AD-A083 318 ROME AIR DEVELOPMENT CENTER GRIFFISS AFB NY F/G 9/2

ITAS - A BETTER WAY OF CODING.(U)

MAR 80 J FELDMAN

UNCLASSIFIED RADC-TR-80-67 NL

1 OF 1

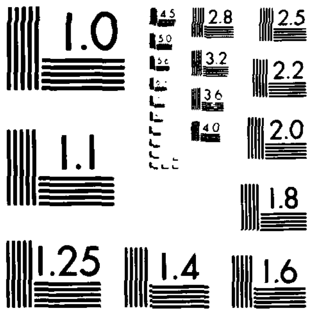
AD A083318

END

DATE FILMED

5-80

DTIC


1.0
4.5
5.0
5.6
6.3
7.1
8.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

RADC-TR-80-67
In-House Report
March 1980

ITAS - A BETTER WAY OF CODING

John Feldman

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

RADC-TR-80-67 has been reviewed and is approved for publication.

APPROVED: [illegible]

THADEUS J. DOMURAT
Chief, Intelligence Applications Branch
Intelligence and Reconnaissance Division

APPROVED: [illegible]

OWEN R. LAWTER, Colonel, USAF
Chief, Intelligence and Reconnaissance Division

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. ~~REPORT NUMBER~~ (14) RADC-TR-8Ø-67	2. GOVT ACCESSION NO. AD-AØ83 318	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) (6) ITAS - A BETTER WAY OF CODING.		5. TYPE OF ~~REPORT & PERIOD COVERED~~ (9) Technical In-House Report,
		6. PERFOR~~MING ORG. RE~~PORT NUMBER N/A
7. AUTHOR(s) (10) John Feldman		8. CONTRACT OR GRANT NUMBER(s) N/A
9. PERFORMING ORGANIZATION NAME AND ADDRESS Rome Air Development Center (IRAE) Griffiss AFB NY 13441		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS (16) 2Ø110001 (17) ØØ 31011G
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (IRAE) Griffiss AFB NY 13441		12. ~~REPORT DATE~~ (11) March 198Ø
		13. NUMBER OF PAGES 55
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Same (12) 61		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Interactive Programming
Modeling Techniques
Programming Languages
Simulation Techniques

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes a software program which can be used as an alternative to print oriented user languages such as BASIC and FORTRAN. The design and use of this "language" is described. The language uses a basic kernel of graphic symbols interconnected by the user to designate data flow within the program. This allows the Itas user to simulate and model scientific and engineering problems directly.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

309Ø5Ø

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(*When Data Entered*)

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(*When Data Entered*)

TABLE OF CONTENTS

		Page
SECTION I		
	The Problem	1
SECTION II		
	The Solution	3
SECTION III		
	How to Use It	5
SECTION IV		
	The Vocabulary	7
	The Arithmetic Operators	7
	Statistical Operators	7
	Boolean/Logical Operators	7
	Control Operators	7
	Complex Operators	8
	Models	8
SECTION V		
	Special Features	9
	Paging	9
	Zooming	9
	Interpolation	10
SECTION VI		
	The Design	11
	The Itas Environment	11
	The Graphics Module	11
	The Graphics Data Base	13
	The Execution Module	13
	Execution Module Data Base	14
SECTION VII		
	The Future	15
	Multiple Outputs	15
	Macroing	15
	Postscript	16

TABLE OF CONTENTS (Cont'd)

		Page
APPENDIX A		
	Examples	17
APPENDIX B		
	The Itas Vocabulary	23
REFERENCES		55

LIST OF ILLUSTRATIONS

FIGURE		Page
1	An Itagram of a first order recursive digital filter	3
2	The Itas virtual paging concept	9
3	An example of interpolation	10
4	A simple Itagram	12
A-1	Itagram of first order recursive filter	17
A-2	Page one of the Itagram	19
A-3	Page two of Itagram	20
A-4	Page three of Itagram	21

PREFACE

This In-House Technical Report cannot be associated with any single contractual effort. It summarizes and generalizes results derived from several RADC contracts. For further information contact RADC (IRAE), Griffiss AFB NY 13441.

This report introduces a different concept in programming developed and implemented under several government contracts. The software described herein was produced by Dr. R. H. Cofer and his associates at the Harris Corporation in Melbourne FL under RADC sponsorship. Currently, this software, Itas is available for use on both the Univac 1110 and the DEC System-10 computers.

The software program described here is an example of the use of a visually oriented highly interactive block diagram user language, to provide an alternative to print oriented user languages such as BASIC FORTRAN, etc. The concept evolved from the need to provide a user transparent programming methodology for non-computer programmers.

The design and use of a symbolic man-machine interface language is described. The language, employing graphical symbols displayed on a cathode ray tube, eliminates the tedium and time delays associated with conventional language programming.

SECTION I

The Problem

Our objective was to develop a computer software system which can be used to stimulate the creativity of engineers and scientists who are engaged in solving data analysis and engineering application problems in a computer environment. It was apparent that the conventional approach these engineers and scientists were employing to interface to the computer, i.e., writing and using English language (e.g. FORTRAN) programs did not satisfy the objective. Characteristically, an engineer would depend on a computer programmer to translate his problem into the appropriate algorithm, debug the program, and provide the polished product to the engineer. This process is cumbersome and inefficient because of the temporal gap between the statement of the problem and the availability of results for analysis. As a result, the engineer's ideas, perception and intuition for the specific problem that he was attempting to solve becomes diluted. This situation is exacerbated by the existence of the programmer middleman who further isolates the engineer from the computer solution implementation. In seeking a method of improving the situation, we naturally focused our attention on visually oriented communications media, being guided by the ageless adage about the relative merits of pictures and words.

SECTION II

The Solution

First, the general approach and procedures followed in implementing specific engineering problems were analyzed. We found that this process can be viewed as a two phased process; a solution formulation phase followed by a solution implementation phase.

Of the two phases, the interesting phase is usually the formulation phase with the actual implementation phase a straightforward, vapid process. The answer to the problem described in Section I above, therefore, should de-emphasize or eliminate the software implementation phase and concentrate on focusing the user's attention on the formulation process. We have designed and coded a software program which achieves this objective. Our program is named Itas.

Itas is a computer program which enables the user to create other computer programs.

The Itas concept derives from the fact that topological graphs (i.e., system block diagrams, flow charts, logic networks, electronic schematics, etc.) are frequently employed in the description of engineering application problems. This is so because topological graphs simultaneously communicate, instruct, report and store information in a very readable and suggestive format.

The Itas system is based on an analog computer metaphor. The graphic elements (building blocks) are arithmetic operators, Boolean operators, calculus operators, control functions, and specific models. Each element has input/output stubs that designate data flows in the program. An example of an Itas program, an Itagram, is shown below:

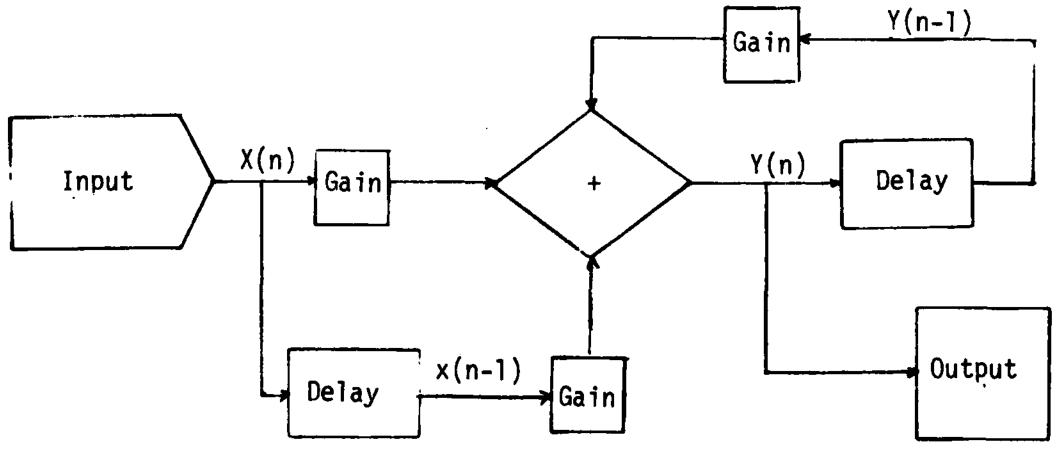


Figure 1 - An Itagram of a first order recursive digital filter.

