HYBRID PROGRAMMING: A COMPREHENSIVE "SOFTWARE" PACKAGE FOR HYDAC ®

INTRODUCTION

The discovery quite early in the "computer age" that a series of programming aids were needed to circumvent or reduce the tedious, time-consuming and expensive operations required in using digital computers — coding, program testing, interpreting results, etc — led to the development of automatic programming language systems, "canned" routines, and other prepared procedures variously referred to as 'software."

Software, in effect, permits the user to 'get at' the computer without 'getting in it'. Through the use of such programming aids, for example, it is possible to introduce to the computer a complete set of instructions regarding mathematical routines, numerical operations, logic and control schemes and data transfer or storage programs *without needing to know* the disposition of each octal subroutine, register, flip-flop, or storage cell. Indeed, the creation of this programmed interface between the user and the machine has made the task of digital computer programming easy for the non-computer-expert, and is responsible for the almost universal acceptance of the digital computer in the scientific field.

The Problem of Communication: Despite the several benefits accrued through the use of such software packages — increased computer utilization, reduced duplication of effort for general utility programs, minimized periods of operator training, etc — the introduction of this interface has erected a barrier which not only restricts the analyst from communicating with his computer model during computation but often limits the programmer himself from taking full advantage of the digital computer's special features.

In the analog computer field the reverse situation exists. No comparable "software" interface has developed and the computer user communicates directly with the machine. There is little need to preserve standard programs; the analyst/programmer is directly involved in building the computer model and maintains rapport with the model throughout the problem.

The Uniqueness of Hybrid Simulation: The role that software must play in hybrid simulation is predicated on the unique combination of digital accuracy and analog speed that is hybrid simulation. . .the combination that permits the use of the best features of each in a particular problem solution. As a result, "hybrid software" must ease the programmer's burden as does digital software so that full utilization can be made of the digital computer's memory, logic and resolution.

At the same time, hybrid software must bring the analyst closer to the model rather than isolating him from it. The interface "barrier" of digital software, which renders the digital machine essentially into a sealed "black box," must be lowered to allow the effective interaction of the analog and digital domains. Accordingly, hybrid software must include coding for the sequential computer to perform internal calculations and make decisions as well as control facets of parallel machine operations and also inter-connection diagrams and prewired patch panels for the parallel machines.

GENERAL REQUIREMENTS

The hybridization of digital and analog techniques into hybrid simulation has not, despite their union in use, altered the specific features of either method of computation. As a result, the capabilities of conventional digital software systems, although useful in hybrid computation, must be extended to provide programming aids for the entire system.

Compilers and Assemblers: Conventional compilers and assemblers which, in digital systems, provide automatic conversion from a symbolic to a machine language program, have a useful function in hybrid simulation as well, but differ in their utilization in four ways:

(1) Program running time is minimized at the expense of compiling time.

Bulletin No. ALHC 64043

Printed in U.S.A. 114

(2) Actual running time for each program statement is pre-calculated or estimated to aid the programmer with the timing aspects of the problem.

(3) Since the programmer must be able to use special machine features in order to program for control of all machine interface operations, a machine-dependent programmer's language is utilized in most cases.

(4) The programer's language is extended to all sections of the hybrid system to aid in programming not only the digital, but also the machine interface and analog sections as a system entity.

Debugging Aids and Diagnostics: Conventional software aids for debugging a digital program include memory dump, updater, and tracer routines; diagnostics usually check out the proper operation of the digital processor hardware. Hybrid software must extend these capabilities for the entire system and provide additional features as follows:

(1) Analog Section Diagnostics — a symbolic program which accepts a problem-oriented language description of the analog flow diagram and automatically produces (a) setup documents (pot sheets) for the analog run, (b) digital static check calculations for the analog program, and (c) possibly an analog program dynamic check calculation.

(2) Logic and Data Transfer Diagnostics — a digital program which is used to ascertain that the linkage, interface, and logic hardware are operating properly and checks out (a) the bidirectional data flow, (b) the bidirectional control signal flow, and (c) the logic and secondary storage components, etc.

Appendix A, Figure 1 shows an excerpt from a representative diagnostic check.

Subroutine Library: The conventional subroutines prepared for a general purpose hybrid computer include:

(1) Mathematical routines (log, ln, exponential, sine, cosine, tangent, arctangent, square root).

(2) Computational routines (single precision floating point, double precision floating point, double precision fixed point).

(3) Conversion routines [(double and single precision) BCD/fixed point, BCD/floating point, fixed point/floating point)].

(4) Input/Output (I/O) routines (typewriter, paper tape, magnetic tape, automatic formatting on output).

(5) Algorithms (a selection of numerical algorithms: integration methods, etc.).

(6) Models (subsystems, environments, etc.).

With regard to Item (5), Algorithms, there is frequent need in a hybrid system for various numerical operations such as definite integrals, the solution of differential equations, etc.; in addition, subroutines mechanizing various integration methods (Runge-Kutta, Adams, Milne, Euler, etc.) have been constructed. These utility routines permit the programmer to choose the method exhibiting the best speed and accuracy for his particular problem.

With regard to Item (6), Models, there is frequent need in a hybrid system for useful subsystem models that have become standard and are used in larger models (specific transfer function routines for use in a typical servo controller, for example, or a function generator program for any number of aerodynamic functions). A standard program for a complete aerodynamic vehicle simulation is of value in many aerospace computing laboratories. These utility routines ease the programmer's task by permitting use of standard models and eliminating duplication of effort.

HYDAC 2400 SOFTWARE REQUIREMENTS: THE EAI SOFTWARE PROGRAM

The EAI Software Program outlined in the block diagram of Figure 1, is the comprehensive hybrid software 'package' offered by EAI to provide the users of the HYDAC 2400 Computing System with the necessary programming tools.

The three essential elements of a hybrid computing system are the parallel computer(s), the sequential computer, and the software or programming systems that transform the former two into an operable simulation tool. Of great importance to the effective use of a hybrid computing system is the software "man-machine interface."

The parallel engineer-oriented aspects of the general purpose analog computer -- pioneered by EAI with the PACE® 231-R Computing System -- have required little of this type of support in the past.

