
PUT OUT CLOCK ERROR MESSAGE
B CYCLON
MESSP DC 7
DC 18
DC C 'CLOCK LOST'

* ON INTERRUPT CHECK IF FINISHES PROG OK
* FINISHED RUNNING PROGRAM
DONE LH R1, ERRORS
BAL 15+5D
DC ?
LH RR,SAVEP RETURN REGISTERS FOR FORTRAN
AH R1+50(R15)
GET RETURN ADDRESS
BR R1
END