Using a light pen, the user selects Itagram elements from an extensive menu, connects them, assigns values to the inputs, and watches the network execute, all in real time. Itas helps the user to visualize several events occurring simultaneously and to evaluate the effects of their interaction with each other. Unlike Itas, conventional computer programming is often tedious work. It requires an intimate knowledge of the capabilities of the programming language, the efficiencies of the compiler/assembler, and sometimes even the organization of the computer in order to generate the desired algorithm. Itas, however, is so close to the process of formulating the solution that the user's interest is maintained through the algorithm implementation process. In fact, the coding implementation process is transparent to the Itas user. The Itagram code is automatically compiled by Itas, and the program is ready for execution.

SECTION III

How to Use It

Itas is designed to closely follow a natural approach to problem solving. The Itas user first describes his problem on paper in terms of a functional diagram consisting of mathematical interrelationships among variables, i.e. in terms of mathematical equations. The result will be a diagram similar to Figure 1, but will probably not use the exact complement of Itas elements. This diagram is next translated into an Itagram using the Itas vocabulary given in Appendix B. This step is critical and is both the strongest and weakest link in applying the Itas technique. This is where the user's ingenuity, inventiveness and skill are put to the test. The variable(s) of interest need not be isolated explicitly because Itas has an implicit equation solution capability. With the paper Itagram complete, the user is ready to use the Itas program.

Sitting at the graphics terminal, the user transcribes the paper Itagram onto the CRT screen. This is accomplished interactively with the aid of an input device such as a light pen. The visual nature of the man-machine interaction causes the user to actively participate in the algorithm implementation phase. In fact, the user is in control, with the Itas software only a tool in his hands. The Itagram is designed using specific rules and conventions. When the Itagram is complete, it is ready for immediate execution at the user's command. The results are available for inspection immediately. Herein lies the most powerful attribute of Itas. We are using the computer in a positive active role, i.e. the user has to determine why an answer is wrong and take action to correct the fault. This is accomplished simply by recalling the Itas diagram, editing its structure, and reexecuting.

The Itagram can be saved for future reference or for archival purposes. A hard copy of the Itagram can be output for use in publications/reports.

SECTION IV

The Vocabulary

Creating a useful capability required a kernel of symbolic elements to be defined. To that end, the initial Itas vocabulary consisted of twenty symbols. The current Itas program comprises an expanded set of fifty-one symbols. For convenience, we will divide the set into six categories: Arithmetic Operators, Statistical Operators, Boolean Operators, Control Operators, Complex Operators, and Models. Most of these operators accept up to four inputs. However, all provide a single output.

The Arithmetic Operators

Nineteen operators are available for use in Itagrams. These are: Addition, Subtraction, Multiplication, Division, Sine, Arcsine, Cosine, Arccosine, Tangent, Arçtangent, Exponentiation, Square Root, Raising to a Power, Natural Logarithm, Common Logarithm, Absolute Value, Constant, Gain, and the Modulus Operation. The functions are self-explanatory.

Statistical Operators

This category comprises of five operators: Summation, Summation of Squares, Summation of Product, Counting, and Random Number Generation. These operators are useful in calculating simple statistical information.

Boolean/Logical Operators

The seven operators in this categroy provide the capability to perform logical operations within Itagrams. These operators are the logical AND, NAND, OR, NOR, NOT functions and the FLIP-FLOP, and COMPARE functions. Boolean operators function with two inputs where applicable.

Control Operators

The seven operators in this category control the flow of data into, within, and out of the Itagram. The operators are: Input, Output, Time Generator, Display Output, Kill, Pause, and Run Time Estimator. The Input/Output elements are the primary vehicles for entering/retrieving data files into/from the Itagram. The data is transferred via file names. The Time Generator provides the incremental cycle time for each execution cycle, i.e. Ti = To + (i+1) T. The Kill and Pause elements allow stopping execution when a prespecified condition occurs. The function of the Run Time Estimator is to generate an estimate of the approximate execution time for the Itagram.

Complex Operators

The complex operators are the following: Integration, Differentiation, Time Delay, Sample and Hold, Implicit Equation Solver, Multiple Arithmetic operator, Function Generator, and Variable Equation operator.

The Implicit Equation Solver consists of two elements used together to provide an interactive solution to an Implicit Equation where it is impossible to segregate the desired output variable on one side of the equation. The Multiple Arithmetic operator provides the means for performing the four simple arithmetic operations, i.e. addition, subtraction, multiplication, and division, in a single graphical entity. The Function Generator allows the user to specify a functional relationship between two parameters creating a piecewise linear approximation to that relationship. The Variable Equation Operator allows the user to designate compound arithmetic functions within one Itagram element. The allowable arithmetic functions are any of the nineteen operators defined in the arithmetic category above, e.g. $Y = \mathrm{SQRT}\ (A) + \mathrm{COS}^2\ (B) + \mathrm{LOG}\ (CD)$ where A, B, C and D are the four inputs to the Variable Equation operator and Y is the output.

Models

This category includes three model elements. These were included to support the specific Itas user. The models include an Atmospheric Model, an Earth Model and a Gravity Model. All models require two inputs representing geometric altitude and geocentric latitude. The model outputs are user selectable and include constants and variables such as atmospheric temperature, atmospheric pressure, density, earth rotation rate, gravitational constant, flattening factor, eccentricity, radial and transverse components of gravitational force, among others.

SECTION V

Special Features

The following features are included in Itas for the convenience of the user: paging, zooming and interpolation.

Paging

The increasing complexity of large Itas networks has been dealt with through the use of a virtual paging concept (Figure 2). The Itas paging concept enables large Itagrams to be displayed on the CRT at one time--a sort of wide-angle look. Employing paging, the user can build an Itas network of several (up to nine) display pages where each display page is identified by its position in a virtual matrix array of pages. The Itas network can be started on a given page and continued into adjoining pages using designated connector elements, which link related elements on different pages. This concept extends the maximum number of Itas elements allowable per Itagram beyond the physical dimension of the CRT display.

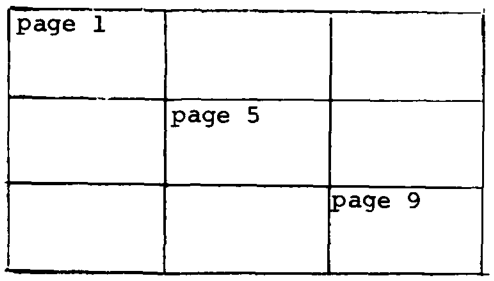


Figure 2 - The Itas virtual paging concept. The pages are numbered and are accessible one at a time in the Edit mode. Each square covers the entire display area.

Zooming

Zooming allows the simultaneous display of any four adjoining pages in an Itagram. While in the zoom mode, the user can also display all nine pages of the Itagram. Editing is not permitted in this mode. Zooming is only provided as an aid to maintaining proper perspective of large Itas networks.

Interpolation

Since Itas uses discrete data files as inputs, it is imperative that proper time synchronization be maintained among the various data inputs. Therefore, data values must be available at given points in the Itagram at identical time intervals. Since this is not always a true condition, the Itas interpolation feature automatically interpolates between data points. Itas interpolation allows the user to specify the closeness of fit desired. An N-th degree Lagrangian polynomial interpolator is provided.