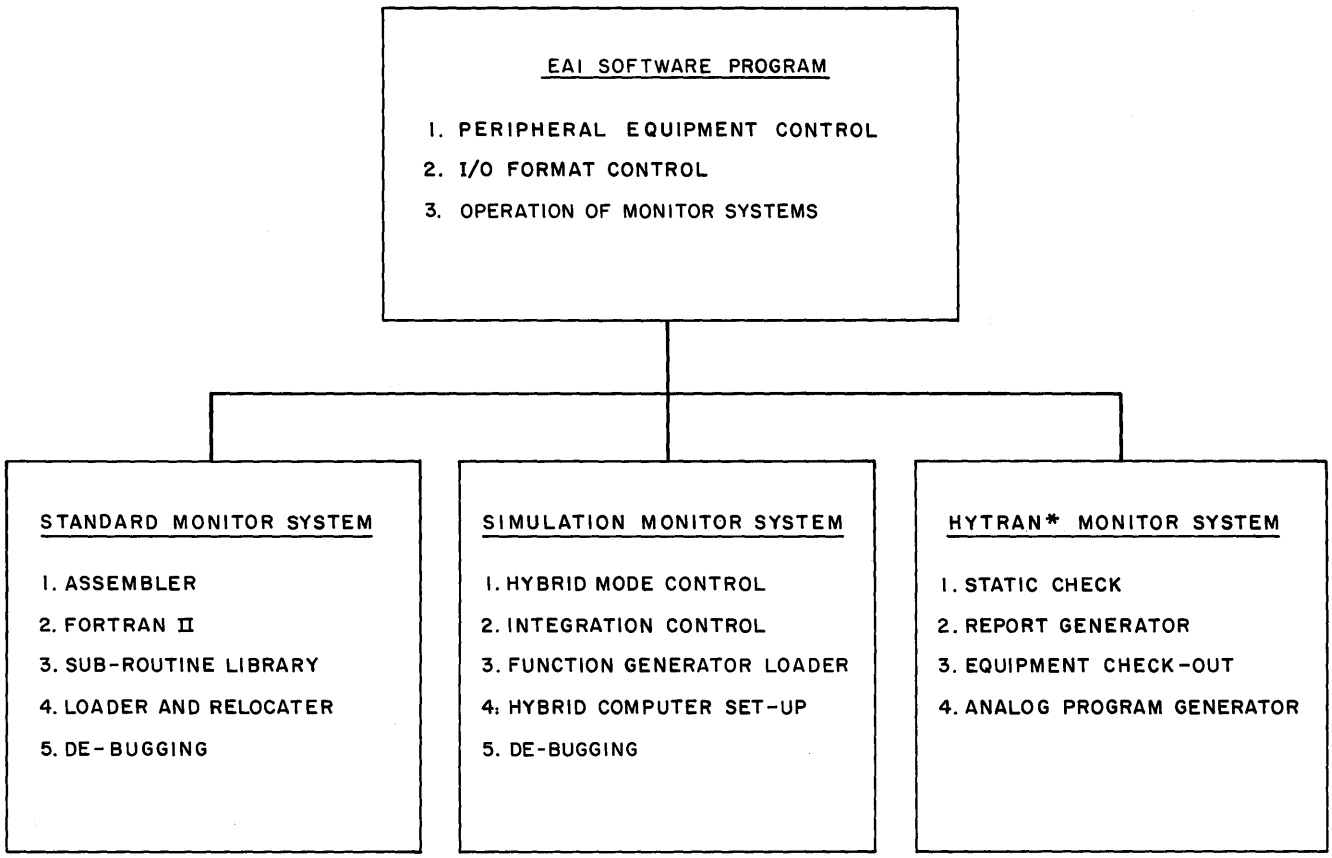


Figure 1: Block Diagram of EAI Operating System

With the introduction of the HYDAC 2400 hybrid computing system, however, having, as a basic component, a general purpose digital computer, EAI has developed the necessary hybrid software capabilities in such a way that full advantage is taken of the unique capabilities of both analog and digital techniques.

In addition, by recognizing the communication difficulties posed by the conventional digital software approach, EAI has formulated this software package to reflect the parallel engineer-oriented aspects of the analog computer. To facilitate this, it becomes necessary only to remember the three phases of problem implementation on the computer. Phase one involves program preparation and is directly related to phase two, program checkout. The third phase consists of simulation operation. It is to these three aspects (program preparation, program checkout, simulation operation) of problem solving in hybrid computation that the EAI Software Program is directed. The purpose of this program is to put the user in control of his model, the computers, and the results of the simulation study. Thus the Standard Monitor System (sequential program preparation and checkout), the Simulation Monitor System, and the HYTRAN Monitor System (parallel program preparation and checkout) operate in such a way as to maintain the integrity of the system.

The following sections will describe the elements of the EAI Software Program.

STANDARD MONITOR SYSTEM

The Standard Monitor System is a complete parallel entity within the EAI Operating System which provides basic digital software for the EAI Series 375 Digital Computer (DDP-24).

1. *Assembler:* This is a metaprogram† which enables the user to write in symbolic machine language. A resultant translation provides the user with an edited listing of his original program and a condensed version of the code on punched paper tape. Several parts of a program may be "assembled" at different times to be linked together later and loaded automatically.

*A Service Mark of EAI.

† A Metaprogram is a program used for the production of other programs.

(a) DAP (Digital Assembly Program) Assembler: a metaprogram based on the SHARE standard assembler.

(i) DAP - Off-Line Output Program punches paper tape instead of listing program on the typewriter during assembly. Paper tape can be used for typewriter listing on standard tape preparation unts or on existing ADIOS* desk units. Procedure reduces digital computer time required for on-line program assembly.

(ii) ADIOS/DDP Conversion Program permits use of ADIOS desk for "off-line" tape preparation, eliminates time-wasting digital computer tape preparation procedures.

(iii) DAP Time Pseudo Operation Program provides estimate of actual machine time required to execute commands as they occur in user's program, provides basis for allocation of tasks in any hybrid simulation.

2. *FORTRAN II:* This is an engineering/ algebraic compiler for flexible, automatic formatting of input-output programs, which is easy to learn, use and debug. Object programs are compatible with DAP condensed output tape format.

(a) In-Line Machine Language Coding with FORTRAN: The 375 FORTRAN Compiler includes special provision for real-time and hardware-oriented programming. A capability to write DAP-like entries "in-line" with the algebraic statements gives the programmer a control over computation not afforded by any competitive FORTRAN System. This ability is a necessity if any compiler metaprogram is to be used to generate the codes for a hybrid computation.

(b) Expanded Features of 375 FORTRAN: Although compatible with the generally-accepted FORTRAN II compiler, the 375 FORTRAN compiler has been extended to include the most desirable features of the new FORTRAN IV version as defined by IBM for its large scale computers.

3. *Sub-Routine Library:* This is a set of subroutines which perform useful or standard calculations or manipulations which can be incorporated easily into the user's own program. This library tends to grow as more and more diversified applications are programmed. A written procedure describing the requirements, inputs, outputs, range of computation, and limitations of each subroutine accompanies a paper tape version (either in symbolic DAP, FORTRAN, or machine code) of the subroutine.

(a) Basic Digital Sub-Routines:

(i) Computational. . .

ADDX, SUBX, MPYX, DIVX: double precision fixed point

FADD, FSUB, FMPY, FDIV: single and double precision floating point

POLY: odd polynomial evaluation (used by many mathematical subroutines)

(ii) Mathematical. . .

(a) fixed point, single and double precision

SQRX, SQDX: square root

EXPX, EXDX: exponential

LG2X, DPL2: logarithm, base 2

LGEX, DPLE: logarithm, base E

LGAX, DPLT: logarithm, base 10

SINX, SNDX: sine X, input in radians

COSX, CODX: cosine X, input in radians

*A Trade Mark of EAI.

TANX, TNDX: tangent X, input in radians

ATNX, ATDX: arctangent X, output in radians

(b) floating point, single and double precision

FSQR: square root

FEXP: exponential

FLG2: logarithm, base 2

FLGE: logarithm, base e

FLGA: logarithm, base 10

FSIN: sine, input in radians

FCOS: cosine, input in radians

FTAN: tangent, input in radians

FATN: arctangent, output in radians

(iii) Conversion (single and double precision). . .

XBXD: binary-to-decimal, fixed point

XDXB: decimal-to-binary, fixed point

FBFD: binary-to-decimal, fixed to floating point

FDFB: decimal-to-binary, floating to fixed point

FBXD: binary-to-decimal, floating point

FDXB: decimal-to-binary, floating point

(iv) Input-Output. . .

PTIN, PTOU: paper tape I/O

TYIN, TYOU: typewriter I/O

Routines for on-line printer, magnetic tape, light pen/scope, and disk also are available.

(v) Algorithms. . .

(a) Integration

- Euler
- Heun
- Milne
- Runge-Kutta
- Adams

(b) Function Generation

Appendix A, Figure 2 shows an excerpt from a representative Model subroutine.

4. ***Loader and Relocator:*** The Loader performs the basic operations of loading absolute and relocatable DAP and/or FORTRAN object (binary output) taped programs in memory, scanning the library tape automatically for standard subroutines, and creating linkages between the subroutines and the main program. (The program also records storage allocations and, through the Relocater, makes the memory assignment or avoids inadvertent destruction of stored information by typing an error message when a relocatable program is loaded. In addition, the Loader/Relocater can be instructed to relocate a program already in memory.)

5. ***De-bugging:*** This is a program-testing communications package for on-line typewriter and console-controlled de-bugging. The program can be used to call for:

   a) point-to-point tracing of a program
   b) clear, dump, and search of memory
   c) display and entry of instructions and data

6. ***Symbolic Addressor:*** This routine provides expanded flexibility to the Loading program and De-bugging routines described above by making use of a symbol chart generated by the assembler or compiler and placed in memory with the object

program. Thus, loading, relocation and de-bugging operations all can be performed with references made to symbolic addresses.

SIMULATION MONITOR SYSTEM

The Simulation Monitor System, illustrated in the flow diagram of Figure 2, is another complete parallel entity within the EAI Software Program which, although it utilizes some of the routines described under the Standard Monitor System, has control of a group of programs designed specifically for more effective control and operation of either all-digital or hybrid scientific situation programs:

1. *Hybrid Mode Control:* This is the program that gives the digital computer the structure of a simulator in like manner to an analog console by exercising interface capabilities (including interrupts, sense lines, delays, and mode control) to achieve real-time synchronization between the stored and the parallel computer programs, and between the various computing system elements needed in the solution of most simulation problems. The program controls a dual-processing mode by which it is possible to work on a second program located in a protected portion of memory when the computing system is not employed on the primary simulation program. (The monitor permits dual processing when the simulation program is not in the "Operate" mode.)

Appendix A, Figure 3 shows an excerpt from a representative linkage program.

2. *Integration Control:* The Digital Integrator Control Program is one that is requested from the monitor by the operator to change the integration algorithm. Selection of algorithms is made from the Subroutine library; initialization and time scale changes can be made also.

3. *Function Generator Loader:* The Digital Function Generator Loading Program permits the operator to load new data or to make changes in data previously stored in tables associated with the particular function generation subroutine. Optionally, "raw" experimental data may be loaded by using the interpolation feature to generate tabular data from it.