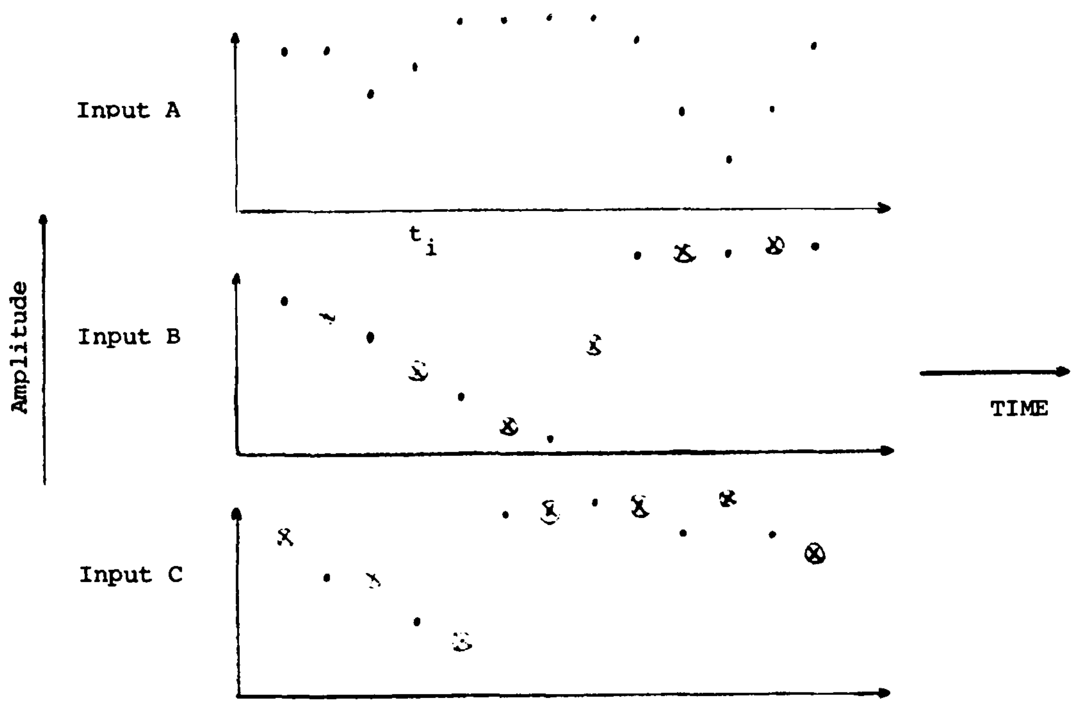


Figure 3 - An example of interpolation. Input B does not have an assigned value at times t_i, t_{i-2}, t_{i+2}. Input C does not have an assigned value at times t_{i-1}, t_{i+1}, t_{i+3}, etc. Itas will automatically assign values at these times (x's). The closeness of fit is specified by the user prior to execution. Itas automatically defaults to a straight line (N=1) interpolation.

$\otimes$= Interpolated Value

SECTION VI

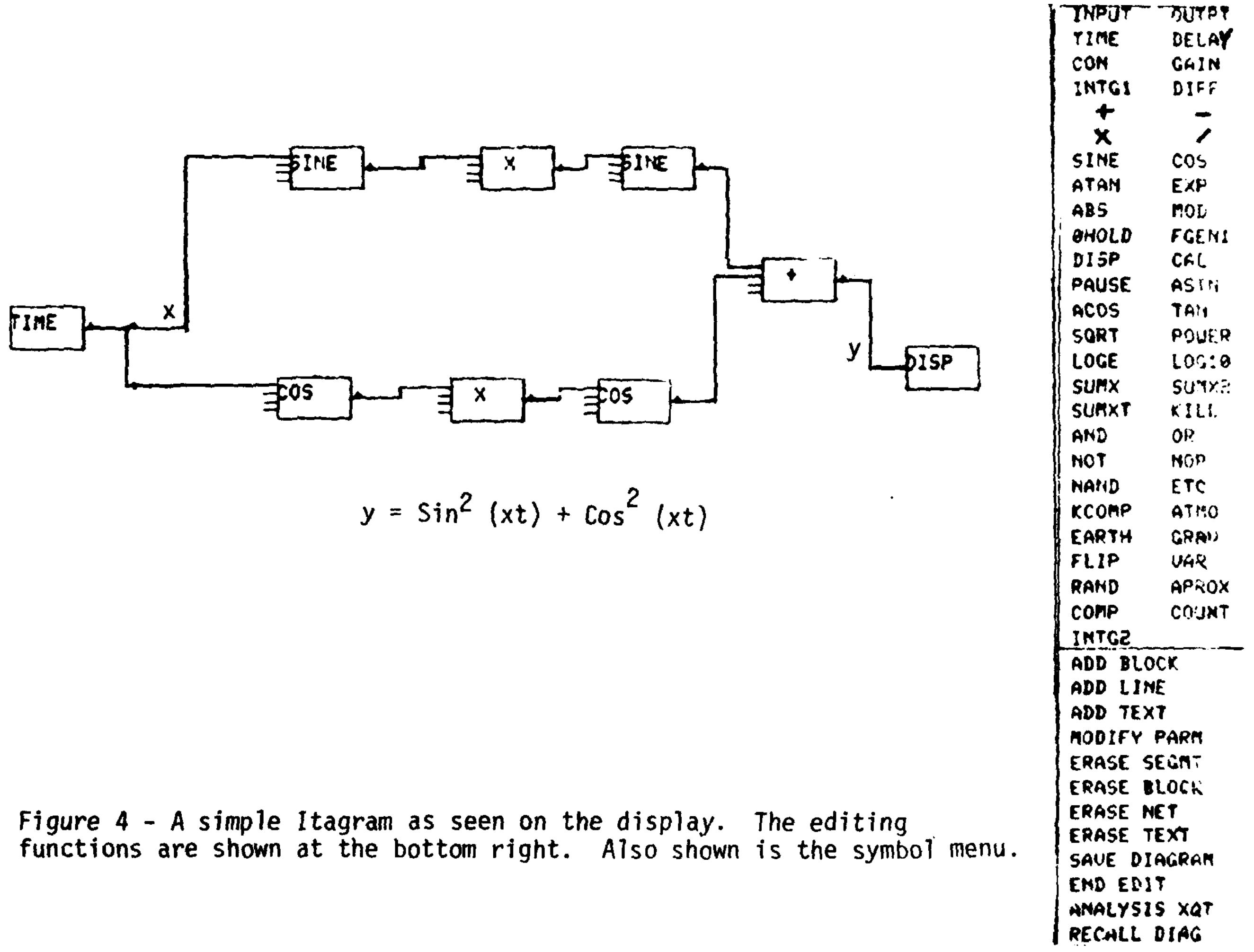
The Design

The Itas Environment

The Itas software was developed for use on a Digital Equipment Corp PDP-10 computer with TOPS-10 time sharing operating system. Itas was required to be imbedded into an existing software program operating under the TOPS-10 monitor. The name of that program is DTEA. Itas is accessed from the first level DTEA menu and returns to DTEA at the same level. The current Itas employs the Sanders Associates ADDS-900 graphic display terminal as the man-machine interface. The software is coded in Fortran IV and is structured and modularized. The software is coded to permit future conversion to reentrant operation. The Itas software consists of two basic modules and supporting data bases. These are the graphics module and the execution module.

The Graphics Module

The graphics module interfaces with the user in generating the Itagram. It maintains the graphical data base necessary for construction of a visual image of the Itagram. The module provides the necessary editing functions (See Figure 4). The generation of a complete Itagram is the process of positioning and joining various Itagram operators and connecting lines into a compilable, executable entity.



$$y = \mathrm{Sin}^2\,(xt) + \mathrm{Cos}^2\,(xt)$$

Figure 4 - A simple Itagram as seen on the display. The editing functions are shown at the bottom right. Also shown is the symbol menu.

The Graphics Data Base

The graphics data base supports the graphics module and is comprised of five data tables: the item table, the symbol table, the net list table, the net store table, and the bulk store table. Entries in all tables are indexed and cross referenced by data pointers. The table contents are as follows:

Item Table - All displayable Itagram related data is composed of units called items. An item in the table can be either a symbol or a line network.

Symbol Table - The symbol table maintains information on each symbol (i.e. operator) in the Itagram. One table entry is included for each symbol.

Net List Table - This table maintains a list of all line segments connected to a given symbol.

Net Store Table - Each entry in this table describes a line network in the Itagram.

Bulk Store Table - This table is used to store all remaining data required to completely reconstruct the Itagram. This includes relative positions of symbols and connecting line networks on the CRT screen, as well as all annotative textual information.

The Execution Module

Diagram execution is accomplished by a table driven compiler. The table driven compiler consists of a main program and all subroutines and functions that could possibly be used in any Itagram. The execution module's preprocessor generates a run table that determines the order of execution of the various operators. The ordering algorithm scans the run table and calls the appropriate operators in the correct sequence.

The execution module is transparent to the user. Its primary function is translation of the graphical data base to an executable computer program. Prior to execution of a given Itagram, various input parameters are specifiable by the user by interacting with the execution module menu. Consequently, a given Itagram can be executed repeatedly with different start/stop times and/or different input data files.

Execution Module Data Base

The components of this data base are the block table, the indirect driver list, the parameter list, and the ordering table. The contents of the block table, the indirect driver list and the parameter list are modified versions of their counterparts in the graphics data base. The block table contains an entry for each graphical element in the Itagram. The indirect driver list associates Itagram input parameters defined at execution time with specific entries in the block table. The parameter list contains all parameters associated with a given graphical element. The ordering table is created within the execution module by the ordering algorithm. The ordering table is the heart of the execution module; it defines the program execution sequence within each time cycle.

SECTION VII

The Future

The following features are potential improvements or extensions of Itas: (1) allowing multiple outputs per symbol, and (2) incorporating a macroing capability.