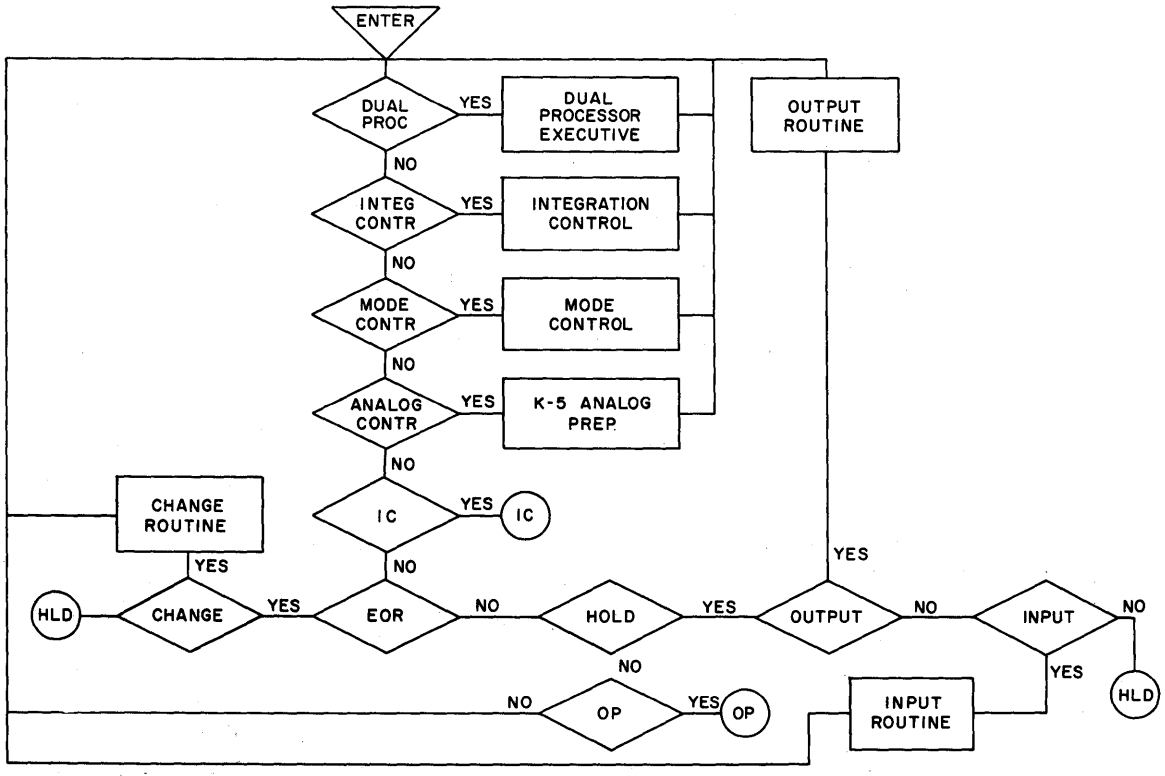


Figure 2: Flow Diagram of Simulation Monitor System

4. *Hybrid Computer Set-up:* The Hybrid Computer Set-up and Check Out Program, selected by typewriter through the monitor, provides the capability for control of analog computer mode and time scale selection, component selection, read-out, pot set and checkout as follows:

(a) Digital-Analog Control. . .

(i) K-5 Input-Output (I/O) Group

- PTIN (CR) — transfers control to the paper tape reader

- OP (CR)
- IC (CR)
- HLD (CR)
- ST (CR)
- PS (CR)

  } mode control commands for the analog computer

- CS (T) X (CR) — select console # X

- LTAB (T) AAAAA-C, AXX-YY, MX'X'-Y'Y', # (CR) — load table of addresses at AAAAA [octal], column C [octal], consisting of amplifiers AXX through (YY-1), etc.

- LTAD (T) AAAAA-C, XX.XX, YY.YY, etc., # (CR) — load table with data

- SETT (T) AAAAA-C, T, .XX (CR) — set table of pots in table starting at AAAAA, column C, type (punch) out results, if setting differs >.XX print error (P)

- STCK (T) AAAAA-C, T, .XX (CR) — check column C data vs. state of machine (P)

- SCAN (T) AAAAA-C, T (CR) — readout component values of column C (P)

- DTAB (T) AAAAA, X, T, .XX (CR) — dump complete table at AAAAA on paper tape for set-up and static check "next time"

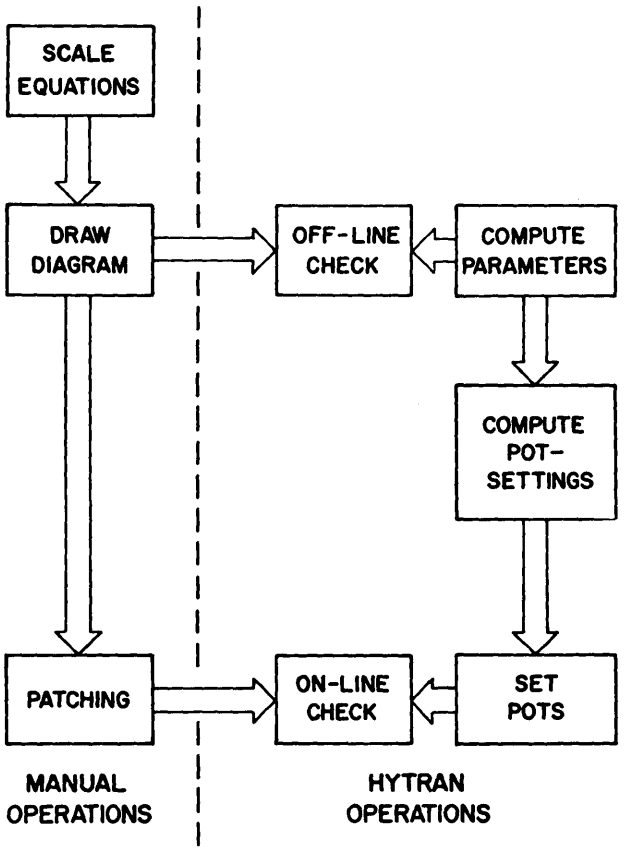


Figure 3: Programming an Analog Problem with HYTRAN

- SET (T) PX (SP) .YYYY, (P) T, .ZZ (CR)— set pot PXX to coefficient YYYY and type (punch) out results; indicate error if differs >.ZZ

- CHK (T) AXX (SP) YYY (P) .YY, T, .ZZ (CR) — check amplifier XX for YYY.YY volts to .ZZ accuracy (P)

- RD (T) MXX, P (CR) — read multiplier XX

- SETG (T) QXX (SP) AYY, (P) T, .ZZ (CR) — same as Set except in G mode set to value of AYY

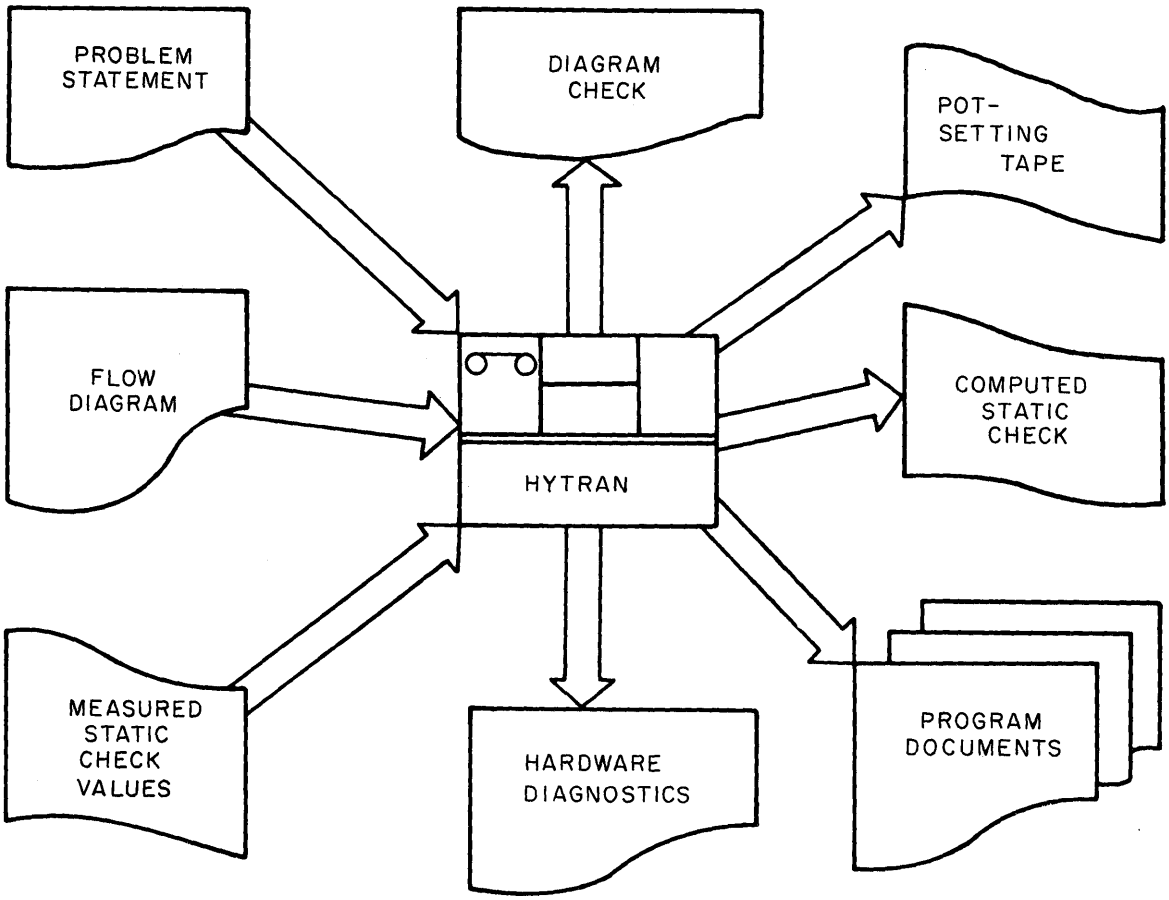


Figure 4: Outputs Obtained from the HYTRAN System

5. *De-bugging:* The Hybrid De-bug Program provides for typewriter control of:

   a. K-5 Input-Output (I/O) Group
   b. Interface ADDA
   c. DDP De-bug
   d. Read Out and Load Memory

HYTRAN MONITOR SYSTEM

The HYTRAN Monitor System is a family of meta-programs being implemented on the HYDAC 2400 Hybrid Digital-Analog Computer as a complete parallel entity of the EAI Operating System to provide digital computer assistance in the programming of its analog system. The scaling of the physical equations and the preparation of the computer diagram are still performed by the programmer who thereby maintains direct control over the analog implementation of the problem. In order to permit calculation of theoretical static check values, HYTRAN also must be given the original problem statement and a set of test initial conditions. Additional data, including patching information, component setting or modes, highest derivatives, and any other expressions representing other appropriate component outputs must be provided as well. In this way, the necessary rapport between the programmer and the machine is kept. Figure 3 shows the steps required to program an analog program with the HYTRAN System.

Program input is punched on paper tape in an analog-oriented language compatible with the Series 375 Digital Computing System. The HYTRAN System then computes potentiometer settings, and both a physical and voltage static check which are tested for consistency. Complete documentation of the analog program is produced (including potentiometer, amplifier, and cross-reference sheets as well as automatic pot-setting and static test tapes) in the ADIOS format. An On-Line Diagnostic Generator checks measured static check voltages against the analog circuit diagram and the specifications of the analog components, providing a rapid means of locating patching errors or component failures.* HYTRAN outputs are shown in Figure 4.

**To run these basic programs, the digital section of the HYDAC 2400 Computer should include three index registers and an 8K memory. With an 8K core, each program can be contained in memory and enough data storage is still available to process an analog computer program requiring three one-hundred-and-twenty-amplifier systems. In general, there are no special requirements on the analog computing system(s) since the programs cover a wide range of component configurations.*

1. *Static Check:* The practice of computing two independent sets of check values has been used as the basis for the HYTRAN *Off-Line* Static Check. The theoretical static-check values in volts are computed from expressions, provided as part of the program input, which specify component outputs in terms of the scale factors, parameters, and variables of the problem. Further input defines the analog component interconnections or patching information which is used to calculate the voltage check values. In this computation, all input voltages to a component, the kind of input to which the component is connected, and the transfer function of the component are used to determine its voltage output.

   When both voltage and theoretical values are available for a component output they are compared. If in agreement, they yield the off-line static check value for that component. If the values are not in agreement, the error is isolated by retaining the theoretical value as the static check input for all subsequent calculations and an error message is given. Values exceeding the voltage range of the computer also will cause error messages but will be retained for further static check calculations.

   The *On-Line* Static Check: While in systems without digital access to the analog computer, the on-line static check must be performed by manual comparison, the availability of a digital input/output system provides the HYTRAN user with a choice of two automatic procedures for such on-line checking. One method is to feed the HYTRAN-generated static-check tape into the ADIOS desk to obtain an automatic comparison between calculated and measured values. This method is used whenever the digital computer is not available at the time of analog on-line check.

   If the digital computer is available, the use of the second HYTRAN method allows an improved consistency check for debugging of complex problems as well as for preventive-maintenance checks. This method is implemented by feeding an ADIOS tape containing the measured pot settings and/or outputs of all components into the digital computer. HYTRAN then checks the transfer value of each individual component and compares it with the measured voltage at the component output. Thus, errors can not propagate but are pinpointed at the component level.