Multiple Outputs

A limitation in the current version of Itas is the inability to designate several outputs from a given element. The restriction to one output from any symbol limits the full exploitation of the Itas concept. For example, availability of multiple outputs per symbol will accommodate a conditional branching capability in Itagrams. More significant, however, is the potential utility of multiple outputs in generating a macro-Itas capability.

Macroing

A macro is a mini-Itagram i.e. a user defined Itas symbol that replaces a set of connected existing Itas symbols. A section of an Itagram could then, for example, be defined by the user as a new Itas macro element, thus becoming an entry in the Itas symbol menu (See Figure 4), which is recallable and useable in other portions of the Itagram. For example, the first order recursive digital filter defined in Figure 1 can be defined as a macro, becoming a simple graphical entity with one input and one output, thus allowing the generation of an uncluttered cascaded higher order digital filter that is simple in appearance, and occupies the identical CRT area currently required to define the first order filter.

Postscript

The Itas software with its graphical Itagram format is successfully implemented in a scientific engineering environment. It is currently operational on two different computer systems, the DEC System 10 and the Univac 1110. Itas is being continually improved and refined while undergoing on-the-job evaluation. Its application to diverse disciplines is easily achievable because of its design which allows, among other things, for the systematic addition of new and special symbols to the basic structure.

APPENDIX A

Examples

Example No. 1 - A first order digital filter.

Consider a first order recursive digital filter (low pass) of the form:

(A-1) $Y(n) = a_0 \; x(n)+a_1 x(n-1)-b_1 y(n-1)$

where $x(n)$ is the value of the n^{th} input sample, the coefficients a_0, a_1 and b_1 determine the response, and $y(n)$ is the filter response to the input, $x(n)$. The implementation of this filter is straight forward and is shown in figure A-1.

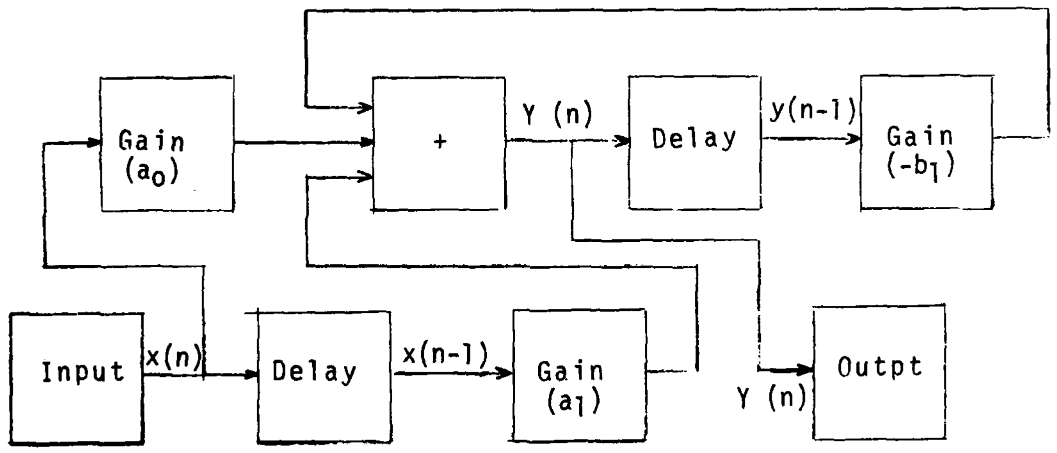


Figure A-1. Itagram of first order recursive filter.

Example No. 2

Given a set of values (z_i, y_i) i+o,1...N, the least square line approximation is given by:

(A-2) $Y=a_0 + a_1 z$

where , $$a_0 = \frac{\left(\sum_{i=0}^{N} y_i\right)\left(\sum_{i=0}^{N} z_i^2\right) - \left(\sum_{i=0}^{N} z_i\right)\left(\sum_{i=0}^{N} z_i y_i\right)}{B}$$

$$a_1 = \frac{N \sum_{i=0}^{N} (z_i y_i) - \left(\sum_{i=0}^{N} z_i\right)\left(\sum_{i=0}^{N} y_i\right)}{B}$$

and $$B = N \sum_{i=0}^{N} z_i^2 - \left(\sum_{i=0}^{N} z_i\right)^2$$

The Itagram which implements this solution is designed on three Itas pages These pages are shown in Figures A-2, A-3, A-4. The paging capability is used here for purposes of illustration. The entire Itagram can be drawn on one page. This Itagram does not represent the most efficient (i.e. least number of elements) Itas format. The use of the Calculator operator, for example, will reduce the number of elements but it would also make the data flow more complicated.

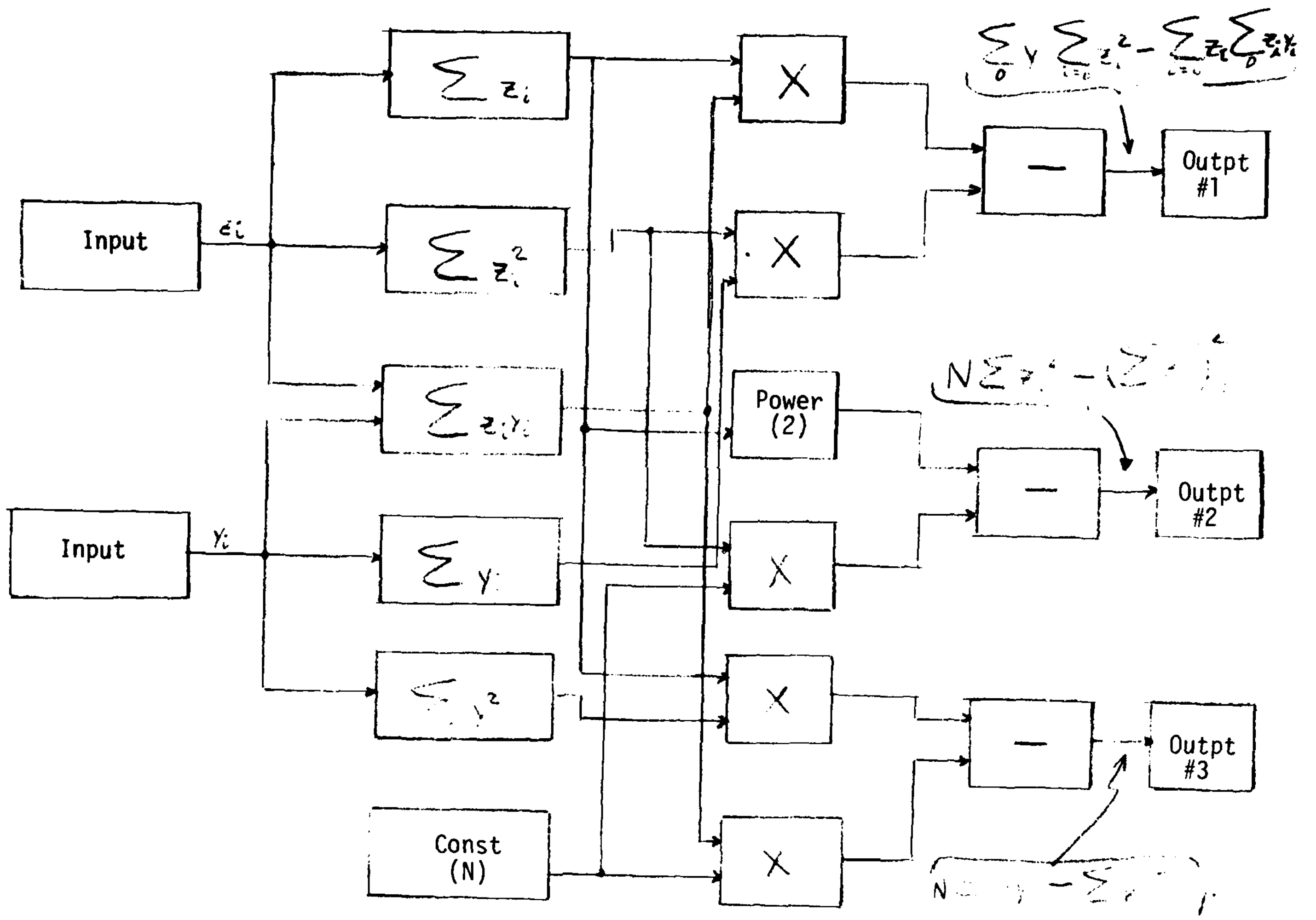


Figure A-2 - This is a portion (one page) of the Itagram which can be used to solve equation A-2.