2. *Report Generator:* This is a Documenting Program which sorts and converts the information from the intermediate tape to component work sheets that contain a list of the analog computing components in an orderly sequence, together with their modes and their outputs or settings in terms of problem parameters, variables, and scale factors. In addition, an alphabetic list of the values of parameters and variables is typed out, and ADIOS tapes are generated.

   One tape contains the potentiometer settings in a format which allows automatic pot setting; the other tape contains the computed static-check values for on-line check with the ADIOS desk.

   The Documenting Program also generates a cross-reference sheet containing an alphabetic list of symbols for parameters and variables. Each of these symbols is followed by a listing of the components whose output or settings include that symbol. This feature provides assistance in changing parameters or scale factors manually.

3. *Equipment Check-out:* This is an On-Line Diagnostic Program utilizing basically the same processing procedure as that of the Off-Line Static Check Generator, described as item 1 above, in the HYTRAN Monitor System. The program accepts an ADIOS tape containing a complete read-out of all static-check voltages and potentiometer settings and generates diagnostics which serve to locate analog components which are either improperly patched or do not perform satisfactorily. Thus, a permanently-programmed analog computer pre-patch panel could be used with this program for daily preventive maintenance checks.

4. *Analog Program Generator* (A proposed further development): This is a program which performs the initial step of reducing equations to a useful analog computer schematic. The three programs described above are designed to process digitally the rather tedious static check and documentation routines of analog programming. The Analog Program Generator goes a step beyond these routine data-checking operations and is utilized directly in writing programs.

Although the digital computer makes use of a special internal code, or machine language, for processing information, the difficulties encountered in writing programs directly in machine coding have resulted in the development of special programming languages. Information and instructions are written first in the programming language and then translated by the computer into coding which the machine can understand. The compiler, or set of instructions, which translates this source program into a machine language (object) program also can perform other clerical functions to improve the efficiency of the procedure and relieve the programmer of many routine tasks. The Analog Program Generator System, therefore, makes it possible to write programs without a detailed knowledge of the internal organization of the computer.

A further convenience is achieved by orienting the programming language toward the procedure rather than the machine. Programs then may be written in a language which closely resembles that of the programmer and his specific area of computer application. For example, the FORTRAN (FORmula TRANslator) programming system includes many statements which resemble algebraic formulae to facilitate programming numerical procedures. The mathematician then may communicate more closely with the computer by programming in FORTRAN rather than in a business-oriented or other programming system.

While FORTRAN programming is suitable for expressing numerical or logical relationships, many of the statements are too general for programming specialized operations conveniently. To improve this situation, a special programming language has been defined for the HYTRAN System which allows the analog computer programmer to write digital computer programs in terms and expressions familiar to him. The HYTRAN language is designed specifically to be compatible with the Series 375 Digital Computing System (3C DDP-24) which forms the arithmetic section of the HYDAC 2400 Hybrid Computer. The HYTRAN language includes statements which are oriented specifically toward analog computer programming, and which make use of the EAI 231R Analog Computer nomenclature in identifying components, modes, etc.

The HYTRAN inputs fall into two main categories: a) inputs which describe the problem to be solved, and b) inputs describing its implementation on the analog computer.

a) The Problem Statement. The problem usually is stated in mathematical notation. The HYTRAN program, therefore, accepts the problem statement in a mathematically-oriented language which bears resemblance to the programming languages ALGOL and FORTRAN.

The problem information to be inputted includes parameter values, variable initial conditions, and a set of algebraic and differential equations — all in terms of physical units. Parameters can be defined by expressions containing numerical values as well as other parameters, while initial conditions of variables can be given in terms of numerical values, parameters, or other initial conditions. All algebraic and differential equations have to be in explicit form with respect to the unknown variable, or the highest derivative, respectively. Otherwise, the equations should be inputted in their original form so that any errors that may occur during further manipulations on the problem equations will show in the theoretical static check.

HYTRAN accepts expressions and equations containing not only the basic algebraic operations, but also the functions *sine, cosine, arc tangent, square root, logarithms* to the bases

ten and *e*, and the *exponent of e*, In addition, the program processes certain discontinuous functions — such as absolute value, sign, limits, and dead zone — which are often used in analog computation.

Relational operators are another means of obtaining step-functions. They are represented by the relations *less than* and *greater than* (for practical purposes, *equal to* does not exist in analog computation). While a logical *AND* can be performed by multiplication, the *OR* is an explicit Boolean operator in HYTRAN.

b) Circuit Diagram Information. An analog program is generated in the form of an analog circuit diagram by which the programmer states the outputs of components, their modes, their interconnections (patching), and, in the case of potentiometers and switches, their setting or position. Inputting this information enables HYTRAN to check the analog program against both the original problem input and the physical set-up on the analog computer.

The general form of a component statement is:

DESIGNATION, MODE = EXPRESSION; CONNECTION

(A console number is given only when a change from one console to another occurs.)

Where

designation generally consists of an identifying letter and two decimal digits,

mode designates the configuration in which a component is used, or any special connections not involving problem variables (for example, amplifiers may be connected in integrator, summer or high gain mode),

expression, in the context of component statements, is a term or collection of terms giving, through problem variables and scale factors, the output of an analog component, and

connection is defined as a sequence of up to thirty-two input statements, each of which consists of an input designation (usually identical with the pre-patch panel input name) followed by the designation of the analog component which is connected to the specified input.

All computer input data is entered from punched paper tape, each type of information being preceded by one of the following keywords: PARAMETERS, VARIABLES, EQUATIONS, COMPONENTS. The keyword sections must be presented in the order given below, although within any section the statements can be in arbitrary order:

i) The PARAMETERS and VARIABLES keywords are followed, respectively, by parameters as used for the static check, and all variable initial conditions. Parameters and variables may be referred to by mnemonic names which, in turn, can be defined in terms of other parameters, initial conditions, or by numerical values. A name thus defined need not be referred to beforehand, but must be specified within the same keyword section.

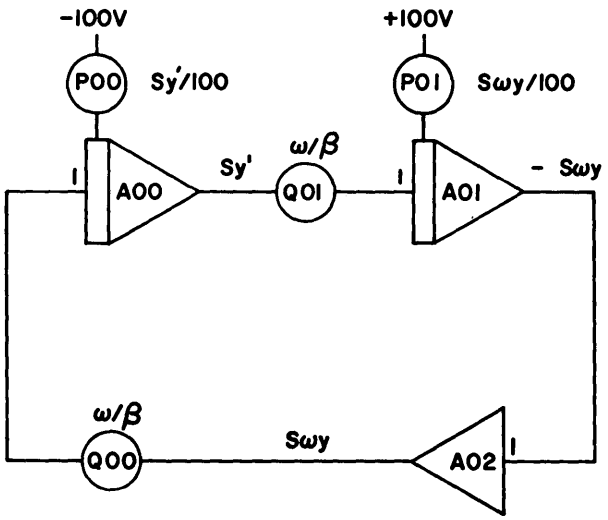
ii) The EQUATIONS section contains the set of theoretical differential and algebraic equations to be implemented on the analog computer.

iii) The COMPONENTS keyword is followed by the patching information contained in the circuit diagram which represents the analog program. These analog patching connections, by means of which the prob- is to be implemented, are entered in any sequence.

As a simple example of how input information to the HYTRAN program is written, consider the case of a second order equation as shown in Figure 5a. The inputs shown in Figure 5b should be provided if a complete check is desired. Note that comments pertinent to the program input, but meaningless to the digital program, can be inserted if preceded by a tab. The input begins with the problem identification, which contains any information the programmer wishes to use as a heading for all typewriter outputs. The resulting outputs are shown in Figure 6.

The HYTRAN System can increase programming efficiency and justify a high degree of confidence in the analog solution for the following reasons:

1. The automatic evaluation of algebraic



ANALOG IMPLEMENTATION OF

$$\frac{d^2y}{dt^2} + \omega^2 y = 0$$

Figure 5a: Sample Second Order Differential Equation

expressions saves programming time and prevents arithmetic errors.

2. The generation of pot-set tape saves time and eliminates the errors which could occur when transferring the numerical settings manually from a desk calculator to the I/O keyboard.

3. The documentation generated provides a complete, error-free, standard documentation of analog programs. Cross-referencing speeds the changing of parameters and scale factors.

4. The off-line static check provides a check of every component in the computer. Automatic static check calculations save time and are error-free. The checking of the component level permits the omission of the check of any selected portions of the computer diagram, such as circuits that must be checked dynamically or that are considered standard routines. Such an omission prevents the checking of other components in the same algebraic chain of components.

5. Simple changes of connections, parameters, or scale factors may change the majority of the static-check values and pot-settings in a program, but only a

TEST PROBLEM #1: 2ND ORDER EQUATION

PARAMETERS

S = 5 S = SCALE FACTOR

A = 20 A = AMPLITUDE

OMEG = 10

BETA = 10 BETA = β = TIME SCALE

TO = 1/10

VARIABLES

Y' = A*COS (OMEG*TO)

Y = A*SIN (OMEG*TO)/OMEG

EQUATIONS

Y'' = -OMEG**2*Y

COMPONENTS

P00 = S*Y'/100; +REF

A00, I = S*Y'; IC:P00

C00 = S*Y''/(BETA*10); 1:Q00

Q01 = OMEG/BETA; A00

P01 = S*OMEG*Y/100; -REF

A01, I = -S*OMEG*Y; IC:P01

C01 = -S*OMEG*Y'/(BETA*10); 1:Q01

A02, S = S*OMEG*Y; 1:A01

Q00 = OMEG/BETA; A02

Figure 5b: Input Information Necessary to Program Sample Equation

simple change is necessary on the HY-TRAN input tape in order to generate a complete, updated set of ADIOS tapes and documents. Obviously, when a new static check is not required, changes can be made in a conventional way and there is no need to update the input tape for every minor change.

TEST PROBLEM #1: 2ND ORDER EQUATION

STATIC CHECK VALUES

A00 = 53.98

A01 = -84.18

A02 = 84.18

C00 = - 8.41

C01 = - 5.40

P00 = -53.98

P01 = 84.18

Q00 = 84.18

Q01 = 53.98

PARAMETERS

A	=	2.00000 E1
BETA	=	1.00000 E1
OMEG	=	1.00000 E1
S	=	5.00000 E0
TO	=	1.00000 E-1

VARIABLES

Y	=	1.68367 E0
Y'	=	1.07963 E1
Y''	=	-1.68367 E2

COMPONENTS

COMP	MODE	EXPR	SETT
A00	I	A*S*Y'	
A01	I	-A*S*Y	
A02	S	A*S*Y	
P00		A*S*Y'/100	.5398
P01		A*S*OMEG*Y/100	.8418

Figure 6: Typical HYTRAN Outputs from Complete Check of Second Order Differential Equation of Figure 5a.

COMP	MODE	EXPR	SETT
Q00		OMEG/BETA	1.0000
Q01		OMEG/BETA	1.0000

CROSS REFERENCE

PARAM	OCCURRENCE
A	P00, P01
OMEG	Q00, Q01
S	P00, P01
Y	P01
Y'	P00

Figure 6: Typical HYTRAN Outputs from Complete Check of Second Order Differential Equation of Figure 5a. (Continued)

6. When the on-line static check is performed by the ADIOS desk, the computational sequence of the static-check values on the HYTRAN-generated tape eliminates tracing for the error sources.
7. The use of the on-line diagnostic program allows pin-pointing of errors on the component level, even for closed algebraic loops of unknown output.
8. In conjunction with a permanently-wired test problem, the on-line diagnostic program can be used for daily maintenance checks.

Manpower savings resulting from these features far outweigh the additional effort involved in preparing the HYTRAN input tape, a task comparable in size to that of preparing potentiometer and amplifier assignment sheets manually.

The system as described can be expanded easily to include new programs (such as a Digital Check Solution, for example), some of which may evolve from the practical use of the present system. Since any such new programs would require little additional input information, their benefits would be available at little extra cost and would, therefore, further increase the overall economy of the system.

CONCLUSION

The EAI Operating System is a comprehensive master program for the HYDAC 2400 Hybrid Computing System prepared to meet the needs of a scientific simulation laboratory. It is designed for use with standard peripheral equipment, and provides program control of all I/O operations, multiprocessing operations, and programming.