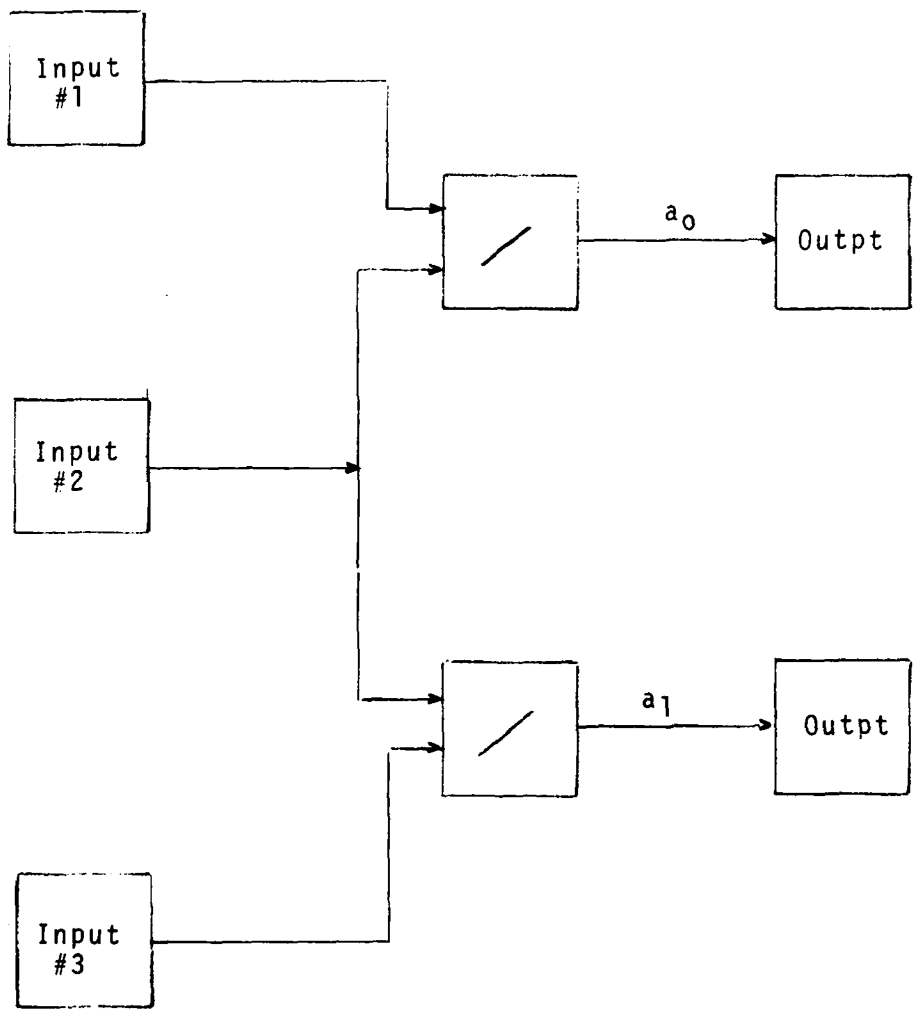


Figure A-3 - This is the second page of the Itagram for solving equation A-1

The KILL block terminates execution in an orderly fashion (i.e. closes all files) whenever the input value is ≥ 0.

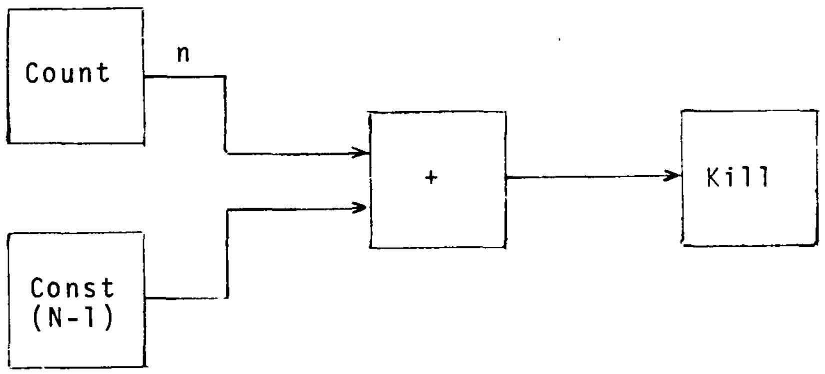


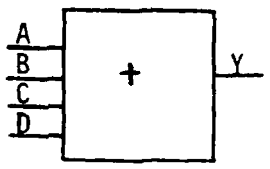
Figure A-4 - This is the final page of the Itagram solving equation A-1. This page is separate from the main flow and its function is to terminate the program when N sets of values have been processed.

APPENDIX B

The Itas Vocabulary

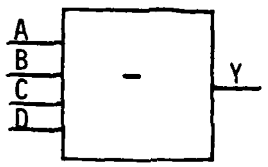
Elementary Arithmetic Group

1. Adder



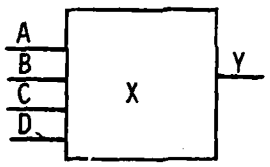
The sum, Y=A$\pm$B$\pm$C$\pm$D is formed where at least input stub A is connected.

2. Subtractor



The difference, Y=A-B-C-D is formed where at least the input stub A is connected.

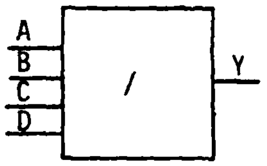
3. Multiplier



The product, Y=AxBxCxD is formed where at least one of the input stubs is connected.

Note: The parameters referred to in this appendix are available for modification via the "Modify Parm" function in the Itas menu (see figure 4). This action is usually accomplished after the Itagram is drawn on the CRT screen. All Itas blocks contain default parameters. Parameters need only be modified when required.

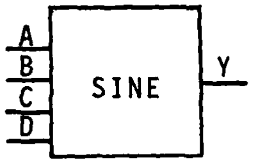
4. Divider



The quotient, Y= A/B/C/D is formed where the input stub A must be connected.

If one or more of the input stubs B, C, or D is connected and sometime during the execution has a value of 0.0, the run is terminated.

5. Sine

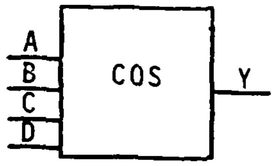


The sine, Y=sin[A*B+C*D] is formed where at least one of the input stubs is connected.

If A and B or C and D are not connected then their product is 0. If either A or B is connected, but not both, then the product A*B = A or B depending on which one is connected; similar logic is applied to C and D.

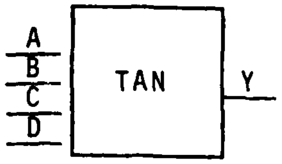
Inputs are in radian measure.

6. Cosine



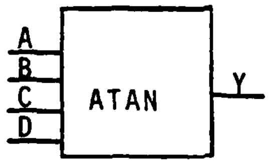
The cosine, Y=cos|A*B+C*D| is formed with logic similar to that used by the SINE symbol discussed previously.

7. Tangent



The tangent, Y=tan |A*B+C*D| is formed by first finding the sin |A*B+C*D| and the cos |A*B+C*D| and then forming Y=sin |A*B+C*D| /cos |A*B+C*D| , where the cos |A*B+C*D| $\neq$ 0. The method used to evaluate the argument is similar to that used by the SINE function.

8. Arctangent

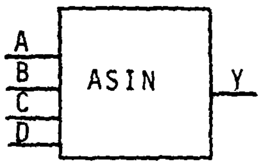


The arctangent, Y=tan^{-1} |A*B+C*D| is formed with restrictions on the input stubs similar to those discussed under the SINE function in this paragraph.

The output Y is in radians and is in the range

$-\frac{\pi}{2} < Y \leq \frac{\pi}{2}$.

9. Arcsine

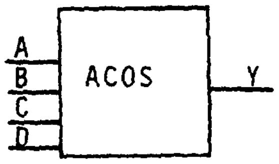


The arcsine, $Y=\sin^{-1}|A*B+C*D|$ is formed by finding $\tan^{-1}\left\{|A*B+C*D| / SQRT\left(1-|A*B+C*D|*|A*B+C*D|\right)\right\}$

The restrictions on the input stubs are similar to those imposed by the SINE function with the additional stipulation that if at any time $|A*B+C*D|*|A*B+C*D|$ is 1 or larger, then the execution is terminated.

A parameter is provided for but not used.

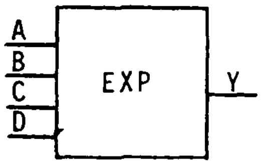
10. Arccosine



The arccosine, $Y=\cos^{-1}|A*B+C*D|$ is formed.

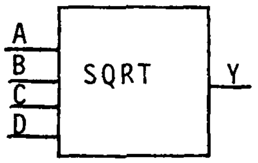
The input argument must be $\leq |1|$.

11. Exponential



The exponential, Y= EXP [A*B+C*D] is formed where the evaluation of the input argument is done in the manner discussed in connection with the SINE function.

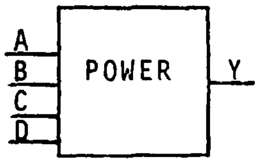
12. Square Root



The square root, Y=[A*B+C*D]**0.5 is formed where the input argument is computed in the same manner as is the input to the SINE function.

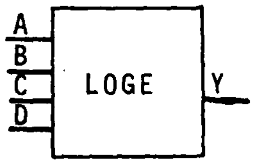
If, during the execution of a diagram, the argument is less than 0.0, the run is aborted.

13. Power



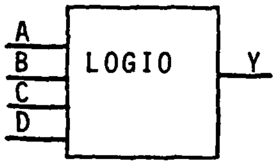
Raises the input argument to a power, $Y=(A*B+C*D)**P_2$ where P_2 is the second parameter entered. The first parameter is ignored and therefore should be left at its default value. The computation of the input argument in the program is rather complex and, consequently, the user must exercise great care when connecting the input stubs A, B, C and D. The program packs the input drivers in order of their placement on the symbol into the stubs A then B then C and then D. It is from this final order, not necessarily the original order, that the input argument is computed. It is best to use the inputs in order and not to skip any.

14. <u>Natural Logarithm</u>



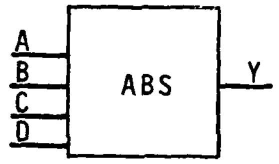
The natural logarithm, $Y=\text{LOGE}(A*B+C*D)$ is formed where the input argument is handled in the same way the SINE function handles it.

15. Common Logarithm



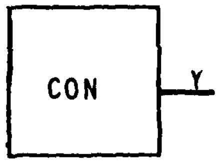
The logarithm base 10, Y=LOG 10(A*B+C*D) is formed where the input argument is determined the same way as the SINE function.

16. Absolute Value



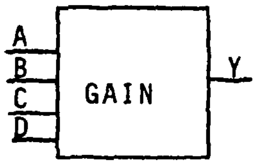
The absolute value, Y= ABS(A*B+C*D) is formed where the input argument is determined in a similar manner to that described under the SINE function.

17. Constant



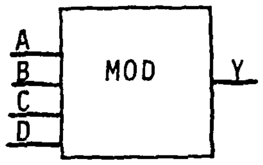
A constant value, Y= P_1 is formed where P_1 is the value of the parameter.

18. Gain



The gain, $Y=P_2*(A+B+C+D)$ is formed where the gain, P_2, is the second parameter, the first parameter is ignored, and any unconnected inputs stubs are ignored.

19. Modulus

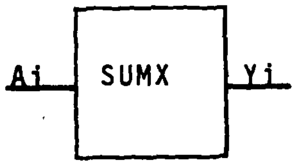


The modulus function, $Y=AMOD(A*B+C*D\ ,P_1)$ is formed where the computation of the input argument follows the rules described for the POWER function.

The modulus value is equal to the input parameter value.

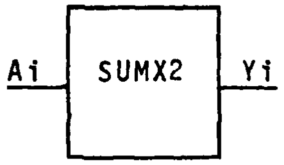
Statistical Group

20. Summation Single



A running sum, $Y_N = \sum_{i=1}^{N} A_i = Y_{N-1} + A_N$ is formed where the present sum is equal to the previous sum, plus the present input.

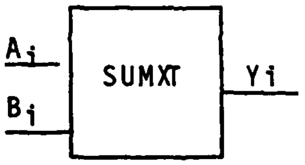
21. <u>Summation of Square</u>



The running sum of the input squared,

$$Y_N = \sum_{i=1}^{N} A_i * A_i = Y_{N-1} + A_N * A_N$$ is formed where the present output is equal to the previous output plus the square of the present input.

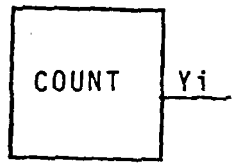
22. <u>Summation of Product</u>



The running sum of the input product,

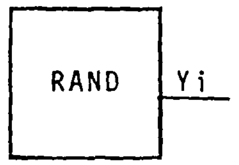
$$Y_N = \sum_{i=1}^{N} A_i * B_i = Y_{N-1} + A_N * B_N$$ is formed where the present output is equal to the previous output plus the product of the present inputs.

23. Cycle Counter



The present cycle number, $Y_i = i$ is formed where i is the present cycle number.

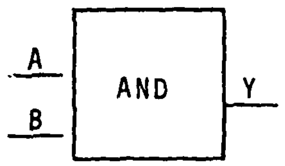
24. Random Number Generator



The pseudo-random number, $Y_i = f(Y_{i-1})$ is formed where the current random number is a function of the previous random number. The first random number Y_1, is a function of the input parameter, which acts as the initial seeding.

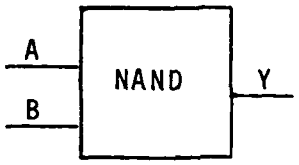
Logical Group

25. AND



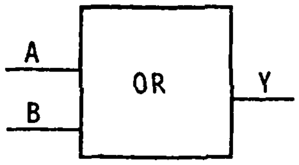
The logical AND, $Y = A \cdot B$ is formed where the inputs A and B are required. An input is treated as a 1 if it is greater than 0.0. Otherwise, it is treated as a 0.

26. NAND



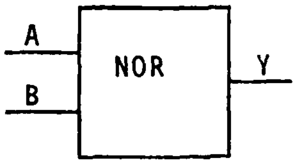
The logical NAND, $Y=\overline{(A \cdot B)}$ is formed where the requirements on, and the meaning of, the inputs are the same as those outlined previously for the AND function.

27. OR



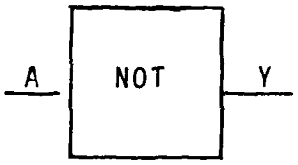
The inclusive OR, $Y=A+B$ is formed where the requirements on, and the meaning of, the inputs are as outlined under the AND function.

28. NOR



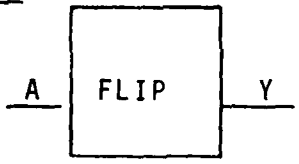
The not inclusive OR, $Y=\overline{(A+B)}$ is formed where the restriction on, and the meaning of, the inputs are as discussed under the AND function.

29. NOT



The NOT function, $Y=\bar{A}$ is formed where the input is a 1 if it is greater than 0.0, else a 0.
The output is the one's complement of the input.

30. Flip-Flop

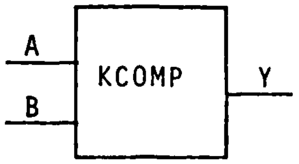


The Flip-Flop function is defined as:

$$Y_i = \begin{cases} \bar{Y}_{i-1} & \text{when } A_{i-1} = 1.0 \text{ and } A_{i-2} = 0.0 \\ Y_{i-1} & \text{all other times} \end{cases}$$

where the input is a 1.0 or 0.0 as explained above.
The initial output of the Flip-Flop is set equal to the parameter value setting (1.0 or 0.0).

31. KCOMP



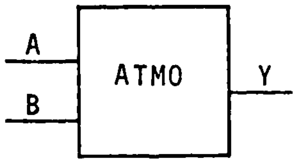
The sum of the inputs is compared to 0.0 as follows:

$$Y = \begin{cases} 0.0 \text{ when A plus B is} \geq 0.0 \\ 1.0 \text{ when A plus B is} < 0.0 \end{cases}$$

where the two inputs must exist; if not, the program is aborted.

Model Group

32. Atmospheric Model



Generates the value of a user specified parameter, chosen from a list of five, at the input altitude and latitude.

$Y = F(A, B, P_2)$. Input A must be connected, indicating the geometric altitude in kilometers.

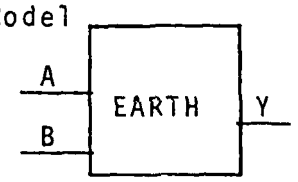
Input driver B is an optional input and identifies the geocentric latitude in radians; when input B is not attached, then a default latitude of $\pi/4$ is used by the model.

Parameter 1 is not used.

Parameter 2 governs the output as follows:

$P_2 = 1$,	Y=	temperature in Kelvin degrees
$P_2 = 2$,	Y=	pressure in Newtons per meter squared
$P_2 = 3$,	Y=	density in kilobars per meter cubed
$P_2 = 4$,	Y=	speed of sound in meters per second
$P_2 = 5$,	Y=	dynamic viscosity coefficient

33. Earth Model



Generates the value of a user specified parameter, chosen from a list of nine parameters.