APPENDIX A

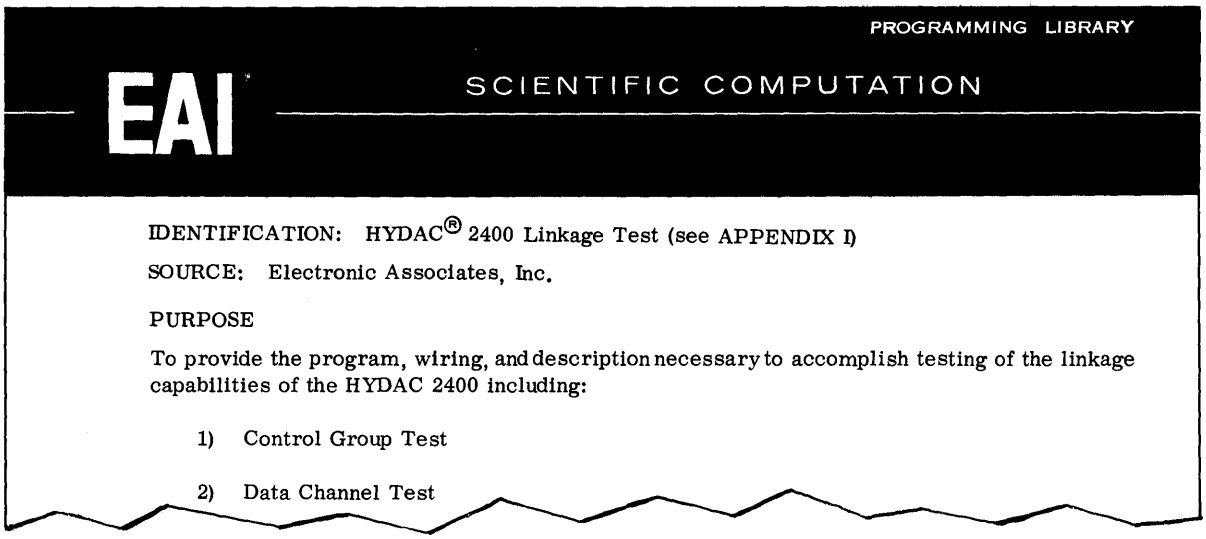


Figure 1: Excerpt from Representative Diagnostic Check

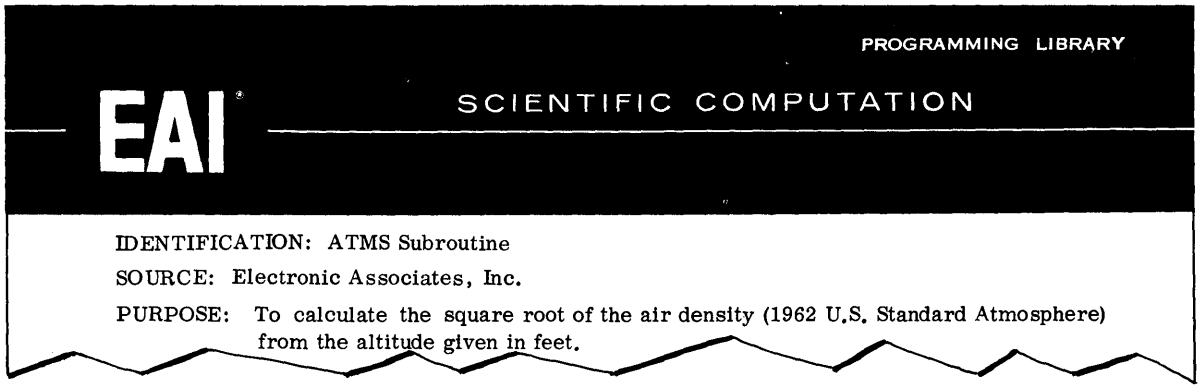


Figure 2: Excerpt from Representative Model Subroutine

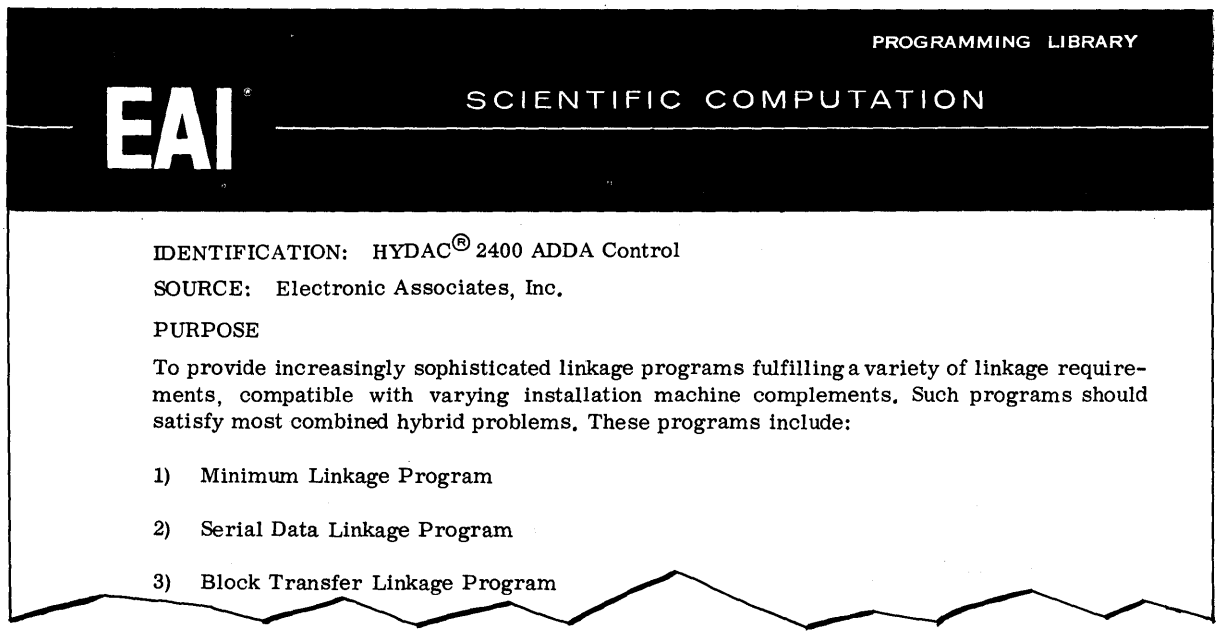


Figure 3: Excerpt from Representative Linkage Program