Driver A inputs the geometric altitude in kilometers and need be present only when the computation requires altitude, as in computing distance to the center of the earth (P_2=8 or P_2=9), for example, Driver B contains geocentric latitude which is only required in the computations associated with

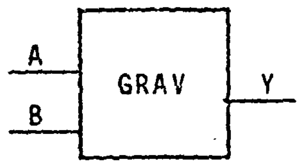
$P_2=8$ or $P_2=9$. If driver B is not present, then a default of $\pi/4$ radians is used if latitude is required.

Parameter 1 is provided for but not used.

Parameter 2 governs the output, Y, as follows:

P_2 = 1, Y =	earth rotation rate in radians per second
P_2 = 2, Y =	gravitational constant in meters cubed per second squared
P_2 = 3, Y =	semi-major axis in meters
P_2 = 4, Y =	semi-minor axis in meters
P_2 = 5, Y =	mass to weight ratio
P_2 = 6, Y =	flattening factor
P_2 = 7, Y =	eccentricity
P_2 = 8, Y =	radial distance to surface point in meters
P_2 = 9, Y =	radial distance to point in atmosphere in meters

34. <u>Gravity Model</u>



Generates the components of the gravitational force based on zonal harmonics through the fourth harmonic component.

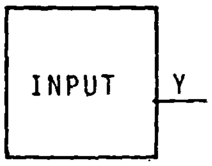
Driver A, is required, and provides the geometric altitude in kilometers.

Driver B, is optional, and provides the geocentric latitude in radians; a default of $\pi/_4$ is used. Parameter 1, the only one provided, determines which component of the gravitational force is output.

P_1 = 0, Y = radial component per unit mass
P_2^1 = 1, Y = transverse component per unit mass

Control and Input/Output Group

35. Input



The Input symbol is used in two different ways by Itas; they will be considered separately.

1. INPUT block as an input file.

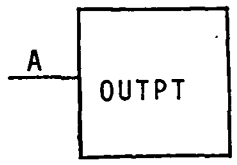
   The output, Y, is equal to the amplitude value of a time-amplitude sample from the requested input file. The user specifies, using options on the Analysis-Execution menu, the stream of expected sample times; if a sample does not occur at an expected time, the system automatically interpolates to the requested time.

The value of the single parameter determines the input file which will be used by an INPUT block. Parameter values of 1 through 4 specify corresponding input files as defined in the Analysis-Execution menu.

2. INPUT block - page connector

   The INPUT block may get its input from another page of the diagram instead of a file. When used in this manner, the INPUT block must be coupled to an OUTPT block of a different page. The OUTPT-INPUT coupling is done by giving each block the same parameter value; negative values of the parameter are used to indicate paging.

36. <u>Output</u>



The OUTPT symbol is used in two different ways by Itas, each will be considered separately.

1. OUTPT block - POT file output

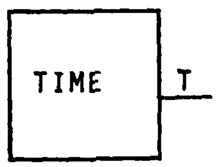
   The output is written to the designated file along with the generated time of the sample, (see INPUT block discussion).

The value of the single parameter determines which output file will be written. Parameter values of 5-8 are used as output file numbers and are defined by the user on the Analysis-Execution menu.

2. OUTPT block - as page connector.

   The OUTPT block may send its output to one or more pages of the diagram, instead of a file. When used in this manner the OUTPT block must be coupled to an INPUT block on another page. The OUTPT-INPUT coupling is done by giving both blocks the same parameter value; negative values are used to indicate that an OUTPT/INPUT block is used as a page connector.

37. <u>Time</u>

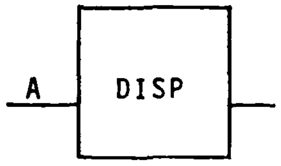


Generates the time, $T_i = T_0+(i-1)*\Delta T$, of the ith cycle of the analysis run.

T_0 = start time as specified by the user on the Analysis-Execution menu

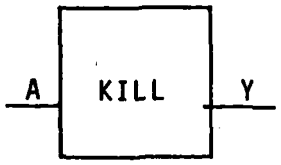
ΔT = time increment as specified by the user on the analysis-Execution menu.

38. Display



Generates a display, on the CRT of the input A to the DISP block. The frequency of the display is controlled by the user through the setting of the display ratio on the Analysis-Execution menu. A display ratio of N means a display every Nth cycle of the analysis run.

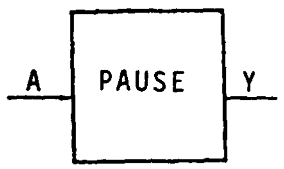
39. Kill



Generates a signal to the system, when the input A changes from negative to positive, that terminates the program run just as if the execution stop time had been reached.

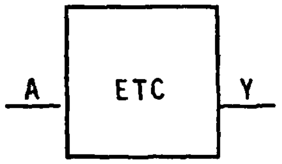
When the kill condition does not exist, Y=A and the KILL block effectively does nothing.
When the run is killed, a termination message is displayed on the CRT. The user can tell from the display when the run was aborted and which KILL block caused it to abort.

40. Pause



Generates a pause in the operation when the input A changes from less than to greater than Parameter 2.
When the pause condition does not exist, Y=A and the PAUSE block effectively does nothing.
At pause time, the CRT display indicates when and where the pause occurred. The user may either continue the run or kill the run. The kill procedure acts as though the stop time has been reached.
Two parameters are provided, the first is ignored and the second determines the pause condition.

41. Run Time Estimator



Estimates the time remaining to completion based on the time to execute ten cycles and the number of cycles in the complete program. An average time per cycle is computed over the first ten cycles and is assumed to remain constant over the run remaining cycles in the run.

$T_r = T_{10} * \left[\frac{Nc}{10} - 1\right]$ where T_r = time remaining after ten cycles.

T_{10} = time for ten cycles

Nc = total number of cycles

The estimated time is written on the CRT after the eleventh cycle is finished.

The output Y=A at all times.

Composite Functions Group

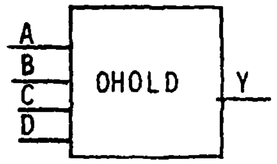
42. Delay

Provides the system with a one cycle delay,

$Y_i = A_{i-1}$ and $Y_0 = P_2$

Parameter 1 is ignored and the initial output is equal to Parameter 2.

43. Sample and Hold



This device holds and outputs a particular value until directed to change it by the input control signal.

Although the symbol indicates four input stubs, only two are used at any one time. Any two of the inputs can be selected each time the block is used. The upper-most connected stub is the control signal and the lower-most connected stub is the new sample value when the control signal indicates a new sample is to be used. The initial sample value is set equal to the value of Parameter 2.

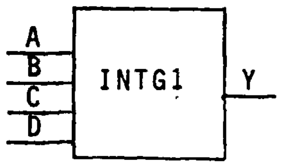
When the control signal A, for example, is 0, then the output Y=0. The value held by the device is not changed.

When the control signal A=0, then the output Y= value in the hold register.

When the control signal $A > 0$, the hold register is set to the value of the lower-most connected input stub; for example C, then the output Y is set to the value of C.

Parameter 1 is ignored by the software.

44. Trapezoidal Integrator



This device performs a trapezoidal integration with a one cycle delay. The input stubs are summed (no more than one need be connected) to form the input value.

The initial output, first cycle, of the integrator Y_0 is equal to Parameter 1.

The next output Y_1 is also equal to Parameter 1 since there is a one cycle delay in the output of the integrator.

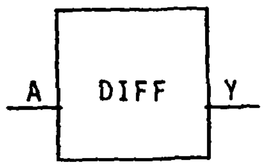
A second parameter is allocated to INTG1 and is used as an internal holding register; the user is not required to set P_2 to an initial value.

After two cycles the output of the integrator is computed by the recursion formula:

$$Y_n = Y_{n-1} + \frac{I_{n-1} + I_{n-2}}{2} * \Delta T$$

where $I_k = A_k + B_k + C_k + D_k$

45. <u>Differentiator</u>



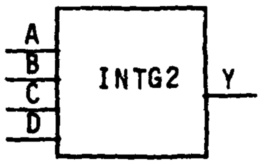
This device is a 3-point differentiator with no delay in its output. It will compute the derivative exactly for first and second degree polynominals.

Three parameters are provided for; the first is ignored, the initial output Y_0 is set to Parameter 2 and the second output Y_1 is set to Parameter 3. Parameters 2 and 3 are subsequently used as holding registers.

In general, the equation for the derivative is:

$$Y_i = (A_{i-2} - 4*A_{i-1} + 3*A_i)/2*\Delta T$$

46. <u>Simpson Integrator</u>



This device is a Simpson integrator where the initial output value is an input parameter and the second output value is computed using the trapezoidal process; subsequent integrations are performed using Simpson's rule.

The output of INTG2 after the second cycle is

$$Y_i = Y_{i-2} + \frac{\Delta T}{3} * \left[X'_{i-2} + 4 * X'_{i-1} + X'_i \right]$$

where $X' = A+B+C+D$.

The input stubs are summed, only one stub must be connected, to form the value of the derivative that is to be integrated.

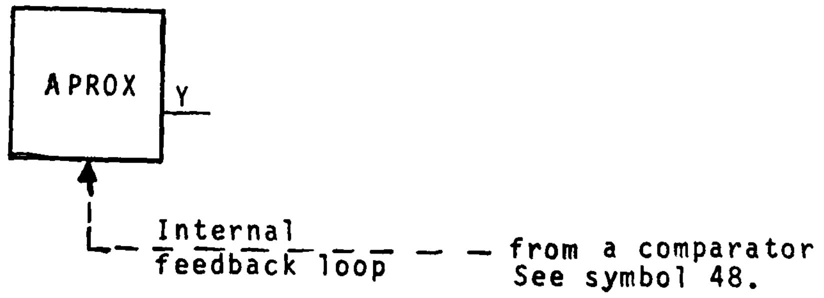
There are fourteen parameters associated with the INTG2 symbol, however, the user only sets Parameter 1, the remaining are used as internal holding registers. Parameter 1 is set by the user to the initial value of the integration.

While Simpson's rule requires only a record of the two previous derivatives and their integrations, it was decided to provide table space (parameter space) and program logic for the possible incorporation of a more complex integrator.

The stability of the two Itas integrators can, in general, be improved by decreasing the step size.

A way of decreasing the step size in Itas integrators is to interpolate the input data stream. However, while integration is improved by interpolation, the other Itas processes within a diagram would also be forced to operate on interpolated data, since all inputs are interpolated the same amount. However, the increase in processing time might prove to be a big drawback to excessive interpolation. It is possible to modify Itas to do the interpolation within the integrators themselves and thus remove the objections to a pre-processor interpolator.

47. <u>Implicit Equation Approximator</u>

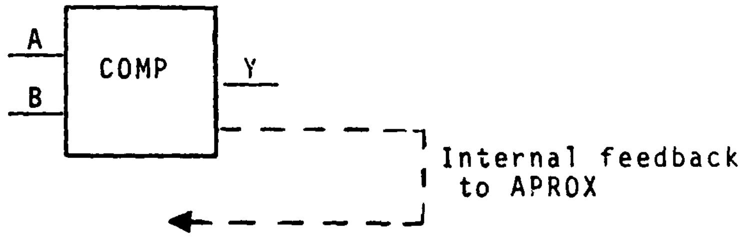


This device provides input approximation values to a set of Itas symbols that form an implicit equation.

There are two parameters associated with the APROX symbol; the first is not used and the second is set to an initial approximation of the solution of the equation. Subsequent cycles of the process are initialized with the solution to the previous cycle; this appears to be a better initialization than reusing the original input parameter value.

In each cycle, successive approximations of the function Y are internally fed back to the APROX block from the comparator until one of the following conditions exists(a) $Y_i = f(X,Y_i)$ and $|Y_i - Y_{i-1}| \leq \delta$ where Y_i is the function value of an iteration, or (b) the number of iterations exceeds the allowable number, in which case the diagram execution is stopped.

48. <u>Implicit Equation Comparator</u>

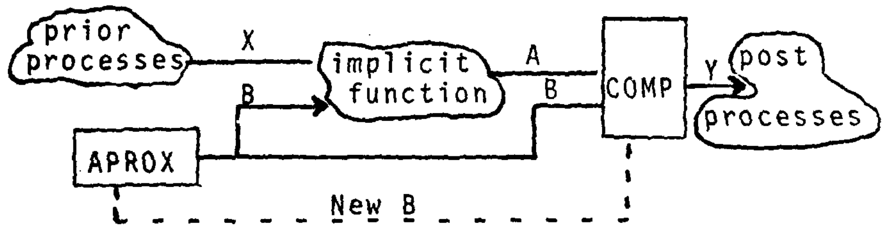


This device performs four functions in the Itas solution of an implicit equation:

1) checks the solution for convergence
2) calculates the next approximation to the solution when the current one is not sufficiently close to the previous solution.

3) feeds the new approximation back to the APROX block,
4) monitors the number of iterations per cycle.

The input stub A contains the present computed value of the implicit function and stub B contains the value used in the computation of A. Pictorially the process appears as:



A = F(X,B) if $|A-B| \leq \delta$ then Y = B

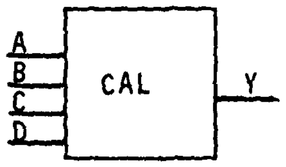
if not, then B = G(A,B)

There are four parameters associated with the COMP symbol:

P_1	not used
P_2	number of allowable iterations
P_3	the convergence delta(δ)
P_4	the fraction alpha used in computing the next value of B to use, as indicated:

$B = \alpha * A + (1-\alpha)*B$

49. 4-Function Calculator



This device computes a resultant value, Y, from up to four inputs, where the computations to be performed are governed by a set of parameters. The input drivers are, in effect, packed in order from top to bottom and unconnected stubs are ignored. In this discussion, if there were any two stubs connected they would be referred to as A and B; if any three were connected they would be called A, B, and C.

There are five parameters associated with the CAL block but, at most, P_2, P_3, and P_4 are used.

The correlation between input stubs and parameters is:

P_2 governs the relationship, f_1, between A and B

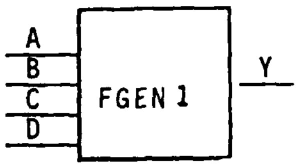
P_3 governs the relationship, f_2, between $f_1(A,B)$ and C

P_4 governs the relationship, f_3, between $f_2\left[f_1(A,B),C\right]$ and D

The functional meaning of the parameter values is:

$P_i = 1$	add the two values defined by $i = 2, 3,$ or 4
$P_i = 2$	subtract the two values defined by $i = 2, 3,$ or 4
$P_i = 3$	multiply the two values defined by $i = 2, 3,$ or 4
$P_i = 4$	divide the two values defined by $i = 2, 3,$ or 4

50. Function Generator



This device performs a linear interpolation, between user-set parameter values, based on the value of the input driver (although four input stubs are shown, only the upper-most one is used in forming the output Y). The output

$Y = f(A, P_{2i}, P_{2i+1})$ where i = 1, 2, 3, ...12

The first parameter is ignored and the remaining ten are paired as follows:

P_2, P_3	first abscissa, first ordinate
P_4, P_5	next abscissa, next ordinate
.	
.	
.	
P_{24} P_{25}	last abscissa, last ordinate

The set of parameters divides the continuum into six subregions as follows:

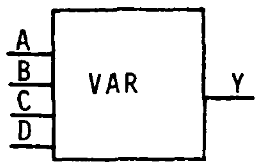
If $A < P_2$ then $Y = P_3$

If $P_2 \leq A < P_4$ then $Y = \frac{P_5 - P_3}{P_4 - P_2} * (A-P_2) + P_3$

.
.
.

If $P_{24} \leq A$ then $Y = P_{25}$

51. Variable Equation Generator



This device solves the equation Y = F(A, B, C, D) where at least one of the independent variables must exist. The function F may be a combination of Fortran operators and simple
algebraic and trancendental functions.
The equation is entered as though it were a string of parameters under the "Modify Parameter Option".
The equation must begin with Y = and may contain 80 characters.
The set of allowable symbols, characters and operators is:
A, B, C, D, SIN, COS, TAN, ASIN, ACOS, ATAN, EXP, ALOG, ALOG10, SQRT, +, -, *, /, **, =, (,), . , 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, and a blank.
The program has limited error checking capabilities; it monitors the equation for illegal functions, non-allowable characters, misplaced parentheses, an inconsistent set of parentheses and an illegal sequence of operators.

REFERENCES

1. H. Feldstein, Final Technical Report, F30602-77-C-0062, RADC-TR-78-271, "ITAS" dated January 1979, Harris Corporation, Electronic Systems Division, (B034567L).

2. D. W. Johnson, Final Technical Report, F30602-77-C-0210, RADC-TR-78-143, "Interactive Programming and Analysis Aids" dated June 1978, Harris Corporation, Electronic Systems Division, (B028910